DOI 10.1007/s00165-007-0056-1
BCS © 2008
Formal Aspects of Computing (2008) 20: 451–479

Formal Aspects of Computing

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, Scotland, UK. E-mail: murdoch.gabbay@gmail.com
[2] Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract. Substitution is fundamental to the theory of logic and computation. Is substitution something that we define on syntax on a case-by-case basis, or can we turn the idea of substitution into a mathematical object? We give axioms for substitution and prove them sound and complete with respect to a canonical model. As corollaries we obtain a useful conservativity result, and prove that equality-up-to-substitution is a decidable relation on terms. These results involve subtle use of techniques both from rewriting and algebra. A special feature of our method is the use of nominal techniques. These give us access to a stronger assertion language, which includes so-called 'freshness' or 'capture-avoidance' conditions. This means that the sense in which we axiomatise substitution (and prove soundness and completeness) is particularly strong, while remaining quite general.

Keywords: Substitution; Nominal techniques; Nominal algebra; Binding; Capture-avoidance; Nominal rewriting; Omega-completeness

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, terms of some process calculus, 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 [BS81].

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

$$v[a \mapsto t][b \mapsto u] = v[b \mapsto u][a \mapsto t[b \mapsto u]] \qquad \text{provided } a \notin fv(u)$$

This is *not* algebraic, because of the side-condition $a \notin fv(u)$. 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?

Correspondence and offprint requests to: M. J. Gabbay, E-mail: murdoch.gabbay@gmail.com

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

Fig. 1. Axioms of SUB

We shall argue that the properties of Fig. 1 axiomatise substitution, all of substitution, and nothing but substitution. We express them in Nominal Algebra as a theory called SUB (given formal meaning in the rest of this paper). Informally, the axioms express the following:

- (**var**$\mapsto$): A variable a with a replaced by T, is T.
- (#$\mapsto$): If a is fresh for X then X with a replaced by T is X.
- (**f**$\mapsto$): Substitution distributes through term-formers; f ranges over them.
- (**abs**$\mapsto$): Substitution of a distributes under an abstraction $[b]U$, provided a capture-avoidance condition is satisfied (b is fresh for T).
- (**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.
- ($\eta\mapsto$): $t[a \mapsto u]$ is sugar for $\mathsf{sub}([a]t, u)$. In ($\eta\mapsto$), X is a term *not* of the form $[a]t$ so we cannot use that sugar. The reader can think of ($\eta\mapsto$) as related to *η-equality* from the world of the λ-calculus, though X is not a function. The reader can also think of ($\eta\mapsto$) as related to a known property of the *atoms-concretion* operation of Gabbay-Pitts abstraction [GP02], though sub is not atoms-concretion. More on this in Subsection 1.2.

Formally, Fig. 1 uses nominal terms [UPG04] as a syntax, and nominal algebra [GM07, GM06b] as an algebraic framework. We describe these below.

A number of **questions** now arise:

1. Is this substitution and if so, in what sense; for what model are the axioms sound?
2. Are the axioms complete for that model?
3. Do other (perhaps unexpected) models exist of the same axioms?
4. Can theories be built on top of this one by adding axioms for predicate logic, functional programming, unification and logic programming, simultaneous equations, and so on?

1.1. Answers to questions and map of the paper

Section 2 defines *nominal terms* and *nominal algebra*, which fix the syntax and judgement forms we use for our axiomatisation. Section 3 creates a concrete model out of syntax and defines a completely standard capture-avoiding substitution action on the model—this is what we want to capture in axioms. Section 4 defines a relatively weak subset of SUB which we call SIMP (for 'simple') which is *sound* for a concrete model, but *not complete*. This answers question 1 above. Section 5 proves useful properties of this simple axiomatisation; here we make heavy use of techniques from (nominal) rewriting.

Section 6 explores the stronger axioms of Fig. 1 formally as a nominal algebra theory. Decidability of these axioms is proved in Subsect. 6.2, and completeness with respect to the concrete model from Sect. 3 is proved in Subsect. 6.3. This answers question 2 above and completes the proof that Fig. 1 *does* represent substitution.

In other as-yet unpublished work the first author shows how Fraenkel–Mostowski set theory (the underlying model of nominal techniques) [GP02] supports a substitution action. In the case of a set representing a term in the concrete model of this paper, the substitution action coincides with capture-avoiding substitution. However, it extends smoothly, in our opinion quite remarkably, to the entire set theoretic universe. This is an answer for question 3 above and, in our opinion, provides independent evidence that substitution has independent mathematical stature.

In other work [GM07, GM06c] we have investigated how the axiomatisation of substitution can be used to develop algebraic theories, for example of first-order logic. This is the start of an answer to question 4.

The conclusions discuss related and future work.

1.2. Relation with the conference version

A conference version of this paper has appeared [GM06a]. In this paper the technical machinery has been revised and considerably simplified by making more use of models.

As part of these changes we identify an *intermediate* axiomatisation SIMP. This captures SUB on closed terms but is weaker than SUB on open terms, in a sense made precise in Theorem 5.24. SIMP can be identified with the usual definition of capture-avoiding substitution.

We give a procedure for converting equalities in SUB into equivalent (in a suitable sense) equalities in SIMP over an extended signature (see Subsect. 6.2). This is part of the proof of decidability of SUB, and it yields a clearer and more efficient algorithm than the one in the conference version.

This paper includes a new axiom $(\eta\mapsto)$, correcting a technical error of the conference paper: $(\eta\mapsto)$ is necessary for ω-completeness (Subsect. 6.3). In fact $(\eta\mapsto)$ is only necessary if we admit unknowns X of abstraction sort $[\mathbb{A}]\mathbb{T}$ (using terminology from the rest of this paper, or from [GM06a]). Although it is not strictly necessary to admit unknowns of abstraction sort, it allows us to talk about substitution at a more *general* level, resulting in a more smooth presentation (see Subsect. 6.4).

2. Nominal algebra

2.1. The syntax of nominal terms

First, we define a syntax of nominal terms. The syntax we use here is tailored to our application of axiomatising substitution; see elsewhere for general treatments [UPG04, FG07].

Definition 2.1 Fix a **base sort** $\mathbb{T}$ and call it the **sort of terms**. Define **sorts** τ by the following grammar:

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

This simple sort system is sufficient to make sure that terms are well-formed; the reader should not necessarily view it as a 'typing system'.

Extensions with pair sorts $\tau \times \tau$, or base sorts other than $\mathbb{T}$ (e.g. to model terms *and formulae* in first-order logic [GM06c]) cause no essential difficulties, aside from inflating the mathematics with extra cases.

Definition 2.2 Fix some countably infinite set of **atoms** $a, b, c, \ldots \in \mathbb{A}$. These model object-level variable symbols (the ones we substitute for).

Fix a countably infinite collection of **unknowns** $X, Y, Z, T, U, \ldots$. These represent unknown terms of sort $\mathbb{T}$ or $[\mathbb{A}]\mathbb{T}$. We assume unknowns are inherently sorted and infinitely many populate each sort. X is shorthand for X_τ, $\tau \in \{\mathbb{T}, [\mathbb{A}]\mathbb{T}\}$. If we write $X_\mathbb{T}$ and $X_{[\mathbb{A}]\mathbb{T}}$ then we have two different unknowns with confusingly similar names. Typically we take T and U to have sort $\mathbb{T}$.

A **permutation** π of atoms is a bijection on atoms 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.

A **term-former** f is a formal symbol to which is associated an **arity** $(\tau_1, \ldots, \tau_n)\tau$. We may write $\mathsf{f} : (\tau_1, \ldots, \tau_n)\tau$ for f **which has arity** $(\tau_1, \ldots, \tau_n)\tau$. We call a collection of term-formers a **signature**.

Atoms, unknowns, signatures and other distinct syntactic classes, are assumed *disjoint*.

Convention 2.3 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.

In the rare circumstances where we do not want this behaviour, we use primes and subscripts, such as a', a_X and a_{Xi} (where X is an unknown and i is an index).

Definition 2.4 Let **terms** t, u, v be inductively defined by:

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

Here a ranges over atoms, X over unknowns, π over permutations, and f over term-formers.

We call $\pi \cdot X$ a **moderated unknown**, representing an unknown term on which a permutation of atoms is performed when it is instantiated. An **abstraction** $[a]t$ represents a term t in which an atom a is abstracted.

Call a term **closed** when it does not mention any unknowns.

Write **syntactic identity** of terms t and u as $t \equiv u$ to distinguish it from provable equality. Note that if $\pi = \pi'$ then $\pi \cdot X \equiv \pi' \cdot X$, since permutations are represented by themselves. *Important:* we do not quotient terms in any way.

Definition 2.5 Let **(valid) sorting assertions** $t : \tau$, read 't has sort τ' be inductively defined by:

$$\frac{}{a : \mathbb{T}} \qquad \frac{}{\pi \cdot X_\tau : \tau} \qquad \frac{t : \tau}{[a]t : [\mathbb{A}]\tau} \qquad \frac{t_1 : \tau_1 \quad \cdots \quad t_n : \tau_n}{\mathsf{f}(t_1, \ldots, t_n) : \tau} \quad (\mathsf{f} : (\tau_1, \ldots, \tau_n)\tau).$$

Note that atom a represents a variable symbol of sort $\mathbb{T}$. Also note that in the rules for $\pi \cdot X_\tau$ and $[a]t$, τ is restricted to $\mathbb{T}$ and $[\mathbb{A}]\mathbb{T}$.

We consider only terms that adhere to the sorting assertions from now on.

Example 2.6 (Lambda calculus) A signature for the lambda calculus consists of the following term-formers:

$$\mathsf{app} : (\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{lam} : ([\mathbb{A}]\mathbb{T})\mathbb{T}.$$

The sorting system is such that a well-sorted term of the form $\mathsf{lam}(t)$ must be of the form $\mathsf{lam}(\pi \cdot X)$ (so $t \equiv \pi \cdot X$) or $\mathsf{lam}([a]t')$ (so $t \equiv [a]t'$). If $\mathsf{lam}(t)$ is closed then it *must* be of the form $\mathsf{lam}([a]t')$.

It may help to show how nominal terms in this signature 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:

$$(a)' = a \qquad (e_1 e_2)' = \mathsf{app}((e_1)', (e_2)') \qquad (\lambda\, a.e)' = \mathsf{lam}([a](e)').$$

We shall see that $\mathsf{lam}([a]X)$ behaves much like the λ-context $\lambda\, a.\text{-}$ where - is a 'hole'.

Substitution is just a term-former, and this is reflected by the notion of a *substitution* signature.

Definition 2.7 We call a signature $\mathcal{T}$ a **substitution signature** when:

- $\mathcal{T}$ contains two term-formers sub, one with arity $([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T}$, and one with arity $([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$;
- for any other term-former f in $\mathcal{T}$, sorting arities are of the form $(\tau_1, \ldots, \tau_n)\mathbb{T}$, where $\tau_i \in \{\mathbb{T}, [\mathbb{A}]\mathbb{T}\}$, for $1 \leqslant i \leqslant n$.

A term $\mathsf{sub}(t, u)$ represents an **explicit substitution**. We write $u[a \mapsto t]$ as shorthand for $\mathsf{sub}([a]u, t)$. We already used this shorthand in Fig. 1.

Note that a well-sorted term of the form $\mathsf{sub}(t, u)$ must be of the form $\mathsf{sub}(\pi \cdot X, u)$, or $\mathsf{sub}(\mathsf{sub}(t', u'), u)$, or $t'[a \mapsto u]$ (without sugar: $\mathsf{sub}([a]t', u)$). If $\mathsf{sub}(t, u)$ is closed then it has the form $t'[a \mapsto u'][a \mapsto u]$, or $t'[a' \mapsto u'][a \mapsto u]$, or $t'[a \mapsto u]$.

We will only consider substitution signatures in the rest of this paper.

Term-formers f other than sub are intuitively 'the language over which substitution for atoms occurs'. Note that sorting assertions allow arguments of abstraction sort $[\mathbb{A}]\mathbb{T}$; this is useful for modelling languages with abstractors, such as λ, $\forall$, fix, ϵ, and so on. Examples follow:

Example 2.8 (Lambda calculus with explicit substitution) A substitution signature for the lambda calculus is:

$$\mathsf{app} : (\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{lam} : ([\mathbb{A}]\mathbb{T})\mathbb{T} \qquad \mathsf{sub} : ([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}.$$

So intuitively, the two term-formers sub take care of substitution, and app and lam take care of there being a term-language to substitute over (in the presence of some axioms, which are our object of study in this paper).

Example 2.9 (Natural numbers with fixpoints and explicit substitutions) We can express natural numbers with fixpoints using the following substitution signature:

$$\mathsf{zero} : ()\mathbb{T} \qquad \mathsf{succ} : (\mathbb{T})\mathbb{T} \qquad \mathsf{plus} : (\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{fix} : ([\mathbb{A}]\mathbb{T})\mathbb{T} \qquad \mathsf{sub} : ([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}.$$

Remark 2.10 Unknowns of sort $[\mathbb{A}]\mathbb{T}$, and the second sub above of arity $([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$, are convenient but not necessary; see SUB' in Subsect. 6.4.

Remark 2.11 The signature is arbitrary and may be empty (aside from sub), except that Theorem 6.29 (a strong completeness result with respect to a single term model) requires a term-former taking at least two arguments (for example app or plus) to avoid a degenerate case. All other results, including Theorem 6.20 (a notion of completeness with respect to term models in extended signatures which are never degenerate) are valid even in a signature with just sub.

2.2. Permutation, substitution and freshness

Before we can introduce an equational logic on nominal terms, we need to be able to

- permute atoms in terms,
- substitute unknowns with terms, and
- decide whether an atom is *fresh* for a term.

In this subsection we elaborate on these elements.

Definition 2.12 In Definition 2.2 we defined permutations as finitely supported bijection on atoms. 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. $id \circ \pi = \pi$ and $\pi \circ id = \pi$. Importantly, we shall write $(a\ b)$ for the permutation that **swaps** a and b, i.e. the permutation that maps a to b and vice versa, and maps all other c to themselves. Also, we usually abbreviate a moderated unknown $id \cdot X$ to X.

Write $a \in \pi$ when $\pi(a) \neq a$. 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 $a \in \pi$, i.e. $\pi(a) \neq a$. Similarly write $a \notin \pi$, $a \notin t$ and $X \notin t$ for 'does not occur in the syntax'.

Definition 2.13 A **permutation action** $\pi \cdot t$ is defined inductively on t by:

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

Intuitively, π propagates through the structure of t until it reaches an atom or a moderated unknown.

Lemma 2.14 $\pi \cdot (\pi' \cdot t) \equiv (\pi \circ \pi') \cdot t$ and $id \cdot t \equiv t$.

Proof. By an easy induction on the structure of t. □

Definition 2.15 Call a **substitution** σ a finitely supported function from unknowns to terms of the same sort. Here, finite support means that $\sigma(X) \equiv id \cdot X$ for all but finitely many unknowns X.

Write $[t_1/X_1, \ldots, t_n/X_n]$ for the substitution σ, defined by $\sigma(X_i) \equiv t_i$ and $\sigma(Y) \equiv id \cdot Y$, for all $Y \not\equiv X_i$, for $1 \leqslant i \leqslant n$.

Definition 2.16 The **substitution action** $t\sigma$ is inductively defined by:

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

We may call $t\sigma$ an **instance** of t.

Intuitively, σ propagates through the structure of t until it reaches an atom or a moderated unknown. σ 'evaporates' on an atom, and acts on the unknown of a moderated unknown. The moderating permutation then passes into the term substituted in that position. We suggest a reading of $\pi \cdot X$ as 'permute π in whatever X eventually becomes'.

Note that meta-level substitution does not avoid capture; $([a]X)[a/X] \equiv [a]a$. In this sense X is 'meta' and really does represent an unknown *term*. There is an exact and deliberate analogy here with context substitution, which is the substitution used when we write 'let - be a in $\lambda a.$-', to obtain $\lambda a.a$.

The following **commutation** is easy to prove [UPG04, FG07]:

Lemma 2.17 $\pi \cdot t\sigma \equiv (\pi \cdot t)\sigma$.

Proof. By straightforward induction on the structure of t. □

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

Fig. 2. Freshness derivation rules for nominal terms

Definition 2.18 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 and ∇ for (possibly infinite) sets of *primitive* freshnesses and call them **freshness contexts**.

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 $a\#X, b\#X$. Furthermore, write $a \in \Delta$ when a occurs anywhere in Δ, and $X \in \Delta$ when X occurs anywhere in Δ.

