MURDOCH J. GABBAY

NOMINAL TERMS AND NOMINAL LOGICS: FROM FOUNDATIONS TO META-MATHEMATICS

ABSTRACT: Nominal techniques concern the study of names using mathematical semantics. Whereas in much previous work names in abstract syntax were studied, here we will study them in meta-mathematics. More specifically, we survey the application of nominal techniques to languages for unification, rewriting, algebra, and first-order logic.

What characterises the languages of this chapter is that they are first-order in character, and yet they can specify and reason on names. In the languages we develop, it will be fairly straightforward to give first-order 'nominal' axiomatisations of name-related things like alpha-equivalence, capture-avoiding substitution, beta- and eta-equivalence, first-order logic with its quantifiers—and as we shall see, also arithmetic. The formal axiomatisations we arrive at will closely resemble 'natural behaviour'; the specifications we see typically written out in normal mathematical usage.

This is possible because of a novel name-carrying semantics in nominal sets, through which our languages will have name-permutations and term-formers that can bind as primitive built-in features.

This chapter draws together material from several papers to deliver a coherent account of a journey from the foundations of a mathematics with names, via logical systems based on those foundations, to concrete applications in axiomatising systems with binding. Definitions and proofs have been improved, generalised, and shortened, and placed into an overall narrative.

On the way we touch on a variety of definitions and results. These include: the nominal unification algorithm; nominal rewriting and its confluence proofs; nominal algebra, its soundness, completeness, and an HSP theorem; permissive-nominal logic and its soundness and completeness; various axiomatisations with pointers to proofs of their correctness; and we conclude with a case study stating and proving correct a finite first-order axiomatisation of arithmetic in permissive-nominal logic.

D. Gabbay and F. Guenthner (eds.),
*Handbook of Philosophical Logic, Volume 17, 1–**??**.*
© 2012, Kluwer Academic Publishers. Printed in the Netherlands.

CONTENTS

1 INTRODUCTION

Nominal sets for meta-mathematics Suppose we want to axiomatise the λ-calculus or first-order logic. Then we need to express properties like this:

- If $y \notin fv(t)$ then $\forall x.t =_\alpha \forall y.(t[y/x])$.
- If $x \notin fv(u)$ then $(\lambda x.t)[u/y] = \lambda x.(t[u/y])$.
- If $x \notin fv(t)$ then $\lambda x.(tx) =_\eta t$.

x, y, t, and u here are what we would call *names*. A linguist might call them *referents*, a mathematician might call them *variables*. But the words ‘referent’ and ‘variable’ carry connotations (a referent should refer to something, a variable should vary), so we prefer the more neutral term ‘name’. So for us, a name is just an atomic symbol, to which we may then associate further properties, at our discretion, using additional axioms.

The axioms above are typical of a certain kind of specification. Mathematical specification is nothing new. First-order logic can specify, to choose a classic trio of examples, groups, rings, and fields. But the λ-calculus, first-order logic itself, the π-calculus, and a very great many other examples, are different. They have *names*.

By adding names to first-order logic in the correct way, we can axiomatise the specifications above, cleanly and in a manner very close to the informal specification. How should we do this? Using a recent application of mathematical foundations originating in computer science: *nominal sets* [GP01; Gab11b], to which we will use *nominal terms* [UPG03; UPG04] as a corresponding formal syntax. To survey and update the state of the art of logics based on nominal terms and taking semantics in nominal sets, is our goal here.

In nominal terms, term-formers can bind names and freshening renamings like the $[y/x]$ or $[u/y]$ above are taken as primitive.

Here are the informal statements above, rewritten in permissive-nominal algebra—an algebraic logic based on nominal terms with a sound and complete semantics in nominal sets:

- If $b \notin \mathit{supp}(X)$ then $\forall([a]X) = \forall([b](b\ a){\cdot}X)$.
- If $a \notin \mathit{supp}(Y)$ then $\lambda([a]X)[b{\mapsto}Y] = \lambda([a](X[b{\mapsto}Y]))$.
- If $a \notin \mathit{supp}(X)$ then $\lambda([a](Xa)) = X$.

In this chapter we will briefly consider nominal sets, then survey nominal terms, unification, rewriting, algebra, and permissive-nominal logic. We cover the nominal unification algorithm, confluence proofs for nominal rewriting, soundness and completeness results for nominal algebra and permissive-nominal logic, an HSP theorem, and a finite axiomatisation of first-order logic.

By doing this we aim to give an overview of the applications of nominal sets to meta-mathematical syntax. We cannot be exhaustive, but we can try to be representative of what can be achieved.

As we shall see, nominal syntax is more expressive than first-order syntax (for instance we can give a finite first-order axiomatisation of arithmetic), because term-formers that can explicitly manipulate names. Yet, it remains first-order in flavour, preserving theoretical and computational properties like completeness and most general unifiers.

A few words on atoms What nominal sets add to 'ordinary' structures is an assumption of a distinguished class of symmetric atomic elements called *atoms*: these are also called *urelemente* or *names*. We will use these terms more-or-less synonymously.

Indeed, nominal sets are a special case Zermelo-Fraenkel sets with atoms, and are instances of the structures considered by Fraenkel and Mostowski in their celebrated independence proof of the the Axiom of Choice from the other axioms of set theory with atoms. For detailed references see [Gab11b, Remark 2.22]. So this chapter really does describe a journey from mathematical foundations to meta-mathematics, and that is representative of how the maths we describe here was arrived at.

We can view the underlying philosophy of nominal techniques is as the following informal inequality, where 'smaller' means 'greater generality':

$$\text{atoms} = \text{urelemente} = \text{names} \leq \text{referents} \leq \text{variables}$$

Discovering to what extent these intuitions can be made precise, concrete, and useful, is the topic of much ongoing research, some of which is reported on here.

Names induce automorphisms generated by permuting them. We shall see that if we model variables as a special case of atoms, then α-renaming becomes a special case of a much more general fact that nominal sets are symmetric under permuting atoms. This generalisation turns out to have powerful consequences, including the atoms-abstraction and И-quantifier introduced by the author with Pitts in [GP01]. So the point of view described above has led to and continues to lead to new reasoning principles.

If we identify a thing with the properties of that thing, then the 'nominal' model suggests that names are equal to the following set of three properties:

$$\text{names} = \{\text{atomic}, \text{symmetric}, \text{generative}\}$$

The reader familiar with nominal techniques can identify these three properties with the use of: atomic symbols a (an atom, name, or urelement, with a distinct existence in the denotation), permutations π (symmetries under permutation of names), and the И-quantifier ('choose a fresh name'). These three properties will appear directly in this chapter as atoms, permutations, and permission sets.[1] Full definitions appear below.

This material in the literature This paper surveys existing literature on logics based on nominal terms, and adds a few new results. Very broadly, Section 2 is based on [GP99; GP01] (nominal sets; they were called *equivariant FM sets* there); Sections 3 and 4 are based on [UPG03; UPG04; DGM09; DGM10; Gab12a] (nominal terms and unification); Sections 5 and 6 are based on [FGM04; FG07; Gab12a] (rewriting and closed terms); Section 7 is based on [Gab05; GM06a; GM07; GM09a] (nominal algebra); Sections 9 to 11 are based on [DG10; DG12a] (permissive-nominal logic).

Definitions and proofs may have changed from the original presentations. In particular:

- The semantics is *permissive*-nominal, meaning that it is based on possibly infinitely supported nominal sets with co-infinite support. In [GP01] a nominal semantics based on finite and co-infinite support was used.
- Unlike [UPG04] and [DGM10] we use nominal abstract syntax to build our nominal terms. That is, in this paper nominal terms atoms-abstraction is directly equal to Gabbay-Pitts atoms-abstraction. Thus, nominal terms here are an instance of nominal abstract syntax and come quotiented by α-equivalence by construction.
- Permutation may be stronger than usual, and we parameterise over the group of permutations.
  We consider (as usual) finite permutations (generated by *swappings*, also called *transpositions*) as standard, but in particular we also find *shift*-permutations δ useful, which shift infinitely many atoms. The *shift*-permutation δ corresponds

[1]In other papers, such as [UPG04], permission sets are presented instead as syntactic freshness assumptions.

to a de Bruijn shift function $\uparrow$ and presheaf reindexing map up, though δ is not equal to them since it is a permutation and so invertible.

- Syntax includes non-equivariant constant symbols. In [UPG04] all term-formers/function-symbols (including 0-ary ones, i.e. constants) were equivariant. This does not matter for finite support but it does make a difference with infinite support.
- Nominal unknowns are modelled as arbitrary elements of a strongly-supported nominal set. This means that the X and Y in this paper correspond to *moderated unknowns* from [UPG04]: see Example 3.1.7.
- Because unknowns have support, there are no freshness contexts and substitutions are characterised as equivariant functions (the freshness conditions normally attached to substitutions follow from equivariance: see Proposition 3.4.3). The theories of nominal unification, rewriting, and algebra are reformulated to reflect this.
- The simplification rules for unification problems (Figure 2) are new and the treatments of closed terms and closed nominal rewriting (Section 6) are entirely revised with respect to [FG07].

Nominal sets and nominal terms

2 NOMINAL SETS

We open with a brief presentation of nominal sets, which are the semantic basis for this work: this is the universe that the logics we define will describe, and be sound (and complete) for.

