Capture-Avoiding Substitution as a Nominal Algebra

Murdoch J. Gabbay[1] and Aad Mathijssen[2]

[1] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, Scotland, Great Britain
murdoch.gabbay@gmail.com
[2] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
A.H.J.Mathijssen@tue.nl

Abstract. Substitution is fundamental to computer science, underlying for example quantifiers in predicate logic and beta-reduction in the lambda-calculus. So is substitution something we define on syntax on a case-by-case basis, or can we turn the idea of 'substitution' into a mathematical object?
We exploit the new framework of Nominal Algebra to axiomatise substitution. We prove our axioms sound and complete with respect to a canonical model; this turns out to be quite hard, involving subtle use of results of rewriting and algebra.

1 Introduction

Substitution is intuitively the operation $v[a \mapsto t]$ meaning:

Replace the variable a by t in v.

Is there an algebra which describes exactly the properties of $v[a \mapsto t]$ independently of what v and t are (λ-terms, formulae of a logic, or any mixture or variation thereof)?

Consider by way of analogy the notion of 'a field'. This has an algebraic characterisation which tells us what properties 'a field' must have, independently of *which* field it is, or *how* it may be implemented (if we are programming). This is useful; for example the definition of 'vector space' is parametric over fields, and this step requires a characterisation of what fields are [1].

When we begin to algebraically axiomatise substitution some unusual difficulties present themselves. Consider the following informally expressed candidate property of substitution:

$$v[a \mapsto t][b \mapsto u] = v[b \mapsto u][a \mapsto t[b \mapsto u]] \qquad \text{provided } \ a \not\in fv(u).$$

This is not algebraic, because of the side-condition $a \not\in fv(u)$. Here $fv(u)$ is 'the free variables of u', which is a property of the syntax of u.

So is it the case that substitution *cannot* be axiomatised, and only exists as an incidental property of syntax used to talk about 'real' mathematical objects? But in that case, what is the status of the intuition which makes us agree that the property above should be satisfied by any self-respecting substitution action?

We shall argue that the following properties axiomatise substitution, all of substitution, and nothing but substitution. We express them in Nominal Algebra (given formal meaning in the rest of this paper):

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

Fig. 1. Axioms of SUB

For convenience we now give an *informal* reading:

($\mathbf{var}\mapsto$): If a is a variable then a with a replaced by T, is T.

($\#\mapsto$): If a is fresh for X then X with a replaced by T is X.

($\mathbf{f}\mapsto$): Substitution distributes through term-formers (we can have as many as we like); f ranges over them.

($\mathbf{abs}\mapsto$): Substitution distributes under abstraction, provided an 'accidental capture-avoidance' condition holds (b is fresh for T).

($\mathbf{ren}\mapsto$): If b is fresh for X then X with a replaced by b is identical to X with a replaced by b and *simultaneously* b replaced by a.

Formally, Fig. 1 uses nominal terms [2] as a syntax, and nominal algebra [3] as an algebraic framework.

A number of questions now arise: a) Is this substitution? In what sense; are the axioms sound, and for what model? b) Are the axioms complete for that model? c) Do other (perhaps unexpected) models exist of the same axioms? d) Can the axioms be used to found theories of predicate logic, λ-calculus, unification, and so on?

Sections 2 and 3 define nominal terms and nominal algebra. Section 4 defines substitution as a nominal algebraic theory. Section 5 develops some highly non-trivial technical results. Section 6 answers a) by showing that on a canonical model, our axioms for substitution give rise to something which is recognisably substitution as we might expect it to behave. Section 7 further shows the harder property mentioned in b) that our axioms for substitution precisely characterise what is true of that concrete model. The Conclusions then describes related and future work. Questions c) and d) are answered positively in other papers [4, 5].

2 Nominal terms

We define a syntax of nominal terms. For simplicity fix a **sort of atoms** $\mathbb{A}$ and a **(base) sort of terms** $\mathbb{T}$. Then **sorts** τ are inductively defined by:[3]

$$\tau ::= \mathbb{T} \mid \mathbb{A} \mid [\mathbb{A}]\tau$$

We could admit more sorts of atoms, and base sorts other than $\mathbb{T}$, if we wished.

Our syntax has term-formers: Fix **term-formers** f_ρ to each of which is associated some unique **arity** $\rho = (\tau_1, \ldots, \tau_n)\tau$. We may write $\mathsf{f} : \rho$ for 'f, which has arity ρ'. We assume term-formers:

$$\mathsf{sub} : ([\mathbb{A}]\tau, \mathbb{T})\tau \quad (\tau \in \{\mathbb{T}, [\mathbb{A}]\mathbb{T}\}) \quad \mathsf{var} : (\mathbb{A})\mathbb{T} \quad \mathsf{pair} : (\mathbb{T}, \mathbb{T})\mathbb{T} \quad \mathsf{binder} : ([\mathbb{A}]\mathbb{T})\mathbb{T}$$

The two subs will be used to represent the substitutions from Fig. 1; the condition on τ is for simplicity only.

var, pair, and binder define a sufficiently rich language for our axioms of substitution to have an interesting action, but as mentioned in the Introduction *almost anything else would do* (e.g. see example signatures in [6]).

Finally, we can define the sorted syntax itself:

Fix some countably infinite set of **atoms** $a, b, c, \ldots \in \mathbb{A}$. These model object-level variable symbols (the ones we axiomatise substitution for).

Fix a countably infinite collection of **unknowns** $X, Y, Z, T, U, \ldots$.[4] Intuitively these represent unknown terms. We assume unknowns are inherently sorted and infinitely many populate each sort: so X is shorthand for X_τ, and $X_\mathbb{A}$ and $X_\mathbb{T}$ are two *different* unknowns with confusingly similar names.

