A Formal Calculus for Informal Equality with Binding

Murdoch J. Gabbay and Aad Mathijssen

Abstract. In informal mathematical usage we often reason using languages with binding. We usually find ourselves placing capture-avoidance constraints on where variables can and cannot occur free. We describe a logical derivation system which allows a direct formalisation of such assertions, along with a direct formalisation of their constraints. We base our logic on equality, probably the simplest available judgement form. In spite of this, we can axiomatise systems of logic and computation such as first-order logic or the lambda-calculus in a very direct and natural way. We investigate the theory of derivations, prove a suitable semantics sound and complete, and discuss existing and future research.

1 Introduction

The theory of equalities $t = u$ is perhaps the simplest of all foundational logical theories. Informal specification of logic and computation often involves equalities with *binding* and subject to conditions about *freshness*. For example:

- λ-calculus: $\lambda x.(tx) = t$ if x is fresh for t
- π-calculus: $\nu x.(P \mid Q) = P \mid \nu x.Q$ if x is fresh for P
- First-order logic: $\forall x.(\phi \supset \psi) = \phi \supset \forall x.\psi$ if x is fresh for ϕ

and for any binder $\zeta \in \{\lambda, \nu, \forall\}$:

- Substitution: $(\zeta y.u)[x \mapsto t] = \zeta y.(u[x \mapsto t])$ if y is fresh for t

It is not hard to extend this short list with many more examples.

In the equalities above there are *two* levels of variable; x and y are variables of the system being axiomatised, we call these **object-level** variables; t, u, P, Q, ϕ, and ψ range over terms of that syntax, we call them **meta-level** variables. Unfortunately these equalities are subject to freshness side-conditions which make them something other than 'just equalities'.

Ways have been developed to attain the simplicity and power of the theory of equality between terms. For example we can work with combinators [1] or combinatory logic [2], cylindric algebra [3], higher-order algebra [4] or higher-order logic [5]. Roughly speaking: combinatory approaches reject object-level variables entirely; cylindric approaches also reject them as independent syntactic entities but enrich the language of term-formers to regain some lost expressivity; higher-order approaches model the difference between the two using a hierarchy of types. These approaches do not permit a *direct* representation of the two-level structure which informal syntax displays in terms such as $\lambda x.t$ or $\forall x.\phi$.

In this paper we describe **Nominal Algebra**. This is a logic based on equality which *embraces* the two-level variable structure by representing it directly in its syntax. Informal equivalences can be represented as axioms almost symbol-for-symbol. For example the equalities above are represented by:

- λ-calculus: $a\#X \vdash \lambda[a](Xa) = X$
- π-calculus: $a\#X \vdash \nu[a](X \mid Y) = X \mid \nu[a]Y$
- First-order logic: $a\#X \vdash \forall[a](X \supset Y) = X \supset \forall[a]Y$
- Substitution: $b\#X \vdash (\zeta[b]Y)[a \mapsto X] = \zeta[b](Y[a \mapsto X])$

Here a and b are distinct *atoms* representing object-level variables; X and Y are *unknowns* representing meta-level variables. Each equality is equipped with a *freshness condition* of the form $a\#X$ that guarantees that X can only be instantiated to a term for which a is fresh. The rest of this paper makes this formal.

In Sect. 2 we introduce the syntax of nominal algebra. In Sect. 3 we define a notion of derivation. In Sect. 4 we define a notion of validity, and in Sect. 5 we prove soundness and completeness. Sections 6 and 7 discuss related work and draw conclusions.

2 Syntax

Fix a countably infinite collection of **atoms** $a, b, c, \ldots$ representing object-level variables. We shall use a *permutative convention* that $a, b, c, \ldots$ range permutatively over atoms, so that for example a and b are always distinct. Also, fix a countably infinite collection of **unknowns** $X, Y, Z, \ldots$ representing meta-level variables. Fix **term-formers** f to each of which is associated some unique **arity** n which is a nonnegative number. Assume these collections are disjoint.

There is no proper sorting system so it will be possible to write 'silly' terms. There is no problem in principle with extending the system with a sort or type system if convenient, perhaps along the lines of other work [6, 7].

Let π range over **(finitely supported) permutations**. So π bijects atoms with themselves and there is a finite set of atoms S such that $\pi(a) = a$ for all atoms *not* in S. Write $\mathbf{Id}$ for the identity permutation such that $\mathbf{Id}(a) = a$ always. Write $\pi \circ \pi'$ for functional composition and write π^{-1} for inverse. This makes permutations into a group — write $\mathbb{P}$ for the set of all permutations.

Then nominal terms t, u, v are inductively defined by:

$$t \quad ::= \quad a \mid \pi \cdot X \mid [a]t \mid \mathsf{f}(t_1, \ldots, t_n)$$

We call $[a]t$ an *abstractor*; it uniformly represents the '$x.t$' or '$x.\phi$' part of expressions such as '$\lambda x.t$' or '$\forall x.\phi$'.

We call $\pi \cdot X$ a **moderated unknown**. We write $\mathbf{Id} \cdot X$ just as X, for brevity. In $\pi \cdot X$, X will get substituted for a term and then π will permute the atoms in that term; see Sect. 3. This notion is grounded in semantics [8] and permits a succinct treatment of α-renaming atoms (see CORE below and [9]).

A **signature** Σ is some set of term-formers with their arities. For example:

– $\{\mathsf{lam}:1, \mathsf{app}:2\}$ is a signature for the λ-calculus; we indicate arities with a colon as a convenient shorthand. When we define axioms for the λ-calculus, we shall extend this signature with a term-former for representing capture-avoiding substitution.
  We generally sugar $\mathsf{lam}([a]t)$ to $\lambda[a]t$ and $\mathsf{app}(t,u)$ to tu.
– $\{\supset:2, \forall:1, \approx:2, \bot:0\}$ is a signature for first-order logic with equality (the symbol for equality inside the logic is $\approx$).
  We sugar $\supset(\phi,\psi)$ to $\phi \supset \psi$, $\forall([a]\phi)$ to $\forall[a]\phi$ and $\approx(t,u)$ to $t \approx u$.

Write $t \equiv u$ for **syntactic identity** of terms. *There is no quotient by abstraction* so for example $[a]a \not\equiv [b]b$. Write $a \in t$ for 'a **occurs in (the syntax of)** t', Occurrence is literal, e.g. $a \in [a]a$ and $a \in \pi \cdot X$ when $\pi(a) \neq a$. Similarly write $a \notin t$ for 'a does not occur in the syntax of t'.