Nominal sets were developed with Pitts and introduced in the author's thesis [Gab01], a conference paper [GP99], and journal paper [GP01]. The nominal sets here are more general than in [GP01]: following [DGM10] we are *permissive*, meaning that we split the set of atoms into two infinite halves and consider infinite support. This specific idea was developed jointly with Dowek,[2] but shades of it appear also in Cheney's paper [Che06] and in the author's study of infinite atoms-abstraction [Gab07b].

In addition we parameterise over a group of permutations which need not just be finitely-supported permutations. This is new.

2.1 Atoms, permutations, permission sets

In Definition 2.1.2 we need several sets of atoms. This is to model the several sorts of names that will appear in our syntax later on.

Following [DGM10] our development will be *permissive*-nominal. A permission set S splits a set of atoms into two halves $\mathbb{A}^<$ and $\mathbb{A}^>$. One intuition for $\mathbb{A}^<$ is 'the atoms that have been generated so far', and for $\mathbb{A}^>$ is 'the atoms that might be generated later'.

DEFINITION 2.1.1. Write $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ for the natural numbers and $\mathbb{Z} = \{0, \text{-}1, 1, \text{-}2, 2, \ldots\}$ for the integers.

DEFINITION 2.1.2. For each $i \in \mathbb{N}$ fix a pair of disjoint countably infinite sets of **atoms** $\mathbb{A}^<_i$ and $\mathbb{A}^>_i$.

Write

$$\mathbb{A}_i = \mathbb{A}^< \uplus \mathbb{A}^> \qquad \mathbb{A}^< = \bigcup \mathbb{A}^<_i \qquad \mathbb{A}^> = \bigcup \mathbb{A}^>_i \qquad \mathbb{A} = \bigcup \mathbb{A}_i$$

$a, b, c, \ldots$ will range over *distinct* atoms: we call this the **permutative** convention.

REMARK 2.1.3 (*Comments on splitting the set of atoms*). The different sets of atoms $\mathbb{A}_i$ are different 'types' of atoms. Thus, later on in Definitions 3.1.1 and 3.2.1 we can give each name sort its own distinct population of atoms.

The reasons for splitting the set of atoms into $\mathbb{A}^<$ and $\mathbb{A}^>$ will become clear as the maths develops. It might help to think of $\mathbb{A}^<$ as 'atoms that can be captured' and of $\mathbb{A}^>$ as 'atoms that cannot be captured', or as 'atoms that might have been generated in the past' and 'atoms that may be generated in the future'—but with reservations. In Definition 2.1.10 we see that this is only true up to permuting atoms.

[2]The development here is a little different from that in [DGM10] because we take permission sets to be sets of the form $\pi\cdot\mathbb{A}^<$ instead of sets of the form $(\mathbb{A}^< \setminus A) \cup B$.

The real purpose of Definition 2.1.2 is to ensure that we have plenty—countably infinitely many—of 'capturable' and 'non-capturable' atoms. Permutations (below) can and will move atoms between these worlds, but no permutation can move them *all at once*. So the interest of $\mathbb{A}^<$ is not just for the set itself but for its orbit under permutations; this is a property of the set as a whole, and not of its individual elements.

REMARK 2.1.4 (*Comments on the permutative convention*). While visiting Tel-Aviv University in 2006 I gave talks on nominal techniques and Arnon Avron asked: "Do a and b refer to specific atoms (e.g. in the axioms in the Introduction), or to any two atoms?". In other words, are a and b constants or variables?

In response I started using a *permutative convention* that a and b are variables, but they range over distinct atoms so that variables with distinct names refer to distinct objects (the first uses were in [GM06c; GM06a]; the convention was explicitly named in [GM08c]).

For a while this was resisted by some anonymous referees. Yet, we typically apply the permutative convention informally; e.g. we silently assume that $\lambda x.\lambda y.xy$ is never the same term as $\lambda x.\lambda x.xx$. I would claim that the permutative convention expresses something about the foundational origins of the nominal view of names as *urelemente*—constants that are distinguishable yet symmetric—in an underlying set theory.

Perhaps this is why the referees did not like it: the permutative convention may seem unnatural if we are committed to standard (nameless) Zermelo-Fraenkel foundations, since names are then just some set, and like any set should be varied over non-permutatively by variables. Thus the fact that we accept that $\lambda x.\lambda y.xy$ and $\lambda x.\lambda x.xx$ always signify distinct λ-terms to us, can be taken as a sign that we inhabit a nameful foundation, so that the permutative convention is a signpost on the way to something more extensive.

A formal reflection of the permutative convention appears explicitly in the formal logics of this paper: it lives in the π of the $\pi{\cdot}X$ in Definition 5.2.1.

DEFINITION 2.1.5. Given $a, b \in \mathbb{A}_i$ for some $i \in \mathbb{N}$ write $(a\ b)$ for the **swapping** bijection on atoms mapping a to b, b to a, and any other $c \in \mathbb{A} \setminus \{a, b\}$ to c.

Another standard name for a swapping is a **transposition**.

By convention $(a\ a)$ will denote the **identity** function on atoms id.

If π is a bijection on atoms define

$$nontriv(\pi) = \{a \mid \pi(a) \neq a\}.$$

DEFINITION 2.1.6. A **nominal permutation group** is any set of bijections $\mathbb{P}$ of $\mathbb{A}$ such that:

1. If $a \in \mathbb{A}_i$ and $b \in \mathbb{A}_i$ then $(a\ b) \in \mathbb{P}$.
2. If $\pi \in \mathbb{P}$ then $a \in \mathbb{A}_i$ if and only if $\pi(a) \in \mathbb{A}_i$.
3. There exists some infinite $S \subseteq \mathbb{A}$ such that $nontriv(\pi) \cap S$ is finite for every $\pi \in \mathbb{P}$.

Call a bijection on atoms π a **finite permutation** when it is in the subgroup generated by swappings. (π is finite when $\pi(a) \in \mathbb{A}_i$ if and only if $a \in \mathbb{A}_i$ and $nontriv(\pi)$ is finite.)

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).

The purpose of conditions 1 to 3 of Definition 2.1.6 are as follows:

1. Swappings make sure we can always rename a to b (and b to a).
2. Condition 2 is a standard typing condition, that we do not try to turn an atom of one sort, into an atom of another sort.
3. This condition guarantees that we can still always choose a fresh atom for any finite set of permutations (see for instance Lemma 3.2.9).

EXAMPLE 2.1.7.

1. The set of all finite permutations is a nominal permutation group.

2. For each i fix a bijection f_i between $\mathbb{A}_i$ and the integers $\mathbb{Z}$, such that $\{f(i) \mid i \leq 0\} = \mathbb{A}_i^<$ and (consequently) $\{f(i) \mid i > 0\} = \mathbb{A}_i^>$. We can do this because we assumed atoms are countable.

   Write δ_i for the permutation mapping

   - $f_i(j)$ to $f_i(j{-}1)$ for $j \leq 0$,
   - $f_i(2j)$ to $f_i(2(j{-}1))$ and $f_i(2j{-}1)$ to $f_i(2j{-}1)$ for $j \geq 1$, and
   - any other $c \in \mathbb{A} \setminus \mathbb{A}_i$ to c.

   This is an example of a *shift*-permutation, considered in more generality in Definition 3.6.1 and throughout Subsection 3.6. We illustrate fragments of the actions of a swapping $(f(0)\ f(1))$ and a δ_i:

$$\begin{tikzcd}
f(\text{-}6) \arrow[loop below] & f(\text{-}5) \arrow[loop below] & f(\text{-}4) \arrow[loop below] & f(\text{-}3) \arrow[loop below] & f(\text{-}2) \arrow[loop below] & f(\text{-}1) \arrow[loop below] & f(0) \arrow[r, bend left, leftrightarrow] & f(1) & f(2) \arrow[loop below] & f(3) \arrow[loop below] & f(4) \arrow[loop below] & f(5) \arrow[loop below] & f(6) \arrow[loop below]
\end{tikzcd}$$

$$\begin{tikzcd}
f(\text{-}6) & f(\text{-}5) \arrow[l, bend right] & f(\text{-}4) \arrow[l, bend right] & f(\text{-}3) \arrow[l, bend right] & f(\text{-}2) \arrow[l, bend right] & f(\text{-}1) \arrow[l, bend right] & f(0) \arrow[l, bend right] & f(1) \arrow[loop below] & f(2) \arrow[ll, bend right] & f(3) \arrow[loop below] & f(4) \arrow[ll, bend right] & f(5) \arrow[loop below] & f(6) \arrow[ll, bend right]
\end{tikzcd}$$

   The atoms corresponding to positive odd integers are taken to be fixed points of δ_i in order to satisfy condition 3 of Definition 2.1.6, so that these atoms can be taken fresh for δ_i if we need to.

   The set of permutations generated by swappings and δ_i, is a nominal permutation group.

REMARK 2.1.8. The nominal permutation group $\mathbb{P}$ determines the symmetries of our nominal syntax and semantics. We consider permutations designed to guarantee (in Definition 2.2.3) symmetry up to equality/inequality of atoms. We will get sets with atoms that are atomic, symmetric (up to equality and inequality of names), and generative—the main further design choice we care about is whether or not to include a *shift* (Example 2.1.7), which goes strictly beyond what can be achieved with finite permutations as considered e.g. in [GP01].

