Permissive nominal terms and their unification: an infinite, co-infinite approach to nominal techniques

Gilles Dowek, Murdoch J. Gabbay, Dominic P. Mulligan

Abstract

Nominal terms extend first-order terms with binding. They lack some properties of first- and higher-order terms: Terms must be reasoned about in a context of 'freshness assumptions'; it is not always possible to 'choose a fresh variable symbol' for a nominal term; it is not always possible to 'α-convert a bound variable symbol' or to 'quotient by α-equivalence'; the notion of unifier is not based just on substitution.

Permissive nominal terms closely resemble nominal terms but they recover these properties, and in particular the 'always fresh' and 'always rename' properties. In the permissive world, freshness contexts are elided, equality is fixed, and the notion of unifier is based on substitution alone rather than on nominal terms' notion of unification based on substitution plus extra freshness conditions.

We prove that expressivity is not lost moving to the permissive case and provide an injection of nominal terms unification problems and their solutions into permissive nominal terms problems and their solutions.

We investigate the relation between permissive nominal unification and higher-order pattern unification. We show how to translate permissive nominal unification problems and solutions in a sound, complete, and optimal manner, in suitable senses which we make formal.

Keywords: Nominal unification, higher-order pattern unification, permissive nominal techniques

[☆] Thanks to the anonymous referees.

Contents

1 Introduction **3**
- 1.1 About nominal terms and permissive nominal terms . . . 3
- 1.2 Difference 1: about fixing freshness contexts . . . 4
- 1.3 Difference 2: about choosing fresh atoms . . . 5
- 1.4 Permissive nominal terms in this paper . . . 7
- 1.5 Map of the paper . . . 7

2 Permissive nominal terms **8**
- 2.1 Foundations of permission sets . . . 12

3 Substitutions, problems, and solutions **12**
- 3.1 Substitutions . . . 12
- 3.2 Unification problems, and solutions . . . 13

4 Relation to nominal terms **14**
- 4.1 Alpha-equivalence between nominal and permissive nominal terms . . . 14
- 4.2 Substitutions and solutions between nominal and permissive nominal unification problems . . . 16

5 Support inclusion problems **17**
- 5.1 Simplification reduction and normal forms . . . 17
- 5.2 Building solutions for support inclusion problems . . . 19
- 5.3 Support reduction example . . . 22

6 Permissive nominal unification problems **22**
- 6.1 The unification algorithm . . . 22
- 6.2 Examples of the algorithm . . . 24
- 6.3 Preservation of solutions . . . 25
- 6.4 Simplification rewrites calculate principal solutions . . . 26

7 Lambda-term syntax **29**

8 Translating nominal terms to lambda-term syntax **32**
- 8.1 The translation, and its soundness . . . 32
- 8.2 Capturable atoms; injectivity and minimality . . . 33

9 Translating substitutions; relating solutions of nominal and pattern unification problems **35**
- 9.1 Translating substitutions . . . 35
- 9.2 Translating permissive nominal unification to pattern unification; soundness, weak completeness . . . 37
- 9.3 Strong Completeness . . . 40

10 Conclusions **43**
- 10.1 Related work . . . 44
- 10.2 Future work . . . 45

1. Introduction

1.1. About nominal terms and permissive nominal terms

Many formal languages feature names and binding: examples include quantification, λ-abstraction, sets comprehension $\{x \mid \phi(x)\}$, and process-calculi name-hiding. Binding is ubiquitous, because variables are there to be bound or substituted.

In contrast, variables cannot be bound in first-order terms: In first-order logic, variables are bound in propositions by quantifiers and not at all in terms; first-order rewriting does not allow binding as it is based on first-order terms; and, many programming languages and proof systems allow datatypes of terms, but only of first-order terms.

This motivates logics where variables can be bound by any function or predicate symbol [9], extensions of rewriting on terms with binders [28, 30, 10], and programming languages and proof systems allowing datatypes with binders [36, 31, 39, 27, 35] and more generally, definitions of a notion of term where variables may be bound.

In particular, this motivates *nominal terms* [40]. They are designed to directly represent informal schematic specifications. For example:

Informal equality:	If $y \notin \mathit{fv}(t)$ then $\lambda x.t$ is α-equivalent to $\lambda y.[y/x]t$.
Nominal terms:	$b\#X \vdash \lambda[a]X = \lambda[b](b\,a) \cdot X$
Permissive:	$\lambda[a]X^S = \lambda[b](b\,a) \cdot X^S$, where $b \notin S$
Informal unification:	Which t and u make $\lambda x.\lambda y.(y\,t)$ equal to $\lambda x.\lambda x.(x\,u)$?
Nominal unification:	$\varnothing \vdash \lambda[a]\lambda[b](b\,X) \stackrel{?}{=} \lambda[a]\lambda[a](a\,Y)$
Permissive:	$\lambda[a]\lambda[b](b\,X^S) \stackrel{?}{=} \lambda[a]\lambda[a](a\,Y^S)$ (here, $a \in S$, $b \in S$)

The first example is part of the usual specification of α-equivalence. The second example is from [40]. Definitions of nominal and permissive nominal terms will follow.

Nominal terms have been explored in logic-programming [7], rewriting [10], logic [19, 21], and elsewhere.

An intuition of the translation from informal specification to (permissive) nominal terms is as follows:

- Object-level variable symbols x and y correspond to *atoms* a and b (Definition 2.1).
- Meta-level variables t and u correspond to *unknowns* X.

  Substitution of unknowns is capturing, which models the effect of writing "set t to be x in $\lambda x.t$; we get $\lambda x.x$". Similarly, $(\lambda[a]X)[X{:=}a] \equiv [a]a$.

  Atoms and unknowns are two distinct *levels* of variable, just as object- and meta-level are two levels.
- Conditions like $x \notin \mathit{fv}(t)$ correspond to *freshness* conditions $b\#X$ in nominal terms.

  $b\#X$ is a restriction, which is enforced on the notion of substitution, on what X may be instantiated to. $b\#X$ is a promise to instantiate X only to terms for which b is fresh (see the conditions '$\nabla' \vdash \theta(\nabla)$' in Lemma 2.14, and '$\nabla \vdash a\#\theta(t)$' in Definition 3.1 of [40]).

- Object-level renaming $[y/x]$ is modelled by *swapping* $(b\ a)$ (which maps b to a, a to b, and all other atoms c to themselves; Definition 2.5).

  Swappings are bijective on atoms and invertible, unlike renamings, and for this reason they are *naturally capture-avoiding*.[1] This gives them attractive mathematical properties which can be exploited in the mathematical theory. Using them to manage renaming in α-equivalence is a conspicuous feature of nominal techniques and follows [25, 40].

  The reader can safely think of $[y/x]$ as corresponding to $(b\ a)$ so long as b (y) is sufficiently fresh. It is notable that freshness side-conditions in informal practice seem to appear exactly in those places in which they would be necessary for this to work.

- λ and application are term-formers (function-symbols). In the examples above we write application infix, as is standard.

Permissive nominal terms differ in two ways from nominal terms:

- *Difference 1.* Freshness information is fixed once and for all, and associated to unknowns in the manner of a sorting or typing annotation.

  For example: $b\#X \vdash X$ is a nominal term-in-freshness context X whose freshness context insists that b not occur free in X. A corresponding permissive nominal term is X^S where S is a set of atoms we call the *permission set* of X, and $b \notin S$.

- *Difference 2.* Permission sets are sets of atoms that are both infinite and co-infinite (see Remark 2.23; $S \subseteq \mathbb{A}$ is co-infinite when $\mathbb{A} \setminus S$ is infinite).

These combine with consequences which we now discuss.

1.2. *Difference 1: about fixing freshness contexts*

Difference 1 on its own is mostly a matter of presentation. It is mentioned already in [40, Remark 2.6] (there, we would obtain 'permission sets' A such that A is infinite and $\mathbb{A} \setminus A$ is finite). It is still worth reflecting briefly on what that difference in presentation means, for some example results. Part 2 of Lemma 2.14 from [40] reads as follows:

> If $\nabla' \vdash \sigma(\nabla)$ and $\nabla \vdash a\#t$, then $\nabla \vdash a\#\sigma(t)$.

The corresponding result in this paper (Lemma 3.3) reads as follows:

> $fa(r\theta) \subseteq fa(r)$.

Parts 1 to 3 of Lemma 2.7 from [40] read as follows:

1. If $\nabla \vdash a\#\pi \cdot t$ then $\nabla \vdash \pi^{-1} \cdot a\#t$.
2. If $\nabla \vdash \pi \cdot a\#t$ then $\nabla \vdash a\#\pi^{-1} \cdot t$.
3. If $\nabla \vdash a\#t$ then $\nabla \vdash \pi \cdot a\#\pi \cdot t$.

[1] For example, $(b\ a) \cdot [a](b\ a) \equiv [b]a$ (Definition 4.3) whereas $(\lambda x.(y\ x))[x/y]$ is equal to $\lambda x'.(x\ x')$ where x' is chosen fresh; with the permutation, the capture-avoidance is automatic, with the renaming we have to deliberately rename. This nice turn of phrase is due to Cheney, as far as we know.

These three statements correspond to one result in this paper (Lemma 2.17) which reads as follows:

$$fa(\pi \cdot r) = \pi \cdot fa(r).$$

To us, the 'permissive' versions seem shorter and clearer.

In fact there is a little more to Difference 1 than presentation. If freshness contexts are fixed then some 'consistency' properties of the same term across different freshness contexts become irrelevant. For instance consider the *weakening* and *strengthening* results for freshness contexts like those in Subsection 3.2 of [24] or Subsection 3.3.2 of [21]; if we had used permissive nominal terms, then these results would have been irrelevant and could have been omitted.

1.3. Difference 2: about choosing fresh atoms

The reader used to nominal terms can think of Difference 2 as allowing and requiring infinite freshness contexts.[2] Together with Difference 1, this gives us two key properties which are absent in [40]:

- An infinite supply of fresh atoms is guaranteed for every term. There is no need to change a freshness context to obtain fresh atoms, because they are already there.
- Terms may be directly quotiented by α-equivalence. They also become susceptible to the nominal inductive reasoning principles of [25].

This matters. The nominal terms as presented in [40] cannot be quotiented by α-equivalence, and cannot be subjected to the nominal inductive principles of nominal abstract syntax [25]. In this respect, they are less tractable than first-order or higher-order terms, which can be.

True enough, it is always possible to 'add a fresh atom to the freshness context' in nominal terms and rename, but this causes similar difficulties to those encountered in α-renaming on name-carrying syntax.

This is potentially a problem for the viability of any reasoning on, programming on, or extensions of nominal terms. We have experienced this ourselves in [24, 23].

It becomes possible to argue against nominal terms because they appear to be harder to manipulate than the language they describe, and in particular, because they reintroduce the problem of α-equivalence which nominal abstract syntax was originally designed to solve.

The existence of permissive nominal terms demonstrates that in fact, the apparent problem is just an artefact of the way matters were set up in [40]. We *can* quotient permissive nominal terms by α-equivalence. In fact we can also use nominal inductive reasoning principles to reason on permissive nominal terms [22, 15].

Turning to the issue of fresh atoms; in the world of finite freshness contexts based on [40], we may always extend freshness contexts if we want more fresh atoms. This is made formal for example by $(\mathbf{fr})$ in [21, Figure 2] and $(\mathbf{Tfr})$ in [24, Figure 1].

We can observe the effects of this solution to the problem of generating fresh atoms, by considering the mathematics in [24]. $(\mathbf{Tfr})$ is not a syntax-directed rule and this complicates case-analysis on derivations. Subsection 5.1 of [24] is devoted to controlling this

[2] An similar idea is behind the notion of freshness contexts with *sufficient freshnesses* in [12, Subsection 3.1].

issue. Further, since freshness contexts may change during a proof, it is necessary to account for these changes in statements and proofs of results; the results in Subsection 5.3 of [24] therefore contain an existential quantification over 'sufficiently freshened' freshness contexts.

In fact we should recognise what happens in [24] because it is typical of something we have seen before. A 'state' of 'generated atoms' is being carried by the freshness context and threaded through all the proofs. This is a situation familiar from implementing languages with binding; we may need to keep track of the 'generated fresh atoms' so that when we generate a new one we can easily pick an 'even fresher atom'. We thread through the program a 'state' of 'generated atoms' (see for instance the 'fresh monad' in Cheney's FreshLib [5]). This is undesirable if it can be avoided, because it interferes with the state-free style of functional programming, forces us to program more sequentially, and it complicates code.[3] This is what motivated FreshML [38, 39].

The paper [24] is not an isolated case. Similar issues have arisen in proofs on the two-level lambda-calculus [23] and in other work [19, 18, 20]. Furthermore, the 'fresh atom' issue is not restricted to proofs:

A prototype implementation of permissive nominal terms and their unification by the third author [33] has a direct mechanism for generating a fresh name for any permissive term. Even in a pure functional language like Haskell, no monadic programming is needed just to generate fresh names.

Calvès recently wrote the Haskell Nominal Toolkit (HNT) [2], which provides efficient implementations of several algorithms on nominal terms. The HNT provides an elegant programming interface for manipulating nominal terms, and any efficient implementation of permissive nominal terms would also likely expose a similar API to the programmer.

However, Calvès' underlying model is the nominal term and this is reflected in the types of programs written with the HNT. Whereas core functions in the implementation of permissive nominal terms (e.g. unification and alpha-equivalence checks) are pure, the types of their HNT counterparts are heavily monadic, and explicit freshness contexts must also be passed around.

As an example, we compare the types (1) of the alpha-equivalence check function from the HNT and (2) of the implementation of permissive nominal terms:[4]

$$
\begin{array}{lll}
(1) & \texttt{alpha'check} & :: \ (\texttt{Show}\ t,\ \texttt{Eq}\ t,\ \texttt{Ord}\ a,\ \texttt{Ord} v) \Rightarrow \texttt{FrsCtxt}\ v\ a \rightarrow \\
 & & \texttt{Term}\ a\ t\ v \rightarrow \texttt{Term}\ a\ t\ v \rightarrow \\
 & & \texttt{CS}\ r\ (\texttt{ExtB}\ l\ e\ (\texttt{ErrorT}\ [\texttt{Char}]))\ m\ () \\
(2) & \texttt{aeq} & :: \ (\texttt{Eq}\ a, \texttt{Permissive}\ b) \Rightarrow \texttt{Term}\ a\ b \rightarrow \texttt{Term}\ a\ b\ \rightarrow \texttt{Bool}
\end{array}
$$

To us, the 'permissive' type seems shorter and clearer. An implementation of the HNT, based on permissive nominal terms, would likely present less daunting types to

[3] Another layer of complexity arises. Some natural definitions rely on a sufficiently large supply of fresh atoms being available. Even if we can always extend the freshness context, for a *fixed* freshness context there may be inputs for which the function cannot be defined, and so is partial. This happens for example to the *canonical form* function in Subsection 5.3 of [24].

[4] See the documentation for module `Nominal.Matching` at http://www.dcs.kcl.ac.uk/pg/calves/hnt/doc/Nominal-Matching.html and module `Terms.Terms` in the permissive nominal terms implementation source code, available at http://www.macs.hw.ac.uk/~dpm8/permissive/

the programmer, and removing the burden of handling freshness contexts would likely make code shorter and neater.

1.4. *Permissive nominal terms in this paper*

Permissive nominal terms are designed to deal with these issues. Their theory of α-equivalence and 'fresh atom of' restore the good features of nominal abstract syntax without losing any expressivity or computational properties.

In fact, permissive nominal terms are 'best possible' in a certain sense: in [22, 15] we demonstrate how the atoms-abstraction construction from [25] can be applied directly to permissive nominal terms syntax. That is, in [22, 15] we show that it is possible to take $[a]r$ to be *literally* the Gabbay-Pitts atoms-abstraction of a in the set that is the abstract syntax tree r, so that α-equivalence is *literal* identity, and the definition of permissive nominal terms syntax (Definition 2.6) becomes a nominal abstract syntax style inductive datatype of syntax-with-binding.

In this paper we introduce permissive nominal terms. We study their theory of unification. Given any new syntax, it is important to connect it to existing denotations. Therefore, we relate permissive nominal terms and their unification to nominal unification, and to unification of higher-order patterns. Denotations in nominal sets will be the topic of a separate manuscript [15] (some elements are also in [22]).

The contributions are as follows:

- We introduce permissive nominal terms Definitions 2.6, along with theories of α-equivalence and freshness with the 'always fresh' and 'always rename' properties discussed above (Corollaries 2.14 and 2.15).
- It may look like permissive nominal terms are infinite, because the S in X^S in Definition 2.6 is an infinite set. In Remark 2.7 we mention that they are just as 'finite', and computable-upon, as nominal terms.
- We develop a notion of unification (Definition 6.6). We use a simplified notion of unifier (Definition 3.9) which is more like the notion of unifier from first- and higher-order unification in that it is based just on a substitution, as compared to the notion of unifier used in [40], which is not.
- We make precise the connection between nominal terms and permissive nominal terms by translating nominal terms, nominal unification problems, and their solutions, to the permissive context (Definition 4.6). We verify that no expressivity is lost (Theorem 4.16).
- We connect permissive nominal terms and higher order patterns [32, 31] (Definition 8.3) by translating permissive nominal terms, unification problems, and their solutions, to a very general notion of untyped pattern unification problems and their solutions (Definition 9.1). We also prove that this translation is 'best possible' and 'complete' in senses which we make formal (Theorems 8.14 and Theorems 9.16 and 9.30).

1.5. *Map of the paper*

The paper is organised as follows:

We introduce permissive nominal terms in Section 2. Notable results are the 'always fresh' and 'always rename' properties for terms (Corollaries 2.14 and 2.15 respectively).

In Section 3 we introduce technical definitions and results which will prove useful in the development of the unification algorithm, including permissive nominal terms substitution (Definition 3.1), unification problems and their solutions (Definition 3.9).

We clarify the relationship between nominal and permissive nominal terms in Section 4, by injecting nominal terms into permissive nominal terms (Definition 4.6 onwards). We also elucidate the relationship between solutions of nominal unification problems, based on subsititution+freshness, and solutions of permissive nominal unification problems, based on substitution alone (Definition 4.12).

In Sections 5 and 6 we present an algorithm to compute most general solutions to permissive nominal unification problems (Definition 6.6). We prove it correct in Theorem 6.25.

We define λ-terms syntax (Definition 7.1) in Section 7, and higher-order patterns (Definition 7.19). Section 8 translates from permissive nominal term syntax to λ-term syntax (Definition 8.3 onwards). We prove that this translation is 'best possible', in a suitable sense which we make formal (Theorem 8.14).

Section 9 relates the solutions of higher-order pattern unification with solutions of permissive nominal unification (Definition 9.1). We show that the instantiation ordering, hence the property of solutions being 'more general', is preserved by the translation (Corollary 9.5). We prove a soundness and weak completeness result (Theorem 9.16). Finally, we refine this to a more complex but more powerful completeness result (Theorem 9.30).

In Section 10, we conclude and suggest ideas for future work.

2. Permissive nominal terms

We set up the syntax of permissive nominal terms (Definition 2.6) and their notion of α-equivalence $=_\alpha$, which does not require a freshness context (Definition 2.13). We prove that permissive nominal terms have the 'always fresh' and 'always rename' properties (Corollaries 2.14 and 2.15). Finally, we verify that $=_\alpha$ is an equivalence relation; this mirrors the result for nominal terms [40, Theorem 2.11], though the proof-method is based on [10, Subsection 3.2]. Where we omit proofs they are routine (or see a technical report [8], or [10]).

Note the *two* notions of 'free variables of'; $fa(r)$ is the free atoms in r, and $fV(r)$ is the free unknowns in r (Definitions 2.11 and 2.12). This reflects the two-level structure of nominal terms familiar from previous work [40].

Definition 2.1. Fix two disjoint countably infinite sets $\mathbb{A}^<$ and $\mathbb{A}^>$ of **atoms** and write

$$\mathbb{A} = \mathbb{A}^< \uplus \mathbb{A}^>.$$

(Here $\uplus$ denotes disjoint set union) $a, b, c, \ldots$ will range over *distinct* elements of $\mathbb{A}$ (we call this the **permutative convention**).

Definition 2.2. Define $\mathcal{P}$ by

$$\mathcal{P} = \{(\mathbb{A}^< \setminus A) \cup B \mid A \subseteq \mathbb{A}^<,\ B \subseteq \mathbb{A}^>,\ A, B \text{ finite}\}.$$

Call elements of $\mathcal{P}$ **permission sets**. S, S', T will range over permission sets.

Call $S \subseteq \mathbb{A}$ **co-infinite** when $\mathbb{A} \setminus S$ is infinite. $\mathcal{P}$ is a set of infinite, co-infinite sets of atoms.[5]

Definition 2.3. For each permission set S fix a disjoint countably infinite set of **unknowns** of sort S. X^S, Y^S, Z^S, will range over distinct unknowns of sort S. If $S \neq S'$ then there is no particular connection between X^S and $X^{S'}$. $\mathcal{V}$ will range over finite sets of unknowns (we use this from Section 5 onwards).

Definition 2.4. Suppose f is a function from atoms to atoms. Define $nontriv(f)$ by:

$$nontriv(f) = \{a \mid f(a) \neq a\}$$

This has also been called the *support* of π [25].

Definition 2.5. A (finite) **permutation** is a bijection on atoms such that $nontriv(\pi)$ is finite. π and π' will range over finite permutations.

Write $\pi \circ \pi'$ for the **composition** of π and π' (so $(\pi \circ \pi')(a) = \pi(\pi'(a))$). Write id for the **identity** permutation (so $id(a) = a$ always). Write $(a\ b)$ for the **swapping** permutation that swaps a and b.

Definition 2.6. Fix a set of **term-formers**. $\mathsf{f}, \mathsf{g}, \mathsf{h}$ will range over distinct term-formers.

Define **(permissive nominal) terms** by:

$$r, s, t, \ldots ::= a \mid \mathsf{f}(r, \ldots, r) \mid [a]r \mid \pi \cdot X^S$$

We write $\equiv$ for syntactic identity; $r \equiv s$ when r and s denote identical terms. Note that X^S (the unknown) is not a term, however $\pi \cdot X^S$ is a term and in particular $id \cdot X^S$ is a term, which we may write as X^S.

Remark 2.7. Permissive-nominal terms are finite. They are finitely branching finitely deep trees.

Equality of permissive-nominal terms may be calculated in finite time. Permission sets trivially admit a finite representation as the pair of finite sets A and B in Definition 2.2.

In an implementation of the algorithms in this paper by Mulligan [33], atoms are implemented concretely as numbers. $\mathbb{A}^<$ is identified with the even numbers. Permission sets are represented finitely as their finite deviation from $\mathbb{A}^<$.

Another, quite elegant, presentation is possible using *exclusive or*; see Remark 2.23.

[5] Other versions of $\mathcal{P}$ are possible.

We believe that a sufficient property for $\mathcal{P}$ is that it be: *co-infinitely down-closed* (so if $S \in \mathcal{P}$ and $S' \subseteq S$ is co-infinite, then $S' \in \mathcal{P}$); and such that for any finite $\{S_1, \ldots, S_n\} \subseteq \mathcal{P}$, the union $S_1 \cup \ldots \cup S_n$ is co-infinite (it does not have to be in $\mathcal{P}$).

The first property is sufficient to build the permission set in (1). The second property is sufficient to guarantee Corollary 2.14. This is similar in spirit to the notion of *support ideal* from [6, Definition 4.1].

Remark 2.8. See Section 4 for a comparison between permissive nominal terms of Definition 2.6 and 'ordinary' nominal terms [40]. Atoms represent *variable symbols*; term-formers *functions*; unknowns *meta-variables*; abstraction $[a]r$ *binding*; and $\pi \cdot X^S$ a meta-variable with a suspended substitution, like '$t[y/x]$'. For example, suppose term-formers app and λ:

– $\mathsf{app}(a, b)$ can represent 'xy' (x applied to y).
– $\mathsf{app}(\mathsf{lam}([a]a), b)$ can represent '$(\lambda x.x)y$' (identity applied to y).
– $\lambda([a]X^S)$ can represent '$\lambda x.t$' if $a \in S$, and '$\lambda x.t$, where $x \notin fv(t)$' if $a \notin S$.

Definition 2.9. Define a **permutation action** by:

$$\begin{array}{cc} \pi \cdot a \equiv \pi(a) & \pi \cdot (\mathsf{f}(r_1, \ldots, r_n)) \equiv \mathsf{f}(\pi \cdot r_1, \ldots, \pi \cdot r_n) \\ \pi \cdot [a]r \equiv [\pi(a)](\pi \cdot r) & \pi \cdot (\pi' \cdot X^S) \equiv (\pi \circ \pi') \cdot X^S \end{array}$$

Definition 2.10. If $S \subseteq \mathbb{A}$, define the **pointwise** action by:

$$\pi \cdot S = \{\pi(a) \mid a \in S\}$$

Definition 2.11. Define **free atoms** $fa(r)$ by:

$$\begin{array}{cc} fa(a) = \{a\} & fa(\mathsf{f}(r_1, \ldots, r_n)) = \bigcup_{1 \leq i \leq n} fa(r_i) \\ fa([a]r) = fa(r) \backslash \{a\} & fa(\pi \cdot X^S) = \pi \cdot S \end{array}$$

Note that $fa(\pi \cdot X^S) = \pi \cdot S$. Thus, an intuition for $fa(r)$ is 'the free atoms we can have after instantiation'.

Definition 2.12. Define **free unknowns** $fV(r)$ by:

$$\begin{array}{cc} fV(a) = \varnothing & fV(\mathsf{f}(r_1, \ldots, r_n)) = fV(r_1) \cup \cdots \cup fV(r_n) \\ fV([a]r) = fV(r) & fV(\pi \cdot X^S) = \{X^S\} \end{array}$$

Definition 2.13. If $A \subseteq \mathbb{A}$, define $\pi|_A$, π **restricted to** A, to be:

$$\begin{array}{lll} \pi|_A(a) = \pi(a) & & \text{when } a \in A \\ \pi|_A(a) & \text{undefined} & \text{when } a \in \mathbb{A} \setminus A \end{array}$$

Define α**-equivalence** $=_\alpha$ inductively by the rules in Figure 1.

Corollaries 2.14 and 2.15 are properties that 'ordinary syntax' has, that nominal terms do not have, and that permissive nominal terms recover; we can always choose a fresh variable, and we can always α-rename with it.

Corollary 2.14. *For any $r_1, \ldots, r_n$ there exist infinitely many b such that $b \notin \bigcup\{fa(r_i) \mid 1 \leq i \leq n\}$.*

$$\frac{}{a =_\alpha a}\,(=_\alpha\mathbf{aa}) \qquad \frac{r_1 =_\alpha s_1 \quad \cdots \quad r_n =_\alpha s_n}{\mathsf{f}(r_1,\ldots,r_n) =_\alpha \mathsf{f}(s_1,\ldots,s_n)}\,(=_\alpha\mathsf{f}) \qquad \frac{r =_\alpha s}{[a]r =_\alpha [a]s}\,(=_\alpha[\mathbf{a}])$$

$$\frac{(b\ a)\cdot r =_\alpha s \qquad (b \notin fa(r))}{[a]r =_\alpha [b]s}\,(=_\alpha[\mathbf{b}]) \qquad \frac{(\pi|_S = \pi'|_S)}{\pi \cdot X^S =_\alpha \pi' \cdot X^S}\,(=_\alpha\mathbf{X})$$

Figure 1: Derivable α-equivalence on permissive nominal terms.