A **freshness (assertion)** is a pair $a\#t$ of an atom a and a term t. An **equality (assertion)** is a pair $t = u$ where t and u are terms. Call a freshness of the form $a\#X$ (so $t \equiv X$) **primitive**. Write Δ for a finite set of *primitive* freshnesses and call it a **freshness context**. We drop set brackets in freshness contexts, e.g. writing $a\#X, b\#Y$ for $\{a\#X, b\#Y\}$.

Nominal algebra has two **judgement forms**, a pair $\Delta \vdash a\#t$ of a freshness context and a freshness assertion, and a pair $\Delta \vdash t = u$ of a freshness context and an equality assertion. We generally write $\Delta \vdash A$ for an arbitrary judgement form, i.e. A is an equality or freshness. We may write $\emptyset \vdash A$ as $\vdash A$.

A **theory** $\mathsf{T} = (\Sigma, Ax)$ is a pair of a signature Σ and a possibly infinite set of *equality* judgement forms Ax in that signature; we call them the **axioms**.

Here are some nominal algebra theories:

– LAM has signature $\{\mathsf{lam}:1, \mathsf{app}:2, \mathsf{sub}:2\}$ and two axioms

$$\begin{array}{lll} (\beta) & \vdash & (\lambda[a]Y)X = Y[a \mapsto X] \\ (\eta) & a\#X \vdash & \lambda[a](Xa) = X \end{array}$$

  where we sugar $\mathsf{sub}([a]t, u)$ to $t[a \mapsto u]$. LAM on its own is not terribly useful because we need to give sub the correct behaviour:
– SUB has axioms

$$\begin{array}{lll} (\mathbf{var}{\mapsto}) & \vdash & a[a \mapsto X] = X \\ (\#{\mapsto}) & a\#Y \vdash & Y[a \mapsto X] = Y \\ (\mathbf{f}{\mapsto}) & \vdash & \mathsf{f}(Y_1,\ldots,Y_n)[a \mapsto X] = \mathsf{f}(Y_1[a \mapsto X],\ldots,Y_n[a \mapsto X]) \\ (\mathbf{abs}{\mapsto}) & b\#X \vdash & ([b]Y)[a \mapsto X] = [b](Y[a \mapsto X]) \\ (\mathbf{ren}{\mapsto}) & b\#Y \vdash & Y[a \mapsto b] = (b\ a) \cdot Y \end{array}$$

  Axiom $(\mathbf{f}{\mapsto})$ represents three axioms, one for each of $\mathsf{f} \in \{\mathsf{lam}, \mathsf{app}, \mathsf{sub}\}$. A theory of substitution for a different signature would have a suitable axiom for each term-former f. Note the heavy use of freshness side-conditions to manage the relationship between atoms and unknowns.
  But there is *one more axiom*. We would like $\lambda[a]a$ to be equal to $\lambda[b]b$. In other words we want α-equivalence:

– CORE has just one axiom

$$(\mathbf{perm}) \quad a\#X, b\#X \vdash (b\ a)\cdot X = X$$

Lemma 3.2 below shows that this axiom with the derivation rules of nominal algebra give the native notion of α-equivalence on nominal terms from previous work [9].

See [10] for an axiomatisation of first-order logic. Similar development for other systems with binding, such as System F [11] and the π-calculus [12], should also be possible.

3 A Derivation System

Now we need a notion of derivation which represents freshness assumptions on meta-variables, and permits axioms involving abstraction and conditioned on freshness assumptions, just like we do in informal reasoning.

We define a **permutation action** $\pi \cdot t$ by:

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X \qquad \pi \cdot [a]t \equiv [\pi(a)](\pi \cdot t)$$
$$\pi \cdot \mathsf{f}(t_1, \ldots, t_n) \equiv \mathsf{f}(\pi \cdot t_1, \ldots, \pi \cdot t_n)$$

A **substitution** σ is a finitely supported function from unknowns to terms. Here, finite support means: for some finite set of unknowns $\sigma(X) \not\equiv X$, and for all other unknowns $\sigma(X) \equiv X$. Write $[t_1/X_1, \ldots, t_n/X_n]$ for the substitution σ such that $\sigma(X_i) \equiv t_i$ and $\sigma(Y) \equiv Y$, for all $Y \not\equiv X_i$, $1 \leq i \leq n$.

We can define a **substitution action** $t\sigma$ on terms by:

$$a\sigma \equiv a \qquad (\pi \cdot X)\sigma \equiv \pi \cdot \sigma(X) \qquad ([a]t)\sigma \equiv [a](t\sigma)$$
$$\mathsf{f}(t_1, \ldots, t_n)\sigma \equiv \mathsf{f}(t_1\sigma, \ldots, t_n\sigma)$$

Substitution for an unknown does not avoid capture with abstraction, for example $([a]X)[a/X] \equiv [a]a$. This is designed to mirror informal practice, where instantiation of meta-variables does not avoid capture with binding.

Extend notation for permutation and substitution action to freshness contexts Δ pointwise to the terms it contains. It reduces parentheses to give substitution a higher priority than permutation and abstraction, so we do. The following **commutation** is easy to prove [9, 6]:

Lemma 3.1. $\pi \cdot t\sigma \equiv (\pi \cdot t)\sigma$.

Define **derivability** on freshnesses (in some signature Σ) by the rules in Fig. 1. Here f ranges over the term-formers of Σ, and in accordance with our permutative convention a and b range over *distinct* atoms. Write $\Delta \vdash a\#t$ when $a\#t$ may be derived from Δ.