Other notions of permutation may lead to other symmetries, so an interesting topic of future research is to weaken the conditions in Definition 2.1.6.

For instance, if we only allow permutations generated by $f(i) \mapsto f(i+1)$ and $f(i) \mapsto f(\text{-}i)$ then we preserve a notion of 'distance' between atoms.[3] In a similar vein, we can identify atoms with points in a plane and consider Euclidian transformations. It is not known how much of 'nominal techniques' would hold of such examples.

More generally of course, presheaves are a forum within which sets with symmetry structure can be expressed. Indeed, nominal sets can be viewed as a category of presheaves [GP99] and a similar presheaf category was considered at the same time [FPT99] (see also the later related *nominal renaming sets* [GH08], which are in some sense half-way in between those two systems).

There is no shortage of research into this kind of structure [MM92]. It remains, however, to understand what are the abstract properties that make a set with a group action, or a presheaf, into something 'nominal'.

DEFINITION 2.1.9. If $A \subseteq \mathbb{A}$ define the **pointwise** action by

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

DEFINITION 2.1.10. A **permission set** S is a set of the form $\pi \cdot \mathbb{A}^{<}$.
S, T will range over permission sets.

REMARK 2.1.11. Some preliminary comments on permission sets:

- The notion of permission set used in some previous work, for instance in [DGM10, Definition 2.2], was slightly different: a permission set was taken to be a set of the form $(S \setminus A) \cup B$ for finite $A \subseteq \mathbb{A}^{<}$ and $B \subseteq \mathbb{A}^{>}$. In the presence of *shift*-permutations we can do this using a permutation, and any $(S \setminus A) \cup B$ can be written as $\pi \cdot S$ for suitable π (cf. Remark 3.6.2 and $\delta_{X\text{-}a} \cdot X$ in (**IF**) of Figure 2).
  Given that the designs are equivalent for the cases we will care about, we chose Definition 2.1.10 because it is somewhat simpler to do mathematics with.
- In the semantics, permission sets are used in the definition of support Definition 2.2.3; if permutations specify *symmetry*, permission sets specify *capturability* and *generativity* (Remark 2.2.4).
- In the syntax, permission sets are used to control capture (see Remark 3.4.9); atoms in S are intuitively 'capturable' and atoms not in S are intuitively 'not capturable'.
  This is reminiscent of some treatments of syntax where a formal distinction is made between 'names that exist to be bound' and 'names that exist to be free'. See for instance the *freie* and *gebundene Gegenstansvariable* of Gentzen [Gen35, Section 1], and the *individual variables* and *parameters* of Prawitz [Pra65, Section 1], or Smullyan [Smu68, Chapter IV, Section 1].
  However, note that here, for any $a \in S$ and $b \notin S$, also $a \notin (b\ a) \cdot S$ and $b \in (b\ a) \cdot S$. That is, for any given atom there is no fixed sense in which it is

[3]This example modified from an example by Bartek Klin; private communication from Alexander Kurz.

capturable or not capturable. Each individual permission sets defines its own world of capturable/non-capturable atoms, which differs by a permutation π from what is really a fixed but entirely arbitrary representative $\mathbb{A}^<$.

2.2 Permissive-nominal sets

DEFINITION 2.2.1. A **set with a ($\mathbb{P}$-)permutation action** X is a pair $(|\mathsf{X}|, \cdot)$ of

- a **carrier set** $|\mathsf{X}|$ and
- a group action $(\mathbb{P} \times |\mathsf{X}|) \to |\mathsf{X}|$, written infix as $\pi \cdot x$.
  So, $id \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$ for every π, π', and $x \in |\mathsf{X}|$.

DEFINITION 2.2.2. Given a set with a $\mathbb{P}$-permutation action X say that $A \subseteq \mathbb{A}$ **supports** $x \in |\mathsf{X}|$ when for all permutations $\pi \in \mathbb{P}$, if $\pi(a) = a$ for all $a \in A$ then $\pi \cdot x = x$.

Also, call $A \subseteq \mathbb{A}$ **small** when $A \subseteq S$ for some permission set S.

DEFINITION 2.2.3. A **permissive-nominal set** is a set with a permutation action such that every element has a unique least small supporting set $supp(x)$. We call this the **support** of x.
X, Y will range over permissive-nominal sets.

Note in Definition 2.2.3 that $supp(x)$ must be *small*, that is, included in some permission set. For instance, $a \in \mathbb{A}$—with $\mathbb{A}$ having the natural permutation action given by $\pi \cdot x = \pi(x)$ for $x \in \mathbb{A}$—is supported by $\{a\}$ and $\mathbb{A} \setminus \{a\}$, but the former is small while the latter is not.

REMARK 2.2.4. The difference between a set with a permutation action and a 'nominal' set is that nominal sets guarantee for any element, infinitely many atoms fresh for that element.

A mild generalisation of Definition 2.2.3 is possible, where we insist there is a supporting set but do not insist on the existence of a unique *least* such set. It is possible to do a surprising amount just with that; see for instance Fiore, Plotkin and Turi's paper [FPT99] based on presheaves, and the 'nominal' study of infinite permutations and infinite atoms-abstraction in [Gab07b].[4]

EXAMPLE 2.2.5.

[4] If all permutations in $\mathbb{P}$ are finite then we have as a *Technical Lemma* that the existence of *some* supporting set implies the existence of a unique least small supporting set.

In the more general case where infinite permutations are allowed, it is possible to construct a set with a permutation action X and $x \in |\mathsf{X}|$ such that x has a supporting set but does not have a unique least small supporting set. See [Gab07b, Lemma 21] for an example.

An intermediate state is to admit infinite permutations but restrict the notion of support to consider only the finite ones. We do this in Definitions 3.1 and 3.2 and Remark 3.3 of [DG12a].

For this paper, none of this will matter directly.

- First-order syntax with variable symbols (modelled as atoms) is a permissive-nominal set, where the permutation action permutes variable symbols directly in syntax so that e.g. $\pi{\cdot}\lambda a.t = \lambda\pi(a).\pi{\cdot}t$.
  A term t is supported by the variable symbols it contains. In this and the following examples the precise nature of the permutation group is not important.
- First-order syntax up to α-equivalence is a permissive-nominal set. The α-equivalence class of t is supported by the free variable symbols of t. A full proof is in [Gab11b, Theorem 5.18].
- Traces of π-calculus processes with channel names (atoms) taken from some permission set S, form a permissive-nominal set. A trace is supported by the set of channel names it mentions (which may be infinite in number).
- Given a permissive-nominal nominal set X the set of subsets $U \subseteq |\mathsf{X}|$ with the pointwise action $\pi{\cdot}U = \{\pi{\cdot}u \mid u \in U\}$ is a set with a permutation action (this generalises Definition 2.1.9).
  The subset of this consisting of those subsets $U \subseteq |\mathsf{X}|$ that have a supporting permission set under this action, forms a permissive-nominal set $pow(\mathsf{X})$.[5]

LEMMA 2.2.6. Suppose X is a permissive-nominal set and $x \in |\mathsf{X}|$. Then $supp(\pi{\cdot}x) = \pi{\cdot}supp(x)$.

Proof. By a routine calculation using the group action. ■

We conclude with a useful condition for checking whether $a \in supp(x)$:

COROLLARY 2.2.7. Suppose X is a permissive-nominal set and $x \in |\mathsf{X}|$. Suppose $b \notin supp(x)$. Then $(b\ a){\cdot}x = x$ if and only if $a \notin supp(x)$.

Proof. Suppose $b \notin supp(x)$. The right-to-left implication is by the definition of support. For the left-to-right implication, we prove the contrapositive. Suppose $a \in supp(x)$. By Lemma 2.2.6 $supp((b\,a){\cdot}x) = (b\,a){\cdot}supp(x)$. By our suppositions, $(b\ a){\cdot}supp(x) \neq supp(x)$. It follows that $(b\ a){\cdot}x \neq x$. ■

2.3 Equivariance

DEFINITION 2.3.1. Suppose X and Y are permissive-nominal sets.

Call $x \in |\mathsf{X}|$ **equivariant** when $supp(x) = \varnothing$. (So x is equivariant when $\pi{\cdot}x = x$ for all π.)

Call $F \in |\mathsf{X}| \to |\mathsf{Y}|$ **equivariant** when

$$\boxed{\forall \pi \in \mathbb{P}.\forall x \in |\mathsf{X}|.\pi{\cdot}(F(x)) = F(\pi{\cdot}x).}$$

F will range over equivariant functions between pairs of permissive-nominal sets.

REMARK 2.3.2. The second notion of equivariance in Definition 2.3.1 is a special case of the first. For details, see e.g. Definition 9.3 and Lemma 9.4 of [Gab11b].

[5]Using possibly repeated powersets, arbitrarily complex structures may be constructed. Thus this example guarantees an inexhaustible supply of arbitrarily large and complex structures with which to model ... almost anything we can imagine. The survey [Gab11b] explores this in detail.

LEMMA 2.3.3. If F from $|\mathsf{X}|$ to $|\mathsf{Y}|$ is equivariant then $supp(F(x)) \subseteq supp(x)$ for all $x \in |\mathsf{X}|$.