Definition 2.19 Define **derivability on freshnesses** in natural deduction style [Hod01] by the rules in Fig. 2. Here:

- a and b *permutatively* range over atoms, i.e. a and b represent any two *distinct* atoms;
- π ranges over permutations
- X ranges over unknowns;
- t and $t_1, \ldots, t_n$ range over nominal terms;
- f ranges over term-formers; there is one copy of the rule for each term-former.

We use similar conventions in the rest of this paper.

The side-condition $\pi \neq id$ of the (**#X**) rule restricts π to non-empty permutations. There is no mathematical reason for this, but there is a nice *computational* one: the algorithm obtained by reading rules bottom-up, must terminate.

Definition 2.20 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**.

We usually write $\emptyset \vdash a\#t$ as $\vdash a\#t$. We will also write $\Delta \vdash S$ for a set of freshnesses S when $\Delta \vdash a\#t$ for each $a\#t \in S$.

For example, in the substitution signature for the lambda calculus (Example 2.8), we have $\vdash a\#\mathsf{lam}([b]b)$ and $a\#X \vdash a\#\mathsf{app}(X, \mathsf{lam}([a]Y))$:

$$\dfrac{\dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{a\#[b]b}\,(\#[]\mathbf{b})}{a\#\mathsf{lam}([b]b)}\,(\#\mathbf{f}) \qquad \dfrac{a\#X \quad \dfrac{\dfrac{}{a\#[a]Y}\,(\#[]\mathbf{a})}{a\#\mathsf{lam}([a]Y)}\,(\#\mathbf{f})}{a\#\mathsf{app}(X, \mathsf{lam}([a]Y))}\,(\#\mathbf{f})$$

The derivation rules are completely syntax-directed, so they also hold in the opposite direction:

Lemma 2.21

1. If $\Delta \vdash a\#X$ then $a\#X \in \Delta$.
2. If $\Delta \vdash a\#\pi \cdot X$ then $\Delta \vdash \pi^{-1}(a)\#X$.
3. If $\Delta \vdash a\#[b]t$ then $\Delta \vdash a\#t$.
4. If $\Delta \vdash a\#\mathsf{f}(t_1, \ldots, t_n)$ then $\Delta \vdash a\#t_i$, for $1 \leqslant i \leqslant n$.

Proof. By an easy induction on the structure of the derivation rules in Fig. 2. □

Sometimes the freshness context Δ may be *strengthened*:

Lemma 2.22 If $c\#Z, \Delta \vdash a\#t$ and $c \notin t$ then $\Delta \vdash a\#t$.

Proof. We transform a derivation of $c\#Z, \Delta \vdash a\#t$ into a derivation of $\Delta \vdash a\#t$:

- If $c\#Z, \Delta \vdash a\#X$ by assumption then $a\#X \in c\#Z, \Delta$. Since we assumed $c \notin a\#X$, we know $c \neq a$. Then $a\#X \in \Delta$, and we conclude $\Delta \vdash a\#X$ by assumption.

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

$$\frac{t=u}{[a]t=[a]u}\,(\mathbf{cong}[]) \qquad \frac{t=u}{\mathsf{f}(t_1,\ldots,t,\ldots,t_n)=\mathsf{f}(t_1,\ldots,u,\ldots,t_n)}\,(\mathbf{congf})$$

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

Fig. 3. Derivation rules for nominal equality

- (#**X**): If $c\#Z, \Delta \vdash a\#\pi\cdot X$ is derived using (#**X**) then $c\#Z, \Delta \vdash \pi^{-1}(a)\#X$. By assumption $c\notin \pi\cdot X$, so $\pi(c)=c$. Since also $a\neq c$, we know $\pi^{-1}(a)\neq c$. By the inductive hypothesis and the simple fact that $c\notin X$ we obtain $\Delta\vdash\pi^{-1}(a)\#X$. We conclude $\Delta\vdash a\#\pi\cdot X$ using (#**X**).
- (#**ab**) and (#[]**a**) carry over directly.
- (#[]**b**) and (#**f**) are straightforward using the inductive hypothesis and the following facts: if $c\notin[b]t$ then $c\notin t$, and if $c\notin\mathsf{f}(t_1,\ldots,t_n)$ then $c\notin t_i$ for $1\leqslant i\leqslant n$.

□

Lemma 2.23 If $\Delta\vdash a\#t$ then $\Delta\vdash\pi(a)\#\pi\cdot t$.

Proof. By an easy induction on freshness derivations (Fig. 2). □

Definition 2.24 Write $\Delta\sigma$ for $\{a\#\sigma(X)\mid a\#X\in\Delta\}$.

Note that $\Delta\sigma$ is not usually a freshness context unless $\sigma(X)$ is an unknown for each $a\#X\in\Delta$.

Theorem 2.25 For any Δ', Δ, σ, if $\Delta\vdash a\#t$ and $\Delta'\vdash\Delta\sigma$ then $\Delta'\vdash a\#t\sigma$.

Proof. The structure of natural deduction derivations is such that the conclusion of one derivation may be 'plugged in' to an assumption in another derivation, if assumption and conclusion are syntactically identical. The structure of all the rules except for (#**X**) is such that if unknowns are instantiated by σ nothing need change. For the case of (#**X**) we use Lemma 2.23. □

The above condition $\Delta'\vdash\Delta\sigma$ ensures that $\Delta\sigma$ is consistent, in the sense that $a\#\sigma(X)$ is derivable from Δ' for each $a\#X\in\Delta$.

2.3. Equality

Definition 2.26 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 the rules in Fig. 3.

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**. We also write $\emptyset\vdash_{\mathsf{CORE}} t=u$ as $\vdash_{\mathsf{CORE}} t=u$.

In (**fr**) square brackets denote *discharge* in the sense of natural deduction (as in implication introduction); Δ denotes the other assumptions of the derivation of $t=u$. 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 sequent style (**fr**) would be

$$\frac{a\#X,\Delta\vdash t=u}{\Delta\vdash t=u}\quad (a\notin t,u).$$

In (**perm**) read $(a\ b)\cdot t$ as 'swap a and b in t'. This single rule expresses α-equivalence. To provide some intuition, here are some examples on closed terms.

Example 2.27 $\vdash_{\mathsf{CORE}} [a]a = [b]b$ is derivable in any signature, and $\vdash_{\mathsf{CORE}} [a][b]\mathsf{app}(a, b) = [b][a]\mathsf{app}(b, a)$ is derivable in the substitution signature of the lambda calculus (Example 2.8):

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

Note that we use $[a]a \equiv (a\ b) \cdot [b]b$ and $[a][b]\mathsf{app}(a, b) \equiv (a\ b) \cdot [b][a]\mathsf{app}(b, a)$ in the conclusions of the derivations.

Example 2.28

- $\vdash_{\mathsf{CORE}} [a]a = [b]b$ is derivable:

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

  (Note here that $[a]a \equiv (a\ b) \cdot [b]b$.)
- In the substitution signature of the lambda calculus (Example 2.8), $\vdash_{\mathsf{CORE}} [a][b]\mathsf{app}(a, b) = [b][a]\mathsf{app}(b, a)$ is derivable.
  Since $[a][b]\mathsf{app}(a, b) \equiv (a\ b) \cdot [b][a]\mathsf{app}(b, a)$ it suffices to show $\vdash_{\mathsf{CORE}} (a\ b) \cdot [b][a]\mathsf{app}(b, a) = [b][a]\mathsf{app}(b, a)$. By (**perm**) this follows from $\Delta \vdash a, b\#[b][a]\mathsf{app}(b, a)$, which follow using (#[]**a**) and (#[]**b**).

The following is an example of α-equivalence on open terms, which shows that we can rename the atom which is substituted for.

Lemma 2.29 $b\#X \vdash_{\mathsf{CORE}} X[a \mapsto T] = ((b\ a) \cdot X)[b \mapsto T]$

Proof. De-sugaring, we derive $\mathsf{sub}([a]X, T) = \mathsf{sub}([b](b\ a) \cdot X, T)$ from $b\#X$ using the rules in Figs. 3 and 2:

$$\dfrac{\dfrac{\dfrac{}{a\#[a]X}(\#[]\mathbf{a}) \qquad \dfrac{b\#X}{b\#[a]X}(\#[]\mathbf{b})}{[b](b\ a) \cdot X = [a]X}(\mathbf{perm})}{[a]X = [b](b\ a) \cdot X}(\mathbf{symm})}{\mathsf{sub}([a]X, T) = \mathsf{sub}([b](b\ a) \cdot X, T)}(\mathbf{congf})$$

□

2.4. Inequality of nominal terms up to CORE

It is important to be able to decide when two nominal terms are *not* equal in CORE because:

- We need this to show that CORE is *consistent* (does not equate *all* terms; Corollary 2.33).
- We need to prove that CORE captures α-equivalence (and *no more than* α-equivalence; see Theorem 3.9).
- We promised to show that equality up to axioms for substitution is decidable. If we are unable to determine equality and inequality of terms *without* any axioms then our project would be doomed from the start.

The rules in Fig. 3 are unsuited to determining inequality. The problem is that (**tran**) is not syntax-directed, in the sense that u appears in the premises and not in the conclusion. This makes derivation-search, and any proof that derivation-search *must fail*, hard. In this subsection we give a syntax-directed version of CORE which is more convenient for proving that derivations cannot exist. It bears an astounding resemblance to the equality on nominal terms introduced in nominal unification [UPG04] (we still need nominal algebra so we can extend CORE with axioms).

$$\frac{}{a \approx_\Delta a}\ (\mathbf{Ax}) \qquad \frac{\Delta \vdash \mathrm{ds}(\pi', \pi)\#X}{\pi \cdot X \approx_\Delta \pi' \cdot X}\ (\mathbf{Ds}) \qquad \frac{t_1 \approx_\Delta u_1 \quad \cdots \quad t_n \approx_\Delta u_n}{\mathsf{f}(t_1, \ldots, t_n) \approx_\Delta \mathsf{f}(u_1, \ldots, u_n)}\ (\mathbf{F})$$

$$\frac{t \approx_\Delta u}{[a]t \approx_\Delta [a]u}\ (\mathbf{Absaa}) \qquad \frac{(b\ a) \cdot t \approx_\Delta u \quad \Delta \vdash b\#t}{[a]t \approx_\Delta [b]u}\ (\mathbf{Absab})$$

Fig. 4. Syntax-directed rules for CORE

The following definition was introduced in [UPG04, Fig. 2]; the proofs are modelled on a method presented in [FG07, p.13]:

Definition 2.30 Let $t \approx_\Delta u$ be an ordered tuple of a term t, a freshness context Δ, and a term u. Let the **derivable equalities of** $t \approx_\Delta u$ be inductively defined by the rules in Fig. 4. Here we write $\mathrm{ds}(\pi, \pi')$ for the set $\{a \mid \pi(a) \neq \pi'(a)\}$, the **difference set** of π and π'. We write $\Delta \vdash \mathrm{ds}(\pi, \pi')\#t$ for a set of proof-obligations $\Delta \vdash a\#t$, one for each $a \in \mathrm{ds}(\pi, \pi')$.

Theorem 2.31 $\Delta \vdash_{\mathsf{CORE}} t = u$ if and only if $t \approx_\Delta u$ is derivable in the sequent system for CORE.

Proof. The left-to-right direction is by induction on the structure of derivations of $\Delta \vdash_{\mathsf{CORE}} t = u$. By the inductive hypothesis it suffices to show:

- Syntax-directed equality $\approx_\Delta$ is an equivalence relation and a congruence. This is [FG07, Theorem 24].
- If $\Delta \vdash a\#t$ and $\Delta \vdash b\#t$ then $(a\ b) \cdot t \approx_\Delta t$. This follows by an induction on the structure of t.
- If $t \approx_{\Delta, a\#X} u$ where $a \notin t, u$ then $t \approx_\Delta u$. By a straightforward induction on the structure of derivations of $t \approx_{\Delta, a\#X} u$. The case of (**Absab**) uses Lemma 2.22 to strengthen the assumption $\Delta, a\#X \vdash c\#t$ to $\Delta \vdash c\#t$.

For the right-to-left direction we work by induction on derivations of $t \approx_\Delta u$. By the inductive hypothesis it suffices to show:

- $\Delta \vdash_{\mathsf{CORE}} a = a$. This is an instance of (**refl**).
- If $\Delta \vdash \mathrm{ds}(\pi, \pi')\#X$ then $\Delta \vdash_{\mathsf{CORE}} \pi \cdot X = \pi' \cdot X$. By induction on the number of elements in $\mathrm{ds}(\pi, \pi')$. If this set is empty then $\pi = \pi'$ and the result follows easily by (**refl**). Now suppose $a \in \mathrm{ds}(\pi, \pi')$. We construct a partial derivation of the proof obligation:

$$\frac{\pi \cdot X = ((\pi(a)\ \pi'(a)) \circ \pi') \cdot X \qquad \dfrac{\pi(a)\#\pi' \cdot X \quad \pi'(a)\#\pi' \cdot X}{((\pi(a)\ \pi'(a)) \circ \pi') \cdot X = \pi' \cdot X}\ (\mathbf{perm})}{\pi \cdot X = \pi' \cdot X}\ (\mathbf{tran})$$

  The remaining equality $\pi \cdot X = ((\pi(a)\ \pi'(a)) \circ \pi') \cdot X$ follows from the assumptions $\mathrm{ds}(\pi, \pi')\#X$ using the inductive hypothesis and the fact that $\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi') = \mathrm{ds}(\pi, \pi') \setminus \{a\}$; the remaining freshnesses $\pi(a)\#\pi' \cdot X$ and $\pi'(a)\#\pi' \cdot X$ follow from the assumptions $\mathrm{ds}(\pi, \pi')\#X$ using Lemma 2.23.
- If $\Delta \vdash_{\mathsf{CORE}} t_i = u_i$ for $1 \leqslant i \leqslant n$, then $\Delta \vdash_{\mathsf{CORE}} \mathsf{f}(t_1, \ldots, t_n) = \mathsf{f}(u_1, \ldots, u_n)$. Using a number of instances of (**tran**) and (**congf**).
- If $\Delta \vdash_{\mathsf{CORE}} t = u$ then $\Delta \vdash_{\mathsf{CORE}} [a]t = [a]u$. This is (**cong**[]).
- If $\Delta \vdash_{\mathsf{CORE}} (b\ a) \cdot t = u$ and $\Delta \vdash b\#t$ then $\Delta \vdash_{\mathsf{CORE}} [a]t = [b]u$. Suppose that Π and Π' are derivations of $\Delta \vdash_{\mathsf{CORE}} (b\ a) \cdot t = u$ and $\Delta \vdash b\#t$ respectively. Then the following is a derivation of $\Delta \vdash_{\mathsf{CORE}} [a]t = [b]u$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Pi'}{b\#t}}{b\#[a]t}\ (\#[]\mathbf{b}) \qquad \dfrac{}{a\#[a]t}\ (\#[]\mathbf{a})}{[b](b\ a) \cdot t = [a]t}\ (\mathbf{perm})}{[a]t = [b](b\ a) \cdot t}\ (\mathbf{symm}) \qquad \dfrac{\Pi}{(b\ a) \cdot t = u}}{[a]t = [b]u}\ (\mathbf{cong}[])\ (\mathbf{tran})$$

□

$$\frac{\nabla\sigma}{\pi\cdot t\sigma=\pi\cdot u\sigma}\quad(\mathbf{ax}_{\nabla\vdash\mathbf{t=u}})$$

Fig. 5. The axiom rule

As corollaries of Theorem 2.31 we obtain syntactic criteria for determining equality in CORE, and consistency of CORE.

Corollary 2.32 (Syntactic criteria for CORE) $\Delta\vdash_{\mathsf{CORE}} t=u$ precisely when one of the following hold:

1. $t\equiv a$ and $u\equiv a$.
2. $t\equiv\pi\cdot X$ and $u\equiv\pi'\cdot X$ and $\Delta\vdash \mathrm{ds}(\pi,\pi')\#X$.
3. $t\equiv[a]t'$ and $u\equiv[a]u'$ and $\Delta\vdash_{\mathsf{CORE}} t'=u'$.
4. $t\equiv[a]t'$ and $u\equiv[b]u'$ and $\Delta\vdash b\#t'$ and $\Delta\vdash_{\mathsf{CORE}}(b\ a)\cdot t'=u'$.
5. $t\equiv \mathsf{f}(t_1,\ldots,t_n)$ and $u\equiv \mathsf{f}(u_1,\ldots,u_n)$ and $\Delta\vdash_{\mathsf{CORE}} t_i=u_i$ for $1\leqslant i\leqslant n$.

Proof. By Theorem 2.31 it suffices to inspect the rules for $t\approx_{\Delta} u$, which are just a rendering of the above criteria in terms of derivation rules. □

Corollary 2.33 (Consistency of CORE) For all Δ there are t and u such that $\Delta\nvdash_{\mathsf{CORE}} t=u$.