Define **derivability** on equalities (between terms in the signature of T) by the rules in Fig. 2. Here (**fr**) is subject to a condition that $a \notin t, u, \Delta$ and the

square brackets denote discharge of assumptions in natural deduction style [13]. Write $\Delta \vdash_{\mathsf{T}} t = u$ when we may derive $t = u$ from Δ, using the signature from theory T and admitting only the axioms it contains. We write $\Delta \vdash_{\mathsf{T}} A$ as a convenient shorthand for $\Delta \vdash_{\mathsf{T}} t = u$ when A is $t = u$ and $\Delta \vdash a\#t$ when A is $a\#t$ (subscript T disappears in the freshness case).

$$\frac{}{a\#b}\,(\#\mathbf{ab}) \qquad \frac{\pi^{-1}(a)\#X}{a\#\pi\cdot X}\,(\#\mathbf{X}) \qquad \frac{}{a\#[a]t}\,(\#[]\mathbf{a}) \qquad \frac{a\#t}{a\#[b]t}\,(\#[]\mathbf{b}) \qquad \frac{a\#t_1 \;\cdots\; a\#t_n}{a\#\mathsf{f}(t_1,\ldots,t_n)}\,(\#\mathbf{f})$$

Fig. 1. Derivation rules for freshness

$$\frac{}{t=t}\,(\mathbf{refl}) \qquad \frac{t=u}{u=t}\,(\mathbf{symm}) \qquad \frac{t=u \quad u=v}{t=v}\,(\mathbf{tran}) \qquad \frac{\pi\cdot\Delta\sigma}{\pi\cdot t\sigma=\pi\cdot u\sigma}\,(\mathbf{ax}_{\boldsymbol{\Delta\vdash t=u}})$$

$$\frac{t=u}{[a]t=[a]u}\,(\mathbf{cong}[]) \qquad \frac{t=u}{\mathsf{f}(\ldots,t,\ldots)=\mathsf{f}(\ldots,u,\ldots)}\,(\mathbf{cong}\mathsf{f}) \qquad \frac{\begin{array}{c}[a\#X_1,\ldots,a\#X_n] \quad \Delta\\ \vdots\\ t=u\end{array}}{t=u}\,(\mathbf{fr})$$

Fig. 2. Derivation rules for equality

$(\mathbf{ax}_{\boldsymbol{\Delta\vdash t=u}})$ allows us to permutatively rename atoms and to instantiate unknowns. This gives the effect that atoms in axioms can be understood to range over *any* (distinct) atoms, and unknowns can be understood to range over *any* terms. So these derivations

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

are valid in theory LAM. The right derivation also shows that substitution of terms for unknowns does not avoid capture, reflecting informal practice.

The use of the $(\mathbf{ax}_{\boldsymbol{\Delta\vdash t=u}})$ rule introduces new proof obligations on the freshness side-conditions in Δ, as the following derivations show:

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

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

Note that we cannot conclude $([c]Y)[c\mapsto X]=[c](Y[c\mapsto X])$ by an application of $(\mathbf{ax}_{\#\mapsto})$ (even if $c\#X$ is derivable), since permutations are bijective: there is no π such that both $\pi(a)=c$ and $\pi(b)=c$.

We conclude these examples with two derivations in theory CORE:

$$\dfrac{\dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{a\#[b]b}\,(\#[]\mathbf{b}) \qquad \dfrac{}{b\#[b]b}\,(\#[]\mathbf{a})}{[a]a=[b]b}\,(\mathbf{ax_{perm}}) \qquad\qquad \dfrac{\dfrac{a\#X}{a\#[b]X}\,(\#[]\mathbf{b}) \qquad \dfrac{}{b\#[b]X}\,(\#[]\mathbf{a})}{[a](b\ a)\cdot X=[b]X}\,(\mathbf{ax_{perm}})$$

So $\vdash_{\mathsf{CORE}} [a]a=[b]b$ and $a\#X \vdash_{\mathsf{CORE}} [a](b\ a)\cdot X=[b]X$. To see that the instances of (**perm**) are valid, we note that $[a]a \equiv (b\ a)\cdot[b]b$ and $[a](b\ a)\cdot X \equiv (b\ a)\cdot[b]X$.

CORE is a novel and pleasingly succinct way to algebraically express the syntax-directed equality given to nominal terms [9]:

Lemma 3.2. *$\Delta \vdash_{\mathsf{CORE}} t=u$ if and only if $\Delta \vdash t=u$ holds in the sense of [9, 6].*

Proof. By induction on derivations of $\Delta \vdash_{\mathsf{CORE}}$ and $\Delta \vdash t=u$. □

We will *always* assume that theories contain the axiom (**perm**) from theory CORE.[1]

The rule (**fr**) introduces a fresh atom into the derivation. To illustrate the extra power this gives, we show that $X[a\mapsto a]=X$ is derivable in SUB:

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

In the above derivation, the superscript number one ^{1} is an annotation associating the instance of the rule (**fr**) with the assumption it discharges in the derivation. Furthermore, the instance of axiom (**ren**$\mapsto$) is valid since we have used the fact that $X \equiv (a\ b)\cdot(b\ a)\cdot X$ in the right-hand side of the equation.

We cannot derive $X[a\mapsto a]=X$ *without* (**fr**); intuitively this is because to α-rename a so that we can use (**ren**$\mapsto$), we need an atom fresh for X. The tools to make this argument formal are elsewhere [14].

Note that (**fr**) mirrors the generation of a fresh name in rules such as the $\forall$ right-introduction rule 'from $\Gamma \vdash \phi$ derive $\Gamma \vdash \forall x.\phi$ provided x is not free in Γ'.

We conclude this section with some proof-theoretical results. We can permute atoms in freshnesses and equations without changing the freshness contexts:

Theorem 3.3. *For any permutation π', if $\Delta \vdash_{\mathsf{T}} A$ then $\Delta \vdash_{\mathsf{T}} \pi'\cdot A$.*

---

[1] Equivalently we could add (**ax$_{\mathbf{perm}}$**) as a derivation rule $\dfrac{a\#t \quad b\#t}{(a\ b)\cdot t=t}$.

Proof. By induction on derivations. For ($\#\mathbf{X}$) we note that permutations are *bijective* and $\pi'(a)\#\pi' \circ \pi \cdot X$ if and only if $\pi^{-1}(a)\#X$. For (**fr**) we might have to rename the freshly chosen atom a if $\pi'(a) \neq a$.[2] □

We can substitute terms for unknowns provided those terms violate no freshness assumptions made on the unknowns:

Theorem 3.4. *If* $\Delta \vdash_{\mathsf{T}} A$ *then* $\Delta' \vdash_{\mathsf{T}} A\sigma$ *for all* Δ' *such that* $\Delta' \vdash_{\mathsf{T}} \Delta\sigma$.

Proof. Natural deduction derivations are such that the conclusion of one derivation may be 'plugged in' to an assumption in another derivation. For ($\#\mathbf{X}$) we use Theorem 3.3. For (**fr**) we might have to rename the freshly chosen atom a if it is mentioned by σ (see footnote 2). □

4 Semantics

A model of a nominal algebra theory T is a nominal set which interprets the term-formers so as to make the axioms valid. We use *nominal* sets [8] because they permit a direct semantic interpretation of freshness judgements $a\#x$ and permutations $\pi \cdot x$, an interpretation which is not conveniently definable on 'ordinary' sets.