Terms t, u, v are inductively defined by the following grammar:

$$t ::= a_\mathbb{A} \mid (\pi \cdot X_\tau)_\tau \mid ([a_\mathbb{A}]t_\tau)_{[\mathbb{A}]\tau} \mid (\mathsf{f}_{(\tau_1,\ldots,\tau_n)\tau}(t^1_{\tau_1}, \ldots, t^n_{\tau_n}))_\tau$$

Here we call $(\pi \cdot X_\tau)_\tau$ a **moderated unknown**; π is described below. We have indicated sorts with a subscript but we shall usually omit them; we repeat the definition above without subscripts, for clarity:

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

A **permutation** π of atoms is a bijection on $\mathbb{A}$ with **finite support** meaning that for some finite set of atoms $\pi(a) \neq a$, and for all other atoms $\pi(a) = a$; in other words, for 'most' atoms π is the identity. As usual write **Id** for the **identity** permutation, π^{-1} for the **inverse** of π, and $\pi \circ \pi'$ for the **composition** of π and π', i.e. $(\pi \circ \pi')(a) = \pi(\pi'(a))$. **Id** is also the identity of composition, i.e. $\mathbf{Id} \circ \pi = \pi$ and $\pi \circ \mathbf{Id} = \pi$. We may abbreviate $\mathbf{Id} \cdot X$ to X. Importantly, we shall write $(a\ b)$ for the permutation which maps a to b and vice versa, and maps all other c to themselves.

[3] So sorts are just $\mathbb{T}$, $[\mathbb{A}]\mathbb{T}$, $[\mathbb{A}][\mathbb{A}]\mathbb{T}$, and so on, and similarly $\mathbb{A}$, $[\mathbb{A}]\mathbb{A}$, and so on.

[4] Unknowns, atoms, and term-formers, are assumed *disjoint*.

In Fig. 1 we have sugared $\mathsf{sub}([a]u, t)$ to $u[a{\mapsto}t]$. We suggestively name a term of this form an **explicit substitution**.

A few more simple notations are useful for later: We call the **size of** t its *inductive rank*.[5] Write $a \in t$ (or $X \in t$) for 'a (or X) **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 $a \notin t$ and $X \notin t$ for 'does not occur in the syntax of t'. Write **syntactic identity** of terms t, u as $t \equiv u$ to distinguish it from provable equality. *Important:* we do not quotient terms in any way.

It may help to show how nominal terms relate to 'ordinary' syntax. For convenience identify atoms with *variable symbols*, then the syntax of the untyped λ-calculus is inductively defined by $e ::= a \mid ee \mid \lambda a.e$. We define a map $(\text{-})'$ to nominal terms by: $a' = \mathsf{var}(a)$, $(e_1 e_2)' = \mathsf{pair}(e_1', e_2')$, $(\lambda a.e)' = \mathsf{binder}([a](e'))$. Even for this simple signature of var, pair, and binder, there are interesting things to say. For example we shall see that $\mathsf{binder}([a]X)$ behaves much like the λ-context $\lambda a.$- where - is a 'hole', and of course sub will allow us to state (and prove!) nontrivial properties of substitution in the presence of those holes. Example 4.1 gives three such properties; there are many more and our main result Theorem 7.5 asserts that we can prove all of them from our axioms.

3 Nominal algebra

We can now do algebra. For us, algebra is the *logic of equality* (no implication, no quantification). We consider a canonical syntax-based model later in Section 6 and find much of interest to say about it in Section 7 — other models of our axioms are the topic of other work [4].

A **freshness (assertion)** is a pair $a\#t$ of an atom and a term. Call a freshness of the form $a\#X$ (so $t \equiv X$) **primitive**. Write Δ for a (possibly infinite) set of *primitive* freshnesses and call it a **freshness context**. We may drop set brackets in freshness contexts, e.g. writing $a\#X, b\#Y$ for $\{a\#X, b\#Y\}$. Also, we may write $a, b\#X$ for $a\#X, b\#X$. Define **derivability** on freshnesses in natural deduction style by:

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

Here f ranges over term-formers,[6] t and $t_1, \ldots, t_n$ range over terms, X ranges over unknowns, and a and b *permutatively* range over atoms, i.e. a and b represent any two *distinct* atoms. We use similar conventions henceforth.

Write $\Delta \vdash a\#t$ when a derivation of $a\#t$ exists using the elements of Δ as assumptions. Say that Δ **entails** $a\#t$ or $a\#t$ **is derivable from** Δ; call this a **freshness judgement**.

[5] In plain english: the depth of a proof that t *is* in the set of terms, using the inductive definition above.

[6] More precisely, f is a *meta-variable* ranging over term-formers.

An **equality (assertion)** is a pair $t = u$ where t and u are terms of the same sort. Define **derivability** on equalities in natural deduction style by:

$$\frac{}{t=t}\ (\mathbf{refl}) \qquad \frac{t=u}{u=t}\ (\mathbf{symm}) \qquad \frac{t=u \quad u=v}{t=v}\ (\mathbf{tran}) \qquad \frac{t_1=u_1 \cdots t_n=u_n}{\mathsf{f}(t_1,\ldots,t_n)=\mathsf{f}(u_1,\ldots,u_n)}\ (\mathbf{congf})$$

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

We may call this the **core theory** and refer to it as CORE. We may write $\Delta \vdash_{\mathsf{CORE}} t = u$ for '$t = u$ is derivable from assumptions Δ in the core theory'; call this an **equality judgement**.

In (**fr**) square brackets denote *discharge* in the sense of natural deduction, as in implication introduction [7]; Δ denotes the other assumptions of the derivation of $t = u$.[7] This is useful because unknowns in a derivation intuitively represent unknown terms, but any finite collection of such terms can mention only finitely many atoms; (**fr**) expresses that we can always find a fresh one.

In (**perm**) read $(a\ b)\cdot t$ as 'swap a and b in t'. It is defined on syntax by:

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

So π propagates through the structure of t until it reaches an atom or a moderated unknown. We can easily verify that $(\pi\circ\pi')\cdot t \equiv \pi\cdot(\pi'\cdot t)$ and $\mathbf{Id}\cdot t \equiv t$.

Here is an example derivation, using the fact that $[a]a \equiv (a\ b)\cdot[b]b$:

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

Provable equality in CORE coincides with provable equality on nominal terms in the sense of nominal unification [2], for details see elsewhere [3]. This corresponds in a suitable sense to α-equivalence, though in a non-trivial way since $\not\vdash_{\mathsf{CORE}} [a]X = [b]X$ does not hold — but $b\#X \vdash_{\mathsf{CORE}} [a]X = [b](b\ a)\cdot X$ does [2].