Proof. Suppose $\pi \in \mathit{fix}(supp(x))$. By assumption $\pi{\cdot}F(x) = F(\pi{\cdot}x)$, and $\pi{\cdot}x = x$. ■

DEFINITION 2.3.4. Write PmsPrm for the category with objects permissive-nominal sets and arrows equivariant functions between them.

So X, Y range over objects in PmsPrm (Definition 2.2.3).

2.4 Examples of permissive-nominal sets

Throughout the rest of this document we will need the following examples of permissive-nominal sets: atoms, booleans, lists, product, equivariant elements, permutation orbits, and atoms-abstraction. We consider each in turn now.

Atoms, Booleans, infinite lists

DEFINITION 2.4.1 (Atoms). $\mathbb{A}$ the set of all atoms can be considered a permissive-nominal set with a natural permutation action $\pi{\cdot}a = \pi(a)$. So can each $\mathbb{A}_\nu$.

DEFINITION 2.4.2. If X is a permissive-nominal set say the permutation action is **trivial** when $\pi{\cdot}x = x$ for all $x \in |\mathsf{X}|$ and all $\pi \in \mathbb{P}$.

So X is trivial if and only if all its elements are equivariant.

DEFINITION 2.4.3. Any 'ordinary' set can be made into a permissive-nominal set by giving it the trivial permutation action such that $\pi{\cdot}x = x$ always.

In particular, the set $\mathbb{B} = \{0, 1\}$ can be considered a permissive-nominal set with the trivial permutation action; so can $\mathbb{N}$ and $\mathbb{Z}$ from Definition 2.1.1.

In the cases of $\mathbb{A}$ and $\{0, 1\}$ only, we will be lax about the distinction between the set, and the permissive-nominal set with its natural permutation action.

DEFINITION 2.4.4 (Infinite lists). Define a permissive-nominal set $\mathbb{L}$ by:

- $|\mathbb{L}|$ is the set of infinite sequences of distinct atoms $L = [a_1, a_2, a_3, \dots]$ such that $atms(L) = \{a_1, a_2, a_3, \dots\}$ is a permission set.
- $\pi{\cdot}L = [\pi(a_1), \pi(a_2), \pi(a_3), \dots]$.

Product

DEFINITION 2.4.5. Suppose I is an indexing set.[6] If X_i are permissive-nominal sets for $i \in I$ then define $\Pi_i\mathsf{X}_i$ by:

- $|\Pi_i\mathsf{X}_i|$ is the set of I-tuples $(x_i)_i$ such that $\forall i.x_i \in |\mathsf{X}_i|$ and there exists a permission set S such that $\forall i.supp(x_i) \subseteq S$.
- $\pi{\cdot}(x_i)_i = (\pi{\cdot}x_i)_i$ (the **elementwise** or **pointwise** action).

[6]For clarity, note that we intend this set to *not* have a permutation action. Or, we can take this to be a nominal set with the trivial action (Definition 2.4.2). We have in mind $\mathbb{N}$.

Permutation orbits

Permutation orbits will serve us later in Definition 3.3.2 (free unknowns of a term).

If X is a nominal set then $orb(\mathsf{X})$ is ‘X quotiented by the permutation action’.

DEFINITION 2.4.6. If X is a permissive-nominal set define $orb(\mathsf{X})$ by:

- If $x \in X$ then define its **permutation orbit** by $orb(x) = \{\pi{\cdot}x \mid \pi \in \mathbb{P}\}$.
- $|orb(\mathsf{X})| = \{orb(x) \mid x \in \mathsf{X}\}$.
- $\pi{\cdot}orb(x) = orb(x)$.

LEMMA 2.4.7.

- $supp(orb(x)) = \varnothing$. That is, $orb(x)$ is *equivariant* (Definition 2.3.1).
- $orb(x) = orb(y)$ if and only if $y = \pi{\cdot}x$ for some π.

Atoms-abstraction

DEFINITION 2.4.8. Suppose X is a permissive-nominal set and $\mathbb{A}_i$ is a set of atoms. Define **atoms-abstraction** $[\mathbb{A}_i]\mathsf{X}$ by:

$$\boxed{\begin{aligned}
[a]x &= \{(a,x)\} \cup \{(b,(b\ a){\cdot}x \mid b \in \mathbb{A}_i \backslash supp(x)\} \\
|[\mathbb{A}_i]\mathsf{X}| = |[\mathbb{A}_i]\mathsf{X}| &= \{[a]x \mid a \in \mathbb{A}_i,\ x \in |\mathsf{X}|\} \\
\pi{\cdot}[a]x &= [\pi(a)]\pi{\cdot}x
\end{aligned}}$$

LEMMA 2.4.9.

1. $[\mathbb{A}_i]\mathsf{X}$ is a permissive-nominal set.
2. $[a]x{=}[a]x'$ if and only if $x{=}x'$, for $a{\in}\mathbb{A}_i$ and $x{\in}|\mathsf{X}|$.
3. $[a]x{=}[a']x'$ if and only if $a'{\notin}supp(x)$ and $(a'\ a){\cdot}x{=}x'$, for $a, a'{\in}\mathbb{A}_i$ and $x, x'{\in}|\mathsf{X}|$.

LEMMA 2.4.10. Suppose a function F from $|\mathbb{A}\times\mathsf{X}|$ to $|\mathsf{Y}|$ is equivariant and suppose $\forall a, x.a \notin supp(F(a,x))$. Then there is a unique equivariant function $\hat{F}$ from $|[\mathbb{A}]\mathsf{X}|$ to $|\mathsf{Y}|$ such that $\forall a, x.\hat{F}([a]x) = F(a,x)$.

Proof. It suffices to show that if $b \notin supp(x) \cup supp(F(a,x))$ then $F(b,(b\ a){\cdot}x) = F(a,x)$. By assumption $a \notin supp(F(a,x))$, so $(b\ a){\cdot}F(a,x) = F(a,x)$. The result follows by equivariance. ■

Here are some basic properties of support:

LEMMA 2.4.11.

- $supp(a) = \{a\}$.
- $supp([a]x) = supp(x) \setminus \{a\}$.
- $supp((x_1,\dots,x_n)) = \bigcup\{supp(x_i) \mid 1 \leq i \leq n\}$.

Proof. Proofs are as in [GP01] or [Gab11b]. ■

The fine design of PmsPrm

Studying PmsPrm (Definition 2.3.4) is not the point of this paper, but for the benefit of the interested reader we will discuss a few aspects of its behaviour.

- If $\mathbb{P}$ consists of finite permutations then PmsPrm is a Boolean topos, directly generalising the category of nominal sets (equivariant FM sets) from [GP01; Gab11b]. The proof proceeds much as in [Gab11b, Corollary 9.11].
- If $\mathbb{P}$ contains infinite permutations then PmsPrm is cartesian (has products) but is not necessarily cartesian closed (may not have exponentials). This is the *fuzzy support* observed in [Gab07b]; see [Gab07b, Lemma 21] for the concrete construction. This is reasonable, and it happens because it is possible to construct a function f on $\omega+\omega$ which satisfies $f(0)=0$ and $f(i{+}1)=f(i)$ yet which is not a constant function (it returns 0 on finite cardinals and 1 on infinite ones).
- If $\mathbb{P}$ contains infinite permutations but we follow [DGM10] and take the notions of support in Definition 2.2.2 and equivariance to consider only *finite* permutations, then the category we obtain is a Boolean topos but we only have $\mathit{supp}(x)\cap \mathit{nontriv}(\pi)=\varnothing$ implies $\pi{\cdot}x=x$ for finite π. In other words, an element can be fixed by all finite permutations and have empty support, but be shifted by some infinite permutation.
  Again, this is reasonable; it is no surprise that infinite permutations can 'observe' more than finite ones.
- If $\mathbb{P}$ contains infinite permutations and we work with presheaves (in essence, we lose the 'unique least supporting set' assumption in Definition 2.2.3), then we get a topos, though it is not Boolean.

In this paper we do not attempt to reason inside PmsPrm so we do not care whether it is a topos; and we do want the possibility of infinite permutations because these let us write nice algorithms and they give our logics some useful extra expressive power (see e.g. rule $(\mathbf{IF})$ of Figure 2, Subsection 3.6, and Remark 9.2.4).

So we admit the possibility of infinite permutations in Definition 2.1.6, we let Definition 2.2.2 consider all $\pi\in\mathbb{P}$ (even infinite ones), and we insist in Definition 2.2.3 that every x have a unique least small supporting set.

In another paper, another set of design decisions might be appropriate.

The reader who does not care about these considerations need not worry; they are all swept under the carpet henceforth.

2.5 *Strong support*

Strong support exists in nominal terms, though this is implicit. Consider in [UPG04] the $\approx$-suspension rule in Figure 2, and Lemma 2.8. We call this *strong support*, following [Tze07, Definition 1].

A possibly useful intuition is that an element $x\in\mathsf{X}$ has strong support when the atoms in its support occur *in order* (a dedicated theoretical study of this is in [Gab07b]). Formally, the notion of strong support enters into the mathematics in this paper via Proposition 2.5.5, Lemma 3.4.6, and Lemma 7.4.4.

DEFINITION 2.5.1. Suppose X is a permissive-nominal set. Say $A \subseteq \mathbb{A}$ **strongly supports** $x \in |\mathsf{X}|$ when $\pi{\cdot}x = x$ if *and only if* $\forall a{\in}A.\pi(a) = a$.