Proof. By Corollary 2.32, $\Delta\vdash_{\mathsf{CORE}} a=b$ is never derivable. □

2.5. Axioms and theories

We now come to the raison d'être of nominal algebra: axioms.

Definition 2.34 Call a triple $\nabla\vdash t=u$ where ∇ is a finite freshness context, an **axiom**. We may write $\vdash t=u$ when ∇ **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 and permutatively renaming atoms such that the hypotheses are corresponding instances of freshness conditions of the axiom.

This is formally expressed by the rule $(\mathbf{ax}_{\nabla\vdash\mathbf{t=u}})$ in Fig. 5. Here π ranges over permutations and σ ranges over substitutions. Recall that we write $\nabla\sigma$ for $\{a\#\sigma(X)\mid a\#X\in\nabla\}$.

The reader might have expected that the premise of the axiom rule should be $\pi\cdot\nabla\sigma$ instead of $\nabla\sigma$. It turns out that both versions are correct, because of Lemma 2.23: $\Delta\vdash\nabla\sigma$ iff $\Delta\vdash\pi\cdot\nabla\sigma$ for any Δ.

Definition 2.35 Call a set of axioms T a **theory**. Write $\Delta\vdash_{\mathsf{T}} t=u$ when $t=u$ can be derived from Δ using only axioms from T.

A number of properties on freshnesses also hold for equations of any theory T.

Lemma 2.36 If $\Delta\vdash_{\mathsf{T}} t=u$ then $\Delta\vdash_{\mathsf{T}}\pi\cdot t=\pi\cdot u$.

Proof. By induction on the structure of the rules of Figs. 3 and 5. □

Theorem 2.37 For any T, Δ', Δ, σ, if $\Delta\vdash_{\mathsf{T}} t=u$ and $\Delta'\vdash\Delta\sigma$ then $\Delta'\vdash_{\mathsf{T}} t\sigma=u\sigma$.

Proof. Analogous to the Proof of Theorem 2.25. □

3. Substitution on ground terms

Definition 3.1 Call terms g, h and k **ground terms** when they do not mention unknowns or explicit substitutions. Ground terms are inductively characterised by

$$g\ ::=\ a\ \mid[a]g\mid\ \mathsf{f}(g_1,\ldots,g_n).$$

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

We now consider the *meaning* of explicit substitution on ground terms, in a suitable formal sense. This will make a connection between $[a \mapsto t]$ and actual capture-avoiding substitution on syntax, and we will find that connection useful later.

Definition 3.2 Define a '**free atoms of**' function $fa(g)$ on ground terms inductively as follows:

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

Lemma 3.3 $\vdash a\#g$ if and only if $a \notin fa(g)$.

Proof. By induction on the structure of g. □

Definition 3.4 Let the **size** of a ground term be inductively defined by:

$$|a| = 1 \qquad |[a]g| = |g| + 1 \qquad |\mathsf{f}(g_1, \ldots, g_n)| = |g_1| + \cdots + |g_n| + 1.$$

Definition 3.5 For each finite set of atoms fix some choice of 'fresh' atom not in that finite set. Then define a **ground substitution action** $g[h/a]$ on ground terms of sort $\mathbb{T}$ and $[\mathbb{A}]\mathbb{T}$ inductively on the *size* of g by:

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

where f ranges over all term-formers excluding sub. By 'c fresh' we mean that c is chosen such that $c \notin \{a, b\} \cup fa(g) \cup fa(h)$ according to our arbitrary choice. We will not mention $c \notin \{a, b\}$ anymore, since this is enforced by our permutative convention (see Convention 2.3).

Note that the ground substitution action is well-defined, since $|g[c/b]| = |g|$ in the penultimate case of Definition 3.5, as can be shown by an induction on the size of g. We will often use this fact that capture-avoiding substitution of atoms for atoms preserves size.

Lemma 3.6 states familiar properties of ground terms. It makes the vital connection between 'substitution as we know it' and the nominal technology we bring to bear on it.

Lemma 3.6 For ground terms g, h, k:

1. **Identity.** $\vdash_{\mathsf{CORE}} g[a/a] = g$.
2. **Swapping.** If $\vdash b\#g$ then $\vdash_{\mathsf{CORE}} g[b/a] = (b\ a) \cdot g$.
3. **Garbage collection.** If $\vdash a\#g$ then $\vdash_{\mathsf{CORE}} g[h/a] = g$.
4. **Non-capturing distributivity.** If $\vdash a\#k$ then $\vdash_{\mathsf{CORE}} g[h/a][k/b] = g[k/b][h[k/b]/a]$.

Proof. Part 1 follows from the stronger property that $g[a/a] \equiv g$, which we can show by an easy induction on the structure of g. The other parts all follow by an induction on the size of g. □

We will now show how equality on ground terms in theory CORE coincides with a straightforward definition of α-equivalence on ground terms: syntactic equality extended with a rule to rename bound variables.

Definition 3.7 Define an α-equivalence relation $g =_\alpha h$ inductively by the rules in Fig. 6. Here 'c fresh' means *any* c such that $c \notin \{a, b\} \cup fa(g) \cup fa(h)$.

Lemma 3.8 If $a, b \notin fa(g)$ then $(a\ b) \cdot g =_\alpha g$.

Proof. All instances of a and b in g must occur in the scope of abstractors $[a]$ and $[b]$. Traverse the structure of g bottom-up using the rules of Fig. 6 to rename abstractions by $[a]$ and $[b]$ to fresh atoms. Call the new term g'. Now $(a\ b) \cdot g' \equiv g'$ because $a, b \notin g'$. Equality is symmetric, so we reverse the process to return to g. □

Lemma 3.8 enables us to prove the main result of this section: in the presence of the equalities of CORE, ground terms g and h are provably equal if and only they are α-equivalent.

Theorem 3.9 $\vdash_{\mathsf{CORE}} g = h$ if and only if $g =_\alpha h$.

$$\frac{}{g =_\alpha g} \qquad \frac{g =_\alpha h}{h =_\alpha g} \qquad \frac{g_1 =_\alpha g_2 \quad g_2 =_\alpha g_3}{g_1 =_\alpha g_3}$$

$$\frac{g =_\alpha h}{[a]g =_\alpha [a]h} \qquad \frac{g_1 =_\alpha h_1 \ \cdots \ g_n =_\alpha h_n}{\mathsf{f}(g_1, \ldots, g_n) =_\alpha \mathsf{f}(h_1, \ldots, h_n)}$$

$$\frac{g[c/a] =_\alpha h[c/b]}{[a]g =_\alpha [b]h} \quad (c \text{ fresh})$$

Fig. 6. α-equivalence $=_\alpha$ on ground terms

$$\begin{array}{lll}
(\mathbf{var}{\mapsto}) & \vdash & a[a \mapsto T] = T \\
(\mathbf{b}{\mapsto}) & \vdash & b[a \mapsto T] = b \\
(\mathbf{f}{\mapsto}) & \vdash & \mathsf{f}(X_1, \ldots, X_n)[a \mapsto T] = \mathsf{f}(X_1[a \mapsto T], \ldots, X_n[a \mapsto T]) \qquad (\mathsf{f} \neq \mathsf{sub}) \\
(\mathbf{abs}{\mapsto}) & c\#T \vdash & ([c]U)[a \mapsto T] = [c](U[a \mapsto T])
\end{array}$$

Fig. 7. Axioms of SIMP

Proof. We prove the left-to-right implication by induction on the structure of g, using the syntactic criteria of Corollary 2.32. The cases of $g \equiv a$ and $g \equiv \mathsf{f}(g_1, \ldots, g_n)$ are easy. Now suppose $g \equiv [a]g'$, then there are two possibilities:

1. $h \equiv [a]h'$ and $\vdash_{\mathsf{CORE}} g' = h'$. Then $g' =_\alpha h'$ by the inductive hypothesis, and we conclude $[a]g' =_\alpha [a]h'$ by congruence.
2. $h \equiv [b]h', \vdash b\#g'$ and $\vdash_{\mathsf{CORE}} (b\ a) \cdot g' = h'$. Then $a, b \notin fa([a]g')$ by Lemma 3.3 and some easy calculations, so $[a]g' =_\alpha [b](b\ a) \cdot g'$ by Lemma 3.8 and symmetry. Also $[b](b\ a) \cdot g' =_\alpha [b]h'$ by congruence and the inductive hypothesis. We conclude $[a]g' =_\alpha [b]h'$ by transitivity.

Conversely suppose that $g =_\alpha h$. It suffices to show that equality in CORE can simulate every derivation rule of $=_\alpha$. We treat the only non-trivial case: $[a]g =_\alpha [b]h$ is deduced using the last rule from Fig. 6. Then for c chosen fresh for $fa(g) \cup fa(h)$, we know $g[c/a] =_\alpha h[c/b]$. By the inductive hypothesis $\vdash_{\mathsf{CORE}} g[c/a] = h[c/b]$. Since $\vdash c\#g$ and $\vdash c\#h$ we obtain $\vdash_{\mathsf{CORE}} g[c/a] = (c\ a) \cdot g$ and $\vdash_{\mathsf{CORE}} h[c/b] = (c\ b) \cdot h$ by part 2 of Lemma 3.6. Using (**symm**), (**tran**) and (**cong**[]) we obtain $\vdash_{\mathsf{CORE}} [c](c\ a) \cdot g = [c](c\ b) \cdot h$. By (**perm**) $\vdash_{\mathsf{CORE}} [c](c\ a) \cdot g = [a]g$ and $\vdash_{\mathsf{CORE}} [c](c\ b) \cdot h = [b]h$, since $\vdash c\#g$ and $\vdash c\#h$. Using (**symm**) and (**tran**) we conclude that $\vdash_{\mathsf{CORE}} [a]g = [b]h$. □

4. The theory SIMP; simply substitution

We introduce nominal algebra theory SIMP which is sound but not complete with respect to the ground term model from Sect. 3. This is an important technical step towards SUB because we shall prove properties of SUB by reducing them to properties of SIMP.

Definition 4.1 Let SIMP be the nominal algebra theory with axioms as in Fig. 7.

Here, f ranges over all term-formers except for sub. Recall that T and U are unknowns of $\mathbb{T}$, and that the X_i are unknowns of sort $\mathbb{T}$ or $[\mathbb{A}]\mathbb{T}$ (which one applies depends on the instance of f).

Lemma 4.2 For ground terms g and h, if $\vdash a\#g$ then $\vdash_{\mathsf{SIMP}} g[a \mapsto h] = g$.

Proof. By induction on the size of g. □

Theorem 4.3 If g and h are ground terms then $\vdash_{\mathsf{SIMP}} g[a \mapsto h] = g[h/a]$ is always derivable.

Proof. By induction on the size of g. We only consider the interesting cases:

- $g \equiv [a]g'$. Then $\vdash_{\mathsf{SIMP}} ([a]g')[a \mapsto t] = [a]g'$ by Lemma 4.2 since $\vdash a\#[a]g'$. Since $([a]g')[h/a] \equiv [a]g'$ we are done.

- $g \equiv [b]g'$, $b \in fa(h)$. Then $([b]g')[h/a] \equiv [c](g[c/b][h/a])$ where c is a fresh atom according to our arbitrary choice, i.e. $c \notin fa(g') \cup fa(h)$. Then by Lemma 3.3 also $\vdash c\#g'$ and $\vdash c\#h$. Now $\vdash_{\mathsf{CORE}} [c](c\ b) \cdot g' = [b]g'$ by (**perm**) since $\vdash b\#[b]g'$ and $\vdash c\#[b]g'$. By (**symm**), (**cong**[]) and (**congf**) we obtain

  $$\vdash_{\mathsf{CORE}} ([b]g')[a \mapsto h] = ([c](c\ b) \cdot g')[a \mapsto h].$$

  By axiom (**abs** $\mapsto$) also

  $$\vdash_{\mathsf{SIMP}} ([c](c\ b) \cdot g')[a \mapsto h] = [c](((c\ b) \cdot g')[a \mapsto h])$$

  since $\vdash c\#h$. By the inductive hypothesis and (**cong**[])

  $$\vdash_{\mathsf{SIMP}} [c](((c\ b) \cdot g')[a \mapsto h]) = [c](((c\ b) \cdot g')[h/a]).$$

  Since $\vdash_{\mathsf{CORE}} g'[c/b] = (c\ b) \cdot g'$ by part 2 of Lemma 3.6, we deduce

  $$\vdash_{\mathsf{SIMP}} [c](((c\ b) \cdot g')[h/a]) = [c](g'[c/b][h/a])$$

  using the rules of equality.
  Using (**tran**) we conclude $\vdash_{\mathsf{SIMP}} ([b]g')[a \mapsto h] = [c](g'[c/b][h/a])$.

□

Definition 4.4 Define the **translation** $\_^{\downarrow}$ of closed terms to ground terms inductively on closed terms by:

$$a^{\downarrow} \equiv a \quad ([a]t)^{\downarrow} \equiv [a](t^{\downarrow}) \quad \mathsf{f}(t_1, \ldots, t_n)^{\downarrow} \equiv \mathsf{f}(t_1^{\downarrow}, \ldots, t_n^{\downarrow}) \quad (\mathsf{f} \neq \mathsf{sub}) \quad \mathsf{sub}(t, u)^{\downarrow} \equiv g[u^{\downarrow}/a] \quad ([a]g \equiv t^{\downarrow}).$$

Note that $(t[a \mapsto u])^{\downarrow} \equiv t^{\downarrow}[u^{\downarrow}/a]$ follows from the sub case, since $[a](t^{\downarrow}) \equiv ([a]t)^{\downarrow}$ by the abstraction case.

Lemma 4.5 For any closed term t, if $\vdash a\#t$ then $\vdash a\#t^{\downarrow}$.

Proof. By induction on the structure of t. □

Note that the converse of Lemma 4.5 does not hold. Take for example $t \equiv b[c \mapsto a]$. Then $\vdash a\#(b[c \mapsto a])^{\downarrow}$ since $(b[c \mapsto a])^{\downarrow} \equiv b$. But $\nvdash a\#b[c \mapsto a]$, since $\nvdash a\#a$.

Theorem 4.6 For any closed term t, $\vdash_{\mathsf{SIMP}} t = t^{\downarrow}$.

Proof. By induction on the structure of t. We consider the only non-trivial case $t \equiv \mathsf{sub}(u, v)$. By definition $\mathsf{sub}(u, v)^{\downarrow} \equiv g[v^{\downarrow}/a]$ where $[a]g \equiv u^{\downarrow}$, and by the inductive hypothesis, $\vdash_{\mathsf{SIMP}} u = u^{\downarrow}$ and $\vdash_{\mathsf{SIMP}} v = v^{\downarrow}$. By (**congf**) and (**tran**), we obtain

$$\vdash_{\mathsf{SIMP}} \mathsf{sub}(u, v) = \mathsf{sub}(u^{\downarrow}, v^{\downarrow}).$$

By Theorem 4.3 we know $\vdash_{\mathsf{SIMP}} g[a \mapsto v^{\downarrow}] = g[v^{\downarrow}/a]$. Since $[a]g \equiv u^{\downarrow}$ and $g[v^{\downarrow}/a] \equiv \mathsf{sub}(u, v)^{\downarrow}$, this is syntactically equivalent to

$$\vdash_{\mathsf{SIMP}} \mathsf{sub}(u^{\downarrow}, v^{\downarrow}) = \mathsf{sub}(u, v)^{\downarrow}.$$

By an application of (**tran**) we conclude $\vdash_{\mathsf{SIMP}} \mathsf{sub}(u, v) = \mathsf{sub}(u, v)^{\downarrow}$ as required. □

As a corollary of Theorem 4.6 all standard properties of capture-avoiding substitution on ground terms carry over to closed terms.

Corollary 4.7 For closed terms t, u, v:

1. $\vdash_{\mathsf{SIMP}} t[a \mapsto a] = t$.
2. If $\vdash b\#t$ then $\vdash_{\mathsf{SIMP}} t[a \mapsto b] = (b\ a) \cdot t$.
3. If $\vdash a\#t$ then $\vdash_{\mathsf{SIMP}} t[a \mapsto u] = t$.
4. If $\vdash a\#v$ then $\vdash_{\mathsf{SIMP}} t[a \mapsto u][b \mapsto v] = t[b \mapsto v][a \mapsto u[b \mapsto v]]$.