Nominal Algebra (**NA**) is the theory outlined above, along with the ability to impose axioms. Call a triple $\Delta \vdash t = u$ where Δ is finite an **axiom**. We may write $\vdash t = u$ when Δ **is empty** (the empty set). Call an **instance** of an axiom a step in a derivation where the conclusion is obtained from an axiom by instantiating unknowns by terms,[8] and permutatively renaming atoms, such that the hypotheses are corresponding instances of freshness conditions of the axiom.

---

[7] In sequent style, (**fr**) would be $\dfrac{\Delta, a\#X_1,\ldots,a\#X_n \vdash t=u}{\Delta\vdash t=u}\ (a\notin t,u,\Delta)$.

[8] Instantiation of unknowns is mostly what the reader would expect: textual replacement of X by t. See Sect. 7.1 for the formal definition.

4 SUB: the theory of explicit substitution

NA substitution allows the axioms in Fig. 1. Here (**f**$\mapsto$) represents a schema of axioms, one for each term-former other than var; one particular example is (**sub**$\mapsto$). We make *concrete* choices of atoms a and b, of an unknown T of sort $\mathbb{T}$, and of unknowns $X, X_1, \ldots, X_n$ of appropriate sorts. We call the axioms and the resulting equality a **theory of substitution** and write it SUB.

Example 4.1. The following judgements are derivable in SUB:

1. $a\#Y \vdash_{\mathsf{SUB}} Z[a \mapsto X][b \mapsto Y] = Z[b \mapsto Y][a \mapsto X[b \mapsto Y]]$
2. $b\#Z \vdash_{\mathsf{SUB}} Z[a \mapsto X] = ((b\ a) \cdot Z)[b \mapsto X]$
3. $\vdash_{\mathsf{SUB}} X[a \mapsto \mathsf{var}(a)] = X$

We give only the first derivation in full. We write σ for $[b \mapsto Y]$ and we use the unsugared syntax for the other substitutions.

$$\dfrac{\dfrac{}{\mathsf{sub}([a]Z, X)\sigma = \mathsf{sub}(([a]Z)\sigma, X\sigma)}\,(\mathbf{f}\mapsto) \quad \dfrac{\dfrac{a\#Y}{([a]Z)\sigma = [a](Z\sigma)}\,(\mathbf{abs}\mapsto) \quad \dfrac{}{X\sigma = X\sigma}\,(\mathbf{refl})}{\mathsf{sub}(([a]Z)\sigma, X\sigma) = \mathsf{sub}([a](Z\sigma), X\sigma)}\,(\mathbf{congf})}{\mathsf{sub}([a]Z, X)\sigma = \mathsf{sub}([a](Z\sigma), X\sigma)}\,(\mathbf{tran})$$

For part 2: We must prove $b\#Z \vdash_{\mathsf{SUB}} \mathsf{sub}([a]Z, X) = \mathsf{sub}([b](b\ a) \cdot Z, X)$. By (**congf**), (**refl**), and (**symm**), it suffices to derive $[b](a\ b) \cdot Z = [a]Z$. Using (**perm**) it suffices to derive $a, b\#[a]Z$, which is easy.

For part 3: By (**fr**) we may assume $b\#X$ for some $b \notin X, X[a \mapsto \mathsf{var}(a)]$, i.e. $b \neq a$. By (**tran**) it suffices to derive $X[a \mapsto \mathsf{var}(a)] = ((b\ a) \cdot X)[b \mapsto \mathsf{var}(a)]$ and $((b\ a) \cdot X)[b \mapsto \mathsf{var}(a)] = X$. The former is an instance of part 2 of this example. For the latter it suffices to derive $a\#(b\ a) \cdot X$, by axiom (**ren** $\mapsto$) and $X \equiv (a\ b) \cdot (b\ a) \cdot X$. By (#**X**), this follows from the assumption $b\#X$.

5 SUBfr: explicit substitution rewritten

Equality has no algorithmic content so we have specified *what* is equal, but not *how* to verify it. Rewriting is algorithmic in that sense, given confluence. It is useful to give a rewrite system for SUB.

Nominal rewriting is like nominal algebra, but with a directed notion of equality; terms are taken up to equality in CORE. A rewrite rule $\nabla \vdash l \to r$ may trigger a rewrite in a term t when (an instance of) l is provably equal in CORE to some subterm of t, and the corresponding instance of ∇ is derivable using the ambient context of freshness assumptions Δ (so ∇ is *freshness conditions* on the rewrite rule) [8, 9, 10]. Rewrite rules are given in Fig. 2. Write $\to_\Delta$ for the rewrite relation induced by rewrites in CORE, given Δ.

Say a freshness context Δ' **freshly extends** Δ when $\Delta' = \Delta \cup \Delta''$ where Δ'' may be empty, but if $a\#X \in \Delta''$ then $a \notin \Delta$. Note that the rule (**fr**) precisely 'introduces a Δ'''. So $b\#X, a\#X$ freshly extends $a\#X$ but $a\#Y, a\#X$ does not.

$$
\begin{array}{lllll}
(\mathbf{Rvar}) & & \vdash \mathsf{var}(a)[a \mapsto X] & \to X & \\
(\mathbf{R\#}) & a\#Z & \vdash Z[a \mapsto X] & \to Z & \\
(\mathbf{Rf}) & & \vdash \mathsf{f}(Z_1, \ldots, Z_n)[a \mapsto X] & \to \mathsf{f}(Z_1[a \mapsto X], \ldots, Z_n[a \mapsto X]) & (\mathsf{f} \neq \mathsf{var}, \mathsf{sub}) \\
(\mathbf{Rsub}) & a\#Y & \vdash Z[a \mapsto X][b \mapsto Y] & \to Z[b \mapsto Y][a \mapsto X[b \mapsto Y]] & \\
(\mathbf{Rabs}) & c\#X & \vdash ([c]Z)[a \mapsto X] & \to [c](Z[a \mapsto X]) & \\
(\mathbf{Rren}) & b\#Z & \vdash Z[a \mapsto \mathsf{var}(b)] & \to (b\ a) \cdot Z &
\end{array}
$$

Fig. 2. Substitution as a rewrite system SUBfr

Lemma 5.1 (Rewriting *is* equality). *$\Delta \vdash_{\mathsf{SUB}} t = u$ is derivable if and only if t is related to u by the symmetric transitive reflexive closure of $\to_{\Delta'}$ for some Δ' freshly extending Δ.*

Say a property holds of a triple (Δ, t, u) '**provided Δ has sufficient freshnesses**' when that property holds of some (Δ', t, u) for Δ' freshly extending Δ. So for Δ, t, and u, equality between t and u SUB coincides with rewritability between them in SUBfr, provided that Δ has sufficient freshnesses.