Proof. Define $atoms(r)$ inductively by:

$$atoms(a) = \{a\} \qquad atoms(\mathsf{f}(r_1,\ldots,r_n)) = atoms(r_1) \cup \ldots \cup atoms(r_n)$$
$$atoms([a]r) = atoms(r) \cup \{a\} \qquad atoms(\pi \cdot X^S) = nontriv(\pi)$$

It is not hard to prove by induction on term syntax that $fa(r_i) \subseteq atoms(r_i) \cup \bigcup\{S \mid X^S \in fV(r_i)\}$ for $1 \leq i \leq n$. The syntax of r_i is finite so $atoms(r_i)$ is finite, and also $fV(r_i)$ is finite. It follows that $\bigcup\{S \mid X^S \in fV(r_i)$ for some $i\}$ is co-infinite. The result follows. □

Later on, we will often need to say 'choose an atom fresh' (see for example Definition 4.14). When we do this, we are using Corollary 2.14.

Corollary 2.15 expresses that we can always α-rename:

Corollary 2.15. *For any r and a there exists infinitely many fresh b (so $b \notin fa(r)$) such that for some s, $[a]r =_\alpha [b]s$.*

Proof. Immediate, by Corollary 2.14 and $(=_\alpha[\mathbf{b}])$. □

Our changes do not affect basic results about nominal terms [40]; the proofs of the following lemmas are by routine inductions (see [8] for details):

Lemma 2.16. 1. $id \cdot r \equiv r$
2. $\pi' \cdot (\pi \cdot r) \equiv (\pi' \circ \pi) \cdot r$

Lemma 2.17. $\pi \cdot fa(r) = fa(\pi \cdot r)$.

Lemma 2.18. *If $r =_\alpha s$ then $\pi \cdot r =_\alpha \pi \cdot s$.*

Lemma 2.19. *If $r =_\alpha s$ then $fa(r) = fa(s)$.*

Lemma 2.20. *If $\pi|_{fa(r)} = \pi'|_{fa(r)}$ then $\pi \cdot r =_\alpha \pi' \cdot r$.*

Proposition 2.21. *$=_\alpha$ is transitive, reflexive, and symmetric.*

Proof. See Appendix Appendix A. We use Lemmas 2.16, 2.17, 2.18, 2.19 and 2.20. □

Remark 2.22. The Fraenkel-Mostowski sets model used in the work which introduced nominal techniques [25] famously does not admit sets like S and T, because they do not have finite support. It is not an issue in this paper because we are not concerned with representing permissive nominal syntax-up-to-binding. Permissive nominal syntax-up-to-binding can be constructed though; see [22, 15]. See also generalisations of nominal sets by the second author [11, 13] or by Cheney ([3, Section 3], or [6]).

Remark 2.23. Define $S \,\Delta\, T$ (the **exclusive or**) by

$$S \,\Delta\, T = (S \setminus T) \cup (T \setminus S).$$

It is a fact that for every $S \in \mathcal{P}$, the set $S \,\Delta\, \mathbb{A}^<$ is a finite set of atoms, and it is the unique finite set such that $\mathbb{A}^< \,\Delta\, (S \,\Delta\, \mathbb{A}^<) = S$. This gives a nice finite representation of permission sets, alternative to the two mentioned in Remark 2.7.

3. Substitutions, problems, and solutions

3.1. Substitutions

The purpose of an unknown X^S is to represent an 'unknown term/unknown element'. We therefore define a substitution action for unknowns. Consistent with nominal terms, substitution for unknowns is capturing for abstraction by atoms.

Definition 3.1. A **substitution** θ is a function from unknowns to terms such that $fa(\theta(X^S)) \subseteq S$ always (so S in X^S describes the 'permission' we have to instantiate X^S, namely to terms with free atoms in S). θ, θ', θ_1, θ_2, will range over substitutions.

Write id for the **identity** substitution mapping X^S to $id \cdot X^S$ always. It will always be clear whether id means the identity substitution or permutation. Suppose $fa(t) \subseteq S$. Write $[X^S{:=}t]$ for the substitution such that

$$[X^S{:=}t](X^S) \equiv t \quad \text{and} \quad [X^S{:=}t](Y^T) \equiv id \cdot Y^T \quad \text{for all other } Y^T.$$

'$fa(\theta(X^S)) \subseteq S$' looks absent in nominal terms theory ([40, Definition 2.13], [10, Definition 4]), yet it is there: see the conditions '$\nabla' \vdash \theta(\nabla)$' in Lemma 2.14, and '$\nabla \vdash a\#\theta(t)$' in Definition 3.1 of [40]. More on this in Section 4.

Definition 3.2. Define a **substitution action** on terms by:

$$\begin{array}{cc} a\theta \equiv a & \mathsf{f}(r_1,\ldots,r_n)\theta \equiv \mathsf{f}(r_1\theta,\ldots,r_n\theta) \\ ([a]r)\theta \equiv [a](r\theta) & (\pi{\cdot}X^S)\theta \equiv \pi{\cdot}\theta(X^S) \end{array}$$

Note that $X^S\theta$ means 'θ acting on $id \cdot X^S$'; $\theta(X^S)$ means 'the value of function θ at X^S'.

Lemma 3.3. $fa(r\theta) \subseteq fa(r)$.

Proof. See Appendix Appendix A. We use Lemma 2.17. □

Lemma 3.4. $\pi \cdot (r\theta) \equiv (\pi \cdot r)\theta$.

Proof. By induction on r. □

Lemma 3.5. *If* $\theta_1(X^S) =_\alpha \theta_2(X^S)$ *for all* $X^S \in fV(r)$*, then* $r\theta_1 =_\alpha r\theta_2$.

Proof. By induction on r.

- The cases a and $\mathsf{f}(r_1, \ldots, r_n)$ are straightforward.
- The case $[a]r$. Suppose $\theta_1(X^S) =_\alpha \theta_2(X^S)$ for every $X^S \in fV([a]r)$. $fV([a]r) = fV(r)$ so by inductive hypothesis $r\theta_1 =_\alpha r\theta_2$. By $(=_\alpha[\mathbf{a}])$ also $[a](r\theta_1) =_\alpha [a](r\theta_2)$. The result follows by Definition 3.2.
- The case $\pi \cdot X^S$. By assumption, $\theta_1(X^S) =_\alpha \theta_2(X^S)$. Using Lemma 2.18, $\pi \cdot (\theta_1(X^S)) =_\alpha \pi \cdot (\theta_2(X^S))$. By Definition 3.2 $(\pi \cdot X^S)\theta_1 =_\alpha (\pi \cdot X^S)\theta_2$, as required.

□

Lemma 3.6. *If* $r =_\alpha s$ *then* $r\theta =_\alpha s\theta$.

Proof. By induction on the derivation of $r =_\alpha s$. We consider one case:

- The case $(=_\alpha[\mathbf{b}])$. Suppose $(b\ a) \cdot r =_\alpha s$ and $b \notin fa(r)$. Then $((b\ a) \cdot r)\theta =_\alpha s\theta$ by assumption. By Lemma 3.4, $(b\ a) \cdot (r\theta) =_\alpha s\theta$. By Lemma 3.3, $b \notin fa(r\theta)$, therefore $[a](r\theta) =_\alpha [b](s\theta)$ by $(=_\alpha[\mathbf{b}])$. By Definition 3.1, $[a](r\theta) \equiv ([a]r)\theta$, and the result follows.

□

Definition 3.7. Define **composition** $\theta_1 \circ \theta_2$ by $(\theta_1 \circ \theta_2)(X^S) \equiv (\theta_1(X^S))\theta_2$.

Lemma 3.8. $(r\theta)\theta' \equiv r(\theta \circ \theta')$.

Proof. By induction on r. We consider one case:

- The case $\pi \cdot X^S$

$$\begin{array}{rll} (\pi \cdot X^S)(\theta \circ \theta') \equiv \pi \cdot (\theta \circ \theta')(X^S) & & \text{Definition 3.2} \\ \equiv \pi \cdot (\theta(X^S)\theta') & & \text{Definition 3.7} \\ \equiv (\pi \cdot \theta(X^S))\theta' & & \text{Lemma 3.4} \\ \equiv ((\pi \cdot X^S)\theta)\theta' & & \text{Lemma 3.4} \end{array}$$

□

3.2. Unification problems, and solutions

This is a brief subsection, but it is useful: A solution to Pr 'makes the equalities valid', as for first- and higher-order unification. This simplifies the nominal unification notion of solution (Definition 4.11 or [40, Definition 3.1]) based on 'a substitution + a freshness context'. We prove results about these definitions in Sections 4 (connection with nominal unification) and 6 (unification algorithm).

Definition 3.9. An **equality** is a pair $r \stackrel{?}{=} s$. A **problem** Pr is a finite multiset of equalities. Define $Pr\theta$ by:

$$Pr\theta = \{r\theta \stackrel{?}{=} s\theta \mid r \stackrel{?}{=} s \in Pr\}$$

Say that θ **solves** Pr when $r \stackrel{?}{=} s \in Pr$ implies $r\theta =_\alpha s\theta$. Write $Sol(Pr)$ for the set of solutions to Pr. Call Pr **solvable** when $Sol(Pr)$ is non-empty.

4. Relation to nominal terms

In Subsection 3.2 we stated what a permissive nominal unification problem is, and what a solution to it is. We now make precise a mathematical sense in which nominal terms and their unification can be considered a subsystem of permissive nominal terms and their unification.

We recall the notions of nominal term and nominal unification problem (Definitions 4.2 and 4.11). We translate from the 'nominal world' to the 'permissive nominal world' (Definition 4.6). Theorem 4.9 expresses how this translation is sound and complete for respective notions of α-equivalence. Theorem 4.16 then shows that furthermore, solutions to unification problems are preserved 1-1 across the translation.

Recall from the Introduction that in nominal terms we often to enrich the freshness context. An interesting feature of Definition 4.6 is how it maps nominal terms to permissive nominal terms with free atoms in $\mathbb{A}^<$, which is an infinite, co-infinite set of atoms. One way to view the interpretation of Definition 4.6 is therefore this: $\mathbb{A}^<$ is 'the atoms we had available so far' (any other permission set would do as well) and $\mathbb{A}^>$ is 'the atoms with which we will extend the freshness context, in the future'. Both these sets are infinite, and syntax is finite, so it is not absolutely necessary to explicitly separate them: permissive nominal terms do this, for each fixed permission set S; nominal terms do not.

4.1. Alpha-equivalence between nominal and permissive nominal terms

Definition 4.1. Fix a countably infinite set of **nominal atoms**, $\dot{\mathbb{A}}$. $\dot{a}, \dot{b}, \dot{c}, \ldots$ will range over distinct nominal atoms.

Fix a bijection ι between $\dot{\mathbb{A}}$ and any permission set. For concreteness we will suppose it is $\mathbb{A}^<$ from Definition 2.2 but any permission set will do as well.

Fix a countably infinite set of **nominal unknowns**. $\dot{X}, \dot{Y}, \dot{Z}, \ldots$ will range over distinct nominal unknowns. A **nominal permutation** is a bijection $\dot{\pi}$ on $\dot{\mathbb{A}}$ such that $nontriv(\dot{\pi})$ is finite. $\dot{\pi}, \dot{\pi}', \dot{\pi}'', \ldots$ will range over permutations.

Write $\dot{\pi}^{-1}$ for the inverse of $\dot{\pi}$, $\dot{id}$ for the identity permutation, and $\dot{\pi} \circ \dot{\pi}'$ for function composition, as is standard. For example, $(\dot{\pi} \circ \dot{\pi}')(\dot{a}) = \dot{\pi}(\dot{\pi}'(\dot{a}))$

Definition 4.2. Define **nominal terms** by:

$$\dot{r}, \dot{s}, \dot{t} ::= \dot{a} \mid \dot{\pi} \cdot \dot{X} \mid [\dot{a}]\dot{r} \mid \mathsf{f}(\dot{r}, \ldots, \dot{r})$$

Definition 4.3. Define a **permutation action** on nominal terms by:

$$\begin{array}{ll} \dot{\pi} \cdot \dot{a} \equiv \dot{\pi}(\dot{a}) & \dot{\pi} \cdot \mathsf{f}(\dot{r}_1, \ldots, r_n) \equiv \mathsf{f}(\dot{\pi} \cdot \dot{r}_1, \ldots, \dot{\pi} \cdot r_n) \\ \dot{\pi} \cdot [\dot{a}]\dot{r} \equiv [\dot{\pi}(\dot{a})](\dot{\pi} \cdot \dot{r}) & \dot{\pi} \cdot (\dot{\pi}' \cdot \dot{X}) \equiv (\dot{\pi} \circ \dot{\pi}') \cdot \dot{X} \end{array}$$

Write $\equiv$ for syntactic identity. f ranges over term-formers (Definition 2.1).

Definition 4.4. A **freshness** is a pair $\dot{a}\#\dot{r}$. A **freshness context** is a finite set of freshnesses of the form $\dot{a}\#\dot{X}$. Define **derivable freshness** on nominal terms by the rules in Figure 2.

$$\frac{}{\Delta \vdash \dot{a}\#\dot{b}}\,(\#\dot{\mathbf{b}}) \qquad \frac{\Delta \vdash \dot{a}\#\dot{r}_i \quad (1 \le i \le n)}{\Delta \vdash \dot{a}\#\mathsf{f}(\dot{r}_1,\ldots,\dot{r}_n)}\,(\#\mathsf{f}) \qquad \frac{}{\Delta \vdash \dot{a}\#[\dot{a}]\dot{r}}\,(\#[\dot{\mathbf{a}}])$$

$$\frac{\Delta \vdash \dot{a}\#\dot{r}}{\Delta \vdash \dot{a}\#[\dot{b}]\dot{r}}\,(\#[\dot{\mathbf{b}}]) \qquad \frac{(\dot{\pi}^{\text{-}1}(\dot{a})\#\dot{X} \in \Delta)}{\Delta \vdash \dot{a}\#\dot{\pi}\cdot\dot{X}}\,(\#\dot{\mathbf{X}})$$

Figure 2: Derivable freshness on nominal terms

$$\frac{}{\Delta \vdash \dot{a} = \dot{a}}\,(=\dot{\mathbf{a}}) \qquad \frac{\Delta \vdash \dot{r}_i = \dot{s}_i \quad (1 \le i \le n)}{\Delta \vdash \mathsf{f}(\dot{r}_1,\ldots,\dot{r}_n) = \mathsf{f}(\dot{s}_1,\ldots,\dot{s}_n)}\,(=\mathsf{f}) \qquad \frac{\Delta \vdash \dot{r} = \dot{s}}{\Delta \vdash [\dot{a}]\dot{r} = [\dot{a}]\dot{s}}\,(=[\dot{\mathbf{a}}])$$

$$\frac{\Delta \vdash (\dot{b}\;\dot{a})\cdot\dot{r} = \dot{s} \quad \Delta \vdash \dot{b}\#\dot{r}}{\Delta \vdash [\dot{a}]\dot{r} = [\dot{b}]\dot{s}}\,(=[\dot{\mathbf{b}}]) \qquad \frac{(\dot{a}\#\dot{X} \in \Delta \text{ for every } \dot{\pi}(\dot{a}) \neq \dot{\pi}'(\dot{a}))}{\Delta \vdash \dot{\pi}\cdot\dot{X} = \dot{\pi}'\cdot\dot{X}}\,(=\dot{\mathbf{X}})$$

Figure 3: Derivable equality on nominal terms

Definition 4.5 repeats [40, Figure 2], up to differences in presentation:

Definition 4.5. An **equality** is a pair $\dot{r} = \dot{s}$. Define **derivable equality** on nominal terms by the rules in Figure 3.

Definition 4.6. Define a mapping $[\![\dot{\pi}]\!]$ from nominal permutations to permissive nominal permutations by:

$$\begin{array}{ll} [\![\dot{\pi}]\!](\iota(\dot{a})) = \iota(\dot{\pi}(\dot{a})) & \\ [\![\dot{\pi}]\!](c) = c & \text{all } c \in \mathbb{A}^{>} \end{array}$$

Define an **interpretation** $[\![\dot{r}]\!]_\Delta$ by:

$$\begin{array}{rll} [\![\dot{a}]\!]_\Delta \equiv \iota(\dot{a}) & & \\ [\![\mathsf{f}(\dot{r}_1,\ldots,\dot{r}_n)]\!]_\Delta \equiv \mathsf{f}([\![\dot{r}_1]\!]_\Delta,\ldots,[\![\dot{r}_n]\!]_\Delta) & & \\ [\![[\dot{a}]\dot{r}]\!]_\Delta \equiv [\iota(\dot{a})][\![\dot{r}]\!]_\Delta & & \\ [\![\dot{\pi}\cdot\dot{X}]\!]_\Delta \equiv [\![\dot{\pi}]\!]\cdot X^S & \text{where } S = \mathbb{A}^{<} \setminus \{\iota(\dot{a}) \mid \dot{a}\#\dot{X} \in \Delta\} & \end{array}$$

Here, we make a fixed but arbitrary choice of X^S for each $\dot{X}$, injectively so that $[\![\dot{X}]\!]_\Delta$ and $[\![\dot{Y}]\!]_\Delta$ are always distinct.

$[\![\dot{r}]\!]_\Delta$ commutes with permutation and it preserves and reflects freshness:

Lemma 4.7. $[\![\dot{\pi}]\!]\cdot[\![\dot{r}]\!]_\Delta \equiv [\![\dot{\pi}\cdot\dot{r}]\!]_\Delta$

Proof. By induction on $\dot{r}$. □

Lemma 4.8. *$\iota(\dot{a}) \notin fa([\![\dot{r}]\!]_\Delta)$ if and only if $\Delta \vdash \dot{a}\#\dot{r}$.*

Proof. See Appendix Appendix A. □

Theorem 4.9 states that α-equivalent nominal-terms-in-context map precisely to α-equivalent permissive nominal terms:

Theorem 4.9. *$[\![\dot{r}]\!]_\Delta =_\alpha [\![\dot{s}]\!]_\Delta$ if and only if $\Delta \vdash \dot{r} = \dot{s}$.*

Proof. See Appendix Appendix A. We use Lemmas 4.7 and 4.8. □

4.2. Substitutions and solutions between nominal and permissive nominal unification problems

Definition 4.10. A **substitution** $\dot{\theta}$ is a function from nominal unknowns to nominal terms such that $\{\dot{X} \mid \dot{\theta}(\dot{X}) \not\equiv \dot{id} \cdot \dot{X}\}$ is finite. $\dot{\theta}, \dot{\theta}', \dot{\theta}'', \ldots$ will range over nominal substitutions.

Write $\dot{id}$ for the **identity**, mapping $\dot{X}$ to $\dot{id} \cdot \dot{X}$.

Define a **substitution action** on nominal terms by:

$$\dot{a}\dot{\theta} \equiv \dot{a} \quad \mathsf{f}(\dot{r}_1, \ldots, \dot{r}_n)\dot{\theta} \equiv \mathsf{f}(\dot{r}_1\dot{\theta}, \ldots, \dot{r}_n\dot{\theta}) \quad ([\dot{a}]\dot{r})\dot{\theta} \equiv [\dot{a}](\dot{r}\dot{\theta}) \quad (\dot{\pi}\cdot\dot{X})\dot{\theta} \equiv \dot{\pi}\cdot\dot{\theta}(\dot{X})$$

Definition 4.11. A **unification problem** $\dot{Pr}$ is a finite multiset of freshnesses and equalities. A **solution** to $\dot{Pr}$ is a pair $(\Delta, \dot{\theta})$ such that $\Delta \vdash \dot{a}\#\dot{r}\dot{\theta}$ for every $\dot{a}\#\dot{r} \in \dot{Pr}$, and $\Delta \vdash \dot{r}\theta = \dot{s}\theta$ for every $\dot{r} = \dot{s} \in \dot{Pr}$. This follows [40, Definition 3.1].

Definition 4.12. We extend the interpretation of Definition 4.6 to solutions of nominal unification problems by:

$$[\![(\Delta, \dot{\theta})]\!](X^S) \equiv [\![\dot{\theta}(X)]\!]_\Delta \text{ if } id\cdot X^S \equiv [\![X]\!]_\Delta \qquad [\![(\Delta, \dot{\theta})]\!](Y^T) \equiv id\cdot Y^T \text{ otherwise}$$

Lemma 4.13. *$[\![\dot{r}]\!]_\Delta [\![(\Delta, \dot{\theta})]\!] \equiv [\![\dot{r}\dot{\theta}]\!]_\Delta$.*

Proof. See Appendix Appendix A. □

Definition 4.14. Define $[\![\dot{Pr}]\!]_\Delta$ by mapping $\dot{r} = \dot{s}$ to $[\![\dot{r}]\!]_\Delta \stackrel{?}{=} [\![\dot{s}]\!]_\Delta$ and mapping $\dot{a}\#\dot{r}$ to $(b\ \iota(\dot{a})) \cdot [\![\dot{r}]\!]_\Delta \stackrel{?}{=} [\![\dot{r}]\!]_\Delta$, for some choice of fresh b (so $b \notin fa([\![\dot{r}]\!]_\Delta)$; in fact, it suffices to choose some $b \notin \mathbb{A}^<$).

Lemma 4.15. *Suppose $b \notin fa(r)$. Then $a \notin fa(r)$ if and only if $(b\ a) \cdot r =_\alpha r$.*

Proof. See Appendix Appendix A. We use Lemmas 2.16 and 2.17, and Proposition 2.21. □

No solutions go missing, moving from the nominal to the permissive nominal world:

Theorem 4.16. *$(\Delta, \dot{\theta})$ solves $\dot{Pr}$ if and only if $[\![(\Delta, \dot{\theta})]\!]$ solves $[\![\dot{Pr}]\!]_\Delta$.*

Proof. *If* $(\Delta,\dot\theta)$ *solves* $\dot Pr$ *then* $[\![(\Delta,\dot\theta)]\!]$ *solves* $[\![\dot Pr]\!]_\Delta$. Suppose $\Delta \vdash \dot r\dot\theta = \dot s\dot\theta$. Using Lemma 4.13 and Theorem 4.9, $[\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!] =_\alpha [\![\dot s]\!]_\Delta[\![(\Delta,\dot\theta)]\!]$.

Suppose $\Delta \vdash a\#\dot r\dot\theta$. Using Lemma 4.8, $\iota(\dot a) \notin fa([\![\dot r\dot\theta]\!]_\Delta)$. By Lemma 4.13, $\iota(\dot a) \notin fa([\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!])$. By Lemma 4.15, $(b\ \iota(\dot a))\cdot[\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!] =_\alpha [\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!]$, where b is fresh (see Definition 4.14). By Lemma 3.4, $((b\ \iota(\dot a))\cdot[\![\dot r]\!]_\Delta)[\![(\Delta,\dot\theta)]\!] =_\alpha [\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!]$. The result follows.

If $[\![(\Delta,\dot\theta)]\!]$ *solves* $[\![\dot Pr]\!]_\Delta$ *then* $(\Delta,\dot\theta)$ *solves* $\dot Pr$. Suppose that $[\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!] =_\alpha [\![\dot s]\!]_\Delta[\![(\Delta,\dot\theta)]\!]$. By Theorem 4.9, $\Delta \vdash r\theta = s\theta$.

Suppose $((b\ \iota(\dot a))\cdot[\![\dot r]\!]_\Delta)[\![(\Delta,\dot\theta)]\!] =_\alpha [\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!]$. By Lemma 3.4, $(b\ \iota(\dot a))\cdot[\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!] =_\alpha [\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!]$. By Lemma 4.15, $\iota(\dot a) \notin fa([\![\dot r]\!]_\Delta[\![(\Delta,\dot\theta)]\!])$. Using Lemma 4.13, $\iota(\dot a) \notin fa([\![\dot r\dot\theta]\!]_\Delta)$. By Lemma 4.8, $\Delta \vdash a\#\dot r\dot\theta$, and the result follows. □

5. Support inclusion problems

The freshness symbol $a\#r$ used in [25] and [40] is ambiguous. Do we mean
– 'a is not free in the syntax of r' or
– 'a is not in the support of the denotation of r'?
The first option can be called *intensional* or *syntactic* freshness; the second option can be called *extensional* or *semantic* freshness.

In nominal terms this question is slightly obscured because a 'free atoms of' function on terms is hard to define. In permissive nominal terms we can easily define a 'free atoms of' function; see Definition 2.11, $fa(r)$.

We have seen in Section 4 how nominal terms' notion of freshness '$a\#r$' translates to a syntactic freshness '$a \notin fa(r)$'.

Nominal terms unification solves equality and freshness problems within a single rewrite system. When we designed the permissive nominal terms unification algorithm, we solve these separately — one algorithm is described in this section, the other (for equalities) is described in Section 6. This is simply a design choice, but it is informed by the translation of $\#$ to a *syntactic* judgement, described above. For, in future it may be useful to consider unification modulo equational theories (and note that semantic freshness can be easily captured using equations; see [17, Theorem 5.5] or [21, Theorem 4.52]). In the presence of equational theories, because we have factored out fragment of the computation that checks '$a \notin fa(r)$', that fragment will remain modular and unaffected by the imposition of equality axioms.

Recall from Definition 3.1 that $fa(\theta(X^S)) \subseteq fa(X^S) = S$, and from Lemma 3.3 that instantiation must reduce the set of free atoms. We will exhibit an algorithm which, intuitively, solves the problem "please make $fa(r\theta) \subseteq T$ true" (Definition 5.9 and Lemma 5.7). In fact the algorithm calculates solutions that are most general, in a sense made formal in Theorem 5.20.

Next, in Section 6, we construct an algorithm to solve equality problems.

5.1. Simplification reduction and normal forms

Definition 5.1. A **support inclusion** is a pair $r \sqsubseteq T$ of a term and a permissions set. A **support inclusion problem** is a finite multiset of support inclusions; Inc will range over support inclusion problems. Call θ a **solution** to Inc when $fa(r\theta) \subseteq T$ for every $r \sqsubseteq T \in$

$$
\begin{array}{lrll}
(\sqsubseteq\mathbf{a}) & a \sqsubseteq T, \mathit{Inc} \Longrightarrow \mathit{Inc} & & (a \in T) \\
(\sqsubseteq\mathsf{f}) & \mathsf{f}(r_1,\ldots,r_n) \sqsubseteq T, \mathit{Inc} \Longrightarrow r_1 \sqsubseteq T, \ldots, r_n \sqsubseteq T, \mathit{Inc} & & \\
(\sqsubseteq[]) & [a]r \sqsubseteq T, \mathit{Inc} \Longrightarrow r \sqsubseteq T \cup \{a\},\ \mathit{Inc} & & \\
(\sqsubseteq\mathbf{X}) & \pi \cdot X^S \sqsubseteq T, \mathit{Inc} \Longrightarrow X^S \sqsubseteq \pi^{-1} \cdot T,\ \mathit{Inc} & & (S \not\subseteq \pi^{-1} \cdot T,\ \pi \neq \mathit{id}) \\
(\sqsubseteq\mathbf{X}') & \pi \cdot X^S \sqsubseteq T, \mathit{Inc} \Longrightarrow \mathit{Inc} & & (S \subseteq \pi^{-1} \cdot T)
\end{array}
$$

Figure 4: Simplification of support inclusion problems

Inc. Write $\mathit{Sol}(\mathit{Inc})$ for the solutions of Inc. Call Inc **solvable** when $\mathit{Sol}(\mathit{Inc}) \neq \varnothing$, and **non-trivial** when $\mathit{nf}(\mathit{Inc}) \neq \varnothing$.