Proof. Using Theorem 4.6 and Lemma 3.6. □

In this section we have established that in the presence of the equalities of SIMP, t and $t^{\downarrow}$ are provably equal, for closed terms t. Yet many questions related to open terms remain:

$$\frac{\Delta \vdash_{\mathsf{CORE}} t = \pi \cdot l\sigma \quad \Delta \vdash_{\mathsf{CORE}} u = \pi \cdot r\sigma \quad \Delta \vdash \nabla\sigma}{\Delta \vdash_{\mathsf{R}} t \to u} \ (\to\mathbf{rew}) \quad (\nabla \vdash l \to r \in \mathsf{R})$$

$$\frac{\Delta \vdash_{\mathsf{R}} t \to u}{\Delta \vdash_{\mathsf{R}} [a]t \to [a]u} \ (\to[]) \qquad \frac{\Delta \vdash_{\mathsf{R}} t \to u}{\Delta \vdash_{\mathsf{R}} \mathsf{f}(t_1, \ldots, t, \ldots, t_n) \to \mathsf{f}(t_1, \ldots, u, \ldots, t_n)} \ (\to\mathbf{f})$$

Fig. 8. Derivation rules for nominal rewriting

1. Is SIMP conservative over CORE? That is, if two terms that do not mention sub are equated in SIMP, are they necessarily equated in CORE?
   (Answer: Yes.)
2. Is equality in SIMP decidable?
   (Answer: Yes.)
3. Is SIMP *sound* for the ground term model? That is, if two terms are equated in SIMP, are all their closed instances α-equivalent, if we interpret every occurrence of sub by capture-avoiding substitution?
   (Answer: Yes.)
4. Is SIMP complete? That is, if all closed instances of two terms are α-equivalent where we interpret sub by capture-avoiding substitution, are the terms themselves provably equal in SIMP?
   (Answer: No, but a more powerful theory SUB *exists which* is *complete, and which retains properties 1 to 3 of this list.)*

The next two sections provide answers to these questions together with detailed proofs.

5. Substitution on open terms using SIMP

In this section we recall a notion of rewriting called *nominal rewriting* [FG07], which is tailored to nominal terms. We will use nominal rewriting to prove properties on open terms of theory SIMP.

5.1. Nominal rewriting

Definition 5.1 A **nominal rewrite rule** $\nabla \vdash l \to r$ is a tuple of a freshness context ∇ and terms l and r of the same sort such that ∇ and r mention only unknowns appearing in l.

A **nominal rewrite system** R is a set of nominal rewrite rules. It determines a set of **nominal rewrites** $\Delta \vdash_{\mathsf{R}} t \to u$ inductively by the rules in Fig. 8.

Write $\Delta \not\vdash_{\mathsf{R}} t \to u$ when the rewrite $\Delta \vdash_{\mathsf{R}} t \to u$ is *not* derivable.

In the $(\to\mathbf{f})$ rule of Fig. 8, f ranges over *all* term-formers of the signature of R (whatever that signature is). The $(\to\mathbf{rew})$ rule is closely related to the $(\mathbf{ax}_{\nabla\vdash\mathbf{t}=\mathbf{u}})$ rule from Fig. 5. We discuss the aspects of this rule in more detail:

- The permutation π allows us to permutatively rename atoms. Consider for instance the substitution signature of the lambda calculus (Example 2.8) extended with a term-former const : $()\mathbb{T}$, and write const() as const. Then the rewrite rule $\mathsf{app}(a, b) \to \mathsf{const}$ generates, for example, rewrites $\mathsf{app}(b, a) \to \mathsf{const}$ and $\mathsf{app}(a, c) \to \mathsf{const}$ but *not* $\mathsf{app}(a, a) \to \mathsf{const}$ because no π can identify a with b.
- The substitution σ gives unknowns X in rules the character of 'unknown terms', and is subject to the freshness conditions formulated by $\Delta \vdash \nabla\sigma$. Note that we could also have used $\Delta \vdash \pi \cdot \nabla\sigma$, which is equivalent to $\Delta \vdash \nabla\sigma$ by Lemma 2.23.
- The use of equality in CORE gives abstractions $[a]$- the character of *real* abstractions. For example the rule $\mathsf{lam}([a]\mathsf{lam}([b]b)) \to \mathsf{const}$ generates rewrites

  $$\mathsf{lam}([a]\mathsf{lam}([a]a)) \to \mathsf{const} \qquad \mathsf{lam}([b]\mathsf{lam}([b]b)) \to \mathsf{const} \qquad \mathsf{lam}([b]\mathsf{lam}([c]c)) \to \mathsf{const}.$$

The following result is easy to prove from the definition of rewriting:

Lemma 5.2 If $\Delta \vdash_{\mathsf{R}} t \to u$ and $\Delta' \vdash \Delta\sigma$, then $\Delta' \vdash_{\mathsf{R}} t\sigma \to u\sigma$.

$$\frac{\Delta \vdash_{\mathsf{R}} t \to u}{\Delta \vdash_{\mathsf{R}} t \to^* u}\ (\to^*\!\to) \qquad \frac{\Delta \vdash_{\mathsf{CORE}} t = u}{\Delta \vdash_{\mathsf{R}} t \to^* u}\ (\to^*\mathbf{refl}) \qquad \frac{\Delta \vdash_{\mathsf{R}} t \to^* u \quad \Delta \vdash_{\mathsf{R}} u \to^* v}{\Delta \vdash_{\mathsf{R}} t \to^* v}\ (\to^*\mathbf{tran})$$

Fig. 9. Derivation rules for the transitive reflexive closure of R

Proof. By induction on the derivation of $\Delta \vdash_{\mathsf{R}} t \to u$ we construct a derivation of $\Delta' \vdash_{\mathsf{R}} t\sigma \to u\sigma$. □

Definition 5.3 Write $\Delta \vdash_{\mathsf{R}} t \to^* u$ for the **transitive reflexive closure** of R defined inductively by the rules in Fig. 9. Write $\Delta \nvdash_{\mathsf{R}} t \to^* u$ when $\Delta \vdash_{\mathsf{R}} t \to^* u$ is *not* derivable.

If a term does not have any rewrites, the transitive reflexive closure of R is precisely CORE-equality:

Lemma 5.4 If there is no t' such that $\Delta \vdash_{\mathsf{R}} t \to t'$ then

$$\Delta \vdash_{\mathsf{R}} t \to^* u \quad \text{iff} \quad \Delta \vdash_{\mathsf{CORE}} t = u.$$

Proof. Suppose there is no t' such that $\Delta \vdash_{\mathsf{R}} t \to t'$. If $\Delta \vdash_{\mathsf{CORE}} t = u$ then by $(\to^*\mathbf{refl})$ also $\Delta \vdash_{\mathsf{R}} t \to^* u$. Conversely if $\Delta \vdash_{\mathsf{R}} t \to^* u$ then this rewrite must be derived using $(\to^*\mathbf{refl})$ by an inductive argument. □

Definition 5.5 Call a nominal rewrite system R **confluent** when if $\Delta \vdash_{\mathsf{R}} t \to^* u$ and $\Delta \vdash_{\mathsf{R}} t \to^* v$ then there is some w such that $\Delta \vdash_{\mathsf{R}} u \to^* w$ and $\Delta \vdash_{\mathsf{R}} v \to^* w$.

Definition 5.6 Call a nominal rewrite system R **strongly normalising** when there is no infinite sequence $t_1, t_2, t_3, \ldots$ such that $\Delta \vdash_{\mathsf{R}} t_i \to t_{i+1}$ for all $1 \leqslant i$.

Lemma 5.7 If a derivation exists of $\Delta \vdash_{\mathsf{R}} t \to u$ then that derivation mentions $(\to\mathbf{rew})$ exactly once.

Proof. By the structure of the derivation rules from Fig. 8. □

Definition 5.8 If a rewrite $\Delta \vdash_{\mathsf{R}} t \to u$ occurs, it must occur *at* some subterm t' of t (the subterm t' where we actually use $(\to\mathbf{rew})$ and prove $\Delta \vdash \nabla\sigma$ and $\Delta \vdash_{\mathsf{CORE}} t' = l\sigma$). We say that the rewrite **occurs at** t' **inside** t. If the derivation tree has just an instance of $(\to\mathbf{rew})$, then we may say the rewrite **occurs at top level**.

Call a pair of nominal rewrites $\Delta \vdash_{\mathsf{R}} t \to u$ and $\Delta \vdash_{\mathsf{R}} t \to v$ a **critical pair** when at least one of the rewrites occurs at top level. If any of the two rewrites occurs at a moderated unknown inside t, call the critical pair **trivial**. Otherwise call it **nontrivial**.

Definition 5.9 Call a rewrite rule $\nabla \vdash l \to r$ **uniform** when $\Delta \vdash a\#l$ and $\Delta \vdash \nabla$ imply $\Delta \vdash a\#r$ for all Δ and a. A rewrite rule $\nabla \vdash l \to r$ is **left-linear** when l does not mention the same unknown more than once. A uniform nominal rewrite system with only left-linear rules and no non-trivial critical pairs is **orthogonal**.

Theorem 5.10 An orthogonal uniform nominal rewrite system is confluent.

Proof. See [FG07, Theorem 65]. □

Remark 5.11 Nominal rewriting is the default notion of rewriting for nominal terms, like higher-order rewriting (such as CRS's [KvOvR93] and HRS's [MN98]) is for higher-order terms. The major differences between the two frameworks can be summarised as follows:

- The default notion of instantiation of meta-variables is capturing for nominal rewriting whereas it is capture-avoiding for higher-order rewriting.
- Nominal rewriting uses unification up to α whereas higher-order rewriting uses unification up $\alpha\beta(\eta)$.

Detailed comparisons are elsewhere [FG07].

5.2. SIMPr: explicit substitution rewritten

Definition 5.12 Let SIMPr be the nominal rewrite system defined in Fig 10.

A basic correctness result is this:

Lemma 5.13 If $\Delta \vdash_{\mathsf{SIMPr}} t \to^* u$ then $\Delta \vdash_{\mathsf{SIMP}} t = u$ is derivable.

$$
\begin{array}{llrcll}
(\mathbf{Rvar}) & & \vdash & a[a \mapsto T] & \to & T \\
(\mathbf{Rb}) & & \vdash & b[a \mapsto T] & \to & b \\
(\mathbf{Rf}) & & \vdash & \mathsf{f}(X_1,\ldots,X_n)[a \mapsto T] & \to & \mathsf{f}(X_1[a \mapsto T],\ldots,X_n[a \mapsto T]) \quad (\mathsf{f} \neq \mathsf{sub}) \\
(\mathbf{Rabs}) & c\#T & \vdash & ([c]U)[a \mapsto T] & \to & [c](U[a \mapsto T])
\end{array}
$$

Fig. 10. Substitution as a rewrite system SIMPr

Proof. By induction on the structure of derivations of $\Delta \vdash_{\mathsf{SIMPr}} t \to^* u$. The case of $(\to^*\to)$ uses the precise correspondence between the rewrites in Fig. 10 and the axioms in Fig. 7. ($\to^*$**refl**) uses (**refl**), and ($\to^*$**tran**) uses (**tran**). □

Remark 5.14 The (**Rb**) rule cannot be represented by a rule in a higher-order rewrite system, since in such a system object-variables only exist when they are bound by a meta-level abstraction. That is, the rule $\mathsf{sub}(\lambda\, x.y,\, T) \to y$ does *not* represent (**Rb**) since y represents a meta-variable instead of an object-variable. It represents the more general rule $a\#X \vdash X[a \mapsto T] \to X$.

Lemma 5.15 All rewrite rules of SIMPr are uniform.

Proof. For each rule, use appropriate instances of Lemma 2.21. For example, for the (**Rabs**) rule, we need to show that $\Delta \vdash a'\#([c]U)[a \mapsto T]$ and $\Delta \vdash c\#T$ imply $\Delta \vdash a'\#[c](U[a \mapsto T])$ for any a' and Δ. From the first assumption we know, by Lemma 2.21:

- if $a' = a$ or $a' = c$ then $a'\#T \in \Delta$;
- if $a' \neq a$ and $a' \neq c$ then $a'\#T \in \Delta$ and $a'\#U \in \Delta$.

Using these assumptions it is easy to show $\Delta \vdash a'\#[c](U[a \mapsto T])$ by case distinction on a'. □

Theorem 5.16 (Confluence of SIMPr) SIMPr is confluent.

Proof. By Lemma 5.15 all rewrite rules of SIMPr are uniform. Also, SIMPr has no non-trivial critical pairs and every rule is left-linear (each unknown is mentioned on the left at most once). Then SIMPr is orthogonal and uniform by Definition 5.9. We conclude that it is confluent by Theorem 5.10. □

Confluence of SIMPr has a number of nice corollaries, which comprise the remainder of this subsection.

Definition 5.17 Call a term t a **SIMPr-normal form with respect to** Δ when there is no u such that $\Delta \vdash_{\mathsf{SIMPr}} t \to u$.

Theorem 5.18 Suppose that t and u are SIMPr-normal forms with respect to Δ. Then

$$\Delta \vdash_{\mathsf{SIMP}} t = u \qquad \text{if and only if} \qquad \Delta \vdash_{\mathsf{CORE}} t = u.$$

Proof. The (empty) set of axioms of CORE is a subset of the axioms of SIMP so a derivation in CORE is also a derivation in SIMP and it follows that if $\Delta \vdash_{\mathsf{CORE}} t = u$ then $\Delta \vdash_{\mathsf{SIMP}} t = u$.

Conversely suppose $\Delta \vdash_{\mathsf{SIMP}} t = u$. By Theorem 5.16 there is some term v such that $\Delta \vdash_{\mathsf{SIMPr}} t \to^* v$ and $\Delta \vdash_{\mathsf{SIMPr}} u \to^* v$. By assumption there can be no t' and u' such that $\Delta \vdash_{\mathsf{SIMPr}} t \to t'$ and $\Delta \vdash_{\mathsf{SIMPr}} u \to u'$. Then $\Delta \vdash_{\mathsf{CORE}} t = v$ and $\Delta \vdash_{\mathsf{CORE}} u = v$ by Lemma 5.4, and we conclude $\Delta \vdash_{\mathsf{CORE}} t = u$ by (**symm**) and (**tran**). □

Corollary 5.19 (Conservativity of SIMP over CORE) Suppose that t and u do *not* mention the term-former sub. Then

$$\Delta \vdash_{\mathsf{SIMP}} t = u \qquad \text{if and only if} \qquad \Delta \vdash_{\mathsf{CORE}} t = u.$$

Proof. This is an instance of Theorem 5.18, since if t and u do not mention sub, then t and u are SIMPr-normal forms with respect to any Δ. □

Recall the notation $t^{\downarrow}$ for closed terms t from Definition 4.4.

Corollary 5.20 For closed terms t and u,

$$\vdash_{\mathsf{SIMP}} t = u \quad \text{if and only if} \quad \vdash_{\mathsf{CORE}} t^{\downarrow} = u^{\downarrow}.$$

Proof. $\vdash_{\mathsf{SIMP}} t = u$ is equivalent to $\vdash_{\mathsf{SIMP}} t^{\downarrow} = u^{\downarrow}$ by Theorem 4.6, (**symm**) and (**tran**). By Corollary 5.19 this is equivalent to $\vdash_{\mathsf{CORE}} t^{\downarrow} = u^{\downarrow}$. □

Corollary 5.21 (Consistency of SIMP) For all Δ there are t and u such that $\Delta \not\vdash_{\mathsf{SIMP}} t = u$.

Proof. A corollary of Corollaries 5.19 and 2.33. □

For the final theorem of this subsection we need a few definitions.

Definition 5.22 Call a substitution σ **closing for an unknown** X when $\sigma(X)$ is a closed term. Call σ **closing for a term** t or **closing for a freshness context** Δ when $\sigma(X)$ is closed for every $X \in t$ or $X \in \Delta$.

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

Theorem 5.23 (Soundness of SIMP) SIMP is sound for the ground term model. That is:
If $\Delta \vdash_{\mathsf{SIMP}} t = u$ then $t\sigma^{\downarrow} =_{\alpha} u\sigma^{\downarrow}$ for all Δ-consistent closing substitutions σ (for Δ, t and u).