If x has some strongly supporting set, call x **strongly supported**.

If every $x \in |\mathsf{X}|$ is strongly supported then call X **strongly supported**.

LEMMA 2.5.2. $x \in \mathsf{X}$ is strongly supported if and only if

$$\boxed{\forall \pi, \pi'.\big(\pi{\cdot}x = \pi'{\cdot}x \Leftrightarrow (\forall a \in \mathit{supp}(x).\pi(a) = \pi'(a))\big).}$$

Proof. From Definition 2.5.1 by considering $\pi^{\text{-}1} \circ \pi'$. ■

EXAMPLE 2.5.3.

- The pair $(a, b) \in \mathbb{A} \times \mathbb{A}$ is strongly supported by $\{a, b\}$.
- The unordered pair $\{a, b\} \subseteq \mathbb{A}$ with the pointwise permutation action (Definition 2.1.9) is not strongly supported, because $(a\ b){\cdot}\{a, b\} = \{a, b\}$.
- The infinite sequences $[a_1, a_2, a_3, \dots]$ in $\mathbb{L}$ from Definition 2.4.4 are strongly supported.

DEFINITION 2.5.4. Suppose X and Y are permissive-nominal sets and X is strongly-supported. Suppose we are given the following data:

- For each $x \in |\mathit{orb}(\mathsf{X})|$ a fixed but arbitrary choice of representative $X_x \in x$.
- For each $x \in |\mathit{orb}(\mathsf{X})|$ a choice of $y_x \in |\mathsf{Y}|$ such that $\mathit{supp}(y_x) \subseteq \mathit{supp}(X_x)$.

Define the **equivariant extension** F of this data, which is a function from $|\mathsf{X}|$ to $|\mathsf{Y}|$, by:

$$\boxed{F(\pi{\cdot}X_x) = \pi{\cdot}y_x}$$

PROPOSITION 2.5.5.

1. The equivariant extension is well-defined and is an equivariant function from $|\mathsf{X}|$ to $|\mathsf{Y}|$.
2. Every equivariant f is an equivariant extension.

Proof. For the first part, by properties of orbits every $x \in |\mathsf{X}|$ has the form $\pi{\cdot}X_x$ for some π and for precisely one X_x. This is equivariant by construction, if it is well-defined. So suppose $\pi{\cdot}X_x = \pi'{\cdot}X_x$. By assumption X_x is strongly supported so $\pi(a) = \pi'(a)$ for every $a \in \mathit{supp}(X_x)$. By assumption $\mathit{supp}(y_x) \subseteq \mathit{supp}(X_x)$. The result follows by the definition of support.

The second part is easy, noting that $\mathit{supp}(F(x)) \subseteq \mathit{supp}(x)$ by Lemma 2.3.3. ■

3 THE SYNTAX OF NOMINAL TERMS

Nominal terms were introduced in [UPG04]. The development here is permissive, following [DGM10], but with some additional ingredients: We allow non-equivariant constant symbols and we parameterise over a set of unknowns which is a *strongly-supported* [Tze07].

Some example permissive-nominal terms are given in Example 3.2.4. See also how nominal terms are used in rewrite theories (Example 5.1.3), algebra (Example 7.1.3), and first-order logic (Subsection 10.1).

3.1 Signatures

DEFINITION 3.1.1. A **sort-signature** is a tuple $(\mathcal{A}, \mathcal{B})$ of **name** and **base** sorts $\mathcal{A} \subseteq \mathbb{N}$ and $\mathcal{B}$.

ν will range over name sorts; τ will range over base sorts.

A **sort language** is defined by

$$\boxed{\alpha ::= \nu \mid \tau \mid (\alpha, \dots, \alpha) \mid [\nu]\alpha.}$$

EXAMPLE 3.1.2. Example base sorts are: 'λ-terms', 'formulae', 'π-calculus processes', and 'program environments', 'functions', 'truth-values', 'behaviours', and 'valuations'.

Base sorts τ are arbitrary; later on when we build denotations they will be populated by elements of arbitrary permissive-nominal sets, see Definition 7.3.1.

Examples of name sorts are 'variable symbols', 'channel names', 'thread identifiers', or 'memory locations'. Name sorts ν are populated by the atoms we fixed in Definition 2.1.2 and which we used to build permutations and permissive-nominal sets.

REMARK 3.1.3. $(\alpha, \dots, \alpha)$ is a product sort and behaves as expected.

$[\nu]\alpha$ is an *atoms-abstraction sort*; this is different. The behaviour of a term of sort $[\nu]\alpha$ corresponds to 'α-abstract a name of sort ν in a term of sort α'. This is *binding without functions*: we will use atoms-abstractions (Definition 2.4.8) to populate atoms-abstraction sorts.

REMARK 3.1.4. In Definition 3.1.1 we insist that a name sort ν is a natural number; this is not necessary but it makes it easier for us to identify name sorts with sets of atoms from Definition 2.1.2, which are also indexed by numbers.

DEFINITION 3.1.5. A **(nominal) term-signature** over a sort-signature $(\mathcal{A}, \mathcal{B})$ is a tuple $(\mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$ where:

- $\mathcal{C}$ is a permissive-nominal set of **constants**.
- $\mathcal{X}$ is a strongly supported (Definition 2.5.1) permissive-nominal set of **unknowns**.
- $\mathcal{F}$ is a set of equivariant **term-formers**.
- ar assigns
  - to each constant $C \in \mathcal{C}$ a base sort τ which may we write $sort(C)$,
  - to each unknown $X \in \mathcal{X}$ a sort α which we may write $sort(X)$, and
  - to each $\mathsf{f} \in \mathcal{F}$ a **term-former arity** $(\alpha)\tau$, where

  α and τ are in the sort-language determined by $(\mathcal{A}, \mathcal{B})$.

A **(nominal terms) signature** Σ is then a tuple $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$.

The support $supp(X)$ of an unknown $X \in \mathcal{X}$ is intuitively the atoms that may occur free in a term we substitute for that unknown, and $\mathbb{A} \setminus supp(X)$ is the atoms which may not occur free. See Proposition 3.4.3.

NOTATION 3.1.6. We may write $((\alpha_1, \dots, \alpha_n))\tau$ just as $(\alpha_1, \dots, \alpha_n)\tau$.

We write $\mathsf{f} : (\alpha)\tau$ for $ar(\mathsf{f}) = (\alpha)\tau$ and similarly we write $\mathsf{P} : \alpha$ for $ar(\mathsf{P}) = \alpha$.

EXAMPLE 3.1.7. Here are some examples of suitable $\mathcal{X}$.

1. For each sort α and permission set S choose a disjoint countably infinite set of **unknown symbols** $\mathtt{X}^{\mathtt{S}}_{\alpha}$, $\mathtt{Y}^{\mathtt{S}}_{\alpha}$, …Define $\pi{\cdot}\mathtt{X}^{\mathtt{S}}_{\alpha} = \{(\pi', \mathtt{X}^{\mathtt{S}}_{\alpha}) \mid \forall \mathtt{a}{\in}\mathtt{S}.\pi(\mathtt{a}) = \pi'(\mathtt{a})\}$. Let $\mathcal{X} = \{\pi{\cdot}\mathtt{X}^{S}_{\alpha} \mid \text{all } \mathtt{X}^{S}_{\alpha}, \pi\}$ with permutation action $\pi{\cdot}(\pi'{\cdot}\mathtt{X}^{S}_{\alpha}) = (\pi \circ \pi'){\cdot}\mathtt{X}^{S}_{\alpha}$. Define $ar(\pi{\cdot}\mathtt{X}^{S}_{\alpha}) = \alpha$.
   Essentially this $\mathcal{X}$ was used in [DGM10].
2. For each sort α choose a disjoint countably infinite set of **unknown symbols** $\mathtt{X}_{\alpha}$, $\mathtt{Y}_{\alpha}$, …Define $\pi{\cdot}\mathtt{X}_{\alpha} = \{(\pi', \mathtt{X}_{\alpha}) \mid \forall \mathtt{a}{\in}\mathbb{A}^{<}.\pi(\mathtt{a}) = \pi'(\mathtt{a})\}$. Let $\mathcal{X} = \{\pi{\cdot}\mathtt{X}_{\alpha} \mid \text{all } \mathtt{X}_{\alpha}, \pi\}$ with permutation action $\pi{\cdot}(\pi'{\cdot}\mathtt{X}_{\alpha}) = (\pi \circ \pi'){\cdot}\mathtt{X}_{\alpha}$. Define $ar(\pi{\cdot}\mathtt{X}_{\alpha}) = \alpha$.
3. Take $X = (\alpha, (a_0, a_1, a_2, \dots))$ where $\{a_i \mid i \in \mathbb{N}\}$ is a permission set and let $\mathcal{X}$ be the set of all possible X. Give this the pointwise permutation action $\pi{\cdot}X = (\alpha, (\pi(a_0), \pi(a_1), \dots))$ and define $ar(X) = \alpha$.
   This $\mathcal{X}$ is mathematically simple, eliminating the need to take quotients over π.
4. Take $\mathcal{X} = \{0, 1, 2, \dots\}$ with the trivial action $\pi{\cdot}x = x$, so every $x \in \mathcal{X}$ has $supp(x) = \varnothing$. This example illustrates that our framework is general enough to include the possibility of unknowns ranging over closed elements (a possibility also mooted in [FG07, Subsection 9.2]). By adding further structure to $\mathcal{X}$, further possibilities can be explored. See also [Gab11c] and [Gab12a].