Definition 5.2. Define a **simplification** rewrite relation by the rules in Figure 4.

Definition 5.2 can easily be expressed as a type I conditional term rewrite system, according to the classification scheme of [1, Definition 7.1.1]. This becomes evident if we bear in mind that we can represent atoms as numbers, $\mathbb{A}^<$ as the even numbers, S as the finite set $S \,\Delta\, \mathbb{A}^<$ (Remark 2.23), and T as the finite set $T \,\Delta\, \mathbb{A}^<$.

Theorem 5.3. *If $\mathit{Inc} \Longrightarrow \mathit{Inc}'$ then $\mathit{Sol}(\mathit{Inc}) = \mathit{Sol}(\mathit{Inc}')$.*

Proof. First, we make the following claims:

Claim 1: If $a \in T$ then $\mathit{fa}(a\theta) \subseteq T$ always. Since $\mathit{fa}(a\theta) = \mathit{fa}(a) = \{a\}$.

Claim 2: $\mathit{fa}(\mathsf{f}(r_1,\ldots,r_n)\theta) \subseteq T$ if and only if $\mathit{fa}(r_i\theta) \subseteq T$ for $1 \leq i \leq n$. Since $\mathit{fa}(\mathsf{f}(r_1,\ldots,r_n)) = \mathit{fa}(r_1) \cup \ldots \cup \mathit{fa}(r_n)$, and $\mathsf{f}(r_1,\ldots,r_n)\theta \equiv \mathsf{f}(r_1\theta,\ldots,r_n\theta)$.

Claim 3: $\mathit{fa}(([a]s)\theta) \subseteq T$ if and only if $\mathit{fa}(s\theta) \subseteq T \cup \{a\}$. Suppose $\mathit{fa}(([a]s)\theta) \subseteq T$, therefore $\mathit{fa}([a]s\theta) \subseteq T$. Then $\mathit{fa}(s\theta) \setminus \{a\} \subseteq T$, therefore $\mathit{fa}(s\theta) \subseteq T \cup \{a\}$ and the result follows. The reverse direction is similar.

Claim 4: $\mathit{fa}((\pi \cdot X^S)\theta) \subseteq T$ if and only if $\mathit{fa}(X^S\theta) \subseteq \pi^{-1} \cdot T$. We consider only one case. Suppose $\theta = [X^S{:=}t]$ and $\mathit{fa}(t) \subseteq S$, therefore $\mathit{fa}((\pi \cdot X^S)[X^S{:=}t]) = \mathit{fa}(\pi \cdot t)$ hence $\mathit{fa}(\pi \cdot t) \subseteq T$ by assumption. By Lemma 2.17, $\pi \cdot \mathit{fa}(t) \subseteq T$, and by Lemma 2.16 and Lemma 2.17, $(\pi^{-1} \circ \pi) \cdot \mathit{fa}(t) \subseteq \pi^{-1} \cdot T$. As $\pi^{-1} \circ \pi = \mathit{id}$, we have $\mathit{fa}(X^S[X^S{:=}t]) \subseteq \pi^{-1} \cdot T$, and the result follows.
Alternatively, suppose $\mathit{fa}(t) \not\subseteq S$. Then $\mathit{fa}((\pi \cdot X^S)[X^S{:=}t]) = \mathit{fa}(\pi \cdot X^S)$ and $\pi \cdot \mathit{fa}(X^S) \subseteq T$ by Lemma 2.17. By Lemmas 2.16 and 2.17, $\mathit{fa}(X^S[X^S{:=}t]) \subseteq \pi^{-1} \cdot T$, and the result follows. The reverse implication is no harder.

Claim 5: By Definition 3.2 we have $\mathit{fa}((\pi \cdot X^S)\theta) = \mathit{fa}(\pi \cdot (X^S\theta))$. If $S \subseteq \pi^{-1} \cdot T$ then $\mathit{fa}(\pi \cdot (X^S\theta)) \subseteq T$ always. Note, $S \subseteq \pi^{-1} \cdot T$ if and only if $\pi \cdot S \subseteq T$ and $\mathit{fa}(\pi \cdot X^S) = \pi \cdot S$. Using Definition 3.2 and Lemma 2.17, $\mathit{fa}(\pi \cdot (X^S\theta)) = \pi \cdot \mathit{fa}(\theta(X^S)) \subseteq \pi \cdot S$. Therefore, $\mathit{fa}((\pi \cdot X^S)\theta) \subseteq T$, and the result follows.

We now proceed by case analysis on $\mathit{Inc} \Longrightarrow \mathit{Inc}'$ (Definition 5.2):

- The case $(\sqsubseteq\mathbf{a})$. Suppose $a \in T$. If $\theta \in \mathit{Sol}(a \sqsubseteq T, \mathit{Inc}')$ then $\theta \in \mathit{Sol}(\mathit{Inc}')$ and the result follows immediately. Conversely, suppose $\theta \in \mathit{Sol}(\mathit{Inc}')$. Using Claim 1, $\mathit{fa}(a\theta) \subseteq T$, and the result follows.
- The case $(\sqsubseteq\mathsf{f})$. From Claim 2.
- The case $(\sqsubseteq[])$. If $\theta \in \mathit{Sol}(r \sqsubseteq T \cup \{a\}, \mathit{Inc}')$ then $\mathit{fa}(r\theta) \subseteq T \cup \{a\}$. By Claim 3, $\mathit{fa}([a](r\theta)) \subseteq T$. As $\mathit{fa}([a](r\theta)) = \mathit{fa}(([a]r)\theta)$ and $\theta \in \mathit{Sol}(\mathit{Inc}')$, the result follows. The reverse implication is similar.

- The case ($\sqsubseteq\mathbf{X}$). Suppose $S \not\subseteq \pi^{-1} \cdot T$, $\pi \neq id$ and $\theta \in Sol(\pi \cdot X^S \sqsubseteq T, Inc)$, so $fa((\pi \cdot X^S)\theta) \subseteq T$. By Claim 4, $fa(X^S\theta) \subseteq \pi^{-1} \cdot T$, and as $\theta \in Sol(Inc')$, the result follows. The reverse implication is similar.
- The case ($\sqsubseteq\mathbf{X}'$). Suppose $S \subseteq \pi^{-1} \cdot T$. If $\theta \in Sol(\pi \cdot X^S, Inc')$ then $\theta \in Sol(Inc')$ and the result follows. Conversely, suppose $\theta \in Sol(Inc')$. By Claim 5, $fa((\pi \cdot X^S)\theta) \subseteq T$. The result follows.

□

Proposition 5.4. *Support inclusion problem simplification is strongly normalising.*

Proof. See Appendix Appendix A. □

We conclude with a few useful observations:

Definition 5.5. For every Inc make a fixed but arbitrary choice of normal form $nf(Inc)$, guaranteed to exist by Proposition 5.4.[6]

Definition 5.6. Call Inc **consistent** when $a \sqsubseteq T \notin nf(Inc)$ for all atoms a and permission sets T.

Lemma 5.7. *If Inc is consistent then all $inc \in nf(Inc)$ have the form $X^S \sqsubseteq T$ where $S \not\subseteq T$.*

5.2. Building solutions for support inclusion problems

Our main results are Theorems 5.13 and 5.20.

Definition 5.8. Define $fV(Inc)$ by $fV(Inc) = \bigcup\{fV(r) \mid \exists T.r \sqsubseteq T \in Inc\}$.
In words, $fV(Inc)$ is "the unknowns appearing in terms appearing in Inc".

Recall from Definition 2.3 that $\mathcal{V}$ ranges over finite sets of unknowns.

Definition 5.9. Let $\mathcal{V}$ range over finite sets of unknowns.
Suppose Inc is consistent. For every $X^S \in \mathcal{V}$ make a fixed but arbitrary choice of $X'^{S'}$ such that $X'^{S'} \notin \mathcal{V}$ and

$$S' = S \cap \bigcap\{T \mid X^S \sqsubseteq T \in nf(Inc)\}. \tag{1}$$

We make our choice injectively; for distinct $X^S \in fV(Inc)$ and $Y^T \in fV(Inc)$, we choose $X'^{S'}$ and $Y'^{T'}$ distinct. It will be convenient to write $\mathcal{V}'^{\mathcal{V}}_{Inc}$ for the set of our choices $\{X'^{S'} \mid X^S \in \mathcal{V}\}$.
Define a substitution $\rho^{\mathcal{V}}_{Inc}$ by:

$$\begin{array}{ll} \rho^{\mathcal{V}}_{Inc}(X^S) \equiv id \cdot X'^{S'} & \text{if } X^S \in \mathcal{V} \\ \rho^{\mathcal{V}}_{Inc}(Y^T) \equiv id \cdot Y^T & \text{otherwise} \end{array}$$

[6] In fact, support inclusion simplification is confluent so $nf(Inc)$ is also unique. A proof of confluence is in a technical report [8]. For our purposes in this paper, it suffices to know that a normal form exists.

Remark 5.10. For example, take $a, b, c \in \mathbb{A}^<$, $S = \mathbb{A}^< \setminus \{c\}$, $T = \mathbb{A}^< \setminus \{a\}$, $U = \mathbb{A}^< \setminus \{b\}$, and $\mathcal{V} = \{X^S\}$.

It is easy to see that the support reduction problem $\{X^S \sqsubseteq T,\ X^S \sqsubseteq U\}$ is in normal form. Let $X'^{S'}$ where $S' = \mathbb{A}^< \setminus \{a, b, c\}$ be the fixed but arbitrary choice of fresh variable made in Definition 5.9. Then:

$$\begin{array}{ll} \rho^{\mathcal{V}}_{Inc}(X^S) = id \cdot X'^{S'} & \text{and} \\ \rho^{\mathcal{V}}_{Inc}(Y^T) = id \cdot Y^T & \text{for all other } Y^T \end{array}$$

Remark 5.11. $\rho^{\mathcal{V}}_{Inc}$ is a substitution that "makes Inc true on $\mathcal{V}$". Nominal unification does not have this notion because nominal terms unknowns are not permanently labelled with freshness information — instead, nominal terms unification emits 'fresh' freshness conditions.

It is easy to verify that

$$fa(\rho^{\mathcal{V}}_{Inc}(X^S)) \subseteq S \quad \text{for all} \quad X^S \in \mathcal{V}.$$

In fact, $\rho^{\mathcal{V}}_{Inc}$ is the most general solution with property; intuitively, all other solutions must factor through $\rho^{\mathcal{V}}_{Inc}$ on $\mathcal{V}$. This is made formal in Theorem 5.20.

Lemma 5.12. *If Inc is consistent then $\rho^{\mathcal{V}}_{Inc} \in Sol(Inc)$. ('$\rho^{\mathcal{V}}_{Inc}$ solves Inc.')*

Proof. Suppose Inc is a $\Longrightarrow$-normal form. If $X^S \sqsubseteq T \in Inc$ then $\rho^{\mathcal{V}}_{Inc}(X) = id \cdot X'^{S'}$ for an S' which satisfies $S' \subseteq T$. The result follows.

More generally, if Inc is not a $\Longrightarrow$-normal form, by Theorem 5.3 $Sol(Inc) = Sol(nf(Inc))$, and we use the previous paragraph. □

Theorem 5.13. *Inc is consistent (Definition 5.6) if and only if Inc is solvable (Definition 5.1).*

Proof. By Theorem 5.3 $Sol(Inc) = Sol(nf(Inc))$, so it suffices to show the result for the case when Inc is a $\Longrightarrow$-normal form.

Suppose Inc is inconsistent, so $nf(Inc)$ contains a support inclusions of the form $a \sqsubseteq T$ where $a \notin T$. Then $a\theta \equiv a$ always, so there is no substitution θ such that $a\theta \subseteq T$. Conversely, if Inc is consistent, the result follows by Lemma 5.12. □

Definition 5.14. Suppose that Inc is consistent, $fV(Inc) \subseteq \mathcal{V}$, and $\theta \in Sol(Inc)$. Define a substitution $\theta{-}\rho^{\mathcal{V}}_{Inc}$ by:

- $(\theta{-}\rho^{\mathcal{V}}_{Inc})(X'^{S'}) \equiv \theta(X^S)$ if $X^S \in \mathcal{V}$ and $\rho^{\mathcal{V}}_{Inc}(X^S) \equiv id \cdot X'^{S'}$.
- $(\theta{-}\rho^{\mathcal{V}}_{Inc})(X^S) \equiv \theta(X^S)$ if $X^S \notin \mathcal{V}$.

We check that Definition 5.14 is well-defined:

Lemma 5.15. *If $\theta{-}\rho^{\mathcal{V}}_{Inc}$ exists then it is well-defined.*

Proof. Suppose $\theta{-}\rho^{\mathcal{V}}_{Inc}$ exists. Then:

- Suppose $X^S \neq Y^T$, $X^S \notin \mathcal{V}$ and $Y^T \notin \mathcal{V}$. By Definition 5.14, $(\theta{-}\rho^{\mathcal{V}}_{Inc})(X^S) \equiv \theta(X^S)$ and $(\theta{-}\rho^{\mathcal{V}}_{Inc})(Y^T) \equiv \theta(Y^T)$. The result follows, as substitutions are well-defined.

- Suppose $X'^{S'} \neq Y'^{T'}$, $\rho^{\mathcal{V}}_{Inc}(X^S) \equiv id \cdot X'^{S'}$, $\rho^{\mathcal{V}}_{Inc}(Y^T) \equiv id \cdot Y'^{T'}$ and $X^S, Y^T \notin \mathcal{V}$. Then $(\theta{-}\rho^{\mathcal{V}}_{Inc})(X'^{S'}) \equiv \theta(X^S)$ and $(\theta{-}\rho^{\mathcal{V}}_{Inc})(Y'^{T'}) \equiv \theta(Y^T)$. Since $X^S \neq Y^T$, the result follows as substitutions are well-defined.
- Suppose $X'^{S'} \neq Y^T$, $\rho^{\mathcal{V}}_{Inc}(X^S) \equiv id{\cdot}X'^{S'}$, $X^S \notin \mathcal{V}$ and $Y^T \in \mathcal{V}$. Then $(\theta{-}\rho^{\mathcal{V}}_{Inc})(Y^T) \equiv \theta(Y^T)$ and $(\theta{-}\rho^{\mathcal{V}}_{Inc})(X'^{S'}) \equiv \theta(X^S)$. By Definition 5.9, $\rho^{\mathcal{V}}_{Inc}(X^S) \not\equiv id{\cdot}Y^T$ as $Y^T \in \mathcal{V}$. The result follows as substitutions are well-defined.

The case $Y'^{T'} \neq X^S$, $\rho^{\mathcal{V}}_{Inc}(Y^T) \equiv id \cdot Y'^{T'}$, $Y^T \notin \mathcal{V}$ and $X^S \in \mathcal{V}$ is similar to the case for $X'^{S'} \neq Y^T$, $\rho^{\mathcal{V}}_{Inc}(X^S) \equiv id \cdot X'^{S'}$, $X^S \notin \mathcal{V}$ and $Y^T \in \mathcal{V}$. □

Lemma 5.16. *If $\theta \in Sol(Inc)$ then $\rho^{\mathcal{V}}_{Inc}$ exists.*

Proof. By assumption, Inc is solvable. By Theorem 5.13, Inc is consistent. Using Definition 5.9, $\rho^{\mathcal{V}}_{Inc}$ exists. □

Lemma 5.17. *If $\rho^{\mathcal{V}}_{Inc}$ exists, then it is well-defined.*

Proof. Suppose $\rho^{\mathcal{V}}_{Inc}$ exists and $X^S \neq Y^T$. Then:

- $X^S \in \mathcal{V}$ and $Y^T \in \mathcal{V}$. Then $\rho^{\mathcal{V}}_{Inc}(X^S) = id \cdot X'^{S'}$ and $\rho^{\mathcal{V}}_{Inc}(Y^T) = id \cdot Y'^{T'}$. By Definition 5.9, $X'^{S'}$ and $Y'^{T'}$ are chosen so $X'^{S'} \neq Y'^{T'}$. The result follows.
- $X^S \notin \mathcal{V}$ and $Y^T \notin \mathcal{V}$. Then $\rho^{\mathcal{V}}_{Inc}(X^S) = id \cdot X^S$ and $\rho^{\mathcal{V}}_{Inc}(Y^T) = id \cdot Y^T$. The result follows.
- $X^S \in \mathcal{V}$ and $Y^T \notin \mathcal{V}$. Then $\rho^{\mathcal{V}}_{Inc}(Y^T) = id \cdot Y^T$ and $\rho^{\mathcal{V}}_{Inc}(X^S) = id \cdot X'^{S'}$ with $X'^{S'} \notin \mathcal{V}$. The result follows.
- The case $X^S \notin \mathcal{V}$ and $Y^T \in \mathcal{V}$ is similar to the case for $X^S \in \mathcal{V}$ and $Y^T \notin \mathcal{V}$.

□

Lemma 5.18. *If $Inc \Longrightarrow Inc'$ then $fV(Inc') \subseteq fV(Inc)$.*

Proof. By case analysis on the rules defining $\Longrightarrow$ (Definition 5.2). □

Lemma 5.19. *If $\theta \in Sol(Inc)$ and $fV(Inc) \subseteq \mathcal{V}$ then $\theta{-}\rho^{\mathcal{V}}_{Inc}$ is a substitution.*

Proof. By Lemma 5.16, $\rho^{\mathcal{V}}_{Inc}$ exists. We show $fa((\theta{-}\rho^{\mathcal{V}}_{Inc})(X'^{S'})) \subseteq S$ by cases:

- The case $id \cdot X'^{S'} \equiv \rho^{\mathcal{V}}_{Inc}(X^S)$ for $X^S \in \mathcal{V}$.
  Using Lemma 5.18, $fV(nf(Inc)) \subseteq fV(Inc)$. Then $fV(nf(Inc)) \subseteq \mathcal{V}$, as $fV(Inc) \subseteq \mathcal{V}$ by assumption. There are two sub-cases:
  - The case $X^S \notin fV(nf(Inc))$. Then $S = S'$ and $(\theta{-}\rho^{\mathcal{V}}_{Inc})(X'^S) = \theta(X^S)$ by Definition 5.14. By assumption $fa(\theta(X^S)) \subseteq S$. The result follows.
  - The case $X^S \in fV(nf(Inc))$. By assumption, $\theta \in Sol(Inc)$ so $\theta \in Sol(nf(Inc))$ using Theorem 5.3. Then $fa(\theta(X^S)) \subseteq T$ for every T such that $X^S \sqsubseteq T \in nf(Inc)$, using Definition 5.1. By Definition 5.14, $S' = \bigcap\{T \mid X^S \sqsubseteq T \in nf(Inc)\}$. The result follows.
- Otherwise, $(\theta{-}\rho^{\mathcal{V}}_{Inc})(X^S) \equiv \theta(X^S)$ and $fa(\theta(X^S)) \subseteq S$ by assumption.

□

Theorem 5.20. *Suppose $\theta \in Sol(Inc)$ and $fV(Inc) \subseteq \mathcal{V}$.*
Then $\theta(X^S) \equiv (\rho^{\mathcal{V}}_{Inc} \circ (\theta{-}\rho^{\mathcal{V}}_{Inc}))(X^S)$ for every $X^S \in \mathcal{V}$.

Proof. For some fresh $X'^S \notin \mathcal{V}$, $\rho(X^S) \equiv id{\cdot}X'^S$, and $(\theta{-}\rho^{\mathcal{V}}_{Inc})(X'^S) \equiv \theta(X^S)$. The result follows by Lemma 2.16. □

5.3. Support reduction example

Suppose $a, c \in \mathbb{A}^<$ and $b, d \notin \mathbb{A}^<$. Take $S = \mathbb{A}^< = T$, $U = \mathbb{A}^< \cup \{b\}$ and $V = \mathbb{A}^< \cup \{d\}$. We first run the support reduction algorithm on $Inc = \{a \sqsubseteq S,\ \mathsf{f}([a]a, id \cdot Y^T) \sqsubseteq T,\ (c\ a) \cdot Z^U \sqsubseteq S,\ (b\ a) \cdot W^V \sqsubseteq T\}$:

$$
\begin{array}{lll}
a \sqsubseteq S,\ \mathsf{f}([a]a, id \cdot Y^T) \sqsubseteq T,\ (c\ a) \cdot Z^U \sqsubseteq S,\ (b\ a) \cdot W^V \sqsubseteq T & \Longrightarrow & (\sqsubseteq \mathbf{a}) \\
\mathsf{f}([a]a, id \cdot Y^T) \sqsubseteq T,\ (c\ a) \cdot Z^U \sqsubseteq S,\ (b\ a) \cdot W^V \sqsubseteq T & \Longrightarrow & (\sqsubseteq \mathsf{f}) \\
[a]a \sqsubseteq T,\ id \cdot Y^T \sqsubseteq T,\ (c\ a) \cdot Z^U \sqsubseteq S,\ (b\ a) \cdot W^V \sqsubseteq T & \Longrightarrow & (\sqsubseteq []) \\
id \cdot Y^T \sqsubseteq T,\ (c\ a) \cdot Z^U \sqsubseteq S,\ (b\ a) \cdot W^V \sqsubseteq T & \Longrightarrow & (\sqsubseteq \mathbf{X}') \\
(c\ a) \cdot Z^U \sqsubseteq S,\ (b\ a) \cdot W^V \sqsubseteq T & \Longrightarrow & (\sqsubseteq \mathbf{X}) \\
Z^U \sqsubseteq (c\ a) \cdot S,\ (b\ a) \cdot W^V \sqsubseteq T & \Longrightarrow & (\sqsubseteq \mathbf{X}) \\
Z^U \sqsubseteq (c\ a) \cdot S,\ W^V \sqsubseteq (b\ a) \cdot T & &
\end{array}
$$

We take $nf(Inc) = \{Z^U \sqsubseteq (c\ a) \cdot S,\ W^V \sqsubseteq (b\ a) \cdot T\}$. By Definition 5.6 we have $nf(Inc)$ is consistent, therefore by Theorem 5.13 a solution for Inc exists.

We now construct $\rho^{\mathcal{V}}_{Inc}$. Take W' and Z' as our injective choices of fresh unknowns. Take $U' = U \cap ((c\ a) \cdot S \cap (b\ a) \cdot T)$ and $V' = V \cap ((c\ a) \cdot S \cap (b\ a) \cdot T)$. An easy calculation shows that $U' = \mathbb{A}^< \setminus \{a\} = V'$. We define $\rho^{\mathcal{V}}_{Inc}$ piecewise by:

$\rho^{\mathcal{V}}_{Inc}(Z^U) = id \cdot {Z'}^{U'}$ and $\rho^{\mathcal{V}}_{Inc}(W^V) = id \cdot {W'}^{V'}$ and $\rho^{\mathcal{V}}_{Inc}(X^S) = id \cdot X^S$ for all other X^S

It is easy to verify that this is in fact a substitution (i.e. $fa(\rho^{\mathcal{V}}_{Inc}(X^S)) \subseteq S$ always).

6. Permissive nominal unification problems

We can now give an algorithm to compute solutions to permissive nominal unification problems (Definition 3.9) and we prove that our algorithm computes most general solutions. Note, as we have observed before, that the notion of problem and solution is based just on equalities and substitutions.

6.1. The unification algorithm

Lemma 6.1. *$\theta \circ \theta' \in Sol(Pr)$ if and only if $\theta' \in Sol(Pr\theta)$.*

Proof. By unpacking Definition 3.9 and using Lemma 3.8. □

Definition 6.2. If Pr is a problem, define a support inclusion problem $Pr_{\sqsubseteq}$ by:

$$Pr_{\sqsubseteq} = \{r \sqsubseteq fa(s),\ s \sqsubseteq fa(r) \mid r \stackrel{?}{=} s \in Pr\}$$

Call a support inclusion problem Inc **non-trivial** when $nf(Inc) \neq \varnothing$.

$$
\begin{array}{lllll}
(\stackrel{?}{=}\mathbf{a}) & \mathcal{V}; a \stackrel{?}{=} a,\ Pr & \Longrightarrow & \mathcal{V}; Pr & \\
(\stackrel{?}{=}\mathsf{f}) & \mathcal{V}; \mathsf{f}(r_1, \ldots) \stackrel{?}{=} \mathsf{f}(s_1, \ldots),\ Pr & \Longrightarrow & \mathcal{V}; r_1 \stackrel{?}{=} s_1,\ \ldots,\ Pr & \\
(\stackrel{?}{=}[\mathbf{a}]) & \mathcal{V}; [a]r \stackrel{?}{=} [a]s,\ Pr & \Longrightarrow & \mathcal{V}; r \stackrel{?}{=} s,\ Pr & \\
(\stackrel{?}{=}[\mathbf{b}]) & \mathcal{V}; [a]r \stackrel{?}{=} [b]s,\ Pr & \Longrightarrow & \mathcal{V}; (b\ a) \cdot r \stackrel{?}{=} s,\ Pr & \\
 & & & (b \notin fa(r)) & \\
(\stackrel{?}{=}\mathbf{X}) & \mathcal{V}; \pi \cdot X^S \stackrel{?}{=} \pi \cdot X^S,\ Pr & \Longrightarrow & \mathcal{V}; Pr & \\
(\mathbf{I1}) & \mathcal{V}; \pi \cdot X^S \stackrel{?}{=} s,\ Pr & \stackrel{[X^S := \pi^{-1} \cdot s]}{\Longrightarrow} & \mathcal{V}; Pr[X^S := \pi^{-1} \cdot s] & \\
 & & & (X^S \notin fV(s),\ fa(s) \subseteq \pi \cdot S) & \\
(\mathbf{I2}) & \mathcal{V}; r \stackrel{?}{=} \pi \cdot X^S,\ Pr & \stackrel{[X^S := \pi^{-1} \cdot r]}{\Longrightarrow} & \mathcal{V}; Pr[X^S := \pi^{-1} \cdot r] & \\
 & & & (X^S \notin fV(r),\ fa(r) \subseteq \pi \cdot S) & \\
(\mathbf{I3}) & \mathcal{V}; Pr & \stackrel{\rho^{\mathcal{V}}_{Pr_\sqsubseteq}}{\Longrightarrow} & \mathcal{V} \cup \mathcal{V}'^{\mathcal{V}}_{Pr_\sqsubseteq}; Pr\ \rho^{\mathcal{V}}_{Pr_\sqsubseteq} & \\
 & & & (Pr_\sqsubseteq \text{ consistent and non-trivial}) &
\end{array}
$$

Figure 5: Simplification rules for problems

Definition 6.3. Define a **simplification** rewrite relation $\mathcal{V}; Pr \Longrightarrow \mathcal{V}'; Pr'$ on unification problems by the rules in Figure 5.