5.1 SUBe and strong normalisation up to SUBe

Substitution has the character of a computation and our re-casting SUB as a rewrite system recognises this. However substitutions also have an awkward 'simultaneous' character. For example

$$\vdash_{\mathsf{SUB}} X[a \mapsto \mathsf{var}(a')][b \mapsto \mathsf{var}(b')][c \mapsto \mathsf{var}(c')] = X[c \mapsto \mathsf{var}(c')][b \mapsto \mathsf{var}(b')][a \mapsto \mathsf{var}(a')]$$

is derivable but there is no obvious direction to the equality and SUBfr does not strongly normalise on the terms.

Call a binary relation (by incredible coincidence write it $\to$) **strongly normalising** when if for $t_1, t_2, \ldots$ we have $t_1 \to t_2 \to t_3 \to \ldots$ then i exists such that if $j \geq i$ then $t_i = t_j$. Clearly this is not the case of any $\to_\Delta$ from SUBfr.

Let SUBe be the NA theory with axioms in Fig. 3.

$$
\begin{array}{lllll}
(\mathbf{Eswap}) & a\#Y, b\#X & \vdash Z[a \mapsto X][b \mapsto Y] & = Z[b \mapsto Y][a \mapsto X] \\
(\mathbf{Egarbage}) & & a\#Z \vdash Z[a \mapsto X] & = Z
\end{array}
$$

Fig. 3. The theory SUBe

Lemma 5.2. *The rules obtained from directing the equalities from* SUBe*, are admissible in* SUBfr*, i.e. every instance of the following can be obtained in* SUBfr*:*

$$
\begin{array}{lllll}
(\mathbf{Rswap}) & a\#Y, b\#X & \vdash Z[a \mapsto X][b \mapsto Y] & \to Z[b \mapsto Y][a \mapsto X] \\
(\mathbf{Rgarbage}) & & a\#Z \vdash Z[a \mapsto X] & \to Z
\end{array}
$$

As a corollary, provable equality in SUBe *implies provable equality in* SUB*.*

We will show that $\rightarrow$ of SUBfr is strongly normalising *up to* provable equality in SUBe. Some auxiliary functions and notations are needed.

Fix Δ and write $a\#v$ for $\Delta \vdash a\#v$ and $\neg a\#v$ for $\Delta \nvdash a\#v$. Let f range over all term-formers excluding sub (but including var), and let . denote the arithmetic product. Define $|v|_b$ by $|v|_b = 0$ if $b\#v$ and *only otherwise* by:

$$|a|_a = 1 \qquad |v[a \mapsto t]|_a = |t|_a + 1 \qquad |v[a \mapsto t]|_b = |v|_b \quad (a\#v)$$
$$|v[a \mapsto t]|_b = |v|_b \quad (\neg a\#v, b\#t) \qquad |v[a \mapsto t]|_b = |v|_b + |t|_b . |v|_a + 1 \quad (\neg a\#v, \neg b\#t)$$
$$|\mathsf{f}(v_1, \ldots, v_n)|_b = |v_1|_b + \cdots + |v_n|_b + 1 \qquad |[a]v|_b = |v|_b + 1 \qquad |\pi \cdot X|_a = 1$$

Then the following inductive definition makes Theorem 5.3 a matter of easy arithmetic:

$$|\mathsf{f}(v_1, \ldots, v_n)| = |v_1| + \cdots + |v_n| \quad (n > 0) \qquad |\mathsf{f}()| = 1 \qquad |[a]v| = |v|$$
$$|\pi \cdot X| = 1 \qquad |a| = 1 \qquad |v[a \mapsto t]| = |t| . |v|_a + |v|$$

Theorem 5.3. *If $t \rightarrow_\Delta t'$ then either $\Delta \vdash_{\mathsf{SUBe}} t = t'$ or $|t|^\Delta > |t'|^\Delta$.*
As a result SUBfr *is strongly normalising up to provable equality in* SUBe.

Write $\rightarrow_\Delta^*$ for the transitive reflexive (but not symmetric) closure of $\rightarrow_\Delta$. We note that $\Delta \vdash_{\mathsf{SUBe}} t = u$ does *not* imply $t \rightarrow_\Delta^* u$ or $u \rightarrow_\Delta^* t$. For a counterexample consider $t \equiv \mathsf{var}(a)[b \mapsto \mathsf{var}(c)]$ and $u \equiv \mathsf{var}(a)[b \mapsto \mathsf{var}(c')]$.

5.2 Garbage and garbage-collection

Call the pair $\Delta \vdash t$ a **(nominal) term-in-context**. Equalities, and rewrites, are on terms *in context*. Only if t is closed is context irrelevant.

Say that a term-in-context $\Delta \vdash t$ has **garbage** when:

- for some subterm $t'[a \mapsto u]$ it is the case that $\Delta \vdash a\#t'$, or
- for some subterm of the form $\pi \cdot X$ and some $a \in \pi \cdot X$, it is the case that both $\Delta \vdash a\#X$ and $\Delta \vdash \pi(a)\#X$ hold.

Otherwise say $\Delta \vdash t$ has **no garbage**.