In all cases it can be verified that $\mathcal{X}$ is strongly supported.

REMARK 3.1.8. In the case that $\mathcal{X}$ the set of unknowns is as described in parts 1 or 2 of Example 3.1.7, $orb(X)$ (Definition 2.4.6) may be identified with $\mathtt{X}^{S}_{\alpha}$ or $\mathtt{X}_{\alpha}$ respectively.

Vanilla	Permissive
X	Unknown with permission set $\mathbb{A}^{<}$
$a\#X$	$a \notin supp(X)$
$a\#r$	$a \notin fa(r)$
$\nabla \vdash r \to s$ or $\Delta \vdash r = s$	$r \to s$ or $r = s$
Extend freshness context	*shift*-permutation (approx)
Finite support	Small support

Figure 1: Cheat sheet relating 'vanilla' nominal terms concepts with 'permissive' ones

The $\mathcal{X}$ of part 1 above may be equivalent to that of $\mathcal{X}$ of part 2, if there exists $\pi \in \mathbb{P}$ bijecting S with $S \setminus \{a\}$ for $a \in S$. This is a *shift*-permutation; see Definition 3.6.1 and subsequent discussion.

For the benefit of the reader familiar with 'vanilla' nominal terms as used e.g. in [UPG04; FG07; GM09a], Figure 1 gives a cheat sheet suggesting how concepts in those papers map to the 'permissive' context.

EXAMPLE 3.1.9. A nominal terms signature for the λ-calculus would have one name sort ν, one base sort τ, and term-formers $\mathsf{lam} : ([\nu]\tau)\tau$, $\mathsf{app} : (\tau, \tau)\tau$, and $\mathsf{var} : (\nu)\tau$. The set of constants is empty, and for unknowns we can consider Example 3.1.7.

Usually we assume 'plenty' of variable symbols. Definition 3.1.10 makes that formal:

DEFINITION 3.1.10. Say that a signature $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$ has **enough unknowns** when for every sort α in $(\mathcal{A}, \mathcal{B})$ and every permission set S, the set $\{orb(X) \mid X \in \mathcal{X},\ sort(X) = \alpha,\ supp(X) = S\}$ is infinite.

All the examples in Example 3.1.7 have enough unknowns.

3.2 Terms

DEFINITION 3.2.1. For each signature $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$ (Definition 3.1.5) define **(permissive-nominal) terms** over Σ by:

$$\frac{(a \in \mathbb{A}_\nu,\ \nu \in \mathcal{A})}{a : \nu} \qquad \frac{(sort(C) = \tau)}{C : \tau} \qquad \frac{(sort(X) = \alpha)}{X : \alpha}$$

$$\frac{r : \alpha \quad (ar(\mathsf{f}) = (\alpha)\tau)}{\mathsf{f}(r) : \tau} \qquad \frac{r_1 : \alpha_1 \ \ldots \ r_n : \alpha_n}{(r_1, \ldots, r_n) : (\alpha_1, \ldots, \alpha_n)} \qquad \frac{r : \alpha \ \ (a \in \mathbb{A}_\nu,\ \nu \in \mathcal{A})}{[a]r : [\nu]\alpha}$$

NOTATION 3.2.2. We may write $\mathsf{f}((r_1, \ldots, r_n))$ as $\mathsf{f}(r_1, \ldots, r_n)$.

REMARK 3.2.3. Definition 3.2.1 is *nominal abstract syntax*: terms come pre-quotiented by α-equivalence by construction by virtue of our use of atoms-abstraction $[a]r$. That is, if $a \in \mathbb{A}_\nu$ and $r : \alpha$ then $[a]r$ is not a pair (a, r), it is a set $\{(a, r)\} \cup \{(b, (b\ a)\cdot r) \mid b \in \mathbb{A}_\nu \setminus \mathit{supp}(r)\}$ (Definition 2.4.8).

EXAMPLE 3.2.4. Recall the signature for the λ-calculus from Example 3.1.9. In that signature we can form terms as illustrated in the following table, where $a : \nu$ and $X : \tau$:

$a : \nu$	This is not a λ-term.
$\mathsf{var}(a) : \tau$	If we want an atom to behave like a λ-term variable, we use var to 'inject' it into τ. This corresponds to 'x'.
$[a]a : [\nu]\nu$	An atoms-abstraction. This is not a λ-term.
$[a]\mathsf{var}(a) : [\nu]\tau$	An atoms-abstraction of a λ-term. This is not a λ-term.
$\mathsf{lam}([a]\mathsf{var}(a)) : \tau$	This corresponds to '$\lambda x.x$'.
$\mathsf{lam}([a]\mathsf{app}(X, \mathsf{var}(a))) : \tau$	An open nominal term. This corresponds to '$\lambda x.tx$, for some t'. Depending on whether $a \notin \mathit{supp}(X)$, we may add a side-condition 'where x is not free in t'.

LEMMA 3.2.5. Support and the permutation action are characterised on terms r as follows:

$$\begin{array}{rclrcl}
\mathit{supp}(a) &=& \{a\} & \mathit{supp}(\mathsf{f}(r)) &=& \mathit{supp}(r) \\
\mathit{supp}(C) &=& \mathit{supp}(C) & \mathit{supp}((r_1, \dots, r_n)) &=& \bigcup_{1\leq i\leq n} \mathit{supp}(r_i) \\
\mathit{supp}(X) &=& \mathit{supp}(X) & \mathit{supp}([a]r) &=& \mathit{supp}(r)\setminus\{a\}
\end{array}$$

$$\begin{array}{rclrcl}
\pi\cdot a &=& \pi(a) & \pi\cdot\mathsf{f}(r) &=& \mathsf{f}(\pi\cdot r) \\
\pi\cdot C &=& \pi\cdot C & \pi\cdot(r_1, \dots, r_n) &=& (\pi\cdot r_1, \dots, \pi\cdot r_n) \\
\pi\cdot X &=& \pi\cdot X & \pi\cdot[a]r &=& [\pi(a)]\pi\cdot r
\end{array}$$

Proof. By facts of the permutation action and Lemma 2.4.11. ■

REMARK 3.2.6. Lemma 3.2.5 is important because it verifies that 'support of r' coincides with the usual definition of 'free variables (atoms) of r'. This is false of nominal terms; for instance the support of the structure $[a]\mathtt{X}$ as constructed in [UPG04] is $\{a\}$, and that of $(a\ b)\cdot\mathtt{X}$ is $\{a, b\}$.

What makes Lemma 3.2.5 work is the very specific way in which we constructed our permissive-nominal terms syntax, so that it coincides with the nominal abstract syntax of [GP01]. In this sense, what Lemma 3.2.5 expresses is a unification (no pun intended) of the mathematics of [GP01] and [UPG04].

In Lemma 3.2.5 the clauses for C and X are uninformative, of course. This is because support and the permutation action are determined by the choice of $\mathcal{C}$ and $\mathcal{X}$. If we assume further internal structure of $C \in \mathcal{C}$ or $X \in \mathcal{X}$ then we can be more specific: for instance in the case of part 1 of Example 3.1.7, $\mathit{fa}(\pi{\cdot}\mathtt{X}^S) = \{\pi(a) \mid a \in S\}$.

Because of Lemma 3.2.5, we are entitled to use the following notation:

NOTATION 3.2.7. In the case of syntax r, we may write $\mathit{fa}(r)$ for $\mathit{supp}(r)$ and call this the **free atoms** of r.

LEMMA 3.2.8. $\mathit{fa}(\pi{\cdot}r) = \pi{\cdot}\mathit{fa}(r)$.

Proof. By a routine induction on r. ■

LEMMA 3.2.9. If $\pi(a) = \pi'(a)$ for all $a \in \mathit{fa}(r)$ then $\pi{\cdot}r = \pi'{\cdot}r$. The reverse implication also holds, provided that all constant symbols in r are strongly supported.

Proof. The first part is immediate from Notation 3.2.7 and the definition of support in Definition 2.2.2.

The reverse implication is by a nominal abstract syntax induction on r. For the case of $r = [a]r'$ we α-convert a to be fresh so that $a \notin \mathit{nontriv}(\pi) \cup \mathit{nontriv}(\pi')$; by assumption 3 in Definition 2.1.6 we can do this. We then use part 2 of Lemma 2.4.9. The case of $r = X \in \mathcal{X}$ uses the assumption of strong support in Definition 3.1.5.[7] ■

3.3 *Free unknowns of a term*

REMARK 3.3.1. Defining a notion of 'the free unknowns of r' is not entirely evident.

Consider for example $[a]X$ where $a \in \mathit{supp}(X)$. If 'X appears in $[a]X$' is true then so is '$(b\ a){\cdot}X$ appears in $[a]X$' for any $b \notin \mathit{supp}(X)$, since $[a]X = [b](b\ a){\cdot}X$. We deal with this in Definition 3.3.2 using *permutation orbits* from Definition 2.4.6; we simply quotient out all permutations. We take a more refined look at this later in Remark 3.7.1.