Call $(\stackrel{?}{=}\mathbf{a})$, $(\stackrel{?}{=}\mathsf{f})$, $(\stackrel{?}{=}[\mathbf{a}])$, $(\stackrel{?}{=}[\mathbf{b}])$, and $(\stackrel{?}{=}\mathbf{X})$ **non-instantiating rules**. Call $(\mathbf{I1})$, $(\mathbf{I2})$, and $(\mathbf{I3})$ **instantiating rules**. Write $\Longrightarrow^*$ for the transitive and reflexive closure of $\Longrightarrow$.

In $(\mathbf{I3})$ we insist that $Pr_\sqsubseteq$ is non-trivial to avoid indefinite rewrites. We insist $Pr_\sqsubseteq$ is consistent so that $\rho^{\mathcal{V}}_{Pr_\sqsubseteq}$ exists. $\rho^{\mathcal{V}}_{Pr_\sqsubseteq}$ and $(\mathcal{V}'^{\mathcal{V}}_{Pr_\sqsubseteq}$ are defined in Definition 5.9.)

Definition 6.4. Define $fV(Pr) = \bigcup\{fV(r) \cup fV(s) \mid r \stackrel{?}{=} s \in Pr\}$.

Definition 6.5. Suppose $\mathcal{V}$ is a set of unknowns. Define $\theta|_{\mathcal{V}}$ by:

$$
\begin{array}{ll}
\theta|_{\mathcal{V}}(X) \equiv \theta(X) & \text{if } X \in \mathcal{V} \\
\theta|_{\mathcal{V}}(X) \equiv id \cdot X & \text{otherwise}
\end{array}
$$

(We overload $|$, for technical convenience: $\pi|_S$ is partial and $\theta|_{\mathcal{V}}$ is total.)

Definition 6.6. If Pr is a problem, define a **unification algorithm** by:

1. Rewrite $fV(Pr); Pr$ using the rules of Definition 6.3 as much as possible, with top-down precedence (so we apply $(\stackrel{?}{=}\mathbf{a})$ before $(\stackrel{?}{=}\mathsf{f})$, and so on down to $(\mathbf{I3})$).
2. If we reduce to $\mathcal{V}'; \varnothing$, we succeed and return $\theta|_{\mathcal{V}}$ where θ is the functional composition of all the substitutions labelling rewrites (we take $\theta = id$ if there are none). Otherwise, we fail.

Proposition 6.7. *The algorithm of Definition 6.6 always terminates.*

Proof. See Appendix Appendix A. □

6.2. Examples of the algorithm

Example one.

Suppose $a, c \in \mathbb{A}^<$ and $d \notin \mathbb{A}^<$. Take $S = \mathbb{A}^<$. Take $\mathcal{V} = \{X^S\}$. Suppose a term-former g. We apply the algorithm to $\{\mathsf{g}([a](id \cdot X^S), [a]a) \stackrel{?}{=} \mathsf{g}([d]c, [d]d)\}$:

$$\begin{array}{lll}
\mathcal{V};\ \mathsf{g}([a](id \cdot X^S), [a]a) \stackrel{?}{=} \mathsf{g}([d]c, [d]d) & \Longrightarrow & (\stackrel{?}{=}\mathsf{f}) \\
\mathcal{V};\ [a](id \cdot X^S) \stackrel{?}{=} [d]c,\ [a]a \stackrel{?}{=} [d]d & \Longrightarrow & (\stackrel{?}{=}[\mathbf{b}]) \\
\mathcal{V};\ (d\ a) \cdot X^S \stackrel{?}{=} c,\ [a]a \stackrel{?}{=} [d]d & \stackrel{[X^S:=c]}{\Longrightarrow} & (\mathbf{I1}) \\
\mathcal{V};\ [a]a \stackrel{?}{=} [d]d & \Longrightarrow & (\stackrel{?}{=}[\mathbf{b}]) \\
\mathcal{V};\ (d\ a) \cdot a \stackrel{?}{=} d & \Longrightarrow & (\stackrel{?}{=}\mathbf{a}) \\
\mathcal{V};\ \varnothing \quad \text{(Success!)} & &
\end{array}$$

The algorithm succeeds and returns the substitution $[X^S{:=}c]$.

Example two.

Suppose $a, c \in \mathbb{A}^<$ and $b, d \notin \mathbb{A}^<$. Take $S = \mathbb{A}^< \cup \{b, d\}$ and $T = \mathbb{A}^< \cup \{f\}$ and $U = \mathbb{A}^<$. Take $\mathcal{V} = \{X^S, Y^T\}$. Suppose a term-former f.

We apply the algorithm to $\{\mathsf{f}([a]b, id \cdot Z^U, id \cdot X^S) \stackrel{?}{=} \mathsf{f}([d]b, [a]a, id \cdot Y^T)\}$:

$$\begin{array}{lll}
\mathcal{V};\ \mathsf{f}([a]b, id \cdot Z^U, id \cdot X^S) \stackrel{?}{=} \mathsf{f}([d]b, [a]a, id \cdot Y^T) & \Longrightarrow & (\stackrel{?}{=}\mathsf{f}) \\
\mathcal{V};\ [a]b \stackrel{?}{=} [d]b,\ id \cdot Z^U \stackrel{?}{=} [a]a,\ id \cdot X^S \stackrel{?}{=} id \cdot Y^T & \Longrightarrow & (\stackrel{?}{=}[\mathbf{b}]) \\
\mathcal{V};\ (d\ a) \cdot b \stackrel{?}{=} b,\ id \cdot Z^U \stackrel{?}{=} [a]a,\ id \cdot X^S \stackrel{?}{=} id \cdot Y^T & \Longrightarrow & (\stackrel{?}{=}\mathbf{a}) \\
\mathcal{V};\ id \cdot Z^U \stackrel{?}{=} [a]a,\ id \cdot X^S \stackrel{?}{=} id \cdot Y^T & \stackrel{[Z^U:=[a]a]}{\Longrightarrow} & (\stackrel{?}{=}\mathbf{I1}) \\
\mathcal{V};\ id \cdot X^S \stackrel{?}{=} id \cdot Y^T & \stackrel{[X^S:=X'^{S'}][Y^T:=Y'^{T'}]}{\Longrightarrow} & (\mathbf{I3}) \\
\mathcal{V} \cup \{X'^{S'}, Y'^{T'}\};\ id \cdot X'^{S'} \stackrel{?}{=} id \cdot Y'^{T'} & \stackrel{[X'^{S'}:=Y'^{T'}]}{\Longrightarrow} & (\mathbf{I1}) \\
\mathcal{V} \cup \{X'^{S'}, Y'^{T'}\};\ id \cdot Y'^{T'} \stackrel{?}{=} id \cdot Y'^{T'} & \Longrightarrow & (\stackrel{?}{=}\mathbf{X}) \\
\mathcal{V} \cup \{X'^{S'}, Y'^{T'}\};\ \varnothing \quad \text{(Success!)} & &
\end{array}$$

Here X' and Y' are the choice of unknown made in Definition 5.9, and $S' = \mathbb{A}^< = T'$.

The algorithm succeeds and returns the substitution

$$[X'^{S'}{:=}Y'^{T'}] \circ [X^S{:=}X'^{S'}] \circ [Y^T{:=}Y'^{T'}] \circ [Z^U{:=}[a]a].$$

Example three.

An example that fails to unify. Take $S = \mathbb{A}^<$. Take $\mathcal{V} = \{X^S\}$. We run the algorithm on $\{[a]([b](id \cdot X^S)) \stackrel{?}{=} [a](id \cdot X^S)\}$:

$$\mathcal{V};\ [a]([b](id \cdot X^S)) \stackrel{?}{=} [a](id \cdot X^S) \qquad \Longrightarrow \qquad (\stackrel{?}{=}[\mathbf{a}])$$

$$\mathcal{V};\ [b](id \cdot X^S) \stackrel{?}{=} id \cdot X^S \qquad \text{(Failure!)}$$

The algorithm fails as the precondition of rule (**I2**), $X^S \notin fV([b](id \cdot X^S))$, the 'occurs check', fails to hold. By Theorem 6.25 there is no solution to the unification problem.

6.3. Preservation of solutions

Lemma 6.8. *If $\mathcal{V}; Pr \Longrightarrow \mathcal{V}; Pr'$ by a non-instantiating rule (Definition 6.3) then $Sol(Pr) = Sol(Pr')$.*

Proof. See Appendix Appendix A. We use Lemmas 3.3 and 3.4, and Proposition 2.21. □

Lemma 6.9. *Suppose $\theta(X^S) =_\alpha \theta'(X^S)$ for all $X^S \in fV(Pr)$. Then $\theta \in Sol(Pr)$ if and only if $\theta' \in Sol(Pr)$.*

Proof. Unpacking Definition 3.9 it suffices to show that $r\theta =_\alpha s\theta$ if and only if $r\theta' =_\alpha s\theta'$, for every $r \stackrel{?}{=} s \in Pr$. This is easy using Lemma 3.5 and the fact by construction (Definition 6.4) that $fV(r) \subseteq fV(Pr)$ and $fV(s) \subseteq fV(Pr)$. □

Definition 6.10. Write $\theta{-}X^S$ for the substitution such that

$$\begin{array}{ll} (\theta{-}X^S)(X^S) \equiv id \cdot X^S & \text{and} \\ (\theta{-}X^S)(Y^T) \equiv \theta(Y^T) & \text{for all other } Y^T. \end{array}$$

In the right circumstances, a substitution θ can be factored as 'a part of θ that does not touch X^S' and 'a single substitution for X^S':

Theorem 6.11. *Suppose $X^S\theta =_\alpha s\theta$ and $X^S \notin fV(s)$. Then*

$$\begin{array}{l} X^S\theta =_\alpha X^S([X^S{:=}s] \circ (\theta{-}X^S)) \quad \textit{and} \\ Y^T\theta =_\alpha Y^T([X^S{:=}s] \circ (\theta{-}X^S)). \end{array}$$

Proof. We reason as follows:

$$\begin{array}{lll} X^S([X^S{:=}s] \circ (\theta{-}X^S)) \equiv s(\theta{-}X^S) & & \text{Definition 3.2} \\ \qquad\qquad \equiv s\theta & & X^S \notin fV(s),\ \text{Lemma 3.5} \\ \qquad\qquad =_\alpha X^S\theta & & \text{Assumption} \\ \\ Y^T([X^S{:=}s] \circ (\theta{-}X^S)) \equiv Y^T(\theta{-}X^S) & & \text{Definition 3.7} \\ \qquad\qquad \equiv Y^T\theta & & \text{Definition 6.10} \end{array}$$

□

6.4. Simplification rewrites calculate principal solutions

Definition 6.12. Write $\theta_1 \leq \theta_2$ when there exists some θ' such that $X^S\theta_2 =_\alpha X^S(\theta_1 \circ \theta')$ always. Call $\leq$ the **instantiation ordering**.

Definition 6.13. A **principal** (or **most general**) solution to a problem Pr is a solution $\theta \in Sol(Pr)$ such that $\theta \leq \theta'$ for all other $\theta' \in Sol(Pr)$.

Our main results are Theorems 6.18 — the unification algorithm from Definition 6.6 calculates a solution — and 6.23 — the solution it calculates, is principal.

Lemma 6.14. *If $fV(Pr) \subseteq \mathcal{V}$ and $\mathcal{V}; Pr \implies \mathcal{V}'; Pr'$ using a non-instantiating rule, then $fV(Pr') \subseteq \mathcal{V}$.*

Proof. See Appendix Appendix A. □

Lemma 6.15. $fV(r[X^S{:=}s]) \subseteq fV(r) \cup fV(s)$.

Proof. By induction on r. □

Lemma 6.16. *If $fV(Pr) \subseteq \mathcal{V}$ and $\mathcal{V}; Pr \overset{\theta}{\implies} \mathcal{V}'; Pr'\theta$ using an instantiating rule, then $fV(Pr'\theta) \subseteq \mathcal{V}'$.*

Proof. There are two cases:

- The case (**I1**). Suppose $fV(\pi \cdot X^S \stackrel{?}{=} s, Pr') \subseteq \mathcal{V}$ and $\mathcal{V}; \pi \cdot X^S \stackrel{?}{=} s, Pr' \overset{[X^S:=\pi^{-1}\cdot s]}{\implies} \mathcal{V}'; Pr'[X^S{:=}\pi^{-1} \cdot s]$ using (**I1**). Using Definition 6.4 and Lemma 6.15, we have $fV(Pr'[X^S{:=}\pi^{-1} \cdot s]) \subseteq fV(Pr') \cup fV(\pi^{-1} \cdot s)$. The result follows.
- The case (**I3**). A corollary of Lemma 5.18.

□

Lemma 6.17. *If $X^S \in \mathcal{V}$ then $([X^S{:=}s]\circ\theta)|_\mathcal{V} = [X^S{:=}s]\circ(\theta|_\mathcal{V})$*

Proof. We consider cases:

- The case X^S with $X^S \in \mathcal{V}$. We reason as follows:

$$\begin{array}{lll} ([X^S{:=}s]\circ\theta)|_\mathcal{V}(X^S) & \equiv ([X^S{:=}s]\circ\theta)(X^S) & \text{Definition 6.5, } X^S \in \mathcal{V} \\ & \equiv s\theta & \text{Definition 3.7} \\ & \equiv s\theta|_\mathcal{V} & \text{Definition 6.5} \end{array}$$

- The case Y^T with $Y^T \in \mathcal{V}$. We reason as follows:

$$\begin{array}{lll} ([X^S{:=}s]\circ\theta)|_\mathcal{V}(Y^T) & \equiv ([X^S{:=}s]\circ\theta)(Y^T) & \text{Definition 6.5, } Y^T \in \mathcal{V} \\ & \equiv \theta(Y^T) & \text{Definition 3.7} \\ & \equiv \theta|_\mathcal{V}(Y^T) & \text{Definition 6.5} \end{array}$$

- The case Y^T with $Y^T \notin \mathcal{V}$. Since $([X^S{:=}s]\circ\theta)|_\mathcal{V}(Y^T) \equiv id \cdot Y^T$ and $\theta|_\mathcal{V} \equiv id \cdot Y^T$.

□

Recall that $\theta|_\mathcal{V}$ is defined in Definition 6.5:

Theorem 6.18. *If $fV(Pr) \subseteq \mathcal{V}$ then $\mathcal{V}; Pr \overset{\theta}{\implies}{}^* \mathcal{V}'; \varnothing$ implies $\theta|_\mathcal{V} \in Sol(Pr)$.*

Proof. By induction on the length of the path in $\stackrel{\theta}{\Longrightarrow}$.

- *Length* 0. Then $Pr = \varnothing$ and $\theta \equiv id$. The result follows.
- *Length* $k+1$. There are three cases:
  - *The non-instantiating case.* Suppose $\mathcal{V}; Pr \Longrightarrow \mathcal{V}; Pr'' \stackrel{\theta}{\Longrightarrow} \mathcal{V}'; \varnothing$. Using Lemma 6.14, $fV(Pr'') \subseteq \mathcal{V}$ and $\theta \in Sol(Pr'')$ by inductive hypothesis. Using Lemma 6.8, $\theta \in Sol(Pr)$, and the result follows.
  - *The case of* (**I1**) *or* (**I2**). Suppose $\mathcal{V}; Pr \stackrel{\chi}{\Longrightarrow} \mathcal{V}; Pr\chi \stackrel{\theta'}{\Longrightarrow} \mathcal{V}'; \varnothing$. Using Lemma 6.16, $fV(Pr\chi) \subseteq \mathcal{V}$. By inductive hypothesis $\theta'|_{\mathcal{V}} \in Sol(Pr\chi)$. By Lemma 6.17, $(\chi \circ \theta')|_{\mathcal{V}} = \chi \circ (\theta'|_{\mathcal{V}})$. By Lemma 6.1, $(\chi \circ \theta')|_{\mathcal{V}} \in Sol(Pr)$.
  - *The case of* (**I3**). Suppose $\mathcal{V}; Pr \stackrel{\rho}{\Longrightarrow} \mathcal{V}'; Pr\rho \stackrel{\theta'}{\Longrightarrow} \mathcal{V}''; \varnothing$. Using Lemma 6.16, $fV(Pr\rho) \subseteq \mathcal{V}'$. By inductive hypothesis $\theta'|_{\mathcal{V}'} \in Sol(Pr\rho)$. By Lemma 6.1, $\rho \circ (\theta'|_{\mathcal{V}'}) \in Sol(Pr)$. By Lemma 6.17, $\rho \circ (\theta'|_{\mathcal{V}'}) = (\rho \circ \theta')|_{\mathcal{V}'}$. By Lemma 6.9, $(\rho \circ \theta')|_{\mathcal{V}} \in Sol(Pr)$.

□

Lemma 6.19. *If* $\theta_1 \leq \theta_2$ *then* $\theta \circ \theta_1 \leq \theta \circ \theta_2$.

Proof. By Definition 6.12, θ' exists such that $X^S\theta_2 =_\alpha X^S(\theta_1 \circ \theta')$ always. Then:

$$\begin{array}{rcll} X^S(\theta \circ \theta_2) & \equiv & (X^S\theta)\theta_2 & \text{Lemma 3.8} \\ & =_\alpha & (X^S\theta)(\theta_1 \circ \theta') & \text{Lemma 3.5} \\ & \equiv & X^S((\theta \circ \theta_1) \circ \theta') & \text{Lemma 3.8} \end{array}$$

□

Lemma 6.20. *Suppose* $X^S\theta_2 =_\alpha X^S\theta_2'$ *always. Then* $\theta_1 \leq \theta_2$ *implies* $\theta_1 \leq \theta_2'$.

Proof. By a routine calculation using Definition 6.12 and using Proposition 2.21. □

Lemma 6.21. *If* $\theta \in Sol(Pr)$ *(Definition 3.9) then* $\theta \in Sol(Pr_{\sqsubseteq})$ *(Definition 5.1).*

Proof. By a routine calculation, using Definitions 3.9 and 6.2, and Lemma 2.19. □

Lemma 6.22. *If* $X^S \in \mathcal{V}$ *then* $(\theta|_{\mathcal{V}} - X^S) = (\theta - X^S)|_{\mathcal{V}}$.

Proof. We consider cases:

- The case X^S. Then $(\theta|_{\mathcal{V}} - X^S)(X^S) = id \cdot X^S$ and $(\theta - X^S)|_{\mathcal{V}}(X^S) = id \cdot X^S$. The result follows.
- The case Y^T with $Y^T \notin \mathcal{V}$. Then $(\theta|_{\mathcal{V}} - X^S)(Y^T) = id \cdot Y^T$ and $(\theta - X^S)|_{\mathcal{V}}(Y^T) = id \cdot Y^T$. The result follows.
- The case Y^T with $Y^T \in \mathcal{V}$. Then $(\theta|_{\mathcal{V}} - X^S)(Y^T) = \theta|_{\mathcal{V}}(Y^T)$ and $(\theta - X^S)|_{\mathcal{V}}(Y^T) = \theta(Y^T)$. As $\theta|_{\mathcal{V}}(Y^T) = \theta(Y^T)$ when $Y^T \in \mathcal{V}$, the result follows.

□

Theorem 6.23. *Suppose* $fV(Pr) \subseteq \mathcal{V}$.

If $\mathcal{V}; Pr \stackrel{\theta}{\Longrightarrow} \mathcal{V}'; \varnothing$ *then* $\theta|_{\mathcal{V}}$ *is a principal solution to* Pr *(Definition 6.13).*

Proof. By Theorem 6.18, $\theta|_{\mathcal{V}} \in Sol(Pr)$. We prove $\theta|_{\mathcal{V}}$ is principal by induction on the path length of $\mathcal{V}; Pr \stackrel{\theta}{\Longrightarrow} \mathcal{V}'; \varnothing$.

- *Length* 0. So $Pr = \varnothing$ and $\theta = id|_{\mathcal{V}}$. By Definition 6.12, $id|_{\mathcal{V}} \leq \theta'|_{\mathcal{V}}$.
- *Length* $k+1$. We consider the rules in Definition 6.3.
  - The non-instantiating case. Suppose

$$\mathcal{V}; Pr \Longrightarrow \mathcal{V}; Pr' \overset{\theta}{\Longrightarrow}{}^{*} \mathcal{V}'; \varnothing$$

    where $\mathcal{V}; Pr \Longrightarrow \mathcal{V}; Pr'$ is a non-instantiating simplification rewrite. By inductive hypothesis $\theta|_{\mathcal{V}}$ is a principal solution of Pr'. By Lemma 6.8 $\theta|_{\mathcal{V}}$ is a principal solution of Pr. The result follows.
  - The case (**I1**). Suppose $fa(s) \subseteq \pi \cdot S$ and $X^S \notin fV(s)$. Write $\chi = [X^S{:=}\pi^{-1} \cdot s]$. Suppose $Pr = \pi \cdot X^S \stackrel{?}{=} s,\ Pr''$ so that

$$\mathcal{V}; \pi \cdot X^S \stackrel{?}{=} s,\ Pr'' \overset{\chi}{\Longrightarrow} \mathcal{V}; Pr''\chi \overset{\theta''}{\Longrightarrow}{}^{*} \mathcal{V}'; \varnothing.$$

    Further, suppose that $\theta'|_{\mathcal{V}} \in Sol(Pr)$.
    By assumption $(\pi \cdot X)\theta' =_{\alpha} s\theta'$, so by Lemma 3.4 $\pi \cdot \theta'(X) =_{\alpha} s\theta'$. By Lemmas 2.18 and 2.16 $\theta'(X) =_{\alpha} \pi^{-1} \cdot (s\theta')$, and by Lemma 3.4 $\theta'(X) =_{\alpha} (\pi^{-1} \cdot s)\theta'$. It follows by Theorem 6.11 and Lemma 6.9 that $\chi \circ (\theta'|_{\mathcal{V}} - X^S) \in Sol(Pr)$. Using Lemma 6.22, $(\theta'|_{\mathcal{V}} - X^S) = (\theta' - X^S)|_{\mathcal{V}}$. By Lemma 6.1, $(\theta' - X^S)|_{\mathcal{V}} \in Sol(Pr''\chi)$.
    By Theorem 6.18, $\theta''|_{\mathcal{V}} \in Sol(Pr''\chi)$. By Lemma 6.16, $fV(Pr''\chi) \subseteq \mathcal{V}$.
    By inductive hypothesis $\theta''|_{\mathcal{V}} \leq (\theta' {-} X^S)|_{\mathcal{V}}$. By Lemma 6.19, $\chi \circ (\theta''|_{\mathcal{V}}) \leq \chi \circ (\theta' {-} X^S)|_{\mathcal{V}}$. Now by assumption $fV(s) \subseteq \mathcal{V}$ and $X^S \in \mathcal{V}$. Using Lemma 6.17 it follows that $\chi \circ (\theta''|_{\mathcal{V}}) = (\chi \circ \theta'')|_{\mathcal{V}}$. By Lemma 6.22, $(\theta' {-} X^S)|_{\mathcal{V}} = \theta'|_{\mathcal{V}} {-} X^S$. By Theorem 6.11 and Lemma 6.20, $(\chi \circ \theta'')|_{\mathcal{V}} \leq \theta'|_{\mathcal{V}}$ as required.
  - The case (**I2**) is similar to the case of (**I1**).
  - The case (**I3**). Suppose $Pr_{\sqsubseteq}$ is consistent and non-trivial. Write $\rho = \rho^{\mathcal{V}}_{Pr_{\sqsubseteq}}$, so that

$$\mathcal{V}; Pr \overset{\rho}{\Longrightarrow} \mathcal{V}''; Pr\rho \overset{\theta''}{\Longrightarrow}{}^{*} \mathcal{V}'; \varnothing,$$

    and suppose that $\theta'|_{\mathcal{V}} \in Sol(Pr)$.
    By Theorem 6.18, $\theta''|_{\mathcal{V}''} \in Sol(Pr\rho)$. It is a fact that $\mathcal{V}'' = \mathcal{V} \cup \mathcal{V}''^{\mathcal{V}}_{Pr_{\sqsubseteq}}$, so $fV(Pr\rho) \subseteq \mathcal{V}''$. By Lemma 6.21, $\theta'|_{\mathcal{V}} \in Sol(Pr_{\sqsubseteq})$. By Theorem 5.20 and Lemma 6.9, $\rho \circ (\theta'|_{\mathcal{V}} {-} \rho) \in Sol(Pr)$. By Lemma 6.1, $\theta'|_{\mathcal{V}} {-} \rho \in Sol(Pr\rho)$.
    By inductive hypothesis $\theta''|_{\mathcal{V}} \leq \theta'|_{\mathcal{V}} {-} \rho$. By Lemma 6.19, $\rho \circ \theta''|_{\mathcal{V}} \leq \rho \circ (\theta'|_{\mathcal{V}} {-} \rho)$. It is a fact that $\rho \circ (\theta''|_{\mathcal{V}}) = (\rho \circ \theta'')|_{\mathcal{V}}$. By Theorem 5.20 and Lemma 6.20, $(\rho \circ \theta'')|_{\mathcal{V}} \leq \theta'|_{\mathcal{V}}$ as required.

□

Lemma 6.24. 1. *Suppose* $fa(s) {\subseteq} \pi \cdot S$ *and* $X^S \notin fV(s)$. *Write* $\chi {=} [X^S{:=}\pi^{-1} \cdot s]$.
If $\mathcal{V}; Pr \overset{\chi}{\Longrightarrow} \mathcal{V}; Pr'$ *with* (**I1**) *or* (**I2**) *then* $\theta \in Sol(Pr)$ *implies* $\theta {-} X^S \in Sol(Pr')$.

2. *If* $\mathcal{V}; Pr \overset{\rho}{\Longrightarrow} \mathcal{V}'; Pr'$ *with* (**I3**) *then* $\theta \in Sol(Pr)$ *implies* $\theta {-} \rho \in Sol(Pr')$.

Proof. 1. We consider the case of (**I1**); the case of (**I2**) is similar. Suppose $Pr = \pi \cdot X^S \stackrel{?}{=} s,\ Pr''$ so that $\mathcal{V}; \pi \cdot X^S \stackrel{?}{=} s,\ Pr'' \overset{\chi}{\Longrightarrow} \mathcal{V}; Pr''\chi$. Now suppose $\theta \in Sol(Pr)$. By Lemma 6.9 and Theorem 6.11, $\chi \circ (\theta {-} X^S)) \in Sol(Pr)$. By Lemma 6.1, $\theta {-} X^S \in Sol(Pr\chi)$. It follows that $\theta {-} X^S \in Sol(Pr''\chi)$ as required.