A nominal set is a pair $(\mathbb{X}, \cdot)$ of a(n ordinary) set $\mathbb{X}$ with a group action by $\mathbb{P}$ such that each $x \in \mathbb{X}$ has **finite support**.[3] This means that there is some *finite* set of atoms $\{a_1, \ldots, a_n\}$ such that for any π if $\pi(a_i) = a_i$ for each $1 \leq i \leq n$, then $\pi \cdot x = x$. It is a fact ([8, Proposition 3.4]) that a unique least such set of atoms exists, we call it the **support** of x. We write $a\#x$ when a is not in the support of x, and call a **fresh for** x. For example:

- The set $\mathbb{A} = \{a, b, c, \ldots\}$ of atoms with action $\pi \cdot a = \pi(a)$ is a nominal set; the support of $a \in \mathbb{A}$ is $\{a\}$, so $b\#a$ but not $a\#a$.
- The powerset $\mathcal{P}(\mathbb{A}) = \{U \mid U \subseteq \mathbb{A}\}$ of $\mathbb{A}$ with action $\pi \cdot U = \{\pi \cdot u \mid u \in U\}$, is *not* a nominal set; $\{a_1, a_3, a_5, \ldots\} \in \mathcal{P}(\mathbb{A})$ does not have finite support, since for no finite set of atoms is it the case that all permutations fixing that set map $\{a_1, a_3, a_5, \ldots\}$ to itself. Note that the support of $\mathbb{A} \in \mathcal{P}(\mathbb{A})$ is $\emptyset$, so $a\#\mathbb{A}$ for any a.
- If $\mathbb{X}$ and $\mathbb{Y}$ are nominal sets write $\mathbb{X} \times \mathbb{Y}$ for $\{(x, y) \mid x \in \mathbb{X},\ y \in \mathbb{Y}\}$ with action $\pi \cdot (x, y) = (\pi \cdot x, \pi \cdot y)$. This is also a nominal set; the support of $(x, y) \in \mathbb{X} \times \mathbb{Y}$ is the union of the supports of x and y.

  It is useful to write $\mathbb{X}^n$ for $\overbrace{\mathbb{X} \times \cdots \times \mathbb{X}}^{n}$, so we do.

[2] It is easy to prove that the resulting 'name-clash-avoiding' derivation *is* a derivation, and we can use induction on *depth* of derivations to preserve the inductive hypothesis. However, we have used the principle of meta-level Fraenkel-Mostowksi equivariance [8], which lets us rename atoms permutatively *without* losing structural inductive properties, so our structural induction is actually perfectly valid.

[3] $\cdot$ is a function $\mathbb{P} \times \mathbb{X} \to \mathbb{X}$ such that $\mathbf{Id} \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$.

We assume these permutation actions on these sets henceforth.

Lemma 4.1. *For any nominal set $\mathbb{X}$ and element $x \in \mathbb{X}$, if $a\#x$ and $b\#x$ then $(a\ b) \cdot x = x$.*

Proof. Since $a, b \notin \mathrm{supp}(x)$, $(a\ b)(c) = c$ for every $c \in \mathrm{supp}(x)$. □

Functions $f \in \mathbb{X} \to \mathbb{Y}$ (on the underlying sets) have a natural conjugation permutation action given by $(\pi \cdot f)(x) = \pi \cdot (f(\pi^{-1} \cdot x))$. Call f **equivariant** if $\pi \cdot (f(x)) = f(\pi \cdot x)$ always.

Lemma 4.2. *For any nominal sets $\mathbb{X}, \mathbb{Y}$, equivariant function $f \in \mathbb{X} \to \mathbb{Y}$ and $x \in \mathbb{X}$, if $a\#x$ then $a\#f(x)$.*

Proof. By an elementary calculation using the fact that $\pi \cdot (f(x)) = f(\pi \cdot x)$. □

We can now give a semantics to nominal algebra theories. An **interpretation** $[\![\_]\!]$ of a signature is a nominal set $\mathbb{T}$ with *equivariant* functions

- $[\![\_]\!]_{\mathbb{T}} \in \mathbb{A} \to \mathbb{T}$ to interpret atoms;
- $[\_]\_ \in \mathbb{A} \times \mathbb{T} \to \mathbb{T}$ such that $a\#[a]x$ always, to interpret abstraction;
- $[\![\mathsf{f}]\!] \in \mathbb{T}^n \to \mathbb{T}$ for each term-former $\mathsf{f} : n$, to interpret term-formers.

A **valuation** ς maps unknowns X to elements $\varsigma(X) \in \mathbb{T}$. The pair of an interpretation and a valuation extend easily to terms $[\![t]\!]_\varsigma$:

$$[\![a]\!]_\varsigma = [\![a]\!]_{\mathbb{T}} \qquad [\![\pi \cdot X]\!]_\varsigma = \pi \cdot \varsigma(X) \qquad [\![[a]t]\!]_\varsigma = [a][\![t]\!]_\varsigma$$
$$[\![\mathsf{f}(t_1, \ldots, t_n)]\!]_\varsigma = [\![\mathsf{f}]\!]([\![t_1]\!]_\varsigma, \ldots, [\![t_n]\!]_\varsigma)$$

This extends to a notion of **validity** for our judgement forms:

$$[\![a\#t]\!]_\varsigma \text{ (is valid) when } a\#[\![t]\!]_\varsigma \qquad [\![t = u]\!]_\varsigma \text{ when } [\![t]\!]_\varsigma = [\![u]\!]_\varsigma$$
$$[\![\Delta]\!]_\varsigma \text{ when } a\#\varsigma(X) \text{ for each } a\#X \in \Delta \qquad [\![\Delta \vdash A]\!]_\varsigma \text{ when } [\![\Delta]\!]_\varsigma \text{ implies } [\![A]\!]_\varsigma$$
$$[\![\Delta \vdash A]\!] \text{ when } [\![\Delta \vdash A]\!]_\varsigma \text{ for all valuations } \varsigma$$

A **model** of a theory T is an interpretation $[\![\_]\!]$ such that $[\![\Delta \vdash t = u]\!]$ for all axioms $\Delta \vdash t = u$ of T — we say the model **validates the axioms**.

Write $\Delta \models_{\mathsf{T}} A$ when $[\![\Delta \vdash A]\!]$ for all models $[\![\_]\!]$ of T.