DEFINITION 3.3.2. Define **(free) unknowns** $\mathit{fU}(r)$ by:

$$\begin{array}{rlrl}
\mathit{fU}(a) &= \varnothing & \mathit{fU}(\mathsf{f}(r)) &= \mathit{fU}(r) \\
\mathit{fU}(C) &= \varnothing & \mathit{fU}((r_1,\dots,r_n)) &= \bigcup_i \mathit{fU}(r_i) \\
\mathit{fU}(X) &= \{\mathit{orb}(X)\} & \mathit{fU}([a]r) &= \mathit{fU}(r)
\end{array}$$

By abuse of notation we write $X \in \mathit{fU}(r)$ for $\mathit{orb}(X) \in \mathit{fU}(r)$ and $X \notin \mathit{fU}(r)$ for $\mathit{orb}(X) \notin \mathit{fU}(r)$, and so forth.

LEMMA 3.3.3. $\mathit{fU}(r)$ is well-defined.

Proof. Using Lemmas 2.4.7 and 2.4.10. ■

NOTATION 3.3.4. Call a term r **ground** when $\mathit{fU}(r) = \varnothing$. Otherwise, call r **open**.

[7]Details of how induction on nominal abstract syntax allows us to α-convert and make freshness assumptions, are the topic of [Gab11b]. A less fancy proof of both implications by a standard induction—so not this new-fangled nominal nonsense—on terms not quotiented by α-equivalence, is in Appendix A of [DGM10], proof of Lemma 4.15 on page 50. We leave it to the reader to judge which is the nicer proof.

3.4 Substitutions

REMARK 3.4.1. Substitutions are of course how unknowns 'stand for' terms. Somewhat later we will develop a denotational theory for nominal terms, and so valuations for unknowns will appear, in Definition 7.3.3. Between now and then, substitutions are king.

The permissive-nominal framework we work with allows us an elegant definition:

DEFINITION 3.4.2. Suppose Σ is a signature. A **substitution** θ in Σ is an equivariant function from $\mathcal{X}$ to terms in Σ such that $sort(\theta(X)) = sort(X)$ always.
θ will range over substitutions.
Write id for the **identity** substitution mapping X to X always. It will always be clear whether id means the identity substitution or permutation.

The reader familiar with nominal terms will expect a 'freshness' condition on substitutions corresponding to '$\nabla' \vdash \nabla\theta$', as in for example Equation (11) or Lemma 2.14 of [UPG04], or '$fa(\theta(X)) \subseteq supp(X)$' as in Definition 3.1 of [DGM10]. This follows immediately from equivariance:

PROPOSITION 3.4.3. If θ is a substitution then $\forall X{\in}\mathcal{X}.fa(\theta(X)) \subseteq supp(X)$.

Proof. Direct from Lemma 2.3.3. ■

Putting Propositions 3.4.3 and 2.5.5 together with a concrete $\mathcal{X}$ recovers the notion of substitution used in [DGM10]:

LEMMA 3.4.4. If $\mathcal{X}$ is equal to example 1 of Example 3.1.7 then the construction in Definition 2.5.4 describes a 1-1 correspondence between substitutions and maps from unknowns $\mathtt{X}^S_\alpha$ to terms $t : \alpha$ such that $fa(t) \subseteq S$.

DEFINITION 3.4.5. Suppose $fa(t) \subseteq supp(X)$ and $sort(t) = sort(X)$. Write $[X{:=}t]$ for the **atomic substitution** equivariantly extending the assignment $X \mapsto t$, so that

$$\begin{array}{lll} [X{:=}t](\pi\cdot X) & = \pi\cdot t & \text{and} \\ [X{:=}t](Y) & = Y & \text{for all other } Y. \end{array}$$

By Proposition 2.5.5 we have:

LEMMA 3.4.6. Definition 3.4.5 is well-defined. That is, if $\pi\cdot X = \pi'\cdot X$ then $\pi\cdot t = \pi'\cdot t$.

REMARK 3.4.7. The 'moderated unknown' $\pi\cdot X$ in Definition 3.4.5 is an artefact of our writing $[X{:=}t]$ instead of a mathematically equal $[\pi\cdot X{:=}\pi\cdot t]$ for some other π.

Since θ is equivariant its behaviour on $\pi\cdot X$ is already determined by its behaviour on X and so we could unambiguously specify $[X{:=}t]$ succinctly as $[X{:=}t](X) = t$ and $[X{:=}t](Y) = Y$.

DEFINITION 3.4.8. Define a **substitution action** on terms by:

$$\boxed{\begin{array}{ll} a\theta = a & \mathsf{f}(r)\theta = \mathsf{f}(r\theta) \\ C\theta = C & (r_1,\dots,r_n)\theta = (r_1\theta,\dots,r_n\theta) \\ X\theta = \theta(X) & ([a]r)\theta = [a](r\theta) \end{array}}$$

Note that $X\theta$ refers to θ acting on X as a term whereas $\theta(X)$ refers the value of the function θ at X. The substitution action is well-defined by Lemmas 2.4.10 and 2.4.11.

REMARK 3.4.9. Famously, the nominal terms substitution is capturing [UPG04, Definition 2.13]. We spell out how this works in our permissive-nominal context: Suppose $supp(X)$ is equal to a permission set S and $a \in S$ and $b \notin S$ (where we assume appropriate sorts). Then:

- $([a]X)[X{:=}a] = [a]a$. The a in the substitution $[X{:=}a]$ has been captured by the $[a]X$.
- $([b]X)[X{:=}a] = [b]a$.
- It is impossible to even ask what $([b]X)[X{:=}b]$ is equal to because $[X{:=}b]$ is not even a substitution, since $b \notin S$. So $b \notin S$ cannot be captured by a substitution $[X{:=}b]$, because that substitution does not exist. This is no *ad hoc* restriction: by Proposition 3.4.3 it *cannot* exist.
- Also, $[b](b\ a){\cdot}X = [a]X$. By construction in Definition 3.4.5

  $$([b](b\ a){\cdot}X)[X{:=}a] = [b](b\ a){\cdot}a = [b]b = [a]a.$$

  Also $[X{:=}a] = [(b\ a){\cdot}X{:=}b]$ and $([b](b\ a){\cdot}X)[(b\ a){\cdot}X{:=}b] = [b]b$.
  That is, the choice of representative of $[a]X$ and $[X{:=}a]$ does not matter for capture to occur.

It is interesting to note that in our setting, $[X{:=}a]$ is *equivariant* and that $a \notin supp([a]X)$. If a is fresh for both $[X{:=}a]$ and $[a]X$, how can it be captured?

What allows a to get captured is the *strong support property* of X. Because X is strongly supported, we can think of it as 'containing' a list of its supporting atoms in some order, so that the a in $[X{:=}a]$ is bound by $supp(X)$ but in being bound it points to a 'position' in X.

Viewed from this interesting perspective, the nominal substitution action is not capturing at all: it is simply a compact way to present an 'infinite raising' (terminology from higher-order logic), or a de Bruijn index.

LEMMA 3.4.10. $\pi{\cdot}(r\theta) = (\pi{\cdot}r)\theta$.

Proof. By a routine induction on r using equivariance. ■

LEMMA 3.4.11. $fa(r\theta) \subseteq fa(r)$.

Proof. From Lemmas 2.3.3 and 3.4.10. ■

LEMMA 3.4.12. $r\theta = r\theta'$ if and only if $\forall X{\in}fU(r).\theta(X) = \theta'(X)$.

Proof. By a routine induction on r. We consider two cases:

- *The case* $[a]r$. Suppose $\theta(X) = \theta'(X)$ for every $X \in fU([a]r)$. $fU([a]r) = fU(r)$ so by inductive hypothesis $r\theta = r\theta'$. The result follows from the definitions.
  The reverse implication is similar.
- *The case* X. Suppose $\theta(\pi \cdot X) = \theta'(\pi \cdot X)$ for all π. Then taking $\pi = id$ we have $X\theta = \theta(X) = \theta'(X) = X\theta'$.
  Conversely if $X\theta = X\theta'$ then using equivariance (Definition 3.4.2) $\theta(\pi \cdot X) = \theta'(\pi \cdot X)$ for all π.

■

REMARK 3.4.13. Recall from Definition 3.3.2 that we write $X \in fU(r)$ for $orb(X) \in fU(r)$. It might seem that the condition $\forall X{\in}fU(r).\theta(X) = \theta'(X)$ in Lemma 3.4.12 would require checking $\theta(X) = \theta'(X)$ for infinitely many X provided that $fU(r) \neq \varnothing$. In fact, this is not the case: by equivariance of θ, we only need to check equality for one representative X of each permutation orbit: $X \in orb(X) \in fU(r)$.

3.5 *Composition and invertibility of substitutions*

DEFINITION 3.5.1. Define **composition** of substitutions $\theta_1 \circ \theta_2$ by

$$(\theta_1 \circ \theta_2)(X) = (\theta_1(X))\theta_2.$$

LEMMA 3.5.2. $(r\theta)\theta' = r(\theta \circ \theta')$.

Proof. By induction on r. ■

DEFINITION 3.5.3. Call θ **invertible** when there exists θ^{-1} such that $\theta \circ \theta^{-1} = \theta^{-1} \circ \theta = id$.