2. Suppose $Pr_{\sqsubseteq}$ is consistent and non-trivial. Write $\rho = \rho^{\mathcal{V}}_{Pr_{\sqsubseteq}}$, so that $\mathcal{V}; Pr \stackrel{\rho}{\Longrightarrow} \mathcal{V}''; Pr\rho$. Now suppose $\theta \in Sol(Pr)$. By Lemma 6.9 and Theorem 5.20, $\rho \circ (\theta - \rho) \in Sol(Pr)$. By Lemma 6.1, $\theta - \rho \in Sol(Pr\rho)$ as required.

□

Theorem 6.25. *Given a problem Pr, if the algorithm of Definition 6.6 succeeds then it returns a principal solution; if it fails then there is no solution.*

Proof. If the algorithm succeeds we use Theorem 6.23. Otherwise, the algorithm generates an element of the form $\mathsf{f}(r_1, \ldots, r_n) \stackrel{?}{=} \mathsf{f}(r'_1, \ldots, r'_{n'})$ where $n \neq n'$, $\mathsf{f}(\ldots) \stackrel{?}{=} \mathsf{g}(\ldots)$, $\mathsf{f}(\ldots) \stackrel{?}{=} [a]s$, $\mathsf{f}(\ldots) \stackrel{?}{=} a$, $[a]r =_\alpha a$, $[a]r =_\alpha b$, $a \stackrel{?}{=} b$, a Pr such that $Pr_{\sqsubseteq}$ is inconsistent, or $\pi \cdot X^S \stackrel{?}{=} r$ or $r \stackrel{?}{=} \pi \cdot X^S$ where $X^S \in fV(r)$. By arguments on syntax and size of syntax, no solution to the reduced problem exists. It follows by Lemma 6.24 that no solution to Pr exists. □

7. Lambda-term syntax

We want to relate permissive nominal unification with higher-order pattern unification as promised in the Introduction. In this section we recall the syntax and operational semantics of the λ-calculus. It is convenient to match the variables of permissive nominal terms with those of the λ-calculus. Therefore, when we define λ-terms' syntax in Definition 7.1, we use the same atoms and unknowns as we used in Definition 2.6; both behave like ordinary variables (with capture-avoiding substitution). As in permissive nominal terms we unify on the unknowns, but we do not care about the permission sets so we will let $X, Y, Z, \ldots$ range over distinct unknowns (without superscripts).

Definition 7.1. Define **λ-terms** by:

$$g, h, \ldots ::= a \mid X \mid \mathsf{f} \mid \lambda a.g \mid g'g$$

Here f ranges over term-formers, a ranges over atoms, and X ranges over unknowns.
g, h, k will range over λ-terms.

Definition 7.2. Define a **permutation action** by:

$$\pi{\cdot}a \equiv \pi(a) \quad \pi{\cdot}X \equiv X \quad \pi{\cdot}\mathsf{f} \equiv \mathsf{f} \quad \pi{\cdot}(\lambda a.g) \equiv \lambda\pi(a).(\pi{\cdot}g) \quad \pi{\cdot}(g'g) \equiv (\pi{\cdot}g')(\pi{\cdot}g)$$

Definition 7.3. Define **free atoms** by:

$$fa(a){=}\{a\} \quad fa(X){=}\varnothing \quad fa(\mathsf{f}){=}\varnothing \quad fa(\lambda a.g){=}fa(g)\backslash\{a\} \quad fa(g'g){=}fa(g')\cup fa(g)$$

Definition 7.4. Define **α-equivalence** $=_\alpha$ inductively by the rules in Figure 6.

$$\frac{}{a =_\alpha a}\ (\lambda{=_\alpha}\mathbf{a}) \qquad \frac{g =_\alpha h}{\lambda a.g =_\alpha \lambda a.h}\ (\lambda{=_\alpha}\lambda\mathbf{aa}) \qquad \frac{(b\ a)\cdot g =_\alpha h \quad b \notin fa(g)}{\lambda a.g =_\alpha \lambda b.h}\ (\lambda{=_\alpha}\lambda\mathbf{ab})$$

$$\frac{}{\mathsf{f} =_\alpha \mathsf{f}}\ (\lambda{=_\alpha}\mathsf{f}) \qquad \frac{}{X =_\alpha X}\ (\lambda{=_\alpha}\mathbf{X}) \qquad \frac{g =_\alpha g' \quad h =_\alpha h'}{gh =_\alpha g'h'}\ (\lambda{=_\alpha}\mathbf{p})$$

Figure 6: α-equivalence on λ-terms.

It is not hard to prove that Definition 7.4 does indeed specify the usual α-equivalence relation on λ-terms. Our definition is designed to match the definition of α-equivalence on nominal terms (Definition 2.13). This makes later results easier to prove (for example Theorem 8.12).

Lemma 7.5 to Proposition 7.12 mirror similar results for permissive nominal terms. The proofs of Lemmas 7.5 to 7.7 are by induction on g.

Lemma 7.5. *If* $\pi|_{fa(g)} = \pi'|_{fa(g)}$ *then* $\pi \cdot g =_\alpha \pi' \cdot g$.

Lemma 7.6. $\pi \cdot (\pi' \cdot g) \equiv (\pi \circ \pi') \cdot g$

Lemma 7.7. $fa(\pi \cdot g) = \pi \cdot fa(g)$.

Lemma 7.8. $g =_\alpha h$ *implies* $\pi \cdot g =_\alpha \pi \cdot h$.

Proof. By induction on the derivation of $g =_\alpha h$. We consider one case:

- The case $(\lambda{=_\alpha}\lambda\mathbf{ab})$. By inductive hypothesis $\pi \cdot ((b\ a) \cdot g) =_\alpha \pi \cdot h$. By Lemma 7.6, $\pi \cdot ((b\ a) \cdot g) \equiv (\pi \circ (b\ a)) \cdot g$. It is a fact that $\pi \circ (b\ a) = (\pi(b)\ \pi(a)) \circ \pi$, therefore $(\pi(b)\ \pi(a)) \cdot (\pi \cdot g) =_\alpha \pi \cdot h$, by Lemma 7.6. By Lemma 7.7, $\pi(b) \notin fa(\pi \cdot g)$. Using $(\lambda{=_\alpha}\lambda\mathbf{ab})$, we obtain $\lambda\pi(a).\pi{\cdot}g =_\alpha \lambda\pi(b).\pi{\cdot}h$. The result follows from Definition 7.2.

□

Lemma 7.9. *If* $g =_\alpha h$ *then* $fa(g) = fa(h)$.

Proof. By induction on the derivation of $g =_\alpha h$. We consider one case;

- The case $(\lambda{=_\alpha}\lambda\mathbf{ab})$. Suppose $\lambda a.g =_\alpha \lambda b.h$ using $(\lambda{=_\alpha}\lambda\mathbf{ab})$, where $b \notin fa(g)$. We aim to show $fa(\lambda a.g) = fa(\lambda b.h)$, or, $fa(g) \setminus \{a\} = fa(h) \setminus \{b\}$. As $b \notin fa(g)$ we have $fa(g) \setminus \{a\} = (b\ a) \cdot fa(g) \setminus \{b\}$. Using Lemma 7.7, $(b\ a) \cdot fa(g) \setminus \{b\} = fa((b\ a) \cdot g) \setminus \{b\}$. By inductive hypothesis $fa((b\ a) \cdot g) = fa(s)$, as required.

□

Definition 7.10. Define a notion of **size** on λ-terms by:

$$size(a) = 0 \quad size(X) = 0 \quad size(\mathsf{f}) = 0 \quad size(g'g) = size(g') + size(g)$$
$$size(\lambda a.g) = 1 + size(g)$$

Lemma 7.11. $size(g) = size(\pi \cdot g)$

Proof. By induction on g. □

Proposition 7.12. $=_\alpha$ *is transitive, reflexive, and symmetric.*

Proof. See Appendix Appendix A. We use Lemmas 7.5, 7.6, 7.7, 7.8, 7.9 and 7.11. □

Definition 7.13. Call a function σ from unknowns to λ-terms a (λ-calculus) **substitution**. σ will range over substitutions (and later so will ρ; Definition 9.17).

We will write $[h/X]$ for the substitution which maps X to h and maps all other Y to Y.

Definition 7.14. Define the **capture-avoiding substitution** action $g\sigma$ on λ-terms by:

$$\begin{array}{llll} a\sigma \equiv a & X\sigma \equiv \sigma(X) & \mathsf{f}\sigma \equiv \mathsf{f} & (g'g)\sigma \equiv (g'\sigma)(g\sigma) \\ (\lambda a.g)\sigma \equiv \lambda a.(g\sigma) & & (a \notin \bigcup\{fa(\sigma(X)) \mid X \in fV(g)\}) & \\ (\lambda a.g)\sigma \equiv \lambda b.(((b\ a)\cdot g)\sigma) & & (a \in \bigcup\{fa(\sigma(X)) \mid X \in fV(g)\}) & \end{array}$$

In the final clause, 'b fresh' denotes a fixed but arbitrary choice of fresh b (so $b \notin fa(g)$ and $b \notin \bigcup\{fa(\sigma(X)) | X \in fV(g)\}$).

Definition 7.15. Define **composition** $\sigma \circ \sigma'$ by: $(\sigma \circ \sigma')(X) \equiv (\sigma(X))\sigma'$. This mirrors the definition for substitutions on permissive terms, given in Definition 3.7.

Lemma 7.16. $g\sigma\sigma' =_\alpha g(\sigma \circ \sigma')$

Proof. By induction on $size(g)$. □

We also need a substitution action on atoms so that we can talk about $\alpha\beta$-convertibility — the distinction between unknowns and atoms is rather artificial here, but in the context of relating to permissive nominal terms it is useful to maintain it:

Definition 7.17. Define a **capture-avoiding substitution** $g[h/a]$ by:

$$\begin{array}{llll} a[h/a] \equiv h & b[h/a] \equiv b & X[h/a] \equiv X & \mathsf{f}[h/a] \equiv \mathsf{f} \\ (g'g)[h/a] \equiv (g'[h/a])(g[h/a]) & & (\lambda a.g)[h/a] \equiv \lambda a.g & \\ (\lambda a.g)[h/a] \equiv \lambda b.(((b\ a)\cdot g)[h/a]) & (b \text{ fresh}) & & \end{array}$$

In the final clause, 'b fresh' denotes a fixed but arbitrary choice of b such that $b \notin fa(h) \cup fa(g)$.

Definition 7.18. Let **$\alpha\beta$-equivalence** $=_{\alpha\beta}$ be the least transitive, reflexive, symmetric relation such that $(\lambda a.g)h =_{\alpha\beta} g[h/a]$ and closed under the rules of Definition 7.4.

We define unification problems as usual and write '$g \stackrel{?}{=} h$' for an equality considered as part of a unification problem. σ solves a problem when $g\sigma =_{\alpha\beta} h\sigma$ for every $g \stackrel{?}{=} h$ in the problem, as usual.

We conclude with definions of *pattern* and *pattern substitution* [32, 31]. Recall that we work in an untyped λ-term syntax.

Definition 7.19. Let ϕ map each unknown X to a natural number which we call its **arity**. Define ϕ**-patterns**, a subset of λ-terms, by:

$$q, r, \ldots ::= a \mid X a_1 \ldots a_{\phi(X)} \mid \mathsf{f} q_1 \ldots q_n \mid \lambda a.q$$

Call q a **pattern** when it is a ϕ-pattern for some ϕ. $q, r, \ldots$ will range over patterns.

Call σ a ϕ**-pattern substitution** when every $\sigma(X)$ is a ϕ-pattern. Call σ a **pattern substitution** when σ is a ϕ-pattern substitution for some ϕ.

So g is a pattern when there exists some ϕ such that every X in g occurs as $X a_1 \ldots a_{\phi(X)}$. This is not quite a typing constraint, but it achieves part of what a typing system would achieve; that within g, for each X, X is consistently applied to a list of atoms of the same length. Note that the translation of Definition 8.3 below, produces terms of this form.

8. Translating nominal terms to lambda-term syntax

8.1. The translation, and its soundness

We define the translation from permissive nominal terms to λ-terms, and show that it is sound in the sense that α-convertible permissive nominal terms map to α-convertible λ-terms (Theorem 8.6). The translation involves a vector of atoms D; we discuss where this comes from in Subsection 8.2.

Definition 8.1. Call a finite list of distinct atoms a **vector**. C, D range over vectors. Write $[a_1, \ldots, a_n]$ for the vector containing $a_1, \ldots, a_n$ in that order.

Definition 8.2. Suppose $A \subseteq \mathbb{A}$. Write $C \cap A$ for the vector of atoms in C that occur in A, in order; thus $[a_1, a_2, a_3] \cap \{a_1, a_3, a_5\} = [a_1, a_3]$. Write $C \subseteq A$ when every atom in C is in A. Write $A \subseteq C$ when every atom in A is in C.

Definition 8.3. Translate a nominal term r to a λ-term $[\![r]\!]^D$ by:

$$[\![a]\!]^D \equiv a \quad [\![\pi \cdot X^S]\!]^D \equiv X^S \pi(d_1) \ldots \pi(d_n) \quad ([d_1, \ldots, d_n] = D \cap S)$$
$$[\![[a]r]\!]^D \equiv \lambda a.[\![r]\!]^D \quad [\![\mathsf{f}(r_1, \ldots, r_n)]\!]^D \equiv \mathsf{f}[\![r_1]\!]^D \ldots [\![r_n]\!]^D$$

Lemma 8.4. $[\![\pi \cdot r]\!]^D \equiv \pi \cdot [\![r]\!]^D$

Proof. By induction on r. □

Lemma 8.5 is useful for the proof of Theorem 8.6:

Lemma 8.5. $fa([\![r]\!]^D) \subseteq fa(r)$.

Proof. By induction on r. We consider only one case:

- The case $\pi \cdot X^S$. We reason as follows:

$$\begin{array}{lll} fa([\![\pi \cdot X^S]\!]^D) = fa(X^S\pi(d_1)\ldots\pi(d_n)) & & \text{Definition 8.3} \\ = fa(\pi(d_1)) \cup \ldots \cup fa(\pi(d_n)) & & \text{Definition 7.3} \\ = \pi \cdot (fa(d_1) \cup \ldots \cup fa(d_n)) & & \text{Fact} \\ \subseteq \pi \cdot fa(X^S) & & \text{Definition 2.11} \\ = fa(\pi \cdot X^S) & & \text{Lemma 2.17} \end{array}$$

  The result follows.

□

Theorem 8.6 (Soundness). *If $r =_\alpha s$ then $[\![r]\!]^D =_\alpha [\![s]\!]^D$.*

Proof. By induction on the size of r (Definition Appendix A.1). We reason by cases on the last rule in the derivation of $r =_\alpha s$:

- The cases $(=_\alpha \mathbf{a})$, $(=_\alpha \mathsf{f})$ and $(=_\alpha [] \mathbf{aa})$ are straightforward.
- The case $(=_\alpha \mathbf{X})$. There are two cases:
  - The case $D \cap S = []$. Then $[\![\pi \cdot X^S]\!]^D = [\![\pi' \cdot X^S]\!]^D = X^S$. We use $(\lambda{=_\alpha}\mathbf{X})$. The result follows.
  - The case $D \cap S = [d_1, \ldots, d_n]$ and $n \geq 1$. By assumption $\pi|_{\delta(X)} = \pi'|_{\delta(X)}$. Then $\pi(d_i) = \pi'(d_i)$ for $1 \leq i \leq n$ and $[\![\pi \cdot X^S]\!]^D \equiv [\![\pi' \cdot X^S]\!]^D \equiv X^S\pi(d_1)\ldots\pi(d_n)$. We use $(\lambda{=_\alpha}\mathbf{p})$, $(\lambda{=_\alpha}\mathbf{X})$, and $(\lambda{=_\alpha}\mathbf{a})$. The result follows.
- The case $(=_\alpha [] \mathbf{ab})$. By assumption $(b\ a) \cdot r =_\alpha s$ and $b \notin fa(r)$. We choose fresh c so $c \notin fa(r) \cup fa(s)$. By Lemma 2.18, $(c\ a) \cdot r =_\alpha (c\ b) \cdot s$. By inductive hypothesis $[\![(c\ a) \cdot r]\!]^D =_\alpha [\![(c\ b) \cdot s]\!]^D$. Using $(\lambda{=_\alpha}\lambda\mathbf{aa})$, $\lambda c.[\![(c\ a) \cdot r]\!]^D =_\alpha \lambda c.[\![(c\ b) \cdot s]\!]^D$. By Lemma 8.4, $\lambda c.((c\ a) \cdot [\![r]\!]^D) =_\alpha \lambda c.((c\ b) \cdot [\![s]\!]^D)$. By Lemma 8.5, $c \notin fa([\![r]\!]^D) \cup fa([\![s]\!]^D)$. By $(\lambda{=_\alpha}\lambda\mathbf{ab})$, $\lambda c.((c\ a) \cdot [\![r]\!]^D) =_\alpha \lambda a.[\![r]\!]^D$ and $\lambda c.((c\ b) \cdot [\![s]\!]^D) =_\alpha \lambda b.[\![s]\!]^D$. Using Proposition 7.12, $\lambda a.[\![r]\!]^D =_\alpha \lambda b.[\![s]\!]^D$. By Definition 8.3, $[\![[a]r]\!]^D =_\alpha [\![[b]s]\!]^D$, as required.

□

8.2. *Capturable atoms; injectivity and minimality*

We investigate the converse to Theorem 8.6; if the translations of two terms are α-convertible, then so are the two terms. This is injectivity (Theorem 8.12). The translation $[\![r]\!]^D$ (Definition 8.3) is parameterised by a vector D. Levy and Villaret introduced a translation [29, Definition 2] (for nominal, not permissive nominal terms); they used *all* the atoms in r. This is a safe choice, but we will also show that the smaller set of *capturable* atoms in r (Definition 8.7) is consistent with injectivity — and that the capturable atoms are the minimal such set (Theorem 8.14).

Definition 8.7. Define the **capturable atoms** of a term (with respect to a set of atoms) $capt_A(r)$ inductively by:

$$\begin{array}{ll} capt_A(a) = \varnothing & capt_A(\pi \cdot X^S) = (nontriv(\pi) \cup A) \cap S \\ capt_A([a]r) = capt_{A \cup \{a\}}(r) & capt_A(\mathsf{f}(r_1, \ldots, r_n)) = \bigcup_{1 \leq i \leq n} capt_A(r_i) \end{array}$$

Write $capt_\varnothing(r)$ as $capt(r)$.

For instance, if $S = (\mathbb{A}^< \cup \{a\}) \setminus \{b\}$, then $capt([a][b]X^S) = \{a\}$ and $capt((b\ a) \cdot X^S) = \{a\}$.

Lemma 8.8. *If $a \notin fa(r)$ then $capt_A(r) = capt_{A \cup \{a\}}(r)$.*

Proof. By induction on r.

- The cases b, $\mathsf{f}(r_1, \ldots, r_n)$ and $[a]r$ are straightforward.
- The case $[b]r$. If $a \notin fa([b]r)$ then $a \notin fa(r)$ by Definition 2.11. We then have:

$$\begin{array}{lll} capt_A([b]r) & = capt_{A \cup \{b\}}(r) & \text{Definition 8.7} \\ & = capt_{A \cup \{b\} \cup \{a\}}(r) & \text{Inductive hypothesis} \\ & = capt_{A \cup \{a\}}([b]r) & \text{Definition 8.7} \end{array}$$

  The result follows.
- The case $\pi \cdot X^S$. If $a \notin fa(\pi \cdot X^S)$ then $a \notin \pi \cdot S$. By Definition 8.7, $capt_A(\pi \cdot X^S) = (nontriv(\pi) \cup A) \cap S$. Further, $capt_{A \cup \{a\}}(\pi \cdot X^S) = (nontriv(\pi) \cup A \cup \{a\}) \cap S$. If $\pi(a) = a$ then $a \notin S$. If $\pi(a) \neq a$ then $a \in nontriv(\pi)$, as required.

□

Lemma 8.9. *If $nontriv(\pi) \subseteq A$ then $capt_A(\pi \cdot r) = capt_A(r)$.*

Proof. See Appendix Appendix A. We use Lemma 2.16. □

Corollary 8.10. *If $b \notin fa(r)$ then $capt_A([a]r) = capt_A([b](b\ a) \cdot r)$.*

Proof. See Appendix Appendix A. We use Lemmas 8.8, 8.9, and 2.17. □

Capturable atoms is a canonical notion, in the sense that it is preserved by α-equivalence:

Lemma 8.11. *If $r =_\alpha s$ then $capt_A(r) = capt_A(s)$.*

Proof. By induction on the derivation of $r =_\alpha s$. We consider one case:

- The case of $(=_\alpha[\mathbf{b}])$. Suppose $b \notin fa(r)$, $(b\ a) \cdot r =_\alpha s$ and $s \equiv [b](b\ a) \cdot r$. The result follows by Corollary 8.10.

□

Provided that D is 'large enough', α-equivalence of translated terms implies α-equivalence of the original terms:

Theorem 8.12 (Injectivity)**.** *Let D be a vector. Let r and s be nominal terms and let $A, B \subseteq \mathbb{A}$ be finite. Suppose $capt_A(r) \cup capt_B(s) \subseteq D$. Then*

$$[\![r]\!]^D =_\alpha [\![s]\!]^D \quad \textit{implies} \quad r =_\alpha s.$$

As a corollary, if $capt(r) \cup capt(s) \subseteq D$ and $[\![r]\!]^D =_\alpha [\![s]\!]^D$ then $r =_\alpha s$ and $capt_A(r) = capt_A(s)$ for all A.

Proof. For the first part, we work by induction on the size of r (Definition Appendix A.1), reasoning by cases on the last rule in the derivation of $[\![r]\!]^D =_\alpha [\![s]\!]^D$:

- The cases $(\lambda{=_\alpha}\mathbf{a})$ and $(\lambda{=_\alpha}\lambda\mathbf{aa})$. Easy.

- The case $(\lambda{=_\alpha}\lambda\mathbf{ab})$. Suppose $(b\ a)\cdot[\![r]\!]^D =_\alpha [\![s]\!]^D$, $b \notin fa([\![r]\!]^D)$ and $capt_A([a]r) \cup capt_B([b]s) \subseteq D$.
  We choose fresh c (so $c \notin fa(r) \cup fa(s)$ and $c \notin fa([\![r]\!]^D) \cup fa([\![s]\!]^D)$). By Lemma 7.5, $(c\ a)\cdot[\![r]\!]^D =_\alpha (c\ b)\cdot[\![s]\!]^D$. By Lemma 8.4:

  $$[\![(c\ a)\cdot r]\!]^D =_\alpha [\![(c\ b)\cdot s]\!]^D.$$

  By Corollary 8.10 and Definition 8.7:

  $$capt_{A\cup\{c\}}((c\ a)\cdot r) \cup capt_{B\cup\{c\}}((c\ b)\cdot s) \subseteq D.$$

  By inductive hypothesis $(c\ a)\cdot r =_\alpha (c\ b)\cdot s$, and using $(=_\alpha[]\mathbf{ab})$, and Proposition 7.12, $[a]r =_\alpha [b]s$ as required.
- The case $(\lambda{=_\alpha}\mathbf{app})$. From Definition 8.3, there are two possibilities:
  - The case $\mathsf{f}(r_1,\ldots,r_n)$ and $\mathsf{f}(s_1,\ldots,s_n)$ and $\mathsf{f}[\![r_1]\!]^D\ldots[\![r_n]\!]^D =_\alpha \mathsf{f}[\![s_1]\!]^D\ldots[\![s_n]\!]^D$. Then $[\![r_i]\!]^D =_\alpha [\![s_i]\!]^D$ for $1 \le i \le n$. By inductive hypothesis $r_i =_\alpha s_i$ for $1 \le i \le n$. The result follows from $(=_\alpha\mathsf{f})$.
  - The case $\pi\cdot X^S$ and $\pi'\cdot X^S$ and $X^S\pi(d_1)\ldots\pi(d_n) =_\alpha X^S\pi'(d_1)\ldots\pi'(d_n)$ where $[d_1,\ldots,d_n] = D\cap S$.
    Then $\pi(d_i) =_\alpha \pi'(d_i)$ for $1 \le i \le n$, and so $\pi|_{D\cap S} = \pi'|_{D\cap S}$. By assumption, $capt(\pi\cdot X^S) \subseteq D$, and by definition, $\pi|_S = \pi'|_S$. We conclude with $(=_\alpha\mathbf{X})$.
- The case $(\lambda{=_\alpha}\mathbf{X})$. From the form of the translation it must be that $r = \pi\cdot X^S$ and $s = \pi'\cdot X^S$ and $nontriv(\pi)\cap S = \varnothing = nontriv(\pi')\cap S$. The result follows from $(=_\alpha\mathbf{X})$.

The corollary follows from the first part, and by Lemma 8.11. □

Lemma 8.13. *$a \in capt_A(r)$ implies $X^S \in fV(r)$ exists such that $a \in S$.*

Proof. See Appendix Appendix A. □

In Theorem 8.12 we used a notion of a 'large enough' vector D, based on capturable atoms. Theorem 8.14 shows how this bound is precise:

Theorem 8.14 (Minimality)**.** *If $capt(r) \not\subseteq D$ then there exists some s such that $r \not=_\alpha s$ and $[\![r]\!]^D =_\alpha [\![s]\!]^D$.*

Proof. Suppose $a \in capt(r)$ and $a \notin D$. By Lemma 8.13 $X^S \in fV(r)$ exists such that $a \in S$. We choose $c \in S\backslash D$ and take $s \equiv r[X^S{:=}(c\ a)\cdot X^S]$. It is a fact that $X^S \not=_\alpha (c\ a)\cdot X^S$ whilst $[\![X^S]\!]^D =_\alpha [\![(c\ a)\cdot X^S]\!]^D$. An easy calculation shows $r \not=_\alpha r[X^S{:=}(c\ a)\cdot X^S]$ and $[\![r]\!]^D =_\alpha [\![r[X^S{:=}(c\ a)\cdot X^S]]\!]^D$. □

9. Translating substitutions; relating solutions of nominal and pattern unification problems

9.1. Translating substitutions

We extend the translation to substitutions. Our main result is Theorem 9.3: we can read this as a compositionality result for permissive nominal substitutions acting on terms with respect to the translation. Given the compositionality result it is easy to

prove that the translation also preserves the natural instantiation ordering of solutions to unification problems (Corollary 9.5).

Translating substitutions introduces a problem: θ may solve Pr but in substituting it may introduce new capturable atoms (consider $\theta = [X^S{:=}[c]Z^S]$ solving $\{X^S \stackrel{?}{=} X^S\}$, where $c \in S$). This motivates introducing a second vector, to account for the capturable atoms 'after' the substitution. Accordingly, we will introduce another vector E that contains at least the capturable atoms of θ.