Example 5.4. Terms in the top line have no garbage, the others do:

$$\vdash \mathsf{var}(a)[a{\mapsto}X] \qquad \vdash X[a{\mapsto}Y] \qquad a\#X \vdash (a\ b){\cdot}X$$
$$\vdash \mathsf{var}(a)[b{\mapsto}X] \quad a\#X \vdash X[a{\mapsto}Y] \quad a, b\#X \vdash (a\,b){\cdot}X \quad a, b\#X \vdash ((a\,b){\cdot}X)[c{\mapsto}\mathsf{var}(c')].$$

Alas the rewrite rule (**Rsub**) in SUBfr may introduce garbage (the innermost $[b \mapsto Y]$ acting on X). Alas also, α-equivalence (provable equality in CORE) may introduce garbage, for example $a, b\#X \vdash_{\mathsf{CORE}} (a\ b) \cdot X = X$.[9]

Say that $\Delta \vdash t$ is a **SUBfr-normal form** when $\Delta \vdash_{\mathsf{SUBe}} t = t'$ for all t' with $t \rightarrow_\Delta t'$. So a SUBfr-normal form may still rewrite with a rule from SUBfr ((**R**#) and (**Rsub**) to be precise) but that rewrite just has no effect up to provable equality in SUBe.

[9] This is one reason SUB is a difficult beast to handle, and it took us quite a while to decide to split it up as a rewrite system over a provable equality, and then which provable equality to use.

Lemma 5.5 (Garbage collection). *For any $\Delta \vdash t$ there is some t' a* SUBfr*-normal form with no garbage such that $t \to_\Delta^* t'$. Furthermore, if $\Delta \vdash_{\mathsf{SUBe}} t = u$ is derivable then there is some* SUBfr*-normal form v with no garbage such that $t \to_\Delta^* v$ and $u \to_\Delta^* v$.*

5.3 Confluence

A standard way to prove confluence is to prove *local* confluence and strong normalisation. But SUBfr is not strongly normalising, see the example of Sect. 5.1. The interest of this proof is that we use SUBe to 'cancel that out'.

Lemma 5.6. *Rewrites of $\Delta \vdash t$ in the nominal rewrite system* SUBfr *are locally confluent in a context with sufficient freshnesses, up to provable equivalence in* CORE*.*

(Recall that the restriction on contexts is not a 'real' one, because we have (**fr**).)

Theorem 5.7. SUBfr *is confluent in a context with sufficient freshnesses.*

Proof. SUBfr is strongly normalising up to provable equality in SUBe by Theorem 5.3. It is locally confluent (in a context with sufficient freshnesses) by Lemma 5.6. By Newman's Lemma [11] it is confluent up to provable equality in SUBe. Finally, we use the second part of Lemma 5.5. □

From this follow:

Theorem 5.8. SUB *is conservative over* CORE*. That is, $\Delta \vdash_{\mathsf{SUB}} t = u$ if and only if $\Delta \vdash_{\mathsf{CORE}} t = u$, assuming that neither t nor u mention explicit substitution.*

Corollary 5.9 (Consistency). *For all Δ there are t, u such that $\Delta \nvdash_{\mathsf{SUB}} t = u$.*

6 Ground terms

Call terms g and h **ground terms** when they do not mention unknowns or explicit substitutions. These are inductively characterised by

$$g ::= a \mid \mathsf{f}(g, \ldots, g) \mid [a]g$$

where f ranges over all term-formers except for sub.

We consider the **meaning of explicit substitution** on ground terms (making a connection between $[a \mapsto t]$ and actual capture-avoiding substitution on syntax).

Define a 'free atoms of' function $fa(g)$ on ground terms inductively as follows:

$$fa(a) = \{a\} \qquad fa(\mathsf{f}(g_1, \ldots, g_n)) = \bigcup_{1 \le i \le n} fa(g_i) \qquad fa([a]g) = fa(g) \setminus \{a\}$$

Define the **support** of g by $\mathrm{supp}(g) = \{a \mid \nvdash a\#g\}$.

Lemma 6.1. *$fa(g) = \mathrm{supp}(g)$.*

For each finite set of atoms arbitrarily choose some canonical 'fresh' atom not in that finite set. Then define a **ground substitution action** $g[h/a]$ on ground terms of sort $\mathbb{T}$ and $[\mathbb{A}]\mathbb{T}$ by

$$\mathsf{var}(b)[h/a] \equiv \mathsf{var}(b) \quad \mathsf{var}(a)[h/a] \equiv h \quad \mathsf{f}(g_1,\ldots,g_n)[h/a] \equiv \mathsf{f}(g_1[h/a],\ldots,g_n[h/a])$$
$$([a]g)[h/a] \equiv [a]g \quad ([b]g)[h/a] \equiv [b](g[h/a]) \quad (b \notin fa(h))$$
$$([b]g)[h/a] \equiv [c](g[\mathsf{var}(c)/b][h/a]) \quad (b \in fa(h),\ c \text{ fresh}),$$

where f ranges over all term-formers excluding var (and sub of course), and 'c fresh' means c is fresh for $\{a,b\} \cup fa(g) \cup fa(h)$ according to our arbitrary choice.

Theorem 6.2. $\vdash_{\mathsf{SUB}} g[h/a] = g[a \mapsto h]$ *is always derivable.*

Proof. By straightforward induction on the structure of g, using Lemma 6.1. □

Define an α-equivalence relation $g =_\alpha h$ inductively by:

$$a =_\alpha a \quad \frac{g_1 =_\alpha h_1 \cdots g_n =_\alpha h_n}{\mathsf{f}(g_1,\ldots,g_n) =_\alpha \mathsf{f}(h_1,\ldots,h_n)} \quad \frac{g =_\alpha h}{[a]g =_\alpha [a]h} \quad \frac{g[\mathsf{var}(c)/a] =_\alpha h[\mathsf{var}(c)/b]}{[a]g =_\alpha [b]h} \quad (c \text{ fresh})$$

Here 'c fresh' means c fresh for $\{a,b\} \cup fa(g) \cup fa(h)$.

Theorem 6.3. $g =_\alpha h$ *if and only if* $\vdash_{\mathsf{SUB}} g = h$ *is derivable.*

Proof. By Theorem 5.8 $\vdash_{\mathsf{SUB}} g = h$ is equivalent to $\vdash_{\mathsf{CORE}} g = h$. By results about nominal terms [2, 3] this happens *precisely* when $g =_\alpha h$. □

So intuitively: *On ground terms* $\vdash_{\mathsf{SUB}} g = h$ is α*-equivalence and explicit substitution* is *capture-avoiding substitution.*

7 ω-completeness

How do we know that SUB *really is* an axiomatisation of substitution? We now give a soundness and completeness result. ω-completeness (notation from [12]) is 'soundness and completeness with respect to the closed term model'.