5 Soundness and Completeness

Derivability of freshness and equality is sound for the semantics presented in the previous section.

Theorem 5.1 (Soundness). *If $\Delta \vdash_{\mathsf{T}} A$ then $\Delta \models_{\mathsf{T}} A$.*

Proof. Let $[\![\_]\!]$ be a model of T. We must show that if A is derived from Δ then $[\![\Delta]\!]_\varsigma$ implies $[\![A]\!]_\varsigma$ for any valuation ς. We work by induction on derivation rules.

– (**#ab**). By definition, $[\![a\#b]\!]_\varsigma$ when $a\#[\![b]\!]_{\mathbb{T}}$. By Lemma 4.2 this follows from $a\#b$, which is a standard property of (semantic) freshness.
– (**#X**). By properties of the group of permutations, $a\#\pi \cdot x$ when $\pi^{-1}(a)\#x$.
– (**#[]a**). $a\#[a]x$ for any $x \in \mathbb{T}$ by construction.
– (**#[]b**). By Lemma 4.2, $a\#x$ implies $a\#[b]x$.
– (**#f**). Take any $\mathsf{f} : n$. If $a\#[\![t_i]\!]_\varsigma$ for $1 \leq i \leq n$ then $a\#[\![\mathsf{f}]\!]([\![t_1]\!]_\varsigma, \ldots, [\![t_n]\!]_\varsigma)$ follows using Lemma 4.2.
– (**refl**), (**symm**), (**tran**), (**cong**[]), (**congf**). By properties of equality.
– ($\mathbf{ax}_{\boldsymbol{\Delta}' \vdash \mathbf{t=u}}$). This follows from the definition of what it is to be a model, and the fact that $[\![\pi \cdot t\sigma]\!]_\varsigma = \pi \cdot [\![t]\!]_{\varsigma'}$ where $\varsigma'(X) = [\![\sigma(X)]\!]_\varsigma$.
– (**fr**). Suppose there is a derivation of $t = u$ from $\Delta, a\#X_1, \ldots, a\#X_n$, and suppose that $a \notin t, u, \Delta$. If a is amongst the atoms in the support of $\varsigma(X_i)$ for any X_i, rename a in the derivation to avoid this. By inductive hypothesis $[\![t]\!]_\varsigma = [\![u]\!]_\varsigma$ follows from $[\![\Delta, a\#X_1, \ldots, a\#X_n]\!]_\varsigma$. So $[\![t]\!]_\varsigma = [\![u]\!]_\varsigma$ as required. □

In order to show completeness of derivability of equality, we need some technical machinery. Given a theory T we construct a **term model** $[\![\_]\!]^{\mathcal{T}}$ of T as follows:

– For each $n > 0$ introduce a term-former $\mathsf{d}_n : n$.
– Take as $\mathbb{T}$ the set of closed terms of sort $\mathbb{T}$ (terms without unknowns) in the enriched signature, quotiented by provable equality, with the permutation action given pointwise.
– Take $[\![a]\!]_{\mathbb{T}} = \{t \mid \vdash_{\mathsf{T}} t = a,\ t \text{ closed}\}$, $[a]x = \{t \mid \vdash_{\mathsf{T}} t = [a]u,\ t \text{ closed},\ u \in x\}$, and for each term-former $\mathsf{f} : n$ take as $[\![\mathsf{f}]\!]^{\mathcal{T}}$ the function defined by

$$[\![\mathsf{f}]\!]^{\mathcal{T}}(x_1, \ldots, x_n) = \{t \mid \vdash_{\mathsf{T}} t = \mathsf{f}(t_1, \ldots, t_n),\ t \text{ closed},\ t_i \in x_i\}.$$

We have to enrich the signature with the d_n to ensure the term model has enough elements; term-formers must be interpreted by *equivariant* functions so the usual method of adding constants c (0-ary term-formers) would add only elements such that $\vdash_{\mathsf{T}} \pi \cdot \mathsf{c} = \mathsf{c}$ always, and this does not suffice. This idea goes back to [15].

It is possible to prove by some elementary calculations that the definition above is well-defined and an interpretation; that is, that $\mathbb{T}$ is a nominal set, that $[\![\_]\!]_{\mathbb{T}}$, $[\_]\_$, and all $[\![\mathsf{f}]\!]^{\mathcal{T}}$ are equivariant, and that $a\#[\![[a]t]\!]^{\mathcal{T}}$ always.

Lemma 5.2. *If $a\#[\![t]\!]^{\mathcal{T}}_\varsigma$ then there is some $t' \in [\![t]\!]^{\mathcal{T}}_\varsigma$ such that $\vdash_{\mathsf{T}} a\#t'$.*

Proof. Take b fresh (so b does not occur in t or ς). Then $b\#[\![t]\!]^{\mathcal{T}}_\varsigma$. Since also $a\#[\![t]\!]^{\mathcal{T}}_\varsigma$ we obtain $(b\ a) \cdot [\![t]\!]^{\mathcal{T}}_\varsigma = [\![t]\!]^{\mathcal{T}}_\varsigma$ by Lemma 4.1. But then also $\vdash_{\mathsf{T}} (b\ a) \cdot t = t$, by construction. Take $t' \equiv (b\ a) \cdot t$. □

To show why Lemma 5.2 is not trivial, consider a theory IOTA with one axiom $\vdash a = b$. It is easy to verify that $a\#[\![a]\!]^{\mathcal{T}}_\varsigma$ (since $[\![a]\!]^{\mathcal{T}}_\varsigma = \mathbb{A}$) but $a\#a$ is not derivable. Of course $a = b$ and $a\#b$ *are* derivable. Similarly in LAM it is a fact that $a\#[\![(\lambda[a]b)a]\!]^{\mathcal{T}}_\varsigma$ but $a\#(\lambda[a]b)a$ is not derivable; of course $(\lambda[a]b)a = b$ and $a\#b$ are derivable.