Definition 9.1. Define $[\![\theta]\!]^E_D$ by:

$$[\![\theta]\!]^E_D(X^S) = \lambda d_1 \ldots \lambda d_n.[\![\theta(X^S)]\!]^E \text{ where } [d_1, \ldots, d_n] = D \cap S.$$

Lemma 9.2 is useful in the proof of Theorem 9.3:

Lemma 9.2. *If* $nontriv(\pi) \cap fa(g) \subseteq \{d_1, \ldots, d_n\}$ *then*

$$(\lambda d_1 \ldots \lambda d_n.g)\pi(d_1) \ldots \pi(d_n) =_{\alpha\beta} \pi \cdot g.$$

Proof. By induction on g. We consider one case:

- The case $\lambda a.g$. Choose a' fresh (so a' does not appear in $\{d_1, \ldots, d_n\}$, $nontriv(\pi)$, or g).

$$\begin{array}{lll}
h =_{\alpha\beta} (\lambda a.g)[\pi(d_1)/d_1] \ldots [\pi(d_n)/d_n] & & \text{Definition 7.18} \\
=_{\alpha} (\lambda a'.(a'\ a) \cdot g)[\pi(d_1)/d_1] \ldots [\pi(d_n)/d_n] & & \text{Definition 7.4} \\
\equiv \lambda a'.((a'\ a) \cdot g)[\pi(d_1)/d_1] \ldots [\pi(d_n)/d_n]) & & \text{Definition 7.17, } a' \text{ fresh} \\
=_{\alpha\beta} \lambda a'.(\pi \cdot ((a'\ a) \cdot g)) & & \text{Inductive hypothesis} \\
\equiv \pi \cdot \lambda a'.((a'\ a) \cdot g) & & \text{Definition 7.2} \\
=_{\alpha} \pi \cdot \lambda a.g & & \text{Definition 7.4}
\end{array}$$

□

Theorem 9.3 is a compositionality result for permissive nominal substitutions acting on terms with respect to the translation:

Theorem 9.3. *If* $capt(r) \subseteq D$ *then* $[\![r\theta]\!]^E =_{\alpha\beta} [\![r]\!]^D[\![\theta]\!]^E_D$.

Proof. By induction on r.

- The cases a and $\mathsf{f}(r_1, \ldots, r_n)$ are routine.
- The case $\pi \cdot X^S$. Let $d_1, \ldots, d_n$ be $D \cap S$ and $[\![\theta]\!]^E_D(X^S) = \lambda d_1 \ldots \lambda d_n.[\![\theta(X^S)]\!]^E$ by Definition 9.1. Then:

$$\begin{array}{lll}
[\![(\pi \cdot X^S)\theta]\!]^E \equiv [\![\pi \cdot \theta(X^S)]\!]^E & & \text{Definition 3.2} \\
\equiv \pi \cdot [\![\theta(X^S)]\!]^E & & \text{Lemma 8.4} \\
=_{\alpha\beta} (\lambda d_1 \ldots \lambda d_n.[\![\theta(X^S)]\!]^E)\pi(d_1) \ldots \pi(d_n) & & \text{Lemma 9.2} \\
\equiv (X^S\pi(d_1) \ldots \pi(d_n))[\![\theta]\!]^E_D & & \text{Definition 9.1} \\
\equiv [\![\pi \cdot X^S]\!]^D[\![\theta]\!]^E_D & & \text{Definition 8.3}
\end{array}$$

The use of Lemma 9.2 above is valid, because: By assumption $capt(\pi \cdot X^S) \subseteq D$. By Definition 8.7 $capt(\pi \cdot X^S) = nontriv(\pi) \cap S$, so $nontriv(\pi) \cap S \subseteq D$. By assumption in Definition 3.1, $fa(\theta(X^S)) \subseteq S$. It follows from Definition 8.3 that $fa([\![\theta(X^S)]\!]^E) \subseteq S$, and so that $nontriv(\pi) \cap fa([\![\theta(X^S)]\!]^E) \subseteq D$.
The result follows.

- The case $[a]r$. Choose b fresh, so $b \notin fa([\![\theta(X^S)]\!]^E_D)$ for every $X^S \in fV(r)$ and $b \notin fa(r)$. Then:

$$\begin{array}{rll}
[\![([a]r)\theta]\!]^E & =_\alpha & [\![([b]((b\ a)\cdot r))\theta]\!]^E \quad \text{Definition 2.13, Theorem 8.6, Lemma 3.6} \\
& \equiv & \lambda b.([\![((b\ a)\cdot r)\theta)]\!]^E \quad \text{Definitions 3.2 and 8.3} \\
& =_{\alpha\beta} & \lambda b.(([\![(b\ a)\cdot r]\!]^D)[\![\theta]\!]^E_D) \quad \text{Inductive hypothesis} \\
& \equiv & (\lambda b.[\![(b\ a)\cdot r]\!]^D)[\![\theta]\!]^E_D \quad \text{Definition 7.17, } b \text{ fresh} \\
& \equiv & [\![[b]((b\ a)\cdot r)]\!]^D[\![\theta]\!]^E_D \quad \text{Definition 8.3} \\
& =_\alpha & [\![[a]r]\!]^D[\![\theta]\!]^E_D \quad \text{Definition 2.13, Theorem 8.6, Lemma 3.6}
\end{array}$$

  The result follows.

□

Recall the instantiation ordering $\theta_1 \leq \theta_2$ from Definition 6.12. Similarly:

Definition 9.4. Write $\sigma_1 \leq \sigma_2$ when there exists some σ' such that $X\sigma_2 =_{\alpha\beta} X(\sigma_1 \circ \sigma')$, for any X. Call $\leq$ the **instantiation ordering**.

We can leverage Theorem 9.3 to prove a corollary, describing a sense in which the instantiation ordering $\theta_1 \leq \theta_2$ of Definition 6.12 translates to the instantiation ordering of Definition 9.4:

Corollary 9.5. *Suppose* $\bigcup_{X^S} capt(\theta_2(X^S)) \subseteq E$.
If $\theta_1 \leq \theta_2$ *then* $[\![\theta_1]\!]^E_D \leq [\![\theta_2]\!]^E_D$.

Proof. Suppose $\theta_1 \leq \theta_2$. By definition (Definition 6.12) there exists some θ' such that $X^S\theta_1 =_\alpha X^S(\theta_2 \circ \theta')$ always. We reason as follows, for any unknown X^S:

$$\begin{array}{rll}
[\![X^S]\!]^D[\![\theta_2]\!]^E_D & =_{\alpha\beta} & [\![X^S\theta_2]\!]^E \quad \text{Theorem 9.3} \\
& =_\alpha & [\![X^S(\theta_1 \circ \theta')]\!]^E \quad \text{Theorem 8.6} \\
& \equiv & [\![(X^S\theta_1)\theta')]\!]^E \quad \text{Lemma 3.8} \\
& =_{\alpha\beta} & [\![X^S\theta_1]\!]^E[\![\theta']\!]^E_E \quad \text{Theorem 9.3, } capt(\theta_1(X^S)) \subseteq E \\
& =_{\alpha\beta} & ([\![X^S]\!]^D[\![\theta_1]\!]^E_D)[\![\theta']\!]^E_E \quad \text{Theorem 9.3} \\
& \equiv & [\![X^S]\!]^D([\![\theta_1]\!]^E_D \circ [\![\theta']\!]^E_E) \quad \text{Lemma 7.16}
\end{array}$$

The result follows. □

In Corollary 9.5, the precondition $\bigcup_{X^S} capt(\theta_2(X^S)) \subseteq E$ is necessary to prevent θ_2 from introducing infinitely many capturable atoms. The 'complexity' of θ_1 is unconstrained. In practice it is likely that we will care about a particular finite set of unknowns $\mathcal{V}$ (for example, $fV(Pr)$ for some Pr), and the precondition can be correspondingly refined to consider just $X^S \in \mathcal{V}$.

9.2. *Translating permissive nominal unification to pattern unification; soundness, weak completeness*

The main result of this subsection is Theorem 9.16.

It says that if D and E are 'large enough', then θ solves Pr if and only if $[\![\theta]\!]^E_D$ solves $[\![Pr]\!]^D$. We call this 'soundness and weak completeness', to distinguish from a stronger completeness result we prove in Subsection 9.3.

Definition 9.6. An **equation** is a pair $r \stackrel{?}{=} s$. A **unification problem** Pr is a finite set of equations. A **solution** to Pr is a θ such that $r\theta =_\alpha s\theta$ for all $r \stackrel{?}{=} s \in Pr$.

Definition 9.7. If $D = [d_1, \ldots, d_n]$ and $Pr = \{r_1 \stackrel{?}{=} s_1, \ldots\}$ then define $[\![Pr]\!]^D$ by:

$$[\![Pr]\!]^D = \left\{\lambda d_1 \ldots \lambda d_n.[\![r]\!]^D \stackrel{?}{=} \lambda d_1 \ldots \lambda d_n.[\![s]\!]^D \mid r \stackrel{?}{=} s \in Pr\right\}$$

For example, if $Pr = \{X^S \stackrel{?}{=} \mathsf{f}(Y^S, a, Z^S)\}$ where $S = \mathbb{A}^< \cup \{a, b\}$, then $[\![Pr]\!]^{[a]} = \{\lambda a.(X^S a) \stackrel{?}{=} \lambda a.(\mathsf{f}\ (Y^S a)\ a\ (Z^S a))\}$.

Definition 9.8 is a technical definition, and the results following it are technical lemmas required for Lemma 9.11, which is a key result for Lemma 9.13, which is itself needed for Theorem 9.16.

Definition 9.8. Define $\mathit{uncapt}(r)$ by:

$$\begin{array}{rl} \mathit{uncapt}(a) & = \varnothing \\ \mathit{uncapt}(\pi \cdot X^S) & = S \setminus \mathit{nontriv}(\pi) \\ \mathit{uncapt}([a]r) & = \mathit{uncapt}(r) \setminus \{a\} \\ \mathit{uncapt}(\mathsf{f}(r_1, \ldots, r_n)) & = \mathit{uncapt}(r_1) \cup \cdots \cup \mathit{uncapt}(r_n) \end{array}$$

$\mathit{uncapt}(r)$ is quite interesting; technically, it came about as the 'right definition' to make Lemmas 9.10, 9.11, and 9.12 work.[7] Intuitively, it may be useful to think of $\mathit{uncapt}(r)$ as collecting those atoms that are not necessarily in $\mathit{capt}(r)$, but which could feature in $\mathit{capt}([a]r)$ for some a.

For example, if $a \in S$ then $a \in \mathit{uncapt}(X^S)$ but

$$a \notin \mathit{uncapt}(a), \quad a \notin \mathit{uncapt}([b][a]X^S), \quad \text{and} \quad a \notin \mathit{uncapt}((b\ a) \cdot X^S).$$

Lemma 9.9. *If $A \subseteq B$ then $\mathit{capt}_A(r) \subseteq \mathit{capt}_B(r)$.*
As a corollary, $\mathit{capt}_A(r) \subseteq \mathit{capt}_A([a]r)$.

Proof. By induction on r. As $\mathit{capt}_A([a]r) = \mathit{capt}_{A\cup\{a\}}(r)$, the corollary follows. □

Lemma 9.10. *Suppose $a \in A$. Then if $a \in \mathit{uncapt}(r)$ then $a \in \mathit{capt}_A(r)$.*
As a corollary, $a \in \mathit{uncapt}(r)$ implies $a \in \mathit{capt}([a]r)$.

Proof. By induction on r.

- The cases a and $\mathsf{f}(r_1, \ldots, r_n)$ are straightforward.
- The case $[a]r$. $a \in \mathit{uncapt}([a]r)$ is impossible.
- The case $[b]r$. Suppose $a \in \mathit{uncapt}([b]r)$. By Definition 9.8 $a \in \mathit{uncapt}(r)$. By inductive hypothesis $a \in \mathit{capt}_A(r)$. By Lemma 9.9 $a \in \mathit{capt}_A([b]r)$ as required.
- The case $\pi \cdot X^S$. Suppose $a \in \mathit{uncapt}(\pi \cdot X^S)$, so $a \in S \setminus \mathit{nontriv}(\pi)$. By assumption $a \in A$ so $a \in (\mathit{nontriv}(\pi) \cup A) \cap S = \mathit{capt}_A(\pi \cdot X^S)$.

□

[7] Thanks to an anonymous referee for spotting the error in a previous proof.

Lemma 9.11. $capt_A(\pi \cdot r) \subseteq ((nontriv(\pi) \cup A) \cap uncapt(r)) \cup capt(r)$.

Proof. See Appendix Appendix A. We use Lemmas 9.9 and 9.10. □

Lemma 9.12. $uncapt(r) \subseteq fa(r)$.

Proof. By a routine induction on r using Definitions 2.11 and 9.8. □

Lemma 9.13. $capt_A(r\theta) \subseteq capt_A(r) \cup \bigcup_{X^S \in fV(r)} capt(\theta(X^S))$.

Proof. By induction on r:

- The cases a and $\mathsf{f}(r_1, \ldots, r_n)$. Straightforward.
- The case $[a]r$. We reason as follows:

$$\begin{array}{lll} capt_A([a](r\theta)) = capt_{A\cup\{a\}}(r\theta) & & \text{Definition 8.7} \\ \subseteq capt_{A\cup\{a\}}(r) \cup \bigcup_{X^S \in fV(r)} capt(\theta(X^S)) & & \text{Ind. hyp.} \\ = capt_A([a]r) \cup \bigcup_{X^S \in fV(r)} capt(\theta(X^S)) & & \text{Definition 8.7} \end{array}$$

  The result follows.
- The case $\pi \cdot X^S$. As $(\pi \cdot X^S)\theta \equiv \pi \cdot \theta(X^S)$, we reason as follows:

$$\begin{array}{lll} capt_A(\pi \cdot \theta(X^S)) \subseteq ((nontriv(\pi) \cup A) \cap uncapt(\theta(X^S))) & & \\ \qquad \cup capt(\theta(X^S)) & & \text{Lemma 9.11} \\ \subseteq ((nontriv(\pi) \cup A) \cap S) \cup capt(\theta(X^S)) & & \text{Lemma 9.12} \\ = capt_A(\pi \cdot X^S) \cup capt(\theta(X^S)) & & \text{Definition 8.7} \end{array}$$

  The result follows.

□

Remark 9.14. $capt(r\theta) \subseteq \bigcup_{fV(r)} capt(\theta(X^S))$ is not true in general. For example if $a \in S$ and $b \in S$ then $capt([a]X^S) = \{a\}$ and $capt([X^S{:=}[b]X^S]) = \{b\}$, and $capt(([a]X^S))\theta = \{a, b\} \not\subseteq \{b\}$.

Lemma 9.15. *Suppose* $capt(Pr) \subseteq D$ *and* $capt(Pr\theta) \subseteq E$. *Then* θ *solves* Pr *if and only if* $[\![\theta]\!]^E_D$ *solves* $[\![Pr]\!]^D$.

Proof. We reason as follows, where $D = [d_1, \ldots, d_n]$:

$$\begin{array}{lll} r\theta =_\alpha s\theta \Leftrightarrow [\![r\theta]\!]^E =_\alpha [\![s\theta]\!]^E & & \text{Theorems 8.6 and 8.12} \\ \Leftrightarrow [\![r]\!]^D[\![\theta]\!]^E_D =_{\alpha\beta} [\![s]\!]^D[\![\theta]\!]^E_D & & \text{Theorem 9.3} \\ \Leftrightarrow \lambda d_1.\ldots.\lambda d_n.[\![r]\!]^D[\![\theta]\!]^E_D =_{\alpha\beta} \lambda d_1.\ldots.\lambda d_n.[\![s]\!]^D[\![\theta]\!]^E_D & & \text{fact of } \lambda\text{-terms} \\ \Leftrightarrow (\lambda d_1.\ldots.\lambda d_n.[\![r]\!]^D)[\![\theta]\!]^E_D =_{\alpha\beta} (\lambda d_1.\ldots.\lambda d_n.[\![s]\!]^D)[\![\theta]\!]^E_D & & \text{no atom of } D \text{ free in } [\![\theta]\!]^E_D \end{array}$$

□

If D and E are 'large enough', then θ solves Pr if and only if the translation $[\![\theta]\!]^E_D$ solves the translation $[\![Pr]\!]^D$:

Theorem 9.16 (Soundness and weak completeness)**.** *Suppose*

$$capt(Pr) \subseteq D \qquad capt(\theta(X^S)) \subseteq E \text{ for all } X^S \in fV(Pr), \quad \text{and} \quad D \subseteq E.$$

Then θ *solves* Pr *if and only if* $[\![\theta]\!]^E_D$ *solves* $[\![Pr]\!]^D$.

Proof. Suppose $capt(Pr) \subseteq D$, $capt(\theta(X^S)) \subseteq E$ for all $X^S \in fV(Pr)$, and $D \subseteq E$. Then:

$$\begin{array}{ll} capt(Pr\theta) \subseteq capt(Pr) \cup \bigcup_{X^S \in fV(Pr)} capt(\theta(X^S)) & \text{Lemma 9.13} \\ \subseteq D \cup \bigcup_{X^S \in fV(Pr)} capt(\theta(X^S)) & capt(Pr) \subseteq D \\ \subseteq D \cup E & capt(\theta(X^S)) \subseteq E \\ \subseteq E & D \subseteq E \end{array}$$

By Lemma 9.15, θ solves Pr if and only if $[\![\theta]\!]^E_D$ solves $[\![Pr]\!]^D$. □

For example, $Pr = \{X^S \stackrel{?}{=} \mathsf{f}(Y^S, a, Z^S)\}$ where $S = \mathbb{A}^< \cup \{a, b\}$ translates to $[\![Pr]\!]^{[a]} = \{\lambda a.(X^S\, a) = \lambda a.(\mathsf{f}\,(Y^S\, a)\, a\, (Z^S\, a))\}$.

The solution $[X^S{:=}\mathsf{f}(W^S, a, b), Y^S{:=}W^S, Z^S{:=}b]$ with $S = \mathbb{A}^< \cup \{a, b\}$ translates to $[\![\theta]\!]^{[a,b]}_{[a]} = [X^S{:=}\lambda a.(\mathsf{f}\,(W^S\, a\, b)\, a\, b), Y^S{:=}\lambda a.(W^S\, a\, b), Z^S{:=}\lambda a.b]$.

9.3. Strong Completeness

The main result of this subsection is Theorem 9.30. This strengthens the completeness result of Theorem 9.16, in a certain sense, by expressing that a class of σ solving $[\![Pr]\!]^D$ all originate from θ solving Pr, in a suitable formal sense.

Definition 9.17. Call a bijection on unknowns a **renaming**. ρ will range over renamings. Each X is also a λ-term (Definition 7.1), so each ρ is also a substitution (Definition 7.13).

Lemma 9.18. *$fa(g) = fa(g\rho)$*

Proof. By induction on g. □

Lemma 9.19. *$g =_\alpha h$ if and only if $g\rho =_\alpha h\rho$.*

Proof. The left to right implication is by induction on the derivation of $g =_\alpha h$; right to left is by induction on the derivation of $g\rho =_\alpha h\rho$. We consider one case:

- The case $(\lambda{=_\alpha}\lambda\mathbf{ab})$. For the left to right implication, suppose $(b\ a) \cdot g =_\alpha h$ and $b \notin fa(g)$. By inductive hypothesis $((b\ a) \cdot g)\rho =_\alpha h\rho$. By Lemma 9.18, $b \notin fa(g\rho)$. It is a fact that $((b\ a) \cdot g)\rho \equiv (b\ a) \cdot (g\rho)$. It follows that $(b\ a) \cdot (g\rho) =_\alpha h\rho$. Using $(\lambda{=_\alpha}\lambda\mathbf{ab})$, $\lambda a.(g\rho) =_\alpha \lambda b.(h\rho)$. The result follows.
  For the right to left implication, suppose $(b\ a) \cdot (g\rho) =_\alpha h\rho$ and $b \notin fa(g\rho)$. It is a fact that $(b\ a) \cdot (g\rho) \equiv ((b\ a) \cdot g)\rho$. It follows by inductive hypothesis that $(b\ a) \cdot g =_\alpha h$. By Lemma 9.18, $b \notin fa(g)$. Using $(\lambda{=_\alpha}\lambda\mathbf{ab})$, $\lambda a.g =_\alpha \lambda b.h$ as required.

□

Definition 9.20. Define the substitution $\pi \cdot \sigma$ by: $(\pi \cdot \sigma)(X) \equiv \pi \cdot \sigma(X)$.

Note that $\pi{\cdot}\sigma$ is a substitution. $g(\pi{\cdot}\sigma)$ is not a shorthand for $\pi{\cdot}(g\sigma)$. $g(\pi{\cdot}\sigma) =_\alpha \pi{\cdot}(g\sigma)$ does not hold in general; for example: $a \equiv a((b\ a) \cdot id) \not\equiv (b\ a) \cdot (a\ id)$. However:

Lemma 9.21. *If $nontriv(\pi) \cap fa(g) = \varnothing$ then $g(\pi \cdot \sigma) =_\alpha \pi \cdot (g\sigma)$.*

Proof. By a routine induction on g. □

Lemma 9.22. *σ solves $[\![Pr]\!]^D$ if and only if $\sigma\circ\rho$ does.*

Suppose $nontriv(\pi)\cap(fa(r)\cup fa(s))=\varnothing$ for every $r \stackrel{?}{=} s \in Pr$. Then σ solves $[\![Pr]\!]^D$ if and only if $\pi\cdot\sigma$ does.

Proof. For the first part, we have two cases:

- The case σ solves $[\![Pr]\!]^D$ implies $\sigma\circ\rho$ solves $[\![Pr]\!]^D$. Suppose $g \stackrel{?}{=} h \in [\![Pr]\!]^D$ and σ solves $[\![Pr]\!]^D$. Then $g\sigma =_\alpha h\sigma$. By Lemma 9.19, $g\sigma\rho =_\alpha h\sigma\rho$. By Lemma 7.16, $g(\sigma\circ\rho) =_\alpha h(\sigma\circ\rho)$, as required.
- The case $\sigma\circ\rho$ solves $[\![Pr]\!]^D$ implies σ solves $[\![Pr]\!]^D$. Suppose $g \stackrel{?}{=} h \in [\![Pr]\!]^D$ and $\sigma\circ\rho$ solves $[\![Pr]\!]^D$. Then $g(\sigma\circ\rho) =_\alpha h(\sigma\circ\rho)$. By Lemma 7.16, $g\sigma\rho =_\alpha h\sigma\rho$. By Lemma 9.19, $g\sigma =_\alpha h\sigma$, as required.

For the second part, suppose $nontriv(\pi)\cap(fa(r)\cup fa(s))=\varnothing$ for every $r \stackrel{?}{=} s \in Pr$ and $D=[d_1,\ldots,d_n]$. Then $[\![Pr]\!]^D = \{\lambda d_1.\ldots.\lambda d_n.[\![r]\!]^D \stackrel{?}{=} \lambda d_1.\ldots.\lambda d_n.[\![s]\!]^D \mid r \stackrel{?}{=} s \in Pr\}$. By Lemma 8.4, $nontriv(\pi)\cap(fa([\![r]\!]^D)\cup fa([\![s]\!]^D))=\varnothing$. The result follows from Lemma 9.21, and using Proposition 7.12 and Lemma 7.8. □

Remark 9.23. Lemma 9.22 expresses an intuition that 'names of atoms and unknowns on the right in a solution, do not matter', which also underlies the π and ρ in Theorem 9.30. ρ is the price we pay for using the same unknowns in Definitions 7.1 and 2.6: This design decision makes Definition 8.3 compact, but it causes technical problems in Lemma 9.29, because $\sigma(X)$ can introduce new unknowns over whose permission sets (back in the nominal world) we have no control. ρ lets us rename those new unknowns as convenient. As for π, we discuss it below.

Another design decision is to work with an untyped λ-terms. This simplifies our presentation and makes our results slightly more powerful (because they apply to more substitutions), but we cannot be *too* liberal: Suppose σ solves $[\![Pr]\!]^D$. Examining Definition 8.3, if X occurs in $[\![Pr]\!]^D$ then it is applied to a number of atoms equal to the length of $D\cap S$. So, we will only be interested in σ that respect this fragment of typability ($\mathcal{V}$ will be $fV(Pr)$):

Definition 9.24. Let $\mathcal{V}$ be a finite set of unknowns. Call σ **D-consistent on $\mathcal{V}$** when for every $X\in\mathcal{V}$, $\sigma(X) =_\alpha \lambda a_1.\ldots.\lambda a_n.q$ where n is the length of $D\cap S$. (So $\sigma(X)$ starts with 'at least' length-$D\cap S$-many λ-abstractions.)

Call σ **strictly** D-consistent when also, for every $X\in\mathcal{V}$, $fa(\sigma(X))\cap D=[]$.

Remark 9.25. Strictness is motivated by the following examples: Take $D=[a]$.

Take $Pr=\{X^S \stackrel{?}{=} \mathsf{f}([a]Y^S, Y^S)\}$ with $S=\mathbb{A}^<$. Then the problem

$$[\![Pr]\!]^D = \{\lambda a.(X^S\,a) \stackrel{?}{=} \lambda a.(\mathsf{f}\,(\lambda a.(Y^S a))\,(Y^S a))\}$$

has the solution $\quad \sigma=[X^S{:=}\lambda c.(\mathsf{f}\,(\lambda c.a)\,a),\ Y^S{:=}\lambda c.a]$.

Now $(\sigma\circ\rho)(Y^S) =_\alpha [\![\theta]\!]^E_D(Y^S)$ is impossible for any ρ, since $\lambda c.a =_\alpha \lambda a.[\![\theta(Y^S)]\!]^E$ is impossible.

Take $Pr=\{X^S \stackrel{?}{=} \mathsf{f}([a]Y^T, Y^T)\}$ with $S=\mathbb{A}^<$ and $T=\mathbb{A}^<\setminus\{a\}$. Then the problem

$$[\![Pr]\!]^D = \{\lambda a.(X^S\,a) \stackrel{?}{=} \lambda a.(\mathsf{f}\,(\lambda a.Y^T)\,Y^T)\}$$

has the solution $\quad \sigma=[X^S{:=}\lambda c.(\mathsf{f}\,(\lambda c.a)\,a), Y^T{:=}a]$.

Now $(\sigma\circ\rho)(Y^T) =_\alpha [\![\theta]\!]^E_D(Y^T)$ is impossible, since $a \in fa(a)$ whereas $a \notin fa([\![\theta(Y^T)]\!]^E)$ by Lemma 8.5.

The a in $\sigma(Y^T)$ for the two σ considered above, has nothing to do with the a in D. We can regard this as an unfortunate 'name-clash' which Lemma 9.22 allows us to eliminate with a permutation π.

More on this in Theorem 9.30. We continue with the proofs:

Definition 9.26. Define the **arguments of unknowns** in a pattern q by:

$$args(a) = \varnothing \qquad args(X) = \varnothing \qquad args(Xa_1 \dots a_n) = \{a_1, \dots, a_n\}$$
$$args(\mathsf{f}q_1 \dots q_n) = \bigcup_{1\le i\le n} args(q_i) \qquad args(\lambda a.q) = args(q)$$

$q =_\alpha r$ does not imply $args(q) = args(r)$. This is by design.