We need a number of technical definitions and lemmas.

7.1 Meta-level substitution

Call a **substitution** σ a finitely supported function from unknowns to terms of the same sort. Here, finite support means that $\sigma(X) \equiv \mathbf{Id} \cdot X$ for all but finitely many unknowns X, i.e. for 'most' X.

Write $[t/X]$ for σ defined by $\sigma(X) \equiv t$ and $\sigma(Y) \equiv \mathbf{Id} \cdot Y$, for all $Y \not\equiv X$.

Let $t\sigma$ ('σ **applied to** t') be inductively defined by:

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

We may call $t\sigma$ an **instance** of t. The substitution action extends to freshness assertions, equalities, and so on. We extend notations and terminologies silently.

Note that this does not avoid capture; $([a]X)[a/X] \equiv [a]a$ and in this formal sense X is 'meta' and really does represent an unknown *term*.

Call σ **nontrivial on** X when $\sigma(X) \not\equiv \mathbf{Id} \cdot X$. By assumption σ is nontrivial for only finitely many unknowns. Say that σ' **extends** σ when $\sigma'(X) \equiv \sigma(X)$ whenever σ is nontrivial on X. Call σ **closing** for some collection of freshness and equality assertions S when $\sigma(X)$ is a closed term for every $X \in S$. We will not mention S when it is clear from the context.

Say a closing σ is **Δ-consistent** when $\vdash a\#\sigma(X)$ for all $a\#X \in \Delta$.

Lemma 7.1. *Fix Δ, X, and closed term v. If $\vdash a\#v$ for every $a\#X \in \Delta$ then there is a Δ-consistent closing σ which extends $[v/X]$.*

7.2 Suspended explicit substitutions

Suppose a term-in-context $\Delta \vdash t \equiv (\pi \cdot t')[a_1 \mapsto t_1] \dots [a_m \mapsto t_m]$ is such that $\Delta \vdash a_i\#t_j$ for all $1 \leq i, j \leq m$. Here m may equal 0, in which case $t \equiv (\pi \cdot t')$.

Then call the partial syntax $(\pi \cdot \text{-})[a_1 \mapsto t_1] \dots [a_m \mapsto t_m]$ a **suspended substitution**; we generally let α and β vary over suspended substitutions. β will typically be $(\pi' \cdot \text{-})[b_1 \mapsto u_1] \dots [b_n \mapsto u_n]$. Suspended substitutions have a natural action on terms $t'\alpha$ given by replacing - by t'.

Lemma 7.2. *Assuming sufficient freshnesses, if $\Delta \vdash t$ is a* SUBfr*-normal form with no garbage then every explicit substitution t mentions is in a subterm u such that $\Delta \vdash_{\mathsf{CORE}} u = X\alpha$ and $\Delta \vdash X\alpha$ has no garbage.*

Proof. Otherwise there is a rewrite such that $\Delta \vdash t \to t'$ where $\Delta \not\vdash_{\mathsf{SUBe}} t = t'$. □

Lemma 7.3. *Suppose t and u are* SUBfr*-normal forms with no garbage. Then $\Delta \vdash_{\mathsf{SUBe}} t = u$* precisely *when one of the following hold:*

1. *$t \equiv a$ and $u \equiv a$.*
2. *$t \equiv \pi \cdot X$ and $u \equiv \pi \cdot X$.*
3. *$t \equiv [a]t'$ and $u \equiv [a]u'$ and $\Delta \vdash_{\mathsf{SUBe}} t' = u'$.*
4. *$t \equiv [a]t'$ and $u \equiv [b]u'$ and $\Delta \vdash_{\mathsf{SUBe}} (b\ a) \cdot t' = u'$ and $\Delta \vdash b\#t'$.*
5. *$t \equiv \mathsf{f}(t_1, \dots, t_n)$ and $u \equiv \mathsf{f}(u_1, \dots, u_n)$ and $\Delta \vdash_{\mathsf{SUBe}} t_i = u_i$ for $1 \leq i \leq n$ ($\mathsf{f} \neq \mathsf{sub}$).*
6. *$t \equiv t'\alpha$ and $u \equiv u'\beta$ and $m = n > 0$ and $\Delta \vdash_{\mathsf{SUBe}} t' = u'$, and for every i there is a unique j such that $\pi^{-1}(a_i) = \pi'^{-1}(b_j)$ and $\Delta \vdash_{\mathsf{SUBe}} t_i = u_j$ — and similarly for every j.*

Theorem 7.4. *Equality in* SUB *is decidable.*

Proof. Given Δ, t and u, we can calculate whether $\Delta \vdash_{\mathsf{SUB}} t = u$ is derivable:

1. Rewrite t and u to SUBfr-normal forms t' and u', using Theorem 5.3.
2. Remove garbage from t' and u', using Lemma 5.5.
3. Check if the top-level term-formers of t' and u' satisfy the criteria stated in Lemma 7.3; for each of the new proof obligations $\Delta \vdash_{\mathsf{SUBe}} t'' = u''$ go to step 2.[10]
4. If all criteria checks were successful, return true; otherwise false. □

[10] Garbage could be introduced by case 4 of Lemma 7.3, so we must remove it.

7.3 ω-completeness

ω-completeness is soundness and completeness with respect to a model made out of closed terms (terms which do not mention unknowns X); since syntax *is* the canonical example on which substitution is defined, soundness *and completeness* with respect to this model is a powerful argument that in theory SUB, we got it right. An NA judgement $\Delta \vdash_{\mathsf{SUB}} t = u$ has the flavour of a universal quantification over the unknowns it mentions. Soundness for the closed terms model means: if $\Delta \vdash_{\mathsf{SUB}} t = u$ is derivable then all *instances* of this equality (subject to Δ) on closed terms are derivable. Much harder to prove is that furthermore if all instances are derivable (subject to Δ), then so is $\Delta \vdash_{\mathsf{SUB}} t = u$.

Call SUB **ω-complete** when if $\vdash_{\mathsf{SUB}} t\sigma = u\sigma$ is derivable for all Δ-consistent closing substitutions σ (for Δ, t and u), then $\Delta \vdash_{\mathsf{SUB}} t = u$ is derivable.