Proof. $\vdash_{\mathsf{SIMP}} t\sigma = u\sigma$ by Theorem 2.37 and our assumption that $\vdash \Delta\sigma$. We obtain $\vdash_{\mathsf{CORE}} t\sigma^{\downarrow} = u\sigma^{\downarrow}$ by Corollary 5.20, since $t\sigma$ and $u\sigma$ are closed. We conclude $t\sigma^{\downarrow} =_{\alpha} u\sigma^{\downarrow}$ by Theorem 3.9, since $t\sigma^{\downarrow}$ and $u\sigma^{\downarrow}$ are ground. □

5.3. Failure of completeness for SIMP

SIMP is not a *complete* theory of substitution on ground terms. Unknowns in nominal algebra represent unknown terms, and here are some examples of statements that are true for every closing σ (for the unknowns in the statements) but which are *not* derivable in SIMP:

Theorem 5.24 (Incompleteness of SIMP)

1. $\not\vdash_{\mathsf{SIMP}} X[a \mapsto a] = X$.
2. $b\#X \not\vdash_{\mathsf{SIMP}} X[a \mapsto b] = (b\ a) \cdot X$.
3. $a\#X \not\vdash_{\mathsf{SIMP}} X[a \mapsto T] = X$.
4. $a\#U \not\vdash_{\mathsf{SIMP}} X[a \mapsto T][b \mapsto U] = X[b \mapsto U][a \mapsto T[b \mapsto U]]$.

Proof. For part 1, we can easily check using the syntactic criteria of Corollary 2.32 that $\vdash_{\mathsf{CORE}} X[a \mapsto a] = X$ is not derivable. Now since $X[a \mapsto a]$ and X have no SIMPr rewrites, we conclude $\not\vdash_{\mathsf{SIMP}} X[a \mapsto a] = X$ by Theorem 5.18. The proofs of the other parts are similar. □

The fact that above assertions are derivable for closing substitutions follows by Corollary 4.7.

Nominal algebra has a model theory and satisfies soundness and completeness [GM06b, GM07]. So SIMP has 'nonstandard' models which contain 'pathological elements' for which the above equalities do not hold. SIMP defines substitution, but it does not express all of the properties which emerge from that definition. To do that we must strengthen the theory. Before that however, it is useful to consider the computational content of SIMP.

5.4. Strong normalisation and decidability

We expect the part of a λ-calculus that handles substitution to be terminating [BR95, Les94]—so is SIMP terminating? After all our syntax contains sub as an explicit term-former, and it contains unknowns so that sub cannot always be completely eliminated. Also, we consider single substitutions and not *simultaneous* substitutions, so that the order of substitutions matters. Perhaps that all makes enough of a difference that reductions could cycle or diverge in some way?

In fact reductions in SIMPr are extremely well-behaved. We show strong normalisation by a standard method: define a well-founded measure $|t|_m$ on terms and show that rewrites reduce it.

Definition 5.25 For a term t let $|t|_m$ be inductively defined by:

$$|a|_m = 1 \qquad |\pi \cdot X|_m = 1 \qquad |[a]t|_m = |t|_m + 1$$
$$|\mathsf{f}(t_1, \ldots, t_n)|_m = |t_1|_m + \cdots + |t_n|_m + n + 1 \qquad (\mathsf{f} \neq \mathsf{sub})$$
$$|\mathsf{sub}(t, u)|_m = |t|_m * (|u|_m + 1)$$

For terms $u[a \mapsto t]$ we have

$$|u[a \mapsto t]|_m \equiv |\mathsf{sub}([a]u, t)|_m = |[a]u|_m * (|t|_m + 1) = (|u|_m + 1) * (|t|_m + 1).$$

Lemma 5.26 For all terms t and permutations π:

1. $|t|_m > 0$.
2. $|t|_m = |\pi \cdot t|_m$. As a corollary, if $\Delta \vdash_{\mathsf{CORE}} t = u$ then $|t|_m = |u|_m$.

Proof. Both parts can be proven by a simple induction on the structure of t. The corollary follows by Corollary 2.32 which states that t and u are renamed versions of each other by means of permutations. □

Lemma 5.27 For terms $t, u, t_1, \ldots, t_n$ and term-formers $\mathsf{f} \neq \mathsf{sub}$:

1. $|a[a \mapsto t]|_m > |t|_m$.
2. $|b[a \mapsto t]|_m > |b|_m$.
3. $|\mathsf{f}(t_1, \ldots, t_n)[a \mapsto t]|_m > |\mathsf{f}(t_1[a \mapsto t], \ldots, t_n[a \mapsto t])|_m \quad (\mathsf{f} \neq \mathsf{sub})$.
4. $|([c]u)[a \mapsto t]|_m > |[c](u[a \mapsto t])|_m$.

Proof. By straightforward calculations using the measure on terms. The last part uses the fact $|t|_m > 0$ (Lemma 5.26 above). □

Lemma 5.28 For terms $t, u, t_1, \ldots, t_n$ and term-formers f:

1. If $|t|_m > |u|_m$ then $|[a]t|_m > |[a]u|_m$.
2. If $|t|_m > |u|_m$ then $|\mathsf{f}(t_1, \ldots, t, \ldots, t_n)|_m > |\mathsf{f}(t_1, \ldots, u, \ldots, t_n)|_m$.

Proof. Again by straightforward calculations. The last part uses the fact that $|t|_m > 0$ when $\mathsf{f} = \mathsf{sub}$. □

Theorem 5.29 (Strong normalisation of SIMPr) SIMPr is strongly normalising.

Proof. It suffices to show that if $\Delta \vdash_{\mathsf{SIMPr}} t \to u$ then $|t|_m > |u|_m$. We proceed by induction on the rules from Fig. 8, using the rewrite rules from Fig. 10.

Suppose $\Delta \vdash_{\mathsf{SIMPr}} t \to u$ is derived using $(\to [])$ or $(\to \mathbf{f})$; then the result follows by Lemma 5.28 and the inductive hypothesis.

Suppose $\Delta \vdash_{\mathsf{SIMPr}} t \to u$ is derived using $(\to \mathbf{rew})$; then for each SIMPr rewrite rule of the form $\nabla \vdash l \to r$ from Fig. 10 we have, for some π and σ,

$$\Delta \vdash_{\mathsf{CORE}} t = \pi \cdot l\sigma \qquad \text{and} \qquad \Delta \vdash_{\mathsf{CORE}} u = \pi \cdot r\sigma \qquad \text{and} \qquad \Delta \vdash \pi \cdot \nabla\sigma.$$

By part 2 of Lemma 5.26, $|t|_m = |\pi \cdot l\sigma|_m = |l\sigma|_m$ and $|u|_m = |\pi \cdot r\sigma|_m = |r\sigma|_m$. So in order to show that $|t|_m > |u|_m$, it suffices to show $|l\sigma|_m > |r\sigma|_m$. For each SIMPr rule this is an instance of Lemma 5.27. □

We have already given an algorithm to compute normal forms on closed terms in Definition 4.4:

Lemma 5.30 If t is closed then $t^{\intercal}$ is a SIMPr-normal form of t with respect to any Δ.

Proof. By an induction on the structure of t. □

Theorem 5.31 (Unique normal forms for SIMPr) SIMPr-normal forms are unique up to equality in CORE.

Proof. Let Δ be a freshness context and t be a term. By Theorem 5.29 t has a normal form, say u, with respect to Δ. Now suppose v is also a normal form of t with respect to Δ. Then $\Delta \vdash_{\mathsf{SIMPr}} t \to^* u$ and $\Delta \vdash_{\mathsf{SIMPr}} t \to^* v$. By confluence (Theorem 5.16) there exists a term w such that $\Delta \vdash_{\mathsf{SIMPr}} u \to^* w$ and $\Delta \vdash_{\mathsf{SIMPr}} v \to^* w$. Since u and v do not have any rewrites $\Delta \vdash_{\mathsf{CORE}} u = w$ and $\Delta \vdash_{\mathsf{CORE}} v = w$ by Lemma 5.4. Using (**symm**) and (**tran**) we conclude $\Delta \vdash_{\mathsf{CORE}} u = v$. □

As a corollary we obtain decidability of theory SIMP:

Theorem 5.32 (Decidability of SIMP) It is decidable whether or not $\Delta \vdash_{\mathsf{SIMP}} t = u$.

Proof. Given Δ, t and u the following procedure decides whether $\Delta \vdash_{\mathsf{SIMP}} t = u$ is derivable:

1. Rewrite t and u to SIMPr-normal forms t' and u' with respect to Δ; by Theorem 5.29 these exist and by Theorem 5.31 they are unique up to equality in CORE.
2. Check whether $\Delta \vdash_{\mathsf{CORE}} t' = u'$ using the syntactic criteria of Corollary 2.32.
3. If $\Delta \vdash_{\mathsf{CORE}} t' = u'$ then return 'true', otherwise return 'false'. □

6. Substitution on open terms using SUB

In this section we focus on theory SUB from Fig. 1. We show that it is decidable and complete with respect to the ground term model by relating it to theory SIMP.

Definition 6.1 The theory of substitution SUB is the equality relation obtained by the rules of nominal algebra with the axioms in Fig. 1. Here f ranges over all term-formers, including sub.

As an example, we show that our standard properties of capture-avoiding substitution are all derivable in SUB.

Lemma 6.2 The following judgements are derivable in SUB:

1. $\vdash_{\mathsf{SUB}} X[a \mapsto a] = X$.
2. $b\#X \vdash_{\mathsf{SUB}} X[a \mapsto b] = (b\ a) \cdot X$.
3. $a\#X \vdash_{\mathsf{SUB}} X[a \mapsto T] = X$.
4. $a\#U \vdash_{\mathsf{SUB}} X[a \mapsto T][b \mapsto U] = X[b \mapsto U][a \mapsto T[b \mapsto U]]$.

Proof. Parts 2 and 3 are direct from (**ren**$\mapsto$) and (#$\mapsto$). For the other two parts we give nominal algebra derivations in the theory SUB.

Derivation for part 1:

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

In the above derivation of $X[a \mapsto a] = X$, the superscript number one ^{1} is an annotation associating the instance of the rule (**fr**) with the assumption it discharges in the derivation. This is standard natural deduction notation.

For the derivation of part 4, we write $\mathfrak{s}$ for $[b \mapsto U]$ and we use the unsugared syntax for the other substitutions.

$$\dfrac{\dfrac{}{\mathsf{sub}([a]X,\,T)\mathfrak{s} = \mathsf{sub}(([a]X)\mathfrak{s},\,T\mathfrak{s})}\,(\mathbf{ax}_{\mathsf{f}\mapsto}) \qquad \dfrac{\dfrac{a\#U}{([a]X)\mathfrak{s} = [a](X\mathfrak{s})}\,(\mathbf{ax}_{\mathbf{abs}\mapsto})}{\mathsf{sub}(([a]X)\mathfrak{s},\,T\mathfrak{s}) = \mathsf{sub}([a](X\mathfrak{s}),\,T\mathfrak{s})}\,(\mathbf{congf})}{\mathsf{sub}([a]X,\,T)\mathfrak{s} = \mathsf{sub}([a](X\mathfrak{s}),\,T\mathfrak{s})}\,(\mathbf{tran})$$

□

We conjecture that it is not possible to derive $\vdash_{\mathsf{SUB}} X[a \mapsto a] = X$ without (**fr**).

6.1. Soundness and the relation to SIMP

SUB can do everything that SIMP can:

Lemma 6.3 If $\Delta \vdash_{\mathsf{SIMP}} t = u$ then $\Delta \vdash_{\mathsf{SUB}} t = u$.

Proof. All the axioms of SIMP are also axioms of SUB, except for (**b**$\mapsto$). However an instance of (**b**$\mapsto$) is *also* an instance of (#$\mapsto$). Therefore any SIMP derivation is also a SUB derivation. The result follows. □

We now show that SIMP can do everything that SUB can—provided that the terms are closed. We need a technical lemma:

Lemma 6.4 If t, u and v are closed, then:

1. $\vdash_{\mathsf{SIMP}} \mathsf{sub}(t,u)[a \mapsto v] = \mathsf{sub}(t[a \mapsto v], u[a \mapsto v])$.
2. If $\vdash a\#t$ then $\vdash_{\mathsf{SIMP}} [a]\mathsf{sub}(t,a) = t$.

Proof. We consider the parts in turn:

1. Since $\vdash_{\mathsf{SIMP}} t = t^{\downarrow}$ by Theorem 4.6, the proof obligation is equivalent to

   $$\vdash_{\mathsf{SIMP}} \mathsf{sub}(t^{\downarrow},u)[a \mapsto v] = \mathsf{sub}(t^{\downarrow}[a \mapsto v], u[a \mapsto v]).$$

   Now $t^{\downarrow}$ is of the form $[a]g$ or $[b]g$, where g is a ground term. We only consider the case of b, the case of a is completely analogous. Take c fresh such that $\vdash c\#g$ and $\vdash c\#v$. Then $\vdash_{\mathsf{CORE}} [b]g = [c](c\ b)\cdot g$ using (**perm**), since $\vdash b\#[b]g$ and $\vdash c\#[b]g$. Then using the rules for equality, the proof obligation is equivalent to

   $$\vdash_{\mathsf{SIMP}} \mathsf{sub}([c](c\ b)\cdot g,u)[a \mapsto v] = \mathsf{sub}(([c](c\ b)\cdot g)[a \mapsto v], u[a \mapsto v]).$$

   Since $\vdash c\#v$, $\vdash_{\mathsf{SIMP}} ([c](c\ b)\cdot g)[a \mapsto v] = [c](((c\ b)\cdot g)[a \mapsto v])$ by axiom (**abs**$\mapsto$), so the proof obligation is equivalent to $\vdash_{\mathsf{SIMP}} \mathsf{sub}([c](c\ b)\cdot g,u)[a \mapsto v] = \mathsf{sub}([c](((c\ b)\cdot g)[a \mapsto v]), u[a \mapsto v])$. Or, using sugar:

   $$\vdash_{\mathsf{SIMP}} ((c\ b)\cdot g)[c \mapsto u][a \mapsto v] = ((c\ b)\cdot g)[a \mapsto v][c \mapsto u[a \mapsto v]].$$

   And this is just an instance of part 4 of Corollary 4.7, since $\vdash c\#v$.
2. We must show $\vdash_{\mathsf{SIMP}} [a]\mathsf{sub}(t,a) = t$. By Theorem 4.6 this is equivalent to $\vdash_{\mathsf{SIMP}} [a]\mathsf{sub}(t^{\downarrow},a) = t^{\downarrow}$. We proceed by case distinction on the structure of $t^{\downarrow}$:

   - $t^{\downarrow} \equiv [a]g$: Then $\vdash_{\mathsf{SIMP}} [a](g[a \mapsto a]) = [a]g$ follows by (**cong**[]) and part 1 of Corollary 4.7.
   - $t^{\downarrow} \equiv [b]g$: Then $\vdash_{\mathsf{SIMP}} [a](g[b \mapsto a]) = [b]g$ follows from

     $$\vdash_{\mathsf{SIMP}} [a](g[b \mapsto a]) = [a](a\ b)\cdot g \quad \text{and} \quad \vdash_{\mathsf{SIMP}} [a](a\ b)\cdot g = [b]g.$$

     by (**tran**). By (**cong**[]), $\vdash_{\mathsf{SIMP}} [a](g[b \mapsto a]) = [a](b\ a)\cdot g$ follows from $\vdash_{\mathsf{SIMP}} g[b \mapsto a] = (a\ b)\cdot g$. By Corollary 4.7, this is when $\vdash a\#g$. By (**perm**) and the rules for freshness also $\vdash_{\mathsf{SIMP}} [a](b\ a)\cdot g = [b]g$ when $\vdash a\#g$. By Lemma 2.21, this is when $\vdash a\#[b]g$. Since $[b]g \equiv t^{\downarrow}$, this follows from the assumption $\vdash a\#t$ by Lemma 4.5. □

On closed terms, SUB is equivalent to SIMP:

Lemma 6.5 For closed terms t, u:

$$\vdash_{\mathsf{SUB}} t = u \quad \text{if and only if} \quad \vdash_{\mathsf{SIMP}} t = u.$$

Proof. The right-to-left part follows by Lemma 6.3.

For the left-to-right part, we must show that SIMP can simulate the axioms of SUB on closed terms. Axioms (**var**$\mapsto$), (**abs**$\mapsto$) and (**f**$\mapsto$), for $\mathsf{f} \neq \mathsf{sub}$, are also present in SIMP. Each instance of axiom (**f**$\mapsto$), where $\mathsf{f} = \mathsf{sub}$, is an instance of part 1 of Lemma 6.4. Instances of axioms (**ren**$\mapsto$) and (#$\mapsto$) are instances of parts 2 and 3 of Corollary 4.7. Finally, each instance of ($\eta\mapsto$) is an instance of part 2 of Lemma 6.4. □

To recap, SIMP is sound for a ground term model by Theorem 4.3, equality in SIMP is decidable by Theorem 5.32, but not complete by the Theorem 5.24.

We now build the tools to prove that SUB is sound, decidable—and also complete for the ground term model. For soundness we have already done all the hard work:

Theorem 6.6 (Soundness of SUB) SUB is sound for the ground term model. That is:
If $\Delta \vdash_{\mathsf{SUB}} t = u$ then $t\sigma^{\downarrow} =_{\alpha} u\sigma^{\downarrow}$ for all Δ-consistent closing substitutions σ (for Δ, t and u).