Theorem 5.3. *The term model* $[\![\_]\!]^{\mathcal{T}}$ *of* T *is a model.*

Proof. We show that if $\Delta \vdash t = u$ is an axiom of T then $[\![\Delta \vdash t = u]\!]^{\mathcal{T}}_{\varsigma}$ is valid for any ς. By definition, we must show that if $a\#\varsigma(X)$ for each $a\#X \in \Delta$, then $[\![t]\!]^{\mathcal{T}}_{\varsigma} = [\![u]\!]^{\mathcal{T}}_{\varsigma}$. Use Lemma 5.2 to choose a term $t' \in \varsigma(X)$ such that $\vdash a\#t'$ for each $a\#X \in \Delta$. By $(\mathbf{ax}_{\Delta \vdash \mathbf{t=u}})$, taking $\pi = \mathbf{Id}$ and $\sigma(X) \equiv t'$ for each $a\#X \in \Delta$ we obtain $\vdash_{\mathsf{T}} t\sigma = u\sigma$, since $\vdash a\#\sigma(X)$. Then $[\![t\sigma]\!]^{\mathcal{T}}_{\varsigma} = [\![u\sigma]\!]^{\mathcal{T}}_{\varsigma}$ by Theorem 5.1. By construction $[\![t]\!]^{\mathcal{T}}_{\varsigma} = [\![t\sigma]\!]^{\mathcal{T}}_{\varsigma}$ and $[\![u]\!]^{\mathcal{T}}_{\varsigma} = [\![u\sigma]\!]^{\mathcal{T}}_{\varsigma}$, so $[\![t]\!]^{\mathcal{T}}_{\varsigma} = [\![u]\!]^{\mathcal{T}}_{\varsigma}$ as required. □

A certain amount of technical subtlety is hidden in Theorem 5.4 and in particular in its use of the extra term-formers d:

Theorem 5.4 (Completeness). *If* $\Delta \models_{\mathsf{T}} t = u$ *then* $\Delta \vdash_{\mathsf{T}} t = u$.

Proof. By assumption $[\![\Delta \vdash t = u]\!]_{\varsigma}$ is valid for any model $[\![\_]\!]$ of T and any valuation ς. Let $[\![\_]\!]$ be $[\![\_]\!]^{\mathcal{T}}$, the term model of T. In order to choose a suitable ς, we introduce the following:

- Let S be the set of atoms mentioned anywhere in Δ, t, and u.
- Let $X_1, \ldots, X_n$ be the set of unknowns mentioned anywhere in Δ, t, or u, (*not* just in Δ!) in some arbitrary order.
- For each $1 \leq i \leq n$ let S_i be the set of all atoms $a \in S$ such that $a\#X_i \notin \Delta$.
- For each $1 \leq i \leq n$ let $\sigma(X_i) \equiv \mathsf{d}_i(a_1, \ldots, a_{n_i})$ where $S_i = \{a_1, \ldots, a_{n_i}\}$.

Here we extend the signature with distinct term-formers d_i. Let ς map X_i to $[\![\sigma(X_i)]\!]^{\mathcal{T}}_{\varsigma}$, for $1 \leq i \leq n$. By construction $[\![\Delta]\!]^{\mathcal{T}}_{\varsigma}$, so by assumption $[\![t]\!]^{\mathcal{T}}_{\varsigma} = [\![u]\!]^{\mathcal{T}}_{\varsigma}$, and by definition this means that $t\sigma = u\sigma$ is derivable. This derivation can be transformed rule by rule into a derivation of $\Delta \vdash_{\mathsf{T}} t = u$, since the only freshnesses and equalities we can assert of the d_i are those we can also assert of the X_i. The only complication is when *perhaps* for some fresh b we use a freshness derivation to derive $b\#v\sigma$ for some fresh b; then we modify the derivation to use (**fr**) instead. □

5.1 The Status of Freshness Derivations

Completeness holds for equalities — but not for freshnesses. That is, $\Delta \models_{\mathsf{T}} b\#t$ does *not* imply $\Delta \vdash b\#t$ necessarily; for counterexamples see the discussion involving theories IOTA and LAM just after Lemma 5.2.

To understand why this is desirable we must draw a distinction between the intension and the extension of a term.

'$\vdash a\#(\lambda[a]b)a$' has the status of 'x is fresh for $(\lambda x.y)x$'; both are false. Yet $(\lambda x.y)x$ is β-convertible to y and x *is* fresh for y. A freshness judgement $\Delta \vdash a\#t$ is a(n intensional) judgement on concrete syntax; call this *syntactic freshness*.[4] All the capture-avoidance side-conditions we know of are in accordance with the slogan 'ϵ away from informal practice', this is what $\Delta \vdash a\#t$ models.

[4] The reader familiar with a theorem-prover such as Isabelle [16] might like to imagine that # maps to *Prop* and = maps to *o*.

Nominal sets is unusual amongst semantics in that it has a(n extensional) *semantic* notion of freshness $a\#x$. In fact, semantic freshness is hiding in nominal algebra — but in a slightly unexpected place; in the theory of *equality*.

Theorem 5.5. *Suppose* $\{X_1, \ldots, X_n\}$ *and* $\{a_1, \ldots, a_m\}$ *are the unknowns and atoms mentioned in* Δ *and* t*. Suppose that* b *is fresh (so* $b \notin \{a_1, \ldots, a_m\}$*).*

$\Delta \models_{\mathsf{T}} a\#t$ *if and only if* $\Delta, b\#X_1, \ldots, b\#X_n \vdash_{\mathsf{T}} (b\ a) \cdot t = t$.

Proof. Choose a model and any ς such that $b\#\varsigma(X_1), \ldots, b\#\varsigma(X_n)$; it follows by an induction on syntax that $b\#[\![t]\!]_\varsigma$. It is a fact that $a\#[\![t]\!]_\varsigma$ if and only if $(b\ a) \cdot [\![t]\!]_\varsigma = [\![t]\!]_\varsigma$ (see [8] for details). Also, since term-formers are interpreted by *equivariant* functions, we rewrite $(b\ a) \cdot [\![t]\!]_\varsigma$ as $[\![(b\ a) \cdot t]\!]_\varsigma$.

Now suppose $\Delta \models_{\mathsf{T}} a\#t$. By definition $a\#[\![t]\!]_\varsigma$ for any $[\![\text{-}]\!]$ and ς such that $[\![\Delta]\!]_\varsigma$. By the arguments above $[\![(b\ a) \cdot t]\!]_\varsigma = [\![t]\!]_\varsigma$ if $b\#\varsigma(X_1), \ldots, b\#\varsigma(X_n)$. By Theorem 5.4 it follows that $\Delta, b\#X_1, \ldots, b\#X_n \vdash_{\mathsf{T}} (b\ a) \cdot t = t$. The reverse implication is similar. □

Recall the examples we used just after Lemma 5.2. Theorem 5.5 tells us that $\models_{\mathsf{IOTA}} a\#a$ if and only if $\vdash_{\mathsf{IOTA}} b = a$ (which follows by the axiom $\vdash a = b$), and that $\models_{\mathsf{LAM}} a\#(\lambda[a]b)a$ if and only if $\vdash_{\mathsf{LAM}} (\lambda[a]b)a = (\lambda[c]b)c$ (which follows since both sides are derivably equal to b).