Theorem 7.5. SUB *is ω-complete.*

Proof. By contraposition. Suppose *not* $\Delta \vdash_{\mathsf{SUB}} t = u$. We construct Δ-consistent σ such that *not* $\vdash_{\mathsf{SUB}} t\sigma = u\sigma$. It suffices to do this for some Δ' which freshly extends Δ; for convenience assume $\Delta = \Delta'$. By the first part of Lemma 5.5 and by Lemma 5.1 we may suppose t and u are SUBfr-normal forms with no garbage. By Lemma 7.2 further assume they have the particular form mentioned in that result. By Lemma 5.2 also $\Delta \vdash_{\mathsf{SUBe}} t = u$ is not derivable. So now we must prove

$$\big(\Delta \nvdash_{\mathsf{SUBe}} t = u\big) \quad \text{implies} \quad \big(\exists \sigma \text{ closing and } \Delta\text{-consistent. } \nvdash_{\mathsf{SUB}} t\sigma = u\sigma\big),$$

where t and u have the structure as described above.

We work by induction on the size of t and u. We proceed by case distinction (we omit routine cases, and calculations concerning size):

- $t \equiv a$ and $u \equiv b$. By Theorem 6.3.
- $t \equiv \mathsf{f}(t_1, \ldots, t_m)$ and $u \equiv \mathsf{g}(u_1, \ldots, u_n)$ $(\mathsf{f}, \mathsf{g} \neq \mathsf{sub})$. Apply *any* closing Δ-consistent σ (easy to manufacture), use Theorem 6.2 to remove all explicit substitutions, and then use Theorem 6.3 to conclude $\nvdash_{\mathsf{SUB}} t\sigma = u\sigma$.
- $t \equiv \mathsf{f}(t_1, \ldots, t_m)$ and $u \equiv \mathsf{f}(u_1, \ldots, u_m)$ $(\mathsf{f} \neq \mathsf{sub})$. By part 5 of Lemma 7.3 $\Delta \nvdash_{\mathsf{SUBe}} t_i = u_i$ for some i. We use the inductive hypothesis and Theorems 6.2 and 6.3.
- $t \equiv [a]t'$ and $u \equiv [b]u'$. Using Lemma 7.3 $\Delta \nvdash_{\mathsf{SUBe}} (b\ a) \cdot t' = u'$ or $\Delta \nvdash b\#t'$. If $\Delta \nvdash b\#t'$, then choose appropriate σ and use Theorems 6.2 and 6.3. If $\Delta \nvdash_{\mathsf{SUBe}} (b\ a) \cdot t' = u'$ then we can remove possible garbage from $(b\ a) \cdot t'$ without increasing size, apply the inductive hypothesis, and finally use Theorems 6.2 and 6.3.
- $t \equiv X\alpha$ and $u \equiv X\beta$ and $m = n > 0$. Using Lemma 7.3 and notation from that result, $\Delta \vdash_{\mathsf{SUBe}} t = u$ precisely when $\Delta \vdash_{\mathsf{SUBe}} t_i = u_j$ for every i, j such that $\pi^{-1}(a_i) = \pi'^{-1}(b_j)$. So suppose i and j are such that $\Delta \nvdash_{\mathsf{SUBe}} t_i = u_j$.
  Now using pair, var, and binder, generate v such that $a_i \in \mathrm{supp}(v)$ and $\mathrm{supp}(v)$ is otherwise fresh (thus, disjoint from atoms mentioned in t and u). Choose Δ-consistent closing σ extending $[v/X]$ using Lemma 7.1. Then by Theorems 6.2 and 6.3 we see that $\nvdash_{\mathsf{SUB}} t\sigma = u\sigma$.

□

8 Conclusions

Substitution underlies quantifiers in predicate logics, the λ-binder of the λ-calculus, unification, and lots more besides. It is a *central*, not incidental, feature of these systems. This paper throws a new and unexpected light on this profoundly important common denominator.

Future work on nominal techniques needs a *nominal* axiomatisation of substitution. This paper provides that and the work has already found application in concurrent work; we use it to 'power' a nominal axiomatisation of first-order logic [5], and we develop abstract (non-term-based) models of SUB in [4]. Such enterprises would not be mathematically secure without Theorem 7.5 to tell us that we got our foundations right.

We have considered *one* substitution and Theorem 7.5 tells us it is capture-avoiding substitution. We can think of other kinds of substitution, for example context substitution which does not avoid capture; the basic principles are clear in this paper and there are a great many possibilities for applying them elsewhere.

Note how we decompose an equational system into a rewrite system over a simpler equational system, and we fully exploit results of nominal rewriting and nominal algebra as well as detailed calculations on terms (such as the notion of measure we use to prove confluence). Similar techniques may be useful for other systems; they seemed to arise in our treatment of first-order logic [5].

We find the nominal terms treatment of binding pleasingly clean; a nominal term such as $\lambda[a]X$ or $\forall[a]X$ corresponds *exactly* and *syntax-for-syntax* to what we intend when we write $\lambda a.t$ and $\forall a.\phi$, right down to the way in which t and ϕ are instantiated. Other approaches to binding involve some degree of emulation (index lifting in de Bruijn [13], type-raising in type-based techniques [14]). Thus a *nominal* treatment of substitution is worthwhile to investigate in itself.

8.1 Related work

Crabbé [15, 16] axiomatises substitution much like us and shares (in our terminology) atoms and freshness conditions. Crabbé does not treat binding.[11] So our substitution is *capture-avoiding*. Also, for us atoms and freshness side-conditions are parts of a broader nominal framework whereas Crabbé expresses them in first-order logic; we feel that nominal techniques have given us a cleaner separation of the layers of complexity hidden in these deceptively simple ideas.