Definition 9.27. Suppose q is a ϕ-pattern and $args(q) \subseteq E$.

Define a nominal term q^{-E} by:

$$a^{-E} \equiv a \qquad (Xb_1 \dots b_{\phi(X)})^{-E} \equiv \pi\cdot X^S$$
$$(\lambda a.q)^{-E} \equiv [a]q^{-E} \qquad (\mathsf{f}q_1 \dots q_n)^{-E} \equiv \mathsf{f}(q_1{}^{-E}, \dots, q_n{}^{-E})$$

Here, for each E and X, π is a fixed but arbitrary choice of permutation of the atoms in E, mapping the i^{th} element of $E \cap S$ (Definition 8.2) to b_i for $1 \le i \le \phi(X)$.

Lemma 9.28. *$args(q) \subseteq E$ implies $[\![q^{-E}]\!]^E \equiv q$.*

Proof. By induction on q. □

Lemma 9.29. *Suppose $\mathcal{V}$ is a finite set of unknowns and σ is a ϕ-pattern substitution, strictly D-consistent on $\mathcal{V}$.*

Then there exist ρ, θ, and E, such that $D \subseteq E$, $\bigcup_{X\in\mathcal{V}} capt(\theta(X)) \subseteq E$, and $(\sigma\circ\rho)(X) =_\alpha [\![\theta]\!]^E_D(X)$ for every $X \in \mathcal{V}$.

Proof. Take any $E = [e_1, ..., e_p]$ which includes all atoms in D and in $\{\sigma(X) \mid X \in \mathcal{V}\}$. Define $\mathcal{V}' = \bigcup_{X\in\mathcal{V}} fV(\sigma(X))$ ('the unknowns in $\sigma(X)$ for $X \in \mathcal{V}$'). For each $Y \in \mathcal{V}'$ choose a fresh Y' such that the length of $E \cap fa(Y')$ is equal to $\phi(Y)$. We do this injectively, so that for distinct $Y, Z \in \mathcal{V}'$, Y' and Z' are also distinct. Let ρ be any renaming such that $\rho(Y) \equiv Y'$ for all $Y \in \mathcal{V}'$.

By assumption $\sigma(X) =_\alpha \lambda a_1.\dots.\lambda a_n.q$ for a ϕ-pattern q, where $[a_1, \dots, a_n] = D \cap S$. Take $\theta(X) \equiv (q\rho)^{-E}$.

We can verify that $\bigcup_{X\in\mathcal{V}} capt(\theta(X)) \subseteq E$. We then reason as follows:

$$\begin{array}{rcll}
[\![\theta]\!]^E_D(X) & \equiv & \lambda a_1.\dots.\lambda a_n.[\![(q\rho)^{-E}]\!]^E & \text{Definition 9.1}\\
& \equiv & \lambda a_1.\dots.\lambda a_n.(q\rho) & \text{Lemma 9.28}\\
& \equiv & (\lambda a_1.\dots.\lambda a_n.q)\rho & \text{Fact of } \lambda\text{-terms}\\
& =_\alpha & (\sigma\circ\rho)(X) & \text{By construction}
\end{array}$$

□

Theorem 9.30 (Strong completeness). *Suppose* $capt(Pr) \subseteq D$.

For σ *strictly* D*-consistent on* $fV(Pr)$ *solving* $[\![Pr]\!]^D$ *there are* ρ, θ, *and* E, *such that*

$$(\sigma \circ \rho)(X) =_\alpha [\![\theta]\!]^E_D(X) \text{ for all } X \in fV(Pr) \text{ and } \theta \text{ solves } Pr.$$

For σ D*-consistent on* $fV(Pr)$ *solving* $[\![Pr]\!]^D$ *there are* π, ρ, θ, *and* E, *such that*

$$\pi \cdot (\sigma \circ \rho)(X) =_\alpha [\![\theta]\!]^E_D(X) \text{ for all } X \in fV(Pr) \text{ and } \theta \text{ solves } Pr.$$

Proof. By Lemma 9.29, there are ρ, θ, and E, such that $(\sigma \circ \rho)(X) =_\alpha [\![\theta]\!]^E_D(X)$ for all $X \in fV(Pr)$, $D \subseteq E$ and $\bigcup_{X \in fV(Pr)} capt(\theta(X)) \subseteq E$. $capt(Pr) \subseteq D$ and $D \subseteq E$, so $capt(Pr) \subseteq E$. By Theorem 9.16, θ solves Pr.

For the second part, write $D = [d_1, \ldots, d_n]$, choose $D' = [d'_1, \ldots, d'_n]$ fresh (so d'_i is not in D, Pr, or $\sigma(X)$ for any $X \in fV(Pr)$), and take $\pi = (d'_1\ d_1) \ldots (d'_n\ d_n)$. $\pi \cdot \sigma$ is strictly D-consistent and the result follows from the first part and Lemma 9.22. □

10. Conclusions

In this paper, we have presented a syntax which slightly generalises nominal terms and obtains significantly enhanced properties. We gain 'always fresh' and 'always rename' properties (Corollaries 2.14 and 2.15) which are present in first- and higher-order syntax, absent in nominal terms, and regained in permissive nominal terms.

We do not claim a telling difference in expressivity in practice between nominal and permissive nominal terms. It may indeed be that permissive nominal terms can express some things that nominal terms cannot[8] but expressivity is not our main motivation in this paper. The issues which motivate us are with the *properties* of these syntaxes. As we have seen in this paper, a significant new body of mathematics follows from these changes, which at first seem so innocuous. It does not stop there; the interested reader can find more in [15].

Permissive nominal terms do not obsolete nominal terms. To discuss 'an arbitrary term', a nominal terms unknown $\dot{X}$ may be more directly useful than a permissive nominal terms unknown $X^{\mathbb{A}^<}$ (which means 'an arbitrary term, mentioning atoms in $\mathbb{A}^<$').

We have leveraged the difference between nominal terms and permissive nominal terms to obtain a new unification algorithm which is more efficient in the sense that it is based just on substitutions, and in that sense is also more like the notion of solution familiar from first- and higher-order syntax. Freshness problems are solved 'all in one go' by a distinct algorithm. We have interpreted nominal unification as a subsystem of permissive nominal unification (Section 4).

One nice way to view this interpretation is that $\mathbb{A}^<$ plays the role of 'the atoms we had so far' and $\mathbb{A}^>$ that of 'the atoms we will generate fresh in the future'. Finally, we have exhibited permissive nominal unification as equivalent to higher-order pattern unification.

[8] Permission sets may be larger than $\mathbb{A}^<$ as well as smaller, whereas intuitively in nominal terms freshness contexts can only make permission sets smaller.

10.1. Related work

Infinite sets of atoms

Permissive nominal terms are based on the idea of infinite and co-infinite sets of atoms S. This is new, but it emerges from a literature rich in precedents. As we noted in Remark 2.22, infinite and co-infinite sets break with the standard nominal sets semantics from [25], which does not admit them because they do not have finite support. This is however not a serious mathematical problem: the idea of relaxing the 'finite support' property of nominal sets to infinite generalisations is natural.

As far as we know this was first discussed in [11], where the second author proposed to identify 'small' sets of atoms not with cardinality but with *well-orderability* of the atoms in the set — so a set of atoms is small when it can be assigned a cardinal size, and large when it cannot be assigned a cardinal size (internally, within the model) — but we do not commit ourselves to how large those cardinals can get, and in particular they could be infinite. See [13] for a more extended treatment of the same ideas.

Pitts referred to the possibility of using nominal sets with infinite support in [37], in order to obtain a complete semantics for nominal logic. This idea was taken up by Cheney in [6]. Thus, in [6] 'small' sets of atoms are identified as elements of a *support ideal* (Definition 4.1 of [6]), which are similar in spirit to the set of permission sets from Definition 2.2. In Definition 2.2 we give a concrete set of permission sets, but in a footnote to that definition we also identify some reasonable abstract conditions for the set of permission sets to satisfy such that the proofs in this paper still work. These conditions are extremely mild; the structure actually required of permission sets is much weaker than that provided by Definition 2.2.

Namespaces

Since this paper was written, the second author has prepared a manuscript treating *permissive nominal algebra* [15] (a precursor is [22]). There, the set of all atoms is taken to be uncountable and permission sets are taken to be all countably infinite sets of atoms. This setting is sufficiently general to accommodate many different notions of permission sets as subsystems. In particular the permission sets of this paper feature as a 'namespace' identified by $\mathbb{A}^<$; thus, the ideas in this paper slot quite nicely into a more abstract setting.

'Free atoms of' as distinct from 'support of the denotation of'

It may be useful to note some work to which this paper is not related. One nice aspect of permissive nominal terms is that they give us a notion of 'free atoms of a term' $fa(r)$. The judgement $\nabla \vdash a\#r$ of nominal terms corresponds to the judgement $a \notin fa(r)$ of permissive nominal terms (see Lemma 4.8) and both correspond to the informal judgement 'a is not free in r'.

Nominal sets have a native notion of semantic freshness; $a\#_{sem}r$ means 'a is not in the support of the denotation of r'. Semantic freshness is a distinct concept. The reader should not confuse semantic freshness with the intensional judgement 'free atoms of' used in this paper.

Semantic freshness may be expressed using 'free atoms of' and equality [17, 21, 15].

Patterns

Patterns emerged by studying Skolemisation of unification problems [32]; they proved useful in the unification of higher-order abstract syntax terms [31].

Cheney proposed a two-stage translation of higher-order pattern unification to nominal unification [4], first by exhibiting a translation of higher-order pattern unification to nominal pattern unification (where nominal patterns are a variant of nominal terms, with a *concretion* operator, where unknowns have empty support), followed by a translation between nominal pattern unification and nominal unification.

Levy and Villaret related nominal unifiability with higher-order pattern unifiability in [29]. Our treatment translates *solutions* as well, handles a more general class of higher-order patterns than considered in [29], and we prove our translation complete (Theorem 9.30) and optimal (Subsection 8.2).

Note that translating solutions really matters: it might have been, for example, that higher-order pattern unification and permissive nominal unification have the same notion of unifiability — but very different notions of solution and sets of solutions. For comparison, the λ-calculus and combinators express similar notions of computability, but have very different notions of reduction and computation.

The version of higher-order pattern unification which we examine is more general than usual, since we do not type our λ-terms. We show how to retain enough of the properties of typing to avoid 'silly' problems. For example, we do not consider untyped higher-order pattern unification problems like $X =_{\alpha\beta} \lambda a.(Xa)$, because this cannot be expressed as a unification of two ϕ-patterns for any ϕ (Definition 7.19) — we impose a structural condition on our patterns that X should be applied to a consistent, fixed number of arguments.

We hypothesise that the results in this paper would work in a typed setting; the conditions which our proofs depend on to work are just structural ones, which would also be guaranteed by types. We have not investigated the effects of η-conversion; this is future work.

The broader literature

Hamana's β_0 unification of λ-terms with holes adds a capturing substitution [26]. Level 2 variables (which are instantiated) are annotated with level 1 variable symbols that *may* appear in them; permissive nominal terms move in this direction in the sense that permission sets also describe which level 1 variable symbols (we call them atoms in this paper) may appear in them, though with our permission sets there are infinitely many that may, and infinitely many that may not. Another significant difference is that the treatment of α-equivalence in Hamana's system is not nominal (not based on permutations) and unlike our systems, Hamana's does not have most general unifiers.

Similarly, Qu-Prolog [34] adds level 2 variables, but does not manage α-conversion in nominal style, and, for better or for worse, the system is more ambitious in what it expresses, and thus loses mathematical properties (unification is semi-decidable, most general unifiers need not exist).

10.2. Future work

We propose permissive nominal terms as a syntax for designing logics and λ-calculi in the spirit of nominal terms.

A first implementation of permissive nominal unification has been made [33] by the third author.

We have begun to apply permissive nominal terms to construct novel logics and λ-calculi, taking advantage of their properties to simplify the theory. It is simply very useful to reason on terms (without a freshness context), to have an inexhaustible supply of fresh names, and to be able to quotient by α-equivalence. We note in particular the papers [24, 23, 19, 18, 20], in which we have struggled with the theory of α-equivalence given to us by nominal terms; these might benefit from the use of permissive nominal terms.

Permissive nominal terms syntax has two levels of variable, atoms a and unknowns X. There is no reason to stop there; we have already considered syntaxes with more than two levels of variable, for example [14, 16]. Again, we had difficulty with α-conversion and fresh atoms. It would be very interesting to revisit this material armed with the permissive ideas of this paper.

Finally, it may be possible to extend the techniques of this paper to biject full higher-order unification with an enrichment of (permissive) nominal unification.

References

[1] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, Great Britain, 1998.

[2] Christophe Calvès. A Haskell Nominal Toolkit. In *2nd International Workshop on Theory and Applications of Abstraction, Substitution and Naming (TAASN 2009)*, 2009. source code available online at: `http://www.dcs.kcl.ac.uk/pg/calves/hnt/`.

[3] James Cheney. *Nominal Logic Programming*. PhD thesis, Cornell University, August 2004.

[4] James Cheney. Relating nominal and higher-order pattern unification. In *Proceedings of the 19th International Workshop on Unification (UNIF 2005)*, pages 104–119, 2005.

[5] James Cheney. Scrap your nameplate: (functional pearl). *SIGPLAN Notices*, 40(9):180–191, 2005.

[6] James Cheney. Completeness and Herbrand theorems for nominal logic. *Journal of Symbolic Logic*, 71:299–320, 2006.

[7] James Cheney and Christian Urban. Alpha-prolog: A logic programming language with names, binding and alpha-equivalence. In Bart Demoen and Vladimir Lifschitz, editors, *Proceedings of the 20th International Conference on Logic Programming (ICLP 2004)*, number 3132 in Lecture Notes in Computer Science, pages 269–283. Springer, 2004.

[8] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. Technical Report HW-MACS-TR-0062, Heriot-Watt University, 2009. Available online at gabbay.org.uk/papers/perntu-tr.pdf.

[9] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Binding logic: Proofs and models. In *LPAR '02: Proceedings of the 9th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, pages 130–144, London, UK, 2002. Springer.

[10] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting (journal version). *Information and Computation*, 205(6):917–965, 2007.

[11] Murdoch J. Gabbay. FM-HOL, a higher-order theory of names. In F. Kamareddine, editor, *35 Years of Automath*, 2002.

[12] Murdoch J. Gabbay. A NEW calculus of contexts. In *PPDP '05: Proc. of the 7th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming*, pages 94–105. ACM, 2005.

[13] Murdoch J. Gabbay. A General Mathematics of Names. *Information and Computation*, 205(7):982–1011, July 2007.
[14] Murdoch J. Gabbay. Hierarchical Nominal Terms and Their Theory of Rewriting. *Electronic Notes in Theoretical Computer Science*, 174(5):37–52, 2007.
[15] Murdoch J. Gabbay. Permissive nominal algebra and semantic nominal unknowns: a new universal algebra for nominal techniques. Manuscript, 2010.
[16] Murdoch J. Gabbay and Stéphane Lengrand. The lambda-context calculus (extended version). *Information and computation*, 207(12):1369–1400, 2009.
[17] Murdoch J. Gabbay and Aad Mathijssen. A Formal Calculus for Informal Equality with Binding. In *WoLLIC'07: 14th Workshop on Logic, Language, Information and Computation*, volume 4576 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 2007.
[18] Murdoch J. Gabbay and Aad Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing*, 20(4-5):451–479, June 2008.
[19] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic. *Journal of Logic and Computation*, 18(4):521–562, August 2008.
[20] Murdoch J. Gabbay and Aad Mathijssen. A nominal axiomatisation of the lambda-calculus. *Journal of Logic and Computation*, September 2009. Online access.
[21] Murdoch J. Gabbay and Aad Mathijssen. Nominal (universal) algebra: equational logic with names and binding. *Journal of Logic and Computation*, 19(6):1455–1508, 2009.
[22] Murdoch J. Gabbay and Dominic P. Mulligan. Semantic nominal terms. In *2nd International Workshop on Theory and Applications of Abstraction, Substitution and Naming (TAASN 2009)*, 2009.
[23] Murdoch J. Gabbay and Dominic P. Mulligan. Two level lambda-calculus. *Electronic Notes in Theoretical Computer Science*, 246:107–129, 2009.
[24] Murdoch J. Gabbay and Dominic P. Mulligan. Curry-Howard for incomplete first-order logic derivations using one-and-a-half level terms. *Information and Computation*, 208:230–258, 2010.
[25] Murdoch J. Gabbay and Andrew M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing*, 13(3–5):341–363, 2001.
[26] Makoto Hamana. A logic programming language based on binding algebras. In *TACS'01*, volume 2215 of *Lecture Notes in Computer Science*, pages 243–262. Springer, 2001.
[27] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. In *Proc. 2nd Annual IEEE Symposium on Logic in Computer Science, LICS'87*, pages 194–204. IEEE Computer Society Press, 1987.
[28] Jan-Willem Klop, Vincent van Oostrom, and Femke van Raamsdonk. Combinatory reduction systems. *Theoretical Computer Science*, 121:279–308, 1993.
[29] Jordi Levy and Mateu Villaret. Nominal unification from a higher-order perspective. In *Proceedings of RTA'08*, volume 5117 of *Lecture Notes in Computer Science*. Springer, 2008.
[30] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.
[31] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497 – 536, 1991.
[32] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14(4):321–358, 1992.
[33] Dominic P. Mulligan. Implementation of permissive nominal terms. source code available online at: `http://www.macs.hw.ac.uk/~dpm8/permissive/`, 2009.
[34] Peter Nickolas and Peter J. Robinson. The Qu-Prolog unification algorithm: formalisation and correctness. *Theoretical Computer Science*, 169(1):81–112, 1996.
[35] Lawrence C. Paulson. Isabelle: the next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[36] Frank Pfenning and Conal Elliot. Higher-order abstract syntax. In *PLDI (Programming Language design and Implementation)*, pages 199–208. ACM Press, 1988.
[37] Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003.
[38] Andrew M. Pitts and Murdoch J. Gabbay. A Metalanguage for Programming with Bound Names Modulo Renaming. In *MPC2000*, volume 1837 of *Lecture Notes in Computer Science*, pages 230–255. Springer, 2000.
[39] Mark R. Shinwell, Andrew M. Pitts, and Murdoch J. Gabbay. FreshML: Programming with Binders Made Simple. In *ICFP'03*, volume 38, pages 263–274. ACM Press, 2003.
[40] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, 2004.
Acknowledgements. Supported by grant RYC-2006-002131 at the Polytechnic University of Madrid.

Appendix A. Supplementary proofs

Definition Appendix A.1 is useful for Proposition 2.21.

Definition Appendix A.1. Define the **size** of a term r by:

$$\begin{array}{cc} size(a) = 0 & size(\mathsf{f}(r_1,\ldots,r_n)) = \sum_{1\leq i\leq n} size(r_i) \\ size([a]r) = 1{+}size(r) & size(\pi{\cdot}X^S) = 0 \end{array}$$

Proof of Proposition 2.21:

Proof. Reflexivity is shown by induction on terms. Symmetry is shown by induction on derivations. Transitivity is shown by induction on the size of a term. We consider one case from the proof of symmetry and one from the proof of transitivity:

- $[a]r =_\alpha [b]s$ *implies* $[b]s =_\alpha [a]r$. Suppose $(b\ a){\cdot}r =_\alpha s$ and $b \notin fa(r)$. By Lemma 2.17 $a \notin fa((a\ b)\cdot r)$. From Lemmas 2.16 and 2.18, $r =_\alpha (a\ b)\cdot s$. By Lemma 2.19, $a \notin fa(s)$. It is a fact that the size of terms is unaffected by permutation, so by inductive hypothesis $(a\ b)\cdot s =_\alpha r$. Extending with $(=_\alpha[\mathbf{b}])$ we obtain $[b]s =_\alpha [a]r$ as required.
- $[a]r =_\alpha [b]s$ *and* $[b]s =_\alpha [c]t$ *imply* $[a]r =_\alpha [c]t$. Suppose $(b\ a)\cdot r =_\alpha s$, $(c\ b)\cdot s =_\alpha t$, $b \notin fa(r)$ and $c \notin fa(s)$. By Lemma 2.18 $(c\ b)\cdot((b\ a)\cdot r) =_\alpha (c\ b)\cdot s$. It is a fact that the size of terms is unaffected by permutation, so by inductive hypothesis also $(c\ b)\cdot((b\ a)\cdot r) =_\alpha t$, and by Lemmas 2.20 and 2.16 $(c\ a)\cdot r =_\alpha t$. By Lemma 2.19 $c \notin fa((b\ a)\cdot r)$, so by Lemma 2.17 $c \notin (b\ a)\cdot fa(r)$. Therefore, $c \notin fa(r)$. We use $(=_\alpha[\mathbf{b}])$ to derive $[a]r =_\alpha [c]t$ as required.

□

Proof of Lemma 3.3:

Proof. By induction on r.

- The cases a and $\mathsf{f}(r_1,\ldots,r_n)$ are routine.

- The case $[a]r$. We reason as follows:

$$\begin{array}{lll} fa(([a]r)\theta) \equiv fa([a]r\theta) & & \text{Definition 3.2} \\ = fa(r\theta) \setminus \{a\} & & \text{Definition 2.11} \\ \subseteq fa(r) \setminus \{a\} & & \text{Inductive hypothesis} \\ = fa([a]r) & & \text{Definition 2.11} \end{array}$$

  The result follows.
- The case $\pi \cdot X^S$. By Definition 2.11, $fa(\pi \cdot X^S) = \pi \cdot S$. By Definition 3.1, $fa(\theta(X^S)) \subseteq S$. Using Lemma 2.17, it follows $fa(\pi \cdot \theta(X^S)) \subseteq \pi \cdot S$.

□

Proof of Lemma 4.8:

Proof. We handle the two implications separately.

- The case $\iota(\dot{a}) \notin fa([\![\dot{r}]\!]_\Delta)$ implies $\Delta \vdash \dot{a}\#\dot{r}$. We proceed by induction on $\dot{r}$.
  - The cases $\dot{b}$ and $\mathsf{f}(\dot{r}_1, \ldots, \dot{r}_n)$ are straightforward.
  - The case $[\dot{a}]\dot{r}$. Since $\Delta \vdash \dot{a}\#[\dot{a}]\dot{r}$ always, using $(\#[\dot{\mathbf{a}}])$.
  - The case $[\dot{b}]\dot{r}$. Suppose $\iota(\dot{a}) \notin fa([\![[\iota(\dot{b})]\dot{r}]\!]_\Delta)$ and $\iota(\dot{a}) \notin fa([\![\dot{r}]\!]_\Delta) \setminus \{\iota(\dot{b})\}$. Then $\iota(\dot{a}) \notin fa([\![\dot{r}]\!]_\Delta)$, therefore $\Delta \vdash \dot{a}\#\dot{r}$ by inductive hypothesis. Using $(\#[\dot{\mathbf{b}}])$, we have $\Delta \vdash \dot{a}\#[\dot{b}]\dot{r}$, and the result follows.
  - The case $\dot{\pi} \cdot \dot{X}$. Suppose $\iota(\dot{a}) \notin fa([\![\dot{\pi} \cdot \dot{X}]\!]_\Delta)$. Then $\iota(\dot{a}) \notin [\![\dot{\pi}]\!] \cdot S$, where $S = \mathbb{A}^< \setminus \{\iota(\dot{a}) \mid \dot{a}\#\dot{X} \in \Delta\}$. But $[\![\dot{\pi}]\!] \cdot \mathbb{A}^< \setminus \{\iota(\dot{a}) \mid \dot{a}\#\dot{X} \in \Delta\}$ is the same as $[\![\dot{\pi}]\!] \cdot \mathbb{A}^< \setminus [\![\dot{\pi}]\!] \cdot \{\iota(\dot{a}) \mid \dot{a}\#\dot{X} \in \Delta\}$. Then $[\![\dot{\pi}]\!] \cdot \{\iota(\dot{a}) \mid \dot{a}\#\dot{X} \in \Delta\} = \{[\![\dot{\pi}]\!] \cdot \iota(\dot{a}) \mid \dot{a}\#\dot{X} \in \Delta\}$. Using Definition 4.6, and the fact permutations are bijective, we have $\{[\![\dot{\pi}]\!] \cdot \iota(\dot{a}) \mid a\#X \in \Delta\} = \{\iota(\dot{\pi}^{-1} \cdot \dot{a}) \mid \dot{\pi}^{-1} \cdot \dot{a}\#X \in \Delta\}$. We use $(\#\dot{\mathbf{X}})$ to obtain $\Delta \vdash \dot{a}\#\dot{X}$, and we have the result.
- The case $\Delta \vdash \dot{a}\#\dot{r}$ implies $\iota(\dot{a}) \notin fa([\![\dot{r}]\!]_\Delta)$. We proceed by induction on the derivation of $\Delta \vdash \dot{a}\#\dot{r}$.
  - The cases $(\#\dot{\mathbf{b}})$ and $(\#\mathsf{f})$ are routine.
  - The case $(\#[\dot{\mathbf{a}}])$. Suppose $\Delta \vdash \dot{a}\#[\dot{a}]\dot{r}$ using $(\#[\dot{\mathbf{a}}])$. Then $[\![[\dot{a}]\dot{r}]\!]_\Delta \equiv [\iota(\dot{a})][\![\dot{r}]\!]_\Delta$. Further, $\iota(\dot{a}) \notin fa([\![\dot{r}]\!]_\Delta) \setminus \{\iota(\dot{a})\}$, and the result follows.
  - The case $(\#[\dot{\mathbf{b}}])$. Suppose $\Delta \vdash \dot{a}\#\dot{r}$ and $\iota(\dot{a}) \notin fa(\dot{r})$ by assumption. Then $\Delta \vdash \dot{a}\#[\dot{b}]\dot{r}$ by $(\#[\dot{\mathbf{b}}])$. Further, $fa([\![[\dot{b}]\dot{r}]\!]_\Delta) = fa([\![\dot{r}]\!]_\Delta) \setminus \{\iota(\dot{b})\}$, and the result follows.
  - The case $(\#\dot{\mathbf{X}})$. Suppose $\dot{\pi}^{-1}(\dot{a})\#\dot{X} \in \Delta$, and $\Delta \vdash \dot{a}\#\dot{\pi} \cdot \dot{X}$ by $(\#\dot{\mathbf{X}})$. Then $[\![\dot{\pi} \cdot \dot{X}]\!]_\Delta = [\![\dot{\pi}]\!] \cdot X^S$ where $S = \mathbb{A}^< \setminus \{\iota(\dot{a}) \mid \dot{a}\#\dot{X} \in \Delta\}$. Further, $fa([\![\dot{\pi}]\!] \cdot X^S) = [\![\dot{\pi}]\!] \cdot S$. The result follows from Definition 4.6.