So the semantic freshness $\Delta \models_{\mathsf{T}} a\#t$ can be expressed as an equality axiom. Any undecidability or algorithmic complexity is isolated in the equality judgement form.

6 Related Work

Nominal algebra is derived from Fraenkel-Mostowski set theory. This provided a semantics for names and gave an unexpected set-theoretic semantics for parse trees (abstract syntax trees) *with* α-equivalence [8]. Existing technology included De Bruijn indexes [17], higher-order abstract syntax [18], the theory of contexts [19], TWELF [20], and other approaches. In this crowded arena approaches based on Fraenkel-Mostowski sets were catchily labelled 'nominal' by Pitts. The derivation rules expressed in this paper by the theory CORE are from this semantics. The name 'nominal algebra' acknowledges this debt.

Nominal unification considers the computational aspects of unifying trees-with-binding [9]. Nominal logic describes inductive programming and reasoning principles for trees-with-binding [21]. FreshML is a programming language... for trees-with-binding [22] in ML [23] style.

Aside from applications to trees-with-binding, ideas from nominal techniques have been used in logic [24] (this was still reasoning on trees, since the logic had a fixed syntactic model) and recently in semantics [25].

Nominal *rewriting* considers equalities between terms *not* directly to do with an underlying model of trees [6]. α-prolog [26] takes a similar tack but in logic programming. Still, the emphasis is squarely on the computational benefits compared to those of other approaches.

Nominal algebra champions nominal techniques as a *logical framework* to represent and reason about semantic structures — i.e. functions, sets, and other mathematical structures. That is new, and this paper sets it on a sound semantic/logical basis.

The full case for nominal algebra as a framework for applications in which higher-order logic can be used [27] remains to be made. However we can make some remarks:

Unification of nominal terms is decidable whereas higher-order unification is not [28]. (First-order) logic can be axiomatised [10] and the treatment of quantification is very smooth. In particular it closely models the informal specification of quantification; the $\forall$-intro rule in Isabelle is

$$\bigwedge x.[P \Longrightarrow \forall x.P]$$

(see [16]) and this is *not* like the usual rule used by logicians

$$\text{'from } \Gamma \vdash P \text{ deduce } \Gamma \vdash \forall x.P \text{ if } x \text{ is fresh for } \Gamma\text{'}.$$

We believe (though we have not yet checked this in detail) that treatments of substructural logics are possible in nominal algebra; they are not necessarily so in higher-order logic because structural properties of the framework's connectives 'infect' those of the logic being axiomatised.

Since the conception of nominal algebra, Pitts and Clouston have derived *nominal equational logic* [29]. Nominal equational logic makes some slightly different design decisions, notably it cannot express syntactic freshness (see Sect. 5.1).

Higher-order algebra [4] uses typed λ-calculus up to $\alpha\beta$-equivalence and can serve as a foundational framework. The simply-typed λ-calculus itself [16, Figures 6 and 7] can also be used. Other algebraic systems, such as Sun's extensional binding algebras [30, Definition 2.2.3] introduce dedicated systems based on functional semantics.

A host of 'cylindric' algebraic techniques exist. These embrace meta-variables and reject object-level variables, preferring to encode their expressive power in the term-formers. Examples are lambda-abstraction algebras [31] for the λ-calculus and cylindric algebras [32, 33] for first-order logic. Polyadic algebras have a slightly more general treatment, for a brief but clear discussion of the design of these and related systems is in [34, Appendix C]. Combinators [1] reject object-level variables altogether. Algebras over (untyped) combinators can then express first-order predicate logic [35]. These systems are effective for their applications but there are things that nominal algebra allows us to say particularly naturally, because of the way it isolates abstraction, has explicit meta-variables, and permits freshness side-conditions on equalities.

7 Conclusions

Nominal terms embrace the difference between the object-level and the meta-level; there are two classes of variables, atoms a and unknowns X. Substitution

for X does not avoid capture in abstraction by a. Freshness side-conditions of the form $a\#X$ manage the interaction between the two levels and 'recover' capture-avoidance where this is required.

Previous 'nominal' work has considered reasoning on datatypes of syntax up to α-equivalence. In this paper we use nominal terms as a foundational syntax for mathematics, and we have constructed a logic for them using a suitable theory of equality. The two-level variable structure permits the framework to get exceptionally close to informal practice; see the end of the Introduction for examples.

Nominal algebra could be suitable to specify and reason algebraically on languages with binding; we have in mind particularly process calculi such as those derived from the π-calculus, which often feature quite complex binding side-conditions and for which algebraic reasoning principles are frequently developed [36, 37, 38].

It is possible to extend nominal algebra with abstractions of the form $[t]u$, and freshness judgements of the form $t\#u$. Other extensions are possible and case studies will tell which of them are most important.

We see nominal algebra as the first of a family of two-level logics as yet to be created. We intend to begin by creating a first-order logic with two levels of variable. For example we would like to be able to write an expression like

$$\forall[X]\forall[a]\big(a\#X \supset (X \Leftrightarrow \forall[a]X)\big).$$

We made a start in that direction by designing one-and-a-halfth order logic [10]. However that system does not allow quantification of unknowns, and we are now working on lifting that restriction.

In our semantics object-level variables are first-class entities in the denotation; a represents an object-level variable symbol in the syntax *and* in the semantics of nominal algebra. Yet we may impose equalities, such as theory SUB from Sect. 2. If we can substitute for a then it has the flavour of a variable 'ranging over' denotational elements; so SUB is a theory of 'variables in denotation'. Nominal algebra is the beginning of an intriguing investigation into what this enrichment of the denotation does to the theory of validity and proof.

We are also interested in developing logics with *hierarchies* of variables. Since nominal algebra offers two levels of variable, why not extend this to allow an infinite hierarchy of variables, by analogy with type hierarchies in the λ-calculus [1], or stratification in set theory [39]? The first author has considered this extension of nominal terms in other publications [40, 41] but introducing such a hierarchy in a *logical* context poses unique challenges, and in particular, we have not yet obtained a satisfactory notion of model. This too is current research.