Feldman [17] gives an algebraic axiomatisation inspired by a concrete model of functions/evaluations. His axioms are closer in spirit to Cylindric Algebras [1] and Lambda Abstraction Algebras [18, 19]. The three approaches share an infinity of term-formers which are 'morally' precisely $\lambda[a]$, $\text{-}[a \mapsto \text{-}]$, and $\exists[a]$. We see the advantage of our treatment as systematising and formalising precisely what rôle the atoms really have. In any case the approaches above *cannot directly*

[11] He declares as much: '... we are not concerned with the notion of bound variable' [16, page 2]. See also the axioms (there is no (**abs**$\mapsto$)) and the soundness and completeness result — Crabbé's model is based on (in our notation) var and pair, whereas we consider a model based on var, pair, binder, and sub.

express (**ren**$\mapsto$), (#$\mapsto$), and (**abs**$\mapsto$), even though instantiations are derivable for closed terms by calculations parametric over their specific structure.

Combinatory Algebra (CA) [20] and related systems implement substitution by 'pipes' (e.g. the translation of λ-terms into CA [20]). General truths such as (#$\mapsto$) are only provable for fixed closed terms by calculations parametric over its specific structure.

Lescanne's classic survey [13] and the thesis of Bloo [21] chart a vast literature on λ-calculi with explicit substitutions. These decompose β-reduction as a rule to introduce explicit substitution ($(\lambda a.u)t \to u[a \mapsto t]$), and explicit rules for that substitution's subsequent behaviour (which is to substitute, of course). These calculi are designed to measure the cost of a β-reduction (in an implementation, which may be based on de Bruijn indexes [22] or on named variable symbols). They do not *axiomatise* substitution, they *implement* it. For example, 'confluence' is a typical correctness criterion for a calculus, and 'ω-completeness' is not.

8.2 Future work

Nominal unification [2] is 'merely' unification of nominal terms up to CORE. Our confluence results are a step towards unification up to SUB. Nominal Unification is to be compared (in a sense we do not discuss here) with higher-order patterns [23]; we now suggest that unification up to SUB is to be compared with higher-order unification [24] — with the difference that SUB is weaker, because there is no λ and application, only their combination as 'substitution'.

There is no obstacle to taking SUB *over itself* — that is, to taking what we write in this paper as, say, $(X[a \mapsto Y])[t/X]$ and expressing it in a stronger axiom system as $(X[a \mapsto Y])[X \mapsto \mathcal{T}]$ where $\mathcal{T}$ is a 'stronger' meta-variable. This relates to the NEW calculus of contexts [25] and hierarchical nominal rewriting [26] investigated by the first author, but much more is possible and there are many substitutions out there which we could axiomatise.

Armed with SUB and the knowledge that it is correct in the sense of Theorem 7.5 we hope to develop logics and λ-calculi with a fundamentally new, beautiful, and mathematically advantageous, way of treating substitution and more generally internalising the meta-level.

References

[1] Burris, S., Sankappanavar, H.: A Course in Universal Algebra. Springer (1981) Available online.

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

[3] Gabbay, M.J., Mathijssen, A.: Nominal algebra. Submitted STACS'07 (2006)

[4] Gabbay, M.J., Marin, A., Rota Bulò, S.: A nominal semantics for simple types. Submitted STACS'07 (2006)

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

[6] 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
[7] Hodges, W.: Elementary predicate logic. In Gabbay, D., Guenthner, F., eds.: Handbook of Philosophical Logic, 2nd Edition. Volume 1. Kluwer (2001) 1–131
[8] Fernández, M., Gabbay, M.J., Mackie, I.: Nominal rewriting systems. In: Proc. 6th Int. ACM SIGPLAN Conf. on Principles and Practice of Declarative Programming (PPDP'2004), ACM (2004) 108–119
[9] Fernández, M., Gabbay, M.J.: Nominal rewriting. Information and Computation (2005) Accepted for publication.
[10] Fernández, M., Gabbay, M.J.: Nominal rewriting with name generation: abstraction vs. locality. In: Proc. 7th Int. ACM SIGPLAN Conf. on Principles and Practice of Declarative Programming (PPDP'2005), ACM (2005) 47–58
[11] Newman, M.: On theories with a combinatorial definition of equivalence. Annals of Mathematics **43**(2) (1942) 223–243
[12] Groote, J.F.: A new strategy for proving omega-completeness applied to process algebra. In Baeten, J., Klop, J., eds.: CONCUR '90: Proceedings of the Theories of Concurrency: Unification and Extension. Volume 458 of LNCS., London, UK, Springer-Verlag (1990) 314–331
[13] Lescanne, P.: From lambda-sigma to lambda-upsilon a journey through calculi of explicit substitutions. In: POPL '94: Proc. 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press (1994) 60–69
[14] Paulson, L.C.: The foundation of a generic theorem prover. Journal of Automated Reasoning **5**(3) (1989) 363–397
[15] Crabbé, M.: Une axiomatisation de la substitution. Comptes rendus de l'Académie des Sciences de Paris, Série I **338** (2004) 433–436
[16] Crabbé, M.: On the notion of substitution. Logic Journal of the IGPL **12 n.2** (2004) 111–124
[17] Feldman, N.: Axiomatization of polynomial substitution algebras. Journal of Symbolic Logic **47**(3) (1982) 481–492
[18] Lusin, S., Salibra, A.: The lattice of lambda theories. Journal of Logic and Computation **14 n.3** (2004) 373–394
[19] Salibra, A.: On the algebraic models of lambda calculus. Theoretical Computer Science **249**(1) (2000) 197–240
[20] Barendregt, H.P.: The Lambda Calculus: its Syntax and Semantics (revised ed.). Volume 103 of Studies in Logic and the Foundations of Mathematics. North-Holland (1984)
[21] Bloo, R.: Preservation of Termination for Explicit Substitution. PhD thesis, Eindhoven University of Technology, Eindhoven (1997)
[22] 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
[23] Miller, D.: A logic programming language with lambda-abstraction, function variables, and simple unification. Journal of Logic and Computation **1**(4) (1991) 497536
[24] Huet, G.: Higher order unification 30 years later. In: TPHOL 2002. Number 2410 in LNCS (2002) 3–12
[25] Gabbay, M.J.: A new calculus of contexts. In: Proc. 7th Int. ACM SIGPLAN Conf. on Principles and Practice of Declarative Programming (PPDP'2005), ACM (2005)
[26] Gabbay, M.J.: Hierarchical nominal rewriting. In: LFMTP'2006. (2006) 32–47