Proof. Since $\Delta \vdash_{\mathsf{SUB}} t = u$ and σ is Δ-consistent, we obtain $\vdash_{\mathsf{SUB}} t\sigma = u\sigma$ by Theorem 2.37. $t\sigma$ and $u\sigma$ are closed so by Lemma 6.5 $\vdash_{\mathsf{SIMP}} t\sigma = u\sigma$. By Corollary 5.20 we obtain $\vdash_{\mathsf{CORE}} t\sigma^{\downarrow} = u\sigma^{\downarrow}$. We conclude $t\sigma^{\downarrow} =_{\alpha} u\sigma^{\downarrow}$ by Theorem 3.9, since $t\sigma^{\downarrow}$ and $u\sigma^{\downarrow}$ are ground. □

6.2. Decidability

In this subsection we will establish that equality in SUB is decidable. We do this by transforming the problem of deciding whether a derivation of $\Delta \vdash_{\mathsf{SUB}} t = u$ exists, into the problem of deciding whether a derivation of $\vdash_{\mathsf{SIMP}} t' = u'$ exists, for some carefully-chosen closed terms t' and u' in an extended signature (to be precise: in a correspondingly extended theory with extra ($\mathbf{f}\mapsto$) axioms for the extra term-formers). We can then exploit decidability of SIMP (Theorem 5.32), and conclude that SUB is decidable. The rest of this subsection makes this formal.

Definition 6.7 Fix some substitution signature $\mathcal{T}$, and fix Δ, t, and u where t and u are terms in $\mathcal{T}$. Let $\mathcal{A}$ be the atoms mentioned anywhere in Δ, t, or u. Let $\mathcal{X}$ be the unknowns mentioned anywhere in Δ, t, or u. For each $X \in \mathcal{X}$ pick the following data:

- Choose an order $a_{X1}, \ldots, a_{Xk_X}$ on the atoms in $\mathcal{A}$ such that $a\#X \notin \Delta$.
- Choose some fresh term-former $\mathsf{d}_X : (\mathbb{T}, \ldots, \mathbb{T})\mathbb{T}$ (so d_X does not occur in $\mathcal{T}$) with k_X arguments.
- Choose some entirely fresh atom c_X when $X : [\mathbb{A}]\mathbb{T}$.

Write $\mathcal{C} = \{c_X \mid X \in \mathcal{X}, X : [\mathbb{A}]\mathbb{T}\}$, write $\mathcal{D} = \{\mathsf{d}_X \mid X \in \mathcal{X}\}$, and let $\mathcal{T}'$ be the signature $\mathcal{T} \cup \mathcal{D}$.

Definition 6.8 Define a substitution ς by:

$$\begin{array}{lcll} \varsigma(X) & = & \mathsf{d}_X(a_{X1}, \ldots, a_{Xk_X}) & (X \in \mathcal{X},\ X : \mathbb{T}) \\ \varsigma(X) & = & [c_X]\mathsf{d}_X(a_{X1}, \ldots, a_{Xk_X}) & (X \in \mathcal{X},\ X : [\mathbb{A}]\mathbb{T}) \\ \varsigma(Y) & = & Y & (Y \notin \mathcal{X}) \end{array}$$

A few words on Definition 6.8 may be useful. Substitution ς maps *possibly open* terms in $\mathcal{T}$ mentioning unknowns in $\mathcal{X}$, to *closed* terms in $\mathcal{T}'$. We design $\varsigma(X)$ to be a closed term which mentions unabstracted those atoms that Δ cannot prove are fresh for X. The term-formers from $\mathcal{D}$ help us to create a syntax in which to write these terms. The atoms from $\mathcal{C}$ help us, where necessary, to place them in the right sort—since according to our sorting system, and consistent with [UPG04], only abstraction can create terms of abstraction sort. We think of $\mathsf{d} \in \mathcal{D}$ and $c_X \in \mathcal{C}$ as *tags*.

Definition 6.9 Call a closed term in $\mathcal{T}'$ **$\mathcal{CD}$-tagged** when for each $X : [\mathbb{A}]\mathbb{T}$ all occurrences of c_X and d_X are restricted to subterms $[c_X]\mathsf{d}_X(t_1, \ldots, t_{k_X})$ where the t_i are also $\mathcal{CD}$-tagged.

ς is Δ-consistent:

Lemma 6.10 $\vdash \Delta\varsigma$ is derivable.

Proof. For each $a\#X \in \Delta$, we need to show $\vdash a\#\varsigma(X)$.

Suppose $a\#X \in \Delta$ and $X : \mathbb{T}$. We must prove $\vdash a\#\mathsf{d}_X(a_{X1}, \ldots, a_{Xk_X})$. By construction $a_{Xi}\#X \notin \Delta$ for $1 \leqslant i \leqslant k_X$, so $a \notin \{a_{X1}, \ldots, a_{Xk_X}\}$, and the result follows.

The case of $X : [\mathbb{A}]\mathbb{T}$ is similar. □

Now it is not hard to show that ς preserves derivability:

Theorem 6.11 If $\Delta \vdash_{\mathsf{SUB}} t = u$ then $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$.

Proof. Suppose $\Delta \vdash_{\mathsf{SUB}} t = u$ in the syntax of $\mathcal{T}$. Then also $\Delta \vdash_{\mathsf{SUB}} t = u$ in the syntax of $\mathcal{T}'$, since $\mathcal{T} \subseteq \mathcal{T}'$. We obtain $\vdash_{\mathsf{SUB}} t\varsigma = u\varsigma$ by Theorem 2.37, using Lemma 6.10. We conclude $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ by Lemma 6.5. □

Recall that we fixed Δ, t, and u. Since t and u mention only unknowns from $\mathcal{X}$, $t\varsigma$ and $u\varsigma$ are closed terms. Then by Lemma 5.30, $t\varsigma^{\downarrow}$ and $u\varsigma^{\downarrow}$ are the SIMPr-normal forms of $t\varsigma$ and $u\varsigma$.

Definition 6.12 Let $\mathcal{A}^+$ be the set of *all* atoms mentioned anywhere in the chain of SIMPr-reductions

$$t\varsigma \equiv t'_1 \to t'_2 \to \cdots \to t'_m \equiv t\varsigma^{\downarrow} \quad \text{and} \quad u\varsigma \equiv u'_1 \to u'_2 \to \cdots \to u'_n \equiv u\varsigma^{\downarrow},$$

extended with a set of fresh atoms $\mathcal{B} = \{b_{Xi} \mid X, i \text{ such that } a_{Xi} \in \mathcal{A}\}$ in bijection with $\mathcal{A}$. Let Δ^+ be Δ enriched with freshness assumptions $a'\#X$ for every $a' \in \mathcal{A}^+ \setminus \mathcal{A}$ and every $X \in \mathcal{X}$.

Without loss of generality we may assume that all terms occurring in the above chains of rewrites are $\mathcal{CD}$-tagged, since the atoms from $\mathcal{C}$ that occur in $t\varsigma$ and $u\varsigma$ are chosen completely fresh (by Definition 6.7), so when rewriting $t\varsigma$ and $u\varsigma$ to normal form it is not necessary to perform α-renamings on these atoms.

Definition 6.13 We let t' and u' range over closed $\mathcal{CD}$-tagged terms in $\mathcal{T}'$ mentioning only atoms in $\mathcal{A}^+ \setminus \mathcal{B}$. The importance of these terms is that they include all of the t'_i and u'_i in the two chains of SIMPr-reductions mentioned above.

Define an **inverse translation** from closed $\mathcal{CD}$-tagged terms in $\mathcal{T}'$ to terms in $\mathcal{T}$, inductively as follows, where we omit $\circ$ between compositions of swappings for brevity:

$$a'^{-1} \equiv a' \qquad ([a']t')^{-1} \equiv [a'](t'^{-1}) \qquad \mathsf{f}(t'_1,\dots,t'_n)^{-1} \equiv \mathsf{f}({t'_1}^{-1},\dots,{t'_n}^{-1}) \quad (\mathsf{f} \notin \mathcal{D})$$
$$\mathsf{d}_X(t'_1,\dots,t'_{k_X})^{-1} \equiv ((b_{Xk_X}\ a_{Xk_X})\cdots(b_{X1}\ a_{X1})\cdot X)[b_{X1} \mapsto {t'_1}^{-1}]\cdots[b_{Xk_X} \mapsto {t'_{k_X}}^{-1}] \qquad (X:\mathbb{T})$$
$$\mathsf{d}_X(t'_1,\dots,t'_{k_X})^{-1} \equiv \mathsf{sub}(((b_{Xk_X}\ a_{Xk_X})\cdots(b_{X1}\ a_{X1})\cdot X)[b_{X1} \mapsto {t'_1}^{-1}]\cdots[b_{Xk_X} \mapsto {t'_{k_X}}^{-1}], c_X) \quad (X:[\mathbb{A}]\mathbb{T})$$

As a first result, we show that $\_^{-1}$ really is the inverse of ς:

Lemma 6.14 $\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{-1} = t$, and $\Delta^+ \vdash_{\mathsf{SUB}} (u\varsigma)^{-1} = u$.

Proof. We prove by induction that if v is a subterm of t or u then $\Delta^+ \vdash_{\mathsf{SUB}} (v\varsigma)^{-1} = v$.

The only interesting case is when $v \equiv \pi \cdot X$. When $X:\mathbb{T}$, we must show

$$\Delta^+ \vdash_{\mathsf{SUB}} \pi \cdot X = ((b_{X1}\ a_{X1})\cdots(b_{Xk_X}\ a_{Xk_X})\cdot X)[b_{X1} \mapsto \pi(a_{X1})]\cdots[b_{Xk_X} \mapsto \pi(a_{Xk_X})].$$

Take $\pi' = (b_{Xk_X}\ \pi(a_{Xk_X}))\cdots(b_{X1}\ \pi(a_{X1}))(b_{X1}\ a_{X1})\cdots(b_{Xk_X}\ a_{Xk_X})$. Then the proof obligation follows from

$$\Delta^+ \vdash_{\mathsf{SUB}} \pi \cdot X = \pi' \cdot X$$

by (**ren**$\mapsto$), since

$$\Delta^+ \vdash \pi(a_{Xi})\#((b_{X1}\ a_{X1})\cdots(b_{Xk_X}\ a_{Xk_X})\cdot X)[b_{X1} \mapsto \pi(a_{X1})]\cdots[b_{Xi-1} \mapsto \pi(a_{Xi-1})]$$

for all i. We can see this as follows: it suffices to show

$$\Delta^+ \vdash \pi(a_{Xi})\#(b_{X1}\ a_{X1})\cdots(b_{Xk_X}\ a_{Xk_X})\cdot X,$$

since the $\pi(a_{Xi})$ are pairwise disjoint and by using the rules for freshnesses. Then there are two possibilities:

- $\pi(a_{Xi}) \neq a_{Xj}$ for all j: then $\pi(a_{Xi})\#X \in \Delta^+$ since $\pi(a_{Xi})\#X \in \Delta$.
- $\pi(a_{Xi}) = a_{Xj}$ for some j: then $b_{Xj}\#X \in \Delta^+$ by definition.

The remaining proof obligation is

$$\Delta^+ \vdash_{\mathsf{SUB}} \pi \cdot X = \pi' \cdot X.$$

It is convenient to show the stronger property $\Delta^+ \vdash_{\mathsf{CORE}} \pi \cdot X = \pi' \cdot X$. By the syntactic criteria of Corollary 2.32 it suffices to show that $\Delta^+ \vdash \mathrm{ds}(\pi,\pi')\#X$. That is, we must show that $\Delta^+ \vdash \mathrm{ds}(\pi,\pi')\#X$ for every a such that $\pi(a) \neq \pi'(a)$. We consider every possible a (every $a \in \pi$ and $a \in \pi'$):

- $a = b_{Xi}$: then $b_{Xi}\#X \in \Delta^+$ by definition, and the result follows.
- $a = a_{Xi}$: then $\pi(a_{Xi}) = \pi'(a_{Xi})$ and there is nothing to prove.
- $a = \pi(a_{Xi})$: then we distinguish two cases:
  - if $\pi(a_{Xi}) = a_{Xj}$ for some j, the result follows by the case of a_{Xi};
  - if $\pi(a_{Xi}) \neq a_{Xj}$ for all j, then $\pi(a_{Xi})\#X \in \Delta^+$ by definition.
- $a \in \pi$, but $a \neq a_{Xj}$ for all j, then $a\#X \in \Delta^+$ by definition.

The case of $X:[\mathbb{A}]\mathbb{T}$ is similar except that we additionally need to prove that

$$\Delta^+ \vdash_{\mathsf{SUB}} [c_X]\mathsf{sub}(\pi \cdot X, c_X) = \pi \cdot X.$$

This follows by axiom ($\eta\mapsto$), since $\Delta^+ \vdash c_X\#\pi \cdot X$. □

Remark 6.15 The reader might wonder why the inverse mapping of the d_X renames the atoms a_{Xi} to the fresh b_{Xi}. Consider for example $(a_{T1}\ a_{T2})\cdot T$ in the empty freshness context $\emptyset$, so we do not know $a_{T1}\#T$ or $a_{T2}\#T$. Then

$$(((a_{T1}\ a_{T2})\cdot T)\varsigma)^{-1} \equiv \mathsf{d}_T(a_{T2}\ a_{T1})^{-1} \equiv ((b_{T2}\ a_{T2})(b_{T1}\ a_{T1})\cdot T)[b_{T1} \mapsto a_{T2}][b_{T2} \mapsto a_{T1}].$$

By calculations we can verify Lemma 6.14:

$$\emptyset^+ \vdash_{\mathsf{SUB}} ((b_{T2}\ a_{T2})(b_{T1}\ a_{T1}) \cdot T)[b_{T1} \mapsto a_{T2}][b_{T2} \mapsto a_{T1}] = (a_{T1}\ a_{T2}) \cdot T,$$

where $\emptyset^+ = \{b_{T1}\#T, b_{T2}\#T, c_T\#T\}$. Had we left out the renaming to fresh atoms then $(((a_{T1}\ a_{T2}) \cdot T)\varsigma)^{-1}$ would be $T[a_{T1} \mapsto a_{T2}][a_{T2} \mapsto a_{T1}]$, which is not equal to $(a_{T1}\ a_{T2}) \cdot T$, since for example

$$\vdash_{\mathsf{SUB}} ((a_{T1}\ a_{T2}) \cdot T)[a_{T1}/T] = a_{T2} \quad \text{but} \quad \vdash_{\mathsf{SUB}} (T[a_{T1} \mapsto a_{T2}][a_{T2} \mapsto a_{T1}])[a_{T1}/T] = a_{T1}.$$

We now build up towards Theorem 6.20, which is our main result. Recall from Definition 6.13 that t' and u' are closed $\mathcal{CD}$-tagged terms in the signature of $\mathcal{T}'$ mentioning only atoms in $\mathcal{A}^+ \setminus \mathcal{B}$.

Lemma 6.16 For any $a' \in \mathcal{A}^+$, if $\vdash a'\#t'$ then $\Delta^+ \vdash a'\#t'^{-1}$.

Proof. By induction on the structure of t'. We treat the hardest case, namely that of $t' \equiv [c_X]\mathsf{d}_X(t'_1, \ldots, t'_{k_X})$. We must show

$$\Delta^+ \vdash a'\#[c_X]\mathsf{sub}((\pi \cdot X)[b_{X1} \mapsto {t'_1}^{-1}] \cdots [b_{Xk_X} \mapsto {t'_{k_X}}^{-1}], c_X),$$

where $\pi = (b_{Xk_X}\ a_{Xk_X}) \cdots (b_{X1}\ a_{X1})$. We distinguish three cases:

- $a' = c_X$: then the result trivially follows by (**#ab**).
- $a' = b_{Xj}$ for some j: then $b_{Xj}\#[b_{Xj}]((\pi \cdot X)[b_{X1} \mapsto {t'_1}^{-1}] \cdots [b_{Xj-1} \mapsto {t'_{j-1}}^{-1}])$ by (#[]**a**), and the result follows by the rules of freshness using the inductive hypothesis.
- $a' \neq b_{Xj}$ for all j and $a' \neq c_X$: then $\pi^{-1}(a') \neq a_{Xj}$ for all j, so $\pi^{-1}(a')\#X \in \Delta^+$ by definition. The result follows using the rules for freshness and the inductive hypothesis.

□

Lemma 6.17 $\Delta^+ \vdash_{\mathsf{CORE}} (\pi \cdot t')^{-1} = \pi \cdot t'^{-1}$ when π mentions only atoms from $\mathcal{A}^+ \setminus (\mathcal{B} \cup \mathcal{C})$.