References

[1] Barendregt, H.P.: The Lambda Calculus: its Syntax and Semantics (revised ed.). North-Holland (1984)

[2] Curry, H.B., Feys, R.: Combinatory Logic. Volume 1. North Holland (1958)

[3] Henkin, L., Monk, J.D., Tarski, A.: Cylindric Algebras. North Holland (1971 and 1985) Parts I and II.

[4] Meinke, K.: Universal algebra in higher types. Theoretical Computer Science **100**(2) (1992) 385–417

[5] Leivant, D.: Higher order logic. In Gabbay, D., Hogger, C., Robinson, J., eds.: Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 2. Oxford University Press (1994) 229–322

[6] Fernández, M., Gabbay, M.J.: Nominal rewriting. Information and Computation (2005) In press.

[7] Fernández, M., Gabbay, M.J.: Curry-style types for nominal rewriting. TYPES'06 (2006)

[8] Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax with variable binding. Formal Aspects of Computing **13**(3–5) (2001) 341–363

[9] Urban, C., Pitts, A.M., Gabbay, M.J.: Nominal unification. Theoretical Computer Science **323**(1–3) (2004) 473–497

[10] Gabbay, M.J., Mathijssen, A.: One-and-a-halfth-order logic. In: PPDP '06: Proc. of the 8th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming, ACM Press (2006) 189–200

[11] Girard, J.Y., Taylor, P., Lafont, Y.: Proofs and types. Cambridge University Press (1989)

[12] Parrow, J.: An introduction to the pi-calculus. In Bergstra, J., Ponse, A., Smolka, S., eds.: Handbook of Process Algebra, Elsevier Science (2001) 479–543

[13] Hodges, W.: Elementary predicate logic. In Gabbay, D., Guenthner, F., eds.: Handbook of Philosophical Logic, 2nd Edition. Volume 1. Kluwer (2001) 1–131

[14] Gabbay, M.J., Mathijssen, A.: Capture-avoiding substitution as a nominal algebra. In: ICTAC'2006: 3rd Int'l Colloquium on Theoretical Aspects of Computing. (2006) 198–212

[15] Gabbay, M.J.: Fresh logic. Journal of Logic and Computation (2006) In press.

[16] Paulson, L.C.: The foundation of a generic theorem prover. Journal of Automated Reasoning **5**(3) (1989) 363–397

[17] de Bruijn, N.G.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. Indagationes Mathematicae **5**(34) (1972) 381–392

[18] Pfenning, F., Elliot, C.: Higher-order abstract syntax. In: PLDI '88: Proc. of the ACM SIGPLAN 1988 conf. on Programming Language design and Implementation, ACM Press (1988) 199–208

[19] Miculan, M.: Developing (meta)theory of lambda-calculus in the theory of contexts. ENTCS **1**(58) (2001)

[20] Pfenning, F., Schürmann, C.: System description: Twelf - a meta-logical framework for deductive systems. In Ganzinger, H., ed.: CADE-16, 16th Int'l Conf. on Automated Deduction, Springer (1999) 202–206

[21] Pitts, A.M.: Nominal logic, a first order theory of names and binding. Information and Computation **186**(2) (2003) 165–193

[22] Shinwell, M.R., Pitts, A.M., Gabbay, M.J.: FreshML: Programming with binders made simple. In: Eighth ACM SIGPLAN Int'l Conf. on Functional Programming (ICFP 2003), Uppsala, Sweden, ACM Press (2003) 263–274

[23] Paulson, L.C.: ML for the working programmer (2nd ed.). Cambridge University Press (1996)
[24] Luís Caires, L.C.: A spatial logic for concurrency (part II). Theoretical Computer Science **322**(3) (2004) 517–565
[25] Benton, N., Leperchey, B.: Relational reasoning in a nominal semantics for storage. In: Proc. of the 7th Int'l Conf. on Typed Lambda Calculi and Applications (TLCA). Volume 3461 of LNCS. (2005) 86–101
[26] Cheney, J., Urban, C.: System description: Alpha-Prolog, a fresh approach to logic programming modulo alpha-equivalence. In: Proc. 17th Int. Workshop on Unification, UNIF'03, Universidad Politecnica de Valencia (2003) 15–19
[27] Paulson, L.C.: Isabelle: the next 700 theorem provers. In Odifreddi, P., ed.: Logic and Computer Science. Academic Press (1990) 361–386
[28] Huet, G.: Higher order unification 30 years later. In: TPHOL 2002. Number 2410 in LNCS (2002) 3–12
[29] Clouston, R.A., Pitts, A.M.: Nominal equational logic. ENTCS **172** (2007) 223–257
[30] Sun, Y.: An algebraic generalization of frege structures - binding algebras. Theoretical Computer Science **211** (1999) 189–232
[31] Salibra, A.: On the algebraic models of lambda calculus. Theoretical Computer Science **249**(1) (2000) 197–240
[32] Burris, S., Sankappanavar, H.: A Course in Universal Algebra. Springer (1981)
[33] Andréka, H., Németi, I., Sain, I.: Algebraic logic. In Gabbay, D., Guenthner, F., eds.: Handbook of Philosophical Logic, 2nd Edition. Volume 2. Kluwer (2001) 133–249
[34] Blok, W.J., Pigozzi, D.: Algebraizable logics. Memoirs of the AMS **77**(396) (1989)
[35] Barendregt, H., Dekkers, W., Bunder, M.: Completeness of two systems of illative combinatory logic for first-order propositional and predicate calculus. Archive für Mathematische Logik **37** (1998) 327–341
[36] Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: the spi calculus. In: CCS '97: Proc. of the 4th ACM conf. on Computer and Communications Security, ACM Press (1997) 36–47
[37] Luttik, B.: Choice Quantification in Process Algebra. PhD thesis, University of Amsterdam (2002)
[38] Katoen, J.P., D'Argenio, P.R.: General distributions in process algebra. In: Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science, Springer (2002) 375–429
[39] Forster, T.: Quine's NF, 60 years on. American Mathematical Monthly **104**(9) (1997) 838–845
[40] Gabbay, M.J.: A new calculus of contexts. In: PPDP '05: Proc. of the 7th ACM SIGPLAN int'l conf. on Principles and Practice of Declarative Programming, ACM Press (2005) 94–105
[41] Gabbay, M.J.: Hierarchical nominal rewriting. In: LFMTP'06: Logical Frameworks and Meta-Languages: Theory and Practice. (2006) 32–47