□

Proof of Theorem 4.9:

Proof. We prove that $[\![\dot{r}]\!]_\Delta =_\alpha [\![\dot{s}]\!]_\Delta$ implies $\Delta \vdash \dot{r} = \dot{s}$ by induction on the derivation of $[\![\dot{r}]\!]_\Delta =_\alpha [\![\dot{s}]\!]_\Delta$:

- The cases $\dot{a}$ and $\mathsf{f}(\dot{r}_1, \ldots, \dot{r}_n)$ are routine.
- The case $(=_\alpha[\dot{\mathbf{a}}])$. Suppose $[\![\dot{r}]\!]_\Delta =_\alpha [\![\dot{s}]\!]_\Delta$ and $\Delta \vdash \dot{r} = \dot{s}$. Then using $(=_\alpha[\dot{\mathbf{a}}])$, $[\iota(\dot{a})][\![\dot{r}]\!]_\Delta =_\alpha [\iota(\dot{a})][\![\dot{s}]\!]_\Delta$ and $\Delta \vdash [\dot{a}]\dot{r} = [\dot{a}]\dot{s}$ also, using $(=[\dot{\mathbf{a}}])$. The result follows, as $[\iota(\dot{a})][\![\dot{r}]\!]_\Delta = [\![[\dot{a}]\dot{r}]\!]_\Delta$.

- The case $(=_\alpha[\mathbf{b}])$. Suppose $(\iota(\dot{b})\ \iota(\dot{a}))\cdot[\![\dot{r}]\!]_\Delta =_\alpha [\![\dot{s}]\!]_\Delta$ and $\iota(\dot{b}) \notin fa([\![\dot{r}]\!]_\Delta)$. By Lemmas 4.7 and 4.8 $[\![(\dot{b}\ \dot{a})\cdot\dot{r}]\!]_\Delta =_\alpha [\![\dot{s}]\!]_\Delta$ and $\Delta \vdash \dot{b}\#\dot{r}$. By inductive hypothesis $\Delta \vdash (\dot{b}\ \dot{a})\cdot\dot{r} = \dot{s}$. We use $(=[\dot{\mathbf{b}}])$.
- The case $(=_\alpha\dot{\mathbf{X}})$. Suppose $[\![\dot{\pi}]\!]|_S = [\![\dot{\pi}']\!]|_S$ where $S = \mathbb{A}^< \backslash \{\iota(\dot{a}) \mid \dot{a}\#\dot{X}\in\Delta\}$. ι is injective, so $a\#\dot{X} \in \Delta$ for all $\dot{a}$ such that $\dot{\pi}(\dot{a}) \neq \dot{\pi}'(\dot{a})$. The result follows by $(=\dot{\mathbf{X}})$.

We prove that $\Delta \vdash \dot{r} = \dot{s}$ implies $[\![\dot{r}]\!]_\Delta =_\alpha [\![\dot{s}]\!]_\Delta$ by induction on the derivation of $\Delta \vdash \dot{r} = \dot{s}$:

- The cases $(=\dot{\mathbf{a}})$, $(=\mathsf{f})$ and $(=[\dot{\mathbf{a}}])$ are straightforward.
- The case $(=[\dot{\mathbf{b}}])$. Suppose $\Delta \vdash (\dot{b}\ \dot{a})\cdot\dot{r} = \dot{s}$ and $\Delta \vdash \dot{b}\#\dot{r}$. By inductive hypothesis and Lemma 4.7, $(\dot{b}\ \dot{a})\cdot[\![\dot{r}]\!]_\Delta =_\alpha [\![\dot{s}]\!]_\Delta$. By Lemma 4.8, $\iota(\dot{b}) \notin fa([\![\dot{r}]\!]_\Delta)$. The result follows by $(=_\alpha[\mathbf{b}])$.
- The case $(=\dot{\mathbf{X}})$. Recall that $[\![\dot{\pi}\cdot\dot{X}]\!]_\Delta = [\![\dot{\pi}]\!]\cdot X^S$ and $[\![\dot{\pi}'\cdot\dot{X}]\!]_\Delta = [\![\dot{\pi}']\!]\cdot X^S$ where $S = \mathbb{A}^< \setminus \{\iota(\dot{a}) \mid \dot{a}\#\dot{X} \in \Delta\}$. Suppose $\dot{\pi}(\dot{a}) \neq \dot{\pi}'(\dot{a})$ implies $\Delta \vdash \dot{a}\#\dot{X}$. Using Lemma 4.8, $[\![\dot{\pi}]\!](\iota(\dot{a})) \neq [\![\dot{\pi}']\!](\iota(\dot{a}))$ implies $\iota(\dot{a}) \notin S$. The result follows by $(=_\alpha\dot{\mathbf{X}})$.

□

Proof of Lemma 4.13:

Proof. By induction on $\dot{r}$. We show two cases:

- The case $[\dot{a}]\dot{r}$. We reason as follows:

$$\begin{array}{rll}
[\![([\dot{a}]\dot{r})\dot{\theta}]\!]_\Delta \equiv [\![[\dot{a}]\dot{r}\dot{\theta}]\!]_\Delta & & \text{Definition 4.10}\\
\equiv [\iota(\dot{a})][\![\dot{r}\dot{\theta}]\!]_\Delta & & \text{Definition 4.6}\\
\equiv [\iota(\dot{a})][\![\dot{r}]\!]_\Delta[\![(\Delta,\dot{\theta})]\!] & & \text{Inductive hypothesis}\\
\equiv ([\iota(\dot{a})][\![\dot{r}]\!]_\Delta)[\![(\Delta,\dot{\theta})]\!] & & \text{Fact}\\
\equiv [\![[\dot{a}]\dot{r}]\!]_\Delta[\![(\Delta,\dot{\theta})]\!] & & \text{Definition 4.6}
\end{array}$$

  The result follows.
- The case $\dot{\pi}\cdot\dot{X}$. We reason as follows:

$$\begin{array}{rll}
[\![(\dot{\pi}\cdot\dot{X})\dot{\theta}]\!]_\Delta \equiv [\![\dot{\pi}\cdot\dot{\theta}(\dot{X})]\!]_\Delta & & \text{Definition 4.10}\\
\equiv [\![\dot{\pi}]\!]\cdot[\![\dot{\theta}(\dot{X})]\!]_\Delta & & \text{Definition 4.6}\\
\equiv [\![\dot{\pi}]\!]\cdot[\![\dot{\theta}]\!]([\![\dot{X}]\!]_\Delta) & & \text{Definition 3.1}
\end{array}$$

  The result follows.

□

Proof of Lemma 4.15:

Proof. We prove by induction on r that $a \notin fa(r)$ implies $(b\ a)\cdot r =_\alpha r$:

- The cases c and $\mathsf{f}(r_1,\ldots,r_n)$. Straightforward.
- The cases $[a]r$. We show $(b\ a)\cdot[a]r =_\alpha [a]r$ where $b \notin fa([a]r)$, hence $b \notin fa(r)$ and $a \notin fa(r)$. By Definition 2.9, $(b\ a)\cdot[a]r =_\alpha [b](b\ a)\cdot r$. By Definition 2.13, we must show $(a\ b)\cdot((b\ a)\cdot r) =_\alpha r$ where $a \notin fa((b\ a)\cdot r)$. By Lemma 2.17, $b \notin fa(r)$ implies $a \notin fa((b\ a)\cdot r)$. By Lemma 2.16, $(a\ b)\cdot((b\ a)\cdot r) =_\alpha ((a\ b)\circ(b\ a))\cdot r$. As $\pi = \pi^{-1}$, we have $r =_\alpha r$. The result follows from Proposition 2.21.
- The case $[b]r$ is similar.

- The case $[c]r$. Suppose $b \notin fa([c]r)$, $a \notin fa([c]r)$ and $a, b \notin fa(r)$. We show $(b\ a) \cdot [c]r =_\alpha [c]r$. By Definition 2.9, $(b\ a) \cdot [c]r \equiv [c](b\ a) \cdot r$. We use $(=_\alpha[\mathbf{a}])$ and the inductive hypothesis to obtain $(b\ a) \cdot r =_\alpha r$.
- The case $\pi \cdot X^S$. Suppose $b \notin fa(\pi \cdot X^S)$, $a \notin fa(\pi \cdot X^S)$ and $a, b \notin \pi \cdot S$. By Definition 2.9, $(b\ a) \cdot (\pi \cdot X^S) \equiv ((b\ a) \circ \pi) \cdot X^S$. Using $(=_\alpha\mathbf{X})$, $((b\ a) \circ \pi) \cdot X^S =_\alpha \pi \cdot X^S$ whenever $((b\ a) \circ \pi)|_S = \pi|_S$. As $a, b \notin \pi \cdot S$, $((b\ a) \circ \pi)|_S = \pi|_S$. The result follows.

We prove by induction on r that $(b\ a) \cdot r =_\alpha r$ implies $a \notin fa(r)$:

- The case a, b, c and $\mathsf{f}(r_1, \ldots, r_n)$ are routine.
- The case $[a]r$. Suppose $(b\ a) \cdot [a]r =_\alpha [a]r$. By Definition 2.9, $(b\ a) \cdot [a]r \equiv [b](b\ a) \cdot r$. By Definition 2.13, $[b](b\ a) \cdot r =_\alpha [a]r$ whenever $(a\ b) \cdot ((b\ a) \cdot r) =_\alpha r$ with $a \notin fa((b\ a) \cdot r)$. By Lemma 2.16, and the fact that swappings are self-inverse, $(a\ b) \cdot ((b\ a) \cdot r) \equiv r$. By assumption, $b \notin fa(r)$. By Lemma 2.17, $a \notin fa((b\ a) \cdot r)$. The result follows.
- The case $[b]r$ is similar.
- The case $[c]r$. By inductive hypothesis $(b\ a) \cdot r =_\alpha r$ implies $a \notin fa(r)$. The result follows from $[c](b\ a) \cdot r \equiv (b\ a) \cdot [c]r$.
- The case $\pi \cdot X^S$. Suppose $(b\ a) \cdot \pi \cdot X^S =_\alpha \pi \cdot X^S$. By Definition 2.9, $(b\ a) \cdot \pi \cdot X^S \equiv ((b\ a) \circ \pi) \cdot X^S$. Using $(=_\alpha\mathbf{X})$, $((b\ a) \circ \pi) \cdot X^S =_\alpha \pi \cdot X^S$ whenever $(b\ a) \circ \pi|_S = \pi|_S$. It is a fact that $(b\ a) \circ \pi|_S = \pi|_S$ only when $b, a \notin \pi \cdot S$. The result follows.

□

The following definition is used in the proof of Proposition 5.4:

Definition Appendix A.2. Define the **size** of a support inclusion problem $size(Inc)$ to be a tuple (T, A, P, S), where:
– T is the number of term-formers appearing within terms in Inc,
– A is the number of abstractions appearing within terms in Inc,
– P is the number of permutations, distinct from the identity permutation, appearing within terms in Inc, and
– S is the number of support inclusions within Inc.
We order tuples lexicographically.

Proof of Proposition 5.4:

Proof. By case analysis, checking all simplification rules reduce the measure defined in Definition Appendix A.2.

- The case $a \sqsubseteq T, Inc'$. Suppose $a \in T$, $size(a \sqsubseteq T, Inc') = (T, A, P, S)$, and $a \sqsubseteq T, Inc' \implies Inc'$ by $(\sqsubseteq\mathbf{a})$. Then $size(Inc') = (T, A, P, S - 1)$. Otherwise, suppose $a \notin T$ so $(\sqsubseteq\mathbf{a})$ is not applicable. No other rule is applicable by assumption, and the result follows.
- The case $\mathsf{f}(r_1, \ldots, r_n) \sqsubseteq T, Inc'$. Suppose $size(\mathsf{f}(r_1, \ldots, r_n) \sqsubseteq T, Inc') = (T, A, P, S)$ and $\mathsf{f}(r_1, \ldots, r_n) \sqsubseteq T, Inc' \implies r_1 \sqsubseteq T, \ldots, r_n \sqsubseteq T, Inc'$ by $(\sqsubseteq\mathsf{f})$. Then $size(r_1 \sqsubseteq T, \ldots, r_n \sqsubseteq T, Inc') = (T - 1, A, P, S + n - 1)$. The result follows from the ordering.
- The case $[a]r \sqsubseteq T, Inc'$. Suppose $size([a]r \sqsubseteq T, Inc') = (T, A, P, S)$ and $[a]r \sqsubseteq T, Inc' \implies r \sqsubseteq T \cup \{a\}, Inc'$ by $(\sqsubseteq[])$. Then $size(r \sqsubseteq T \cup \{a\}, Inc') = (T, A - 1, P, S)$ and the result follows.
- The case $\pi \cdot X^S \sqsubseteq T, Inc'$. Suppose $size(\pi \cdot X^S \sqsubseteq T, Inc') = (T, A, P, S)$. Then, if $S \subseteq \pi^{-1} \cdot T$, we have $\pi \cdot X^S \sqsubseteq T, Inc' \implies Inc'$ by $(\sqsubseteq\mathbf{X}')$, with measure $(T, A, P, S-1)$. Otherwise, if we have $S \not\subseteq \pi^{-1} \cdot T$ and $\pi \neq id$, we have $\pi \cdot X^S \sqsubseteq T, Inc' \implies$

$X^S \sqsubseteq \pi^{-1} \cdot T, Inc'$ with measure $(T, A, P-1, S)$. By assumption, no other rules are applicable.

□

The following definition is used in the proof of Proposition 6.7:

Definition Appendix A.3. Define the **size** of a unification problem $size(Pr)$ to be a tuple (E, T, A), where:
– E is the number of equalities appearing in the unification problem,
– T is the number of term-formers appearing within terms in the equalities of the unification problem,
– A is the number of abstractions appearing within terms in the equalities of the unification problem.
We order tuples lexicographically.

Proof of Proposition 6.7:

Proof. By case analysis, checking all simplification rules reduce the measure defined in Definition Appendix A.3. We consider three cases:
– The case ($\stackrel{?}{=}$[**b**]). Suppose $b \notin fa(r)$ and $\mathcal{V}; [a]r \stackrel{?}{=} [b]s, Pr \Longrightarrow \mathcal{V}; (b\ a) \cdot r \stackrel{?}{=} s, Pr$ by ($\stackrel{?}{=}$[**b**]). Suppose further that $size([a]r \stackrel{?}{=} [b]s, Pr) = (E, T, A)$. Then $size((b\ a) \cdot r \stackrel{?}{=} s, Pr) = (E, T, A-1)$, and the result follows.
– The case (**I1**). Suppose $X^S \notin fV(s)$ and $fa(S) \subseteq \pi \cdot S$, and $\mathcal{V}; \pi \cdot X^S \stackrel{?}{=} s, Pr \stackrel{[X^S := \pi^{-1} \cdot s]}{\Longrightarrow} \mathcal{V}; Pr[X^S := \pi^{-1} \cdot s]$ by (**I1**). Suppose further that $size(\pi \cdot X^S \stackrel{?}{=} s, Pr) = (E, T, A)$. Then $size(Pr[X^S := \pi^{-1} \cdot s]) = (E-1, T, A)$, and the result follows.
– The case (**I3**). It is a fact that rewriting with (**I3**) terminates, because of the condition that $Pr_{\sqsubseteq}$ is non-trivial.

□

Proof of Lemma 6.8:

Proof. The empty set cannot be simplified, so suppose $Pr = r \stackrel{?}{=} s, Pr'$ where the simplification rule acts on $r \stackrel{?}{=} s$. We reason by cases:

- The cases ($\stackrel{?}{=}$**a**), ($\stackrel{?}{=}$**f**) and ($\stackrel{?}{=}$**X**) are straightforward.
- The case ($\stackrel{?}{=}$[**a**]). Suppose $Pr = [a]r \stackrel{?}{=} [a]s, Pr'$ and $[a]r \stackrel{?}{=} [a]s, Pr' \Longrightarrow r \stackrel{?}{=} s, Pr'$ by ($\stackrel{?}{=}$[**a**]). Then:
  - Suppose $([a]r)\theta =_\alpha ([a]s)\theta$. By Definition 3.2, $[a](r\theta) =_\alpha [a](s\theta)$. By the rules in Definition 2.13, $r\theta =_\alpha s\theta$. The result follows.
  - Suppose $r\theta =_\alpha s\theta$. By the rules in Definition 3.2, $[a](r\theta) =_\alpha [a](s\theta)$. By Definition 3.2, $([a]r)\theta =_\alpha ([a]s)\theta$, as required.
- The case ($\stackrel{?}{=}$[**b**]). Suppose $Pr = [a]r \stackrel{?}{=} [b]s, Pr'$, $b \notin fa(r)$ and $Pr \Longrightarrow (b\ a) \cdot r \stackrel{?}{=} s, Pr'$ with ($\stackrel{?}{=}$[**b**]). Then:
  - Suppose $([a]r)\theta =_\alpha ([b]s)\theta$. By Definition 3.2, $[a](r\theta) =_\alpha [b](s\theta)$. By the rules in Definition 2.13, $(b\ a) \cdot (r\theta) =_\alpha s\theta$. By Lemma 3.4 and Proposition 2.21, $((b\ a) \cdot r)\theta =_\alpha s\theta$. The result follows.

- Suppose $((b\ a)\cdot r)\theta =_\alpha s\theta$. By Lemma 3.4 and Proposition 2.21, $(b\ a)\cdot(r\theta) =_\alpha s\theta$. By Lemma 3.3, $b \notin fa(r\theta)$. Using $(=_\alpha[\mathbf{b}])$, $[a](r\theta) =_\alpha [b](s\theta)$. By Definition 3.2 $[a](r\theta) =_\alpha [b](s\theta)$, as required.

□

Proof of Lemma 6.14:

Proof. As the empty set cannot be simplified, it must be that $Pr = r \stackrel{?}{=} s, Pr'$. It therefore suffices to perform case analysis on the simplification of $r \stackrel{?}{=} s$. At each stage, without loss of generality, assume Pr' has been simplified by non-instantiating rules as much as possible.

- The cases $(\stackrel{?}{=}\mathbf{a})$, $(\stackrel{?}{=}\mathsf{f})$ and $(=_\alpha\mathbf{X})$ are routine.
- The case $(\stackrel{?}{=}[\mathbf{a}])$. Suppose $\mathcal{V}; [a]r \stackrel{?}{=} [a]s, Pr'$ and $fV([a]r \stackrel{?}{=} [a]s, Pr') \subseteq \mathcal{V}$, then $\mathcal{V}; [a]r \stackrel{?}{=} [a]s, Pr' \Longrightarrow \mathcal{V}; r \stackrel{?}{=} s, Pr'$ by $(\stackrel{?}{=}[\mathbf{a}])$. By Definitions 2.12 and 6.4, $fV(r \stackrel{?}{=} s, Pr') \subseteq \mathcal{V}$, and the result follows.
- The case $(\stackrel{?}{=}[\mathbf{b}])$. Suppose $\mathcal{V}; [a]r \stackrel{?}{=} [b]s, Pr'$, $b \notin fa(r)$ with $fV([a]r \stackrel{?}{=} [b]s, Pr') \subseteq \mathcal{V}$, then $\mathcal{V}; [a]r \stackrel{?}{=} [b]s, Pr' \Longrightarrow \mathcal{V}; (b\ a)\cdot r \stackrel{?}{=} s, Pr'$ by $(\stackrel{?}{=}[\mathbf{a}])$. By Definitions 2.12 and 6.4 it follows that $fV((b\ a)\cdot r) \subseteq \mathcal{V}$, and the result follows.

□

Proof of Proposition 7.12:

Proof. We prove reflexivity by induction on terms; symmetry by induction on derivations; transitivity by induction on the size of a term. We include one case from the proof of symmetry, and one from the proof of transitivity:

- $\lambda a.g =_\alpha \lambda b.h$ implies $\lambda b.h =_\alpha \lambda a.g$. Suppose $(b\ a)\cdot g =_\alpha h$ and $b \notin fa(g)$. By Lemma 7.7, $a \notin fa((b\ a)\cdot h)$. By Lemmas 7.8 and 7.6, $g =_\alpha (b\ a)\cdot h$. By Lemma 7.9, $a \notin fa(h)$. By inductive hypothesis $(b\ a)\cdot h =_\alpha g$. Extending with $(\lambda=_\alpha\lambda[\mathbf{b}])$ we obtain $\lambda b.h =_\alpha \lambda a.g$, as required.
- $\lambda a.g =_\alpha \lambda b.h$ and $\lambda b.h =_\alpha \lambda c.k$ implies $\lambda a.g =_\alpha \lambda c.k$. Suppose $(b\ a)\cdot g =_\alpha h$, $(c\ b)\cdot h =_\alpha k$, $b \notin fa(g)$ and $c \notin fa(h)$. By Lemma 7.6, $(c\ b)\cdot((b\ a)\cdot g) =_\alpha (c\ b)\cdot h$. By Lemmas 7.11, 7.6 and 7.5 we have $(c\ a)\cdot g =_\alpha k$. By Lemma 7.9, $c \notin fa((b\ a)\cdot g)$. By Lemma 7.8, $c \notin fa(g)$. We use $(\lambda=_\alpha\lambda\mathbf{ab})$ to obtain $\lambda a.g =_\alpha \lambda c.k$, and the result follows.

□

Proof of Lemma 8.9:

Proof. By induction on r.

- The cases a and $\mathsf{f}(r_1,\ldots,r_n)$ are routine.
- The case $[a]r$. We reason as follows:

$$\begin{array}{rll} capt_A(\pi\cdot[a]r) & = capt_A([\pi(a)](\pi\cdot r)) & \text{Definition 2.9} \\ & = capt_{A\cup\{\pi(a)\}}(\pi\cdot r) & \text{Definition 8.7} \end{array}$$

There are now two cases:

- The case $\pi(a) = a$. Then:

$$\begin{aligned} capt_{A\cup\{\pi(a)\}}(\pi \cdot r) &= capt_{A\cup\{a\}}(\pi \cdot r) && \text{Assumption} \\ &= capt_{A\cup\{a\}}(r) && \text{Inductive hypothesis} \\ &= capt_A([a]r) && \text{Definition 8.7} \end{aligned}$$

  The result follows.
- The case $\pi(a) \neq a$. Then:

$$\begin{aligned} capt_{A\cup\{\pi(a)\}}(\pi \cdot r) &= capt_A(\pi \cdot r) && \text{Assumption, } \pi(a) \neq a \\ &= capt_A(r) && \text{Inductive hypothesis} \\ &= capt_A([a]r) && \text{Definition 8.7} \end{aligned}$$

  The result follows.

- The case $\pi' \cdot X^S$. We reason as follows:

$$\begin{aligned} capt(\pi \cdot (\pi' \cdot X^S)) &= capt_A((\pi \circ \pi') \cdot X^S) && \text{Lemma 2.16} \\ &= (nontriv(\pi \circ \pi') \cup A) \cap S && \text{Definition 8.7} \\ &= (nontriv(\pi) \cup nontriv(\pi') \cup A) \cap S && \text{Fact} \\ &= (nontriv(\pi') \cup A) \cap S && \text{Assumption} \\ &= capt_A(\pi' \cdot X^S) && \text{Definition 8.7} \end{aligned}$$

  The result follows.

□

Proof of Corollary 8.10:

Proof. We reason as follows:

$$\begin{aligned} capt_A([b]r) &= capt_{A\cup\{b\}}(r) && \text{Definition 8.7} \\ &= capt_{A\cup\{a,b\}}(r) && \text{Lemma 8.8, } a \notin fa(r) \\ &= capt_{A\cup\{a,b\}}((b\ a) \cdot r) && \text{Lemma 8.9} \\ &= capt_{A\cup\{a\}}((b\ a) \cdot r) && \text{Lemmas 8.8 and 2.17} \\ &= capt_A([a](b\ a) \cdot r) && \text{Definition 8.7} \end{aligned}$$

The result follows. □

Proof of Lemma 8.13:

Proof. By induction on r.

- The cases a and $\mathsf{f}(r_1, \ldots, r_n)$ are straightforward.
- The case $[a]r$. Suppose $a \in capt_A([a]r)$. Then $a \in capt_{A\cup\{a\}}(r)$, and by inductive hypothesis $X^S \in fV(r)$ exists such that $a \in S$. As $fV([a]r) = fV(r)$, the result follows.
- The case $[b]r$. Suppose $a \in capt_A([b]r)$. Then $a \in capt_{A\cup\{b\}}(r)$, and by inductive hypothesis $X^S \in fV(r)$ exists such that $a \in S$. As $fV([b]r) = fV(r)$, the result follows.
- The $\pi \cdot X^S$. The result follows immediately by Definition 8.7.

□

Proof of Lemma 9.11:

Proof. By induction on r.

- The cases a and $\mathsf{f}(r_1, \ldots, r_n)$. Routine.
- The case $[a]r$. By Definitions 2.9 and 8.7

$$capt_A(\pi \cdot [a]r) = capt_{A \cup \{\pi(a)\}}(\pi \cdot r).$$

By inductive hypothesis

$$capt_{A \cup \{\pi(a)\}}(\pi \cdot r) \subseteq ((nontriv(\pi) \cup A \cup \{\pi(a)\}) \cap uncapt(r)) \cup capt(r).$$

By Definition 9.8 $uncapt([a]r) = uncapt(r) \setminus \{a\}$, so it suffices to show that

$$\big((nontriv(\pi) \cup A \cup \{\pi(a)\}) \cap uncapt(r)\big) \cup capt(r) \subseteq \big((nontriv(\pi) \cup A) \cap (uncapt(r) \setminus \{a\})\big) \cup capt([a]r).$$

By Lemma 9.9 $capt(r) \subseteq capt([a]r)$. Therefore, we only need concern ourselves with the 'extra $\pi(a)$' on the left, and the 'missing a' on the right.
We consider cases for the 'extra $\pi(a)$':

- Suppose $\pi(a) \neq a$. Then $\pi(a) \in nontriv(\pi)$.
- Suppose $\pi(a) = a$ and $a \notin uncapt(r)$. Then $\{\pi(a)\} \cap uncapt(r) = \varnothing$.
- Suppose $\pi(a) = a$ and $a \in uncapt(r)$. Then by Lemma 9.10 $a \in capt([a]r)$.

We consider cases for the 'missing a':

- Suppose $a \notin uncapt(r)$. Then $uncapt(r) \setminus \{a\} = uncapt(r)$.
- Suppose $a \in uncapt(r)$. Then by Lemma 9.10 $a \in capt([a]r)$.

In all cases, the result follows.

- The case $\pi' \cdot X^S$. Then:

$$\begin{aligned} capt_A((\pi \circ \pi') \cdot X^S) &= (nontriv(\pi \circ \pi') \cup A) \cap S \\ &\subseteq (nontriv(\pi) \cup nontriv(\pi') \cup A) \cap S \\ &= (((nontriv(\pi) \cup A) \setminus nontriv(\pi')) \cap S) \cup (nontriv(\pi') \cap S) \\ &= (((nontriv(\pi) \cup A) \setminus nontriv(\pi')) \cap S) \cup capt(\pi' \cdot X^S) \\ &= ((nontriv(\pi) \cup A) \cap (S \setminus nontriv(\pi'))) \cup capt(\pi' \cdot X^S) \\ &= ((nontriv(\pi) \cup A) \cap uncapt(\pi' \cdot X^S)) \cup capt(\pi' \cdot X^S) \end{aligned}$$

The result follows.

□