Proof. By induction on the structure of t'. In the case of $t \equiv [c_X]\mathsf{d}_X(t'_1, \ldots, t'_{k_X})$ we use the fact that $\pi(b) = b$ for all $b \in \mathcal{B}$ and $\pi(c) = c$ for all $c \in \mathcal{C}$. □

Lemma 6.18 If $\vdash_{\mathsf{CORE}} t' = u'$ then $\Delta^+ \vdash_{\mathsf{CORE}} t'^{-1} = u'^{-1}$.

Proof. By induction on the structure of t', using the syntactic criteria of Corollary 2.32. The only non-trivial case is when

$$t' \equiv [a']v', \quad u' \equiv [b']w', \quad \vdash b'\#v', \quad \text{and} \quad \vdash_{\mathsf{CORE}} (b'\ a') \cdot v' = w'.$$

By Lemma 6.16 we obtain $\Delta^+ \vdash b'\#v'^{-1}$, and by the inductive hypothesis $\Delta^+ \vdash_{\mathsf{CORE}} ((b'\ a') \cdot v')^{-1} = w'^{-1}$.

We now prove that:

- $a' \notin \mathcal{C}$ and $b' \notin \mathcal{C}$: Suppose $a' = c_X$ for some $c_X \in \mathcal{C}$. Now t' is $\mathcal{CD}$-tagged so $t' \equiv [c_X]\mathsf{d}_X(t'_1, \ldots, t'_{k_X})$. We have assumed that $\vdash_{\mathsf{CORE}} t' = u'$ so by the syntactic criteria of Corollary 2.32 it must be that $u' \equiv [c_X]\mathsf{d}_X(u'_1, \ldots, u'_{k_X})$ (and $\vdash_{\mathsf{CORE}} t_i = u_i$ for $1 \leqslant i \leqslant k_X$). This is impossible because $a' \neq b'$.
  We deduce that $b' \notin \mathcal{C}$ in a similar manner.
- $a' \notin \mathcal{B}$ and $b' \notin \mathcal{B}$: By our assumption in Definition 6.13.

By Lemma 6.17 we obtain $\Delta^+ \vdash_{\mathsf{CORE}} ((b'\ a') \cdot v')^{-1} = (b'\ a') \cdot v'^{-1}$.

By standard reasoning using the rules for freshness and equality we conclude $\Delta^+ \vdash_{\mathsf{CORE}} [a']v'^{-1} = [b']w'^{-1}$ as required. □

Lemma 6.19 If $\vdash_{\mathsf{SIMPr}} t' \to u'$ then $\Delta^+ \vdash_{\mathsf{SUB}} t'^{-1} = u'^{-1}$.

Proof. We work by induction on the derivation of $\vdash_{\mathsf{SIMPr}} t' \to u'$ to show that $\Delta^+ \vdash_{\mathsf{SUB}} t'^{-1} = u'^{-1}$ is derivable.

If the derivation concludes in ($\to$[]) or ($\to$**f**) we may use the inductive hypothesis and extend the derivation with (**cong**[]) or (**congf**) respectively.

Suppose the derivation concludes in ($\to$**rew**). Then there are various cases depending on which rewrite rule is used:

- (**Rvar**). $\Delta^+ \vdash_{\mathsf{SUB}} a'[a' \mapsto v'^{-1}] = v'^{-1}$ is derivable using axiom (**var**$\mapsto$).

- (**Rb**). $\Delta^+ \vdash_{\mathsf{SUB}} b'[a' \mapsto v'^{-1}] = b'$ is derivable using axiom $(\#\mapsto)$, since $\Delta^+ \vdash a'\#b'$.
- (**Rf**), $\mathsf{f} \notin \mathcal{D}$. $\Delta^+ \vdash_{\mathsf{SUB}} \mathsf{f}(v_1'^{-1}, \ldots, v_n'^{-1})[a' \mapsto v'^{-1}] = \mathsf{f}(v_1'^{-1}[a' \mapsto v'^{-1}], \ldots, v_n'^{-1}[a' \mapsto v'^{-1}])$ is derivable using axiom $(\mathbf{f}\mapsto)$.
- (**Rf**), $\mathsf{f} \in \mathcal{D}$. In case $X : \mathbb{T}$, we must show

$$\begin{aligned}&\Delta^+ \vdash_{\mathsf{SUB}} (\pi \cdot X)[b_{X1} \mapsto v_1'^{-1}] \cdots [b_{Xk_X} \mapsto v_{k_X}'^{-1}][a' \mapsto v'^{-1}]\\ &\quad = (\pi \cdot X)[b_{X1} \mapsto v_1'^{-1}[a' \mapsto v'^{-1}]] \cdots [b_{Xk_X} \mapsto v_{k_X}'^{-1}[a' \mapsto v'^{-1}]],\end{aligned}$$

  where $\pi = (b_{Xk_X}\ a_{Xk_X}) \cdots (b_{X1}\ a_{X1})$. Since $b_{Xi} \notin v'$ for all i, we know $\vdash b_{Xi}\#v'$. Then also $\Delta^+ \vdash b_{Xi}\#v'^{-1}$ by Lemma 6.16. Now we apply part 4 of Lemma 6.2, such that the left-hand-side of the proof obligation is SUB-equal to

$$(\pi \cdot X)[a' \mapsto v'^{-1}][b_{X1} \mapsto v_1'^{-1}[a' \mapsto v'^{-1}]] \cdots [b_{Xk_X} \mapsto v_{k_X}'^{-1}[a' \mapsto v'^{-1}]].$$

  By the rules for equality, this is equal to the right-hand-side of the proof obligation when

$$\Delta^+ \vdash_{\mathsf{SUB}} (\pi \cdot X)[a' \mapsto v'^{-1}] = \pi \cdot X.$$

  By axiom $(\#\mapsto)$ this is when $\Delta^+ \vdash a'\#\pi \cdot X$, i.e. when $\pi^{-1}(a')\#X \in \Delta^+$. There are two possibilities:
  - $a' = a_{Xi}$ for some i: then $\pi^{-1}(a') = b_{Xi}$, and $b_{Xi}\#X \in \Delta^+$ by definition.
  - $a' \neq a_{Xi}$ for all i: then $\pi^{-1}(a') = a'$, and $a'\#X \in \Delta^+$ by definition.

  The result follows.
  The case of $X : [\mathbb{A}]\mathbb{T}$ is similar.
- (**Rabs**). Suppose $\vdash c\#v'$. Then $\Delta^+ \vdash c\#v'^{-1}$ by Lemma 6.16. By $(\mathbf{ax}_{\mathbf{abs}\mapsto})$, we obtain

$$\Delta^+ \vdash_{\mathsf{SUB}} ([c]w'^{-1})[a' \mapsto v'^{-1}] = [c](w'^{-1}[a' \mapsto v'^{-1}])$$

  as required.

The result follows. □

Theorem 6.20 $\Delta \vdash_{\mathsf{SUB}} t = u$ if and only if $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$.

Proof. The left-to-right part is Theorem 6.11.

For the right-to-left part, suppose that $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$. We have observed that there are SIMPr rewrites

$$t\varsigma \equiv t_1 \to t_2 \to \cdots \to t_m \equiv t\varsigma^{\downarrow} \quad \text{and} \quad u\varsigma \equiv u_1 \to u_2 \to \cdots \to u_n \equiv u\varsigma^{\downarrow}.$$

Then by Lemma 6.19, we know

$$\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{-1} \equiv t_1^{-1} = t_2^{-1} = \cdots = t_m^{-1} \equiv (t\varsigma^{\downarrow})^{-1} \text{ and } \Delta^+ \vdash_{\mathsf{SUB}} (u\varsigma)^{-1} \equiv u_1^{-1} = u_2^{-1} = \cdots = u_n^{-1} \equiv (u\varsigma^{\downarrow})^{-1},$$

so $\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{-1} = (t\varsigma^{\downarrow})^{-1}$ and $\Delta^+ \vdash_{\mathsf{SUB}} (u\varsigma)^{-1} = (u\varsigma^{\downarrow})^{-1}$ by transitivity.

We supposed that $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ so by Corollary 5.20 it must be that $\vdash_{\mathsf{CORE}} t\varsigma^{\downarrow} = u\varsigma^{\downarrow}$. By Lemma 6.18 also $\Delta^+ \vdash_{\mathsf{CORE}} (t\varsigma^{\downarrow})^{-1} = (u\varsigma^{\downarrow})^{-1}$.

Then $\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{-1} = (u\varsigma)^{-1}$ using symmetry and transitivity. By Lemma 6.14 then also $\Delta^+ \vdash_{\mathsf{SUB}} t = u$. Since Δ^+ extends Δ with atoms not in t, u, we conclude $\Delta \vdash_{\mathsf{SUB}} t = u$ using (**fr**). □

As a simple corollary of Theorem 6.20, we obtain decidability of SUB.

Corollary 6.21 (Decidability of SUB) It is decidable whether $\Delta \vdash_{\mathsf{SUB}} t = u$.

Proof. By Theorem 6.20, $\Delta \vdash_{\mathsf{SUB}} t = u$ is equivalent to $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$. By Theorem 5.32, this is decidable. □

Combining Theorem 6.20 and Corollary 5.20, we extract the following algorithm that decides whether $\Delta \vdash_{\mathsf{SUB}} t = u$:

1. Map possibly open terms t and u to closed terms $t\varsigma$ and $u\varsigma$.
2. Evaluate $t\varsigma^{\downarrow}$ and $u\varsigma^{\downarrow}$.
3. Check whether $\vdash_{\mathsf{CORE}} t\varsigma^{\downarrow} = u\varsigma^{\downarrow}$ using the syntactic criteria of Corollary 2.32.

We can extract a witnessing derivation in SUB of $\Delta \vdash_{\mathsf{SUB}} t = u$ from the algorithm above: the meat of work is done in the proofs of Lemma 6.19 and Lemma 6.14.

By a similar method to the one we used to prove Theorem 6.20 we can prove conservativity of SUB over CORE. We use the notation and machinery of this subsection in the following proof:

Theorem 6.22 (Conservativity of SUB) Suppose that t and u do *not* mention term-former sub. Then

$$\Delta \vdash_{\mathsf{SUB}} t = u \qquad \text{if and only if} \qquad \Delta \vdash_{\mathsf{CORE}} t = u.$$

Proof. A derivation in CORE is also a derivation in SUB so the right-to-left implication is immediate.

Now suppose that $\Delta \vdash_{\mathsf{SUB}} t = u$. We take a suitably chosen enriched signature and suitably chosen ς, as in the proofs above. By Theorem 6.11, $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$. By construction $t\varsigma$ and $u\varsigma$ do not mention sub, therefore by Corollary 5.19 also $\vdash_{\mathsf{CORE}} t\varsigma = u\varsigma$.

Given the derivability of $\vdash_{\mathsf{CORE}} t\varsigma = u\varsigma$ we can prove the derivability of $\Delta \vdash_{\mathsf{CORE}} t = u$ by exploiting the syntactic criteria of Corollary 2.32. The proof is by induction on t using detailed but entirely routine calculations. We consider just one case, the hardest one:

Suppose $t \equiv \pi \cdot X$ and $X : [\mathbb{A}]\mathbb{T}$. Then

$$t\varsigma \equiv [c_X]\mathsf{d}_X(\pi(a_{X1}), \ldots, \pi(a_{Xk_X})).$$

By the syntactic criteria of Corollary 2.32 if $\vdash_{\mathsf{CORE}} t\varsigma = u\varsigma$ it *must* be that

$$u\varsigma \equiv [b]\mathsf{d}_X(\pi(a_{X1}), \ldots, \pi(a_{Xk_X})).$$

Here b is not equal to $\pi(a_{Xi})$ for $1 \leqslant i \leqslant k_X$. By the construction of $u\varsigma$ and the way we chose $a_{X1}, \ldots, a_{Xk_X}$ to be the atoms mentioned in Δ, t, or u which are *not* provably fresh for X in Δ, it follows that u *must* have been equal to $\pi' \cdot X$ for some π' such that $\Delta \vdash \mathrm{ds}(\pi, \pi')\#X$. It follows that $\Delta \vdash_{\mathsf{CORE}} t = u$ as required. □

6.3. ω-completeness

We consider completeness with respect to the ground term model (see Definition 6.23 below for a formal definition). This is also called *ω-completeness*. Before we go into the proof of ω-completeness, it is useful to mention why this is an appropriate notion to consider.

Nominal algebra enjoys a general completeness result [GM07] with respect to a standard class of semantics in *nominal sets* [GP02]. SUB is a nominal algebra theory, so it is automatically sound and complete with respect to the standard class of nominal sets semantics.

A completeness result is *weaker*, the *larger* the class of semantics that it uses. Can we strengthen this general result to some more specific class than the nominal sets models?

Theorem 6.20 can be read as a completeness result with respect to a class of models built out of syntax enriched with finitely but unboundedly many extra term-formers d_X. We could stop there, *however*, we would have an even more powerful completeness result if we could strengthen this to completeness with respect to terms of substitution signature $\mathcal{T}$ itself.

In fact, so long as $\mathcal{T}$ contains a term-former which takes more than one argument (like app or plus), then we can nail SUB down to *the* theory of capture-avoiding substitution on ground terms of $\mathcal{T}$. We now have all the machinery we need to do this quite easily:

Definition 6.23 Call σ a **ground substitution** for Δ, t and u when for every unknown X in Δ, t, and u, $\sigma(X)$ is a ground term (it mentions no unknowns and does not mention sub).

Call SUB **ω-complete** when for all Δ, t and u, if $t\sigma^{\downarrow} =_\alpha u\sigma^{\downarrow}$ for all Δ-consistent ground substitutions σ (for Δ, t and u), then $\Delta \vdash_{\mathsf{SUB}} t = u$.

We shall prove the contrapositive. Supposing $\Delta \nvdash_{\mathsf{SUB}} t = u$, we will exhibit some Δ-consistent ground substitution σ such that $t\sigma^{\downarrow} \neq_\alpha u\sigma^{\downarrow}$.

We cannot use ς from Definition 6.8 because ς maps to an extended signature $\mathcal{T}'$ with extra term-formers d_X. But we can use ς to construct another substitution with the right properties, as we now see.

Recall from the previous subsection the chain of rewrites

$$t\varsigma \equiv t'_1 \to t'_2 \to \cdots \to t'_m \equiv t\varsigma^{\downarrow} \quad \text{and} \quad u\varsigma \equiv u'_1 \to u'_2 \to \cdots \to u'_n \equiv u\varsigma^{\downarrow}.$$

Note that $t\varsigma^{\downarrow}$ and $u\varsigma^{\downarrow}$ are ground.

Definition 6.24 Let $\mathcal{A}'$ be the set of all atoms mentioned in the above chains, let t' and u' range over closed terms in $\mathcal{T}'$ mentioning only atoms from $\mathcal{A}'$, and choose atoms $\{a_X \mid X \in \mathcal{X}\}$ completely fresh from $\mathcal{A}'$.

Assuming that $\mathcal{T}$ contains a binary term-former, say $\mathsf{pair} : (\mathbb{T}, \mathbb{T})\mathbb{T}$), define a translation -* from closed terms in $\mathcal{T}'$ to closed terms in $\mathcal{T}$ by:

$$a'^* \equiv a' \qquad ([a']t')^* \equiv [a']t'^* \qquad \mathsf{f}(t'_1, \ldots, t'_n)^* \equiv \mathsf{f}(t'^*_1, \ldots, t'^*_n) \quad (\mathsf{f} \notin \mathcal{D})$$
$$\mathsf{d}_X()^* \equiv \mathsf{pair}(a_X, a_X) \qquad \mathsf{d}_X(t'_1)^* \equiv \mathsf{pair}(a_X, t'^*_1)$$
$$\mathsf{d}_X(t'_1, \ldots, t'_{k_X})^* \equiv \mathsf{pair}(a_X, \mathsf{pair}(t'^*_1, \mathsf{pair}(t'^*_2, \ldots, \mathsf{pair}(t'^*_{k_X-1}, t'^*_{k_X})))) \quad (k_X > 1)$$

It is not hard to verify the following properties:

Lemma 6.25

1. $\vdash (t'^*)^{\downarrow} \equiv (t'^{\downarrow})^*$.
2. $\vdash_{\mathsf{CORE}} t' = u'$ if and only if $\vdash_{\mathsf{CORE}} t'^* = u'^*$.
3. $\vdash_{\mathsf{SIMP}} t' = u'$ if and only if $\vdash_{\mathsf{SIMP}} t'^* = u'^*$.

Proof. Part 1 is by induction on the syntax of t'. For the case of $t' \equiv \mathsf{sub}(u', v')$, we use the property that $g'[h'/a']^* \equiv g'^*[h'^*/a']$ (where g', h', a' and $g'[h'/a']$ mention only atoms from $\mathcal{A}'$), which is easy to verify.

Part 2 is by induction on the syntax of t', using the syntactic criteria from Corollary 2.32.

For part 3, by Corollary 5.20 it suffices to prove the equivalent

$$\vdash_{\mathsf{CORE}} t'^{\downarrow} = u'^{\downarrow} \quad \text{if and only if} \quad \vdash_{\mathsf{CORE}} (t'^*)^{\downarrow} = (u'^*)^{\downarrow},$$

which follows by parts 1 and 2. □

Definition 6.26 Define the substitution ς^* by:

$$\varsigma^*(X) = \varsigma(X)^* \quad (X \in \mathcal{X}),$$
$$\varsigma^*(Y) = Y \quad (Y \notin \mathcal{X}).$$

It is a fact of the construction that ς^* maps every X appearing in Δ, t, or u, to a *ground term in* $\mathcal{T}$. This is morally 'the same' as $\varsigma(X)$, but we map each extra term-former d_X to a collection of instances of pair.

Lemma 6.27 If v is a subterm of t or v is a subterm of u, $\quad v(\varsigma^*) \equiv (v\varsigma)^*$.

Proof. By an easy induction on the structure of v. □

Corollary 6.28 $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ if and only if $\vdash_{\mathsf{SIMP}} t(\varsigma^*) = u(\varsigma^*)$.

Proof. By Lemma 6.27 and part 3 of Lemma 6.25. □

Theorem 6.29 SUB is ω-complete.

Proof. We show that if $\Delta \nvdash_{\mathsf{SUB}} t = u$ then there exists a Δ-consistent ground substitution σ (for Δ, t and u) such that $t\sigma^{\downarrow} \neq_\alpha u\sigma^{\downarrow}$.

Suppose $\Delta \nvdash_{\mathsf{SUB}} t = u$. Then also $\nvdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ by Theorem 6.20. Now $\nvdash_{\mathsf{SIMP}} t(\varsigma^*) = u(\varsigma^*)$ by Corollary 6.28. Then we obtain $(t\varsigma^*)^{\downarrow} \neq_\alpha (u\varsigma^*)^{\downarrow}$ by Theorem 3.9 and Corollary 5.20.

Now take $\sigma = \varsigma^*$. Since ς^* maps to ground terms in $\mathcal{T}$, the result follows. □

6.4. Restricting the sort system

We now turn to our discussion in Subsect. 2.1 on unknowns of abstraction sort $[\mathbb{A}]\mathbb{T}$ and term-formers $\mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$ (Remark 2.10). We mentioned that they are convenient, but *not* strictly necessary.

To see what precisely we mean by this, we consider a typical signature for languages with binding, namely that of the lambda calculus (see Examples 2.6 and 2.8).

Definition 6.30 Consider the following signature:

$$\mathsf{app} : (\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{lam} : ([\mathbb{A}]\mathbb{T})\mathbb{T} \qquad \mathsf{sub} : ([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T}.$$

Let theory SUB' over this signature have axioms

$$\begin{array}{rll}
(\mathbf{var}{\mapsto}') & \vdash & a[a \mapsto T] = T \\
(\#{\mapsto}') & a\#U \vdash & U[a \mapsto T] = U \\
(\mathbf{app}{\mapsto}') & \vdash & \mathsf{app}(U, V)[a \mapsto T] = \mathsf{app}(U[a \mapsto T], V[a \mapsto T]) \\
(\mathbf{lam}{\mapsto}') & b\#T \vdash & \mathsf{lam}([b]U)[a \mapsto T] = \mathsf{lam}([b](U[a \mapsto T])) \\
(\mathbf{sub}{\mapsto}') & b\#T \vdash & V[b \mapsto U][a \mapsto T] = V[a \mapsto T][b \mapsto U[a \mapsto T]] \\
(\mathbf{ren}{\mapsto}') & b\#T \vdash & T[a \mapsto b] = (b\ a) \cdot T.
\end{array}$$

Theorem 6.31 Suppose SUB is the theory of substitution over the signature of the lambda calculus from Example 2.8. Then for any Δ, t and u (in this signature) not mentioning unknowns of sort $[\mathbb{A}]\mathbb{T}$ or terms of sort $[\mathbb{A}][\mathbb{A}]\mathbb{T}$:

$$\Delta \vdash_{\mathsf{SUB}} t = u \quad \text{if and only if} \quad \Delta \vdash_{\mathsf{SUB}'} t = u.$$

Proof. For the right-to-left part it suffices to show that each axiom of SUB' can be derived in SUB. Both axiom (**lam**$\mapsto'$) and (**sub**$\mapsto'$) follow by an instance of (f$\mapsto$) and (#$\mapsto$), the other axioms of SUB' follow directly from their corresponding axioms in SUB.

For the left-to-right part, suppose $\Delta \vdash_{\mathsf{SUB}} t = u$. Then also $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ by Theorem 6.11. We observe that there are SIMPr rewrites

$$t\varsigma \equiv t_1 \to t_2 \to \cdots \to t_m \equiv t\varsigma^{\downarrow} \quad \text{and} \quad u\varsigma \equiv u_1 \to u_2 \to \cdots \to u_n \equiv u\varsigma^{\downarrow},$$

such that whenever a term of sort $[\mathbb{A}][\mathbb{A}]\mathbb{T}$ is introduced by a rewrite, it is removed in the next step. More precisely, only an application of rewrite rule (**Rf**), where $\mathsf{f} = \mathsf{lam}$, can introduce such a term, but we can always apply the (**Rabs**) rule to get rid of it.

We can use properties similar to Lemma 6.19 on these chains of rewrites to obtain derivations of

$$\Delta^+ \vdash_{\mathsf{SUB}'} (t\varsigma)^{-1} = (t\varsigma^{\downarrow})^{-1} \quad \text{and} \quad \Delta^+ \vdash_{\mathsf{SUB}'} (u\varsigma)^{-1} = (u\varsigma^{\downarrow})^{-1}.$$

That is, we need one such property for direct rewrites and another one for two-step rewrites. The inverse translation never introduces unknowns of sort $[\mathbb{A}]\mathbb{T}$ or terms of sort $[\mathbb{A}][\mathbb{A}]\mathbb{T}$.

We also have $\Delta^+ \vdash_{\mathsf{CORE}} (t\varsigma^{\downarrow})^{-1} = (u\varsigma^{\downarrow})^{-1}$ by Corollary 5.20, Lemma 6.18 and our assumption $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$. Then $\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{-1} = (u\varsigma)^{-1}$ using (**symm**) and (**tran**). Using a property similar to Lemma 6.14 we obtain $\Delta^+ \vdash_{\mathsf{SUB}'} t = u$. We conclude $\Delta \vdash_{\mathsf{SUB}'} t = u$ using (**fr**). □

The presentation of theory SUB' is somewhat more specific and longer than SUB, since we cannot use a meta-variable f to range over term-formers. For this reason we preferred SUB in this paper.

7. Related work and conclusions

Substitution underlies the quantifiers in predicate logics and the λ-binder of the λ-calculus ... and lots more besides. Quantification and binding are *central* features of these systems. This paper discusses their common denominator, substitution.

Future work using nominal techniques seems likely to require an axiomatisation of substitution within the nominal style. This paper provides that, and proves a precise sense in which that axiomatisation can be considered *the right* one, namely soundness and completeness with respect to a canonical term model (Theorem 6.29). It also provides a precise sense in which that axiomatisation can be considered *tractable*, namely decidability of equality up to the axioms for substitution.

Crabbé [Cra04b, Cra04a] axiomatises substitution much like us and shares (in our terminology) atoms and freshness conditions. However, his axiomatisation is not capture-avoiding from the simple fact that he does not treat binding: '... we are not concerned with the notion of bound variable' [Cra04a, p. 2].

Feldman [Fel82] gives an algebraic axiomatisation inspired by a concrete model of functions/evaluations. His axioms are closer in spirit to Cylindric Algebras [BS81] and Lambda Abstraction Algebras [LS04, Sal00]. 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 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) [Bar84] and related systems implement substitution by 'pipes' (e.g. the translation of λ-terms into CA [Bar84]). There is no native notion of binder, nor of capture-avoidance. General truths such as (#$\mapsto$) are not provable as equalities between combinators, though they remain true and can be proved informally by calculations parametric over specific structure.

Lescanne's classic survey [Les94] and the thesis of Bloo [Blo97] 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 [dB72] 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.

Sun has investigated Binding Algebras [Sun99]. As far as we understand, binding algebras implement binding in the style of higher-order or first-order logic—using variables and binders — but without committing to a functional semantics or to higher orders. Put another way, binding algebra enriches the language of algebra with binding, substitution, and α-conversion—but leaves out λ-abstraction and β-conversion. (This has much in common with Binding Logic [DHK02], which does something very similar to first-order logic; neither thread of research cites the other so they appear to have developed independently.)

Nominal algebra is in this spirit. In fact it does not commit to *substitution*, but by design SUB does and SUB is the topic of this paper. We note that in Sun's work that every variable must be explicitly accounted for in some binder or some evaluation, some where. That is, variables have no independent denotational existence analogous to that of the atoms in nominal sets. Our best guess is that binding algebras correspond, in our world, to elements with empty support of models of SUB. We do not intend to investigate a connection with binding algebra but we do plan to consider binding logic.

There is a close connection between nominal sets [GP02] and categories of presheaves used in another thread of work [FT01, FPT99, TP05]. This uses ideas from categorical algebra [LS86] and applies presheaves to give just enough extra structure to model names and name-binding. Nominal sets, the canonical semantics for nominal terms, are the Schanuel topos; they can be viewed as a category of pullback-preserving presheaves. Being pullback-preserving (to be more precise, preserving pullbacks of pairs of monos) does not correspond to having finite support—it corresponds, in the terminology of nominal techniques, to assuming a *unique least supporting set*. This is not a vital assumption, but without a unique least supporting set, the freshness judgement $a\#X$ of nominal terms is meaningless in its current form. It is not clear how [FT01, FPT99, TP05] would give a direct semantics to nominal terms. Thus an easy and direct connection cannot be made at the moment.

A direct connection could be made by relating the general class of models of substitution determined by SUB (which we do not consider in this paper) with the classes of models in presheaves—or alternatively, if the work based on presheaves could include a completeness result for some canonical model (which to our knowledge has not yet been done); this could then be conveniently compared to the ground term model of SUB. Since our models would be in nominal sets and would have unique least supporting sets, and the presheaf models do not, our best guess is that a canonical model for the presheaf work, if it exists, would be a 'ground term model enriched with non-unique least supporting sets'. It remains to make that formal, and non-syntactic models of SUB remain to be investigated.

There is much possible and interesting future work:

Decidability of *unification* up to SUB, the axioms for substitution, remains an open problem. Nominal unification [UPG04] (in our terminology, unification of nominal terms up to CORE) *is* decidable. Nominal unification is to be compared with higher-order patterns [Mil91]. We suggest that unification up to SUB is to be compared with higher-order unification [Hue02], and the technique of Huet's algorithm could perhaps be imported.

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 directly to the Lambda-Context Calculus [GL07] and Hierarchical Nominal Rewriting [Gab07] both of which feature hierarchies of 'increasingly meta-'variables plus operational notions of semantics for substituting those variables.

Finally, SUB is just an axiom system and it has interesting non-syntactic models. Exploring them is current research, and we hope it will be possible to exploit those models to design interesting new classes of logics and lambda-calculi.

References

[Bar84] Barendregt HP (1984) The lambda calculus: its syntax and semantics (revised edn). North-Holland, Amsterdam

[Blo97] Bloo R (1997) Preservation of termination for explicit substitution. PhD Thesis, Eindhoven University of Technology, Eindhoven

[BR95] Bloo R, Rose KH (1995) Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In: CSN-95: computing science in the Netherlands, Amsterdam. Stichting Mathematisch Centrum, pp 62–72

[BS81] Burris S, Sankappanavar H (1981) A course in universal algebra. Springer, Berlin

[Cra04a] Crabbé M (2004) On the notion of substitution. Logic J IGPL 12(2):111–124

[Cra04b] Crabbé M (2004) Une axiomatisation de la substitution. Comptes Rendus l'Académie Sci Paris Série I 338:433–436

[dB72] de Bruijn NG (1972) Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem. Indagationes Math 5(34):381–392

[DHK02] Dowek G, Hardin T, Kirchner C (2002) Binding logic: proofs and models. In: LPAR'02: 9th international conference on logic for programming, artificial intelligence, and reasoning. Vol 2514 of LNCS. Springer, Berlin, pp 130–144

[Fel82] Feldman N (1982) Axiomatization of polynomial substitution algebras. J Symbol Logic 47(3):481–492

[FG07] Fernández M, Gabbay MJ (2007) Nominal rewriting. Inf Comput 205:917–965

[FPT99] Fiore M, Plotkin G, Turi D (1999) Abstract syntax and variable binding. In: 14th Annual symposium on logic in computer science, Brussels. IEEE Computer Society Press, New York, pp 193–202

[FT01] Fiore M, Turi D (2001) Semantics of name and value passing. In: 16th annual symposium on logic in computer science, Los Alamitos. IEEE Computer Society Press, New York, pp 93–104

[Gab07] Gabbay MJ (2007) Hierarchical nominal terms and their theory of rewriting. ENTCS 174(5):37–52

[GL07] Gabbay MJ, Lengrand S (2007) The lambda-context calculus. In: LFMTP'07: international workshop on logical frameworks and meta-languages, to be published in ENTCS

[GM06a] Gabbay MJ, Mathijssen A (2006) Capture-avoiding substitution as a nominal algebra. In: Theoretical aspects of computing: ICTAC 2006. Vol 4281 of LNCS. Springer, Berlin, pp 198–212

[GM06b] Gabbay MJ, Mathijssen A (2006) Nominal algebra. Technical Report HW-MACS-TR-0045, Heriot-Watt

[GM06c] Gabbay MJ, Mathijssen A (2006) 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, New York, pp 189–200

[GM07] Gabbay MJ, Mathijssen A (2007) A formal calculus for informal equality with binding. In: WoLLIC'07: 14th workshop on logic, language, information and computation. Vol 4576 of LNCS. Springer, Berlin, pp 162–176

[GP02] Gabbay MJ, Pitts AM (2002) A new approach to abstract syntax with variable binding. Form Aspects Comput 13(3–5):341–363

[Hod01] Hodges W (2001) Elementary predicate logic. In: Gabbay DM, Guenthner F (eds) Handbook of philosophical logic, 2nd edn, Vol 1. Kluwer, Dordrecht, pp 1–131

[Hue02] Huet G (2002) Higher order unification 30 years later. In: TPHOLs 2002: theorem proving in higher order logics, number 2410 in LNCS. Springer, Berlin, pp 241–258

[KvOvR93] Klop JW, van Oostrom V, van Raamsdonk F (1993) Combinatory reduction systems, introduction and survey. Theor Comput Sci 121:279–308

[Les94] Lescanne P (1994) From $\lambda\sigma$ to $\lambda\upsilon$ a journey through calculi of explicit substitutions. In: POPL'94: Proceedings of 21st ACM SIGPLAN-SIGACT symposium on principles of programming languages. ACM Press, New York, pp 60–69

[LS86] Lambek J, Scott PJ (1986) Introduction to higher order categorical logic. Cambridge University Press, Cambridge

[LS04] Lusin S, Salibra A (2004) The lattice of lambda theories. J Logic Comput 14(3):373–394

[Mil91] Miller D (1991) A logic programming language with lambda-abstraction, function variables, and simple unification. Extensions Logic Program 475:253–281

[MN98] Mayr R, Nipkow T (1998) Higher-order rewrite systems and their confluence. Theor Comput Sci 192:3–29

[Sal00] Salibra A (2000) On the algebraic models of lambda calculus. Theor Comput Sci 249(1):197–240

[Sun99] Sun Y (1999) An algebraic generalization of Frege structures—binding algebras. Theor Comput Sci 211:189–232

[TP05] Tanaka M, Power J (2005) A unified category-theoretic formulation of typed binding signatures. In: MERLIN'05: Proceedings of the ACM SIGPLAN workshop on mechanized reasoning about languages with variable binding. ACM Press, New York, pp 13–24

[UPG04] Urban C, Pitts AM, Gabbay MJ (2004) Nominal unification. Theor Comput Sci 323(1–3):473–497

Received 2 May 2007

Accepted in revised form 1 November 2007 by K. Barkaoui, M. Broy, A. Cavalcanti and A. Cerone

Published online 15 January 2008