LEMMA 3.5.4. θ is invertible if and only if θ is a bijection on $\mathcal{X}$ the set of all unknowns. Furthermore, if θ is invertible then $supp(\theta(X)) = supp(X)$ always.

Proof. Substitution cannot make syntax smaller, or (by Lemma 3.4.11) make free atoms larger. ■

So an invertible θ must biject unknowns of a particular sort and permission set with other unknowns of that same sort and permission set. So, like atoms, we can rename unknowns to 'be fresh' (provided we have given ourselves enough of them). Invertible substitutions will be useful later, and they are also one manifestation of a more general framework of *two-level nominal sets* [Gab11c].

3.6 shift-*permutations*

The reader may be familiar with nominal freshness conditions $a\#\mathtt{X}$ from [UPG04]. In that paper, $a\#\mathtt{X}$ indicated that $\mathtt{X}$ should be substituted only for terms for which a is fresh.

In [UPG04; FG07], we might have to extend a freshness context in order to give ourselves more fresh atoms. This is what rules like (**Fr**) from [GM08c, Figure 2] or (**fr**) from [GM09a, Figure 2] do; see also [FG10] where the issue of extending nominal freshness contexts is made very explicit.

In principle, permission sets guarantee an infinite supply of fresh atoms, so the problem of extending a freshness context should not arise. But this may rely on oracular knowledge of what the permission set should be, which we might prefer not to assume. The choice of nominal permutation group $\mathbb{P}$ gives us the power to implicitly parameterise over this decision.

Suppose we have some X such that $a \in \mathit{supp}(X)$ and we perhaps we are solving a unification problem and the information that a should be fresh for X has just been revealed by an algorithm; so we want to remove a from the permission set of X. This arises in the unification algorithm of Section 4.

Suppose alternatively we would like to make the permission set *larger*, e.g. if we know $\forall X.\phi$ and want to deduce $\phi[X{:=}t]$ where $\mathit{fa}(t) \not\subseteq \mathit{supp}(X)$, or we have a rewrite rule $X \to X$ and want to deduce $t \to t$ where again $\mathit{fa}(t) \not\subseteq \mathit{supp}(X)$. This arises in the nominal rewriting, algebra and permissive-nominal logic which we construct later.

This is where *shift*-permutations can help.

DEFINITION 3.6.1. Call a permutation $\delta \in \mathbb{P}$ a ***shift*-permutation** when there exists a permission set S and atom $a \in S$ such that $S \setminus \{a\} = \delta{\cdot}S$.
Say that a nominal permutation group $\mathbb{P}$ has ***shift*-permutations** when for every permission set S and atom $a \in \mathbb{A}$ there exists a permutation $\pi \in \mathbb{P}$ such that $\pi{\cdot}S = S\setminus\{a\}$.

REMARK 3.6.2. Another way to read Definition 3.6.1 is that $\mathbb{P}$ has *shift*-permutations when, if S is a permission set and A is finite, then $S \setminus A$ and $S \cup A$ are permission sets. Stronger versions allowing infinite A are certainly imaginable.

EXAMPLE 3.6.3. The nominal permutation group in part 2 of Example 2.1.7 has *shift*-permutations.

δ_i bijects $\mathbb{A}_i^{<}$ with $\mathbb{A}_i^{<} \setminus \{f(0)\}$. Using swappings we can now generate a π to biject any permission set S with $S\setminus\{a\}$ for $a \in S$. We give the concrete constructions below, culminating with Lemma 3.6.10.

For the rest of this subsection we work concretely with the nominal permutation group from part 2 of Example 2.1.7; the reader only interested in the high-level picture can skip this. Recall the bijections f_i from integers to atoms from part 2 of Example 2.1.7. For simplicity drop the subscript i and consider just one set of atoms.

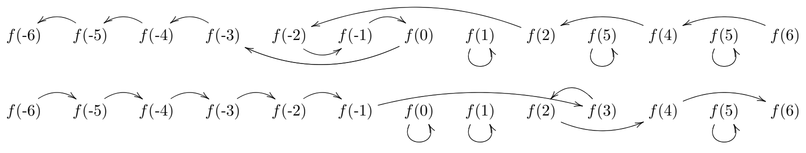
NOTATION 3.6.4. By abuse of notation write 0 for the atom $f(0)$.

DEFINITION 3.6.5.

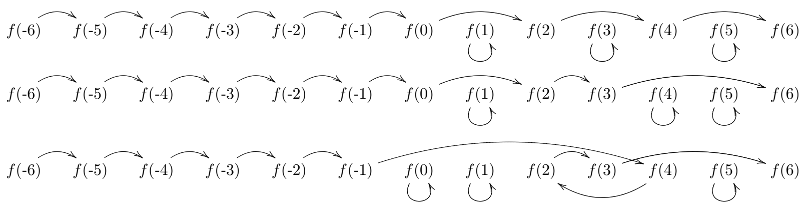
1. If $a \in \mathbb{A}^{<}$ then define $\delta^{\text{-}a}$ by:
$$\delta^{\text{-}a} = (a\,0) \circ \delta \circ (a\,0)$$

2. If $b \in \mathbb{A}^{>}$ then for some fixed but arbitrary choice of $c \in \mathbb{A}^{>}$ such that $\delta(c) = c$ (and so also $c \notin \mathbb{A}^{<}$), define δ^{+b} by:
$$\delta^{+b} = (b\,0) \circ (c\,b) \circ \delta^{-1} \circ (c\,b) \circ (b\,0)$$

EXAMPLE 3.6.6. We illustrate δ^{-a} and δ^{+b} where $a = f(\text{-}2)$ and $b = f(3)$ and where we take $c = b$:


f(-6) f(-5) f(-4) f(-3) f(-2) f(-1) f(0) f(1) f(2) f(5) f(4) f(5) f(6)
f(-6) f(-5) f(-4) f(-3) f(-2) f(-1) f(0) f(1) f(2) f(3) f(4) f(5) f(6)

We also consider the slightly more complex example of δ^{+d} where $d = f(4)$, and again we take $c = f(3)$. We do this in three steps, where we illustrate δ^{-1}, then $(c\ d) \circ \delta^{-1} \circ (c\ d)$, and finally δ^{+d}:


f(-6) f(-5) f(-4) f(-3) f(-2) f(-1) f(0) f(1) f(2) f(3) f(4) f(5) f(6)
f(-6) f(-5) f(-4) f(-3) f(-2) f(-1) f(0) f(1) f(2) f(3) f(4) f(5) f(6)
f(-6) f(-5) f(-4) f(-3) f(-2) f(-1) f(0) f(1) f(2) f(3) f(4) f(5) f(6)

LEMMA 3.6.7.

1. If $a \in \mathbb{A}^{<}$ then δ^{-a} bijects $\mathbb{A}^{<}$ with $\mathbb{A}^{<} \setminus \{a\}$.
2. If $b \in \mathbb{A}^{>}$ then δ^{+b} bijects $\mathbb{A}^{<}$ with $\mathbb{A}^{<} \cup \{b\}$.

Proof. For the first part, suppose $a \in \mathbb{A}^{<}$. Then $\mathbb{A}^{<} = (a\,0)\cdot\mathbb{A}^{<}$. We reason as follows:

$$\begin{aligned} \delta^{-a}\cdot((a\,0)\cdot\mathbb{A}^{<}) &= ((a\,0) \circ \delta \circ (a\,0) \circ (a\,0))\cdot\mathbb{A}^{<} \\ &= ((a\,0) \circ \delta)\cdot\mathbb{A}^{<} = (a\,0)\cdot(\mathbb{A}^{<}\setminus\{0\}) = \mathbb{A}^{<}\setminus\{a\} \end{aligned}$$

Now suppose $b \in \mathbb{A}^{>}$. It is easier to work with $(\delta^{+b})^{-1}$, to keep the parallel with the previous case. So $\mathbb{A}^{<} \cup \{b\} = ((b\,0)\cdot\mathbb{A}^{<}) \cup \{0\}$. We reason as follows:

$$\begin{aligned} (\delta^{+b})^{-1}\cdot(((b\,0)\cdot\mathbb{A}^{<})\cup\{0\}) &= ((b\,0) \circ (c\,b) \circ \delta \circ (c\,b) \circ (b\,0))\cdot(((b\,0)\cdot\mathbb{A}^{<}) \cup \{0\}) \\ &\qquad \text{Def. 3.6.5} \\ &= \big(((b\,0) \circ (c\,b) \circ \delta \circ (c\,b) \circ (b\,0))\cdot((b\,0)\cdot\mathbb{A}^{<})\big) \cup \{0\} \\ &\qquad \delta(c){=}c \\ &= \big(((b\,0) \circ (c\,b) \circ \delta \circ (c\,b))\cdot\mathbb{A}^{<}\big) \cup \{0\} \\ &\qquad \text{Fact} \\ &= \big(((b\,0) \circ (c\,b) \circ \delta)\cdot\mathbb{A}^{<}\big) \cup \{0\} \\ &\qquad b, c \notin \mathbb{A}^{<} \\ &= \big(((b\,0) \circ (c\,b))\cdot(\mathbb{A}^{<}\setminus\{0\})\big) \cup \{0\} \\ &\qquad \delta\cdot\mathbb{A}^{<}{=}\mathbb{A}^{<}\setminus\{0\} \\ &= (\mathbb{A}^{<}\setminus\{0\}) \cup \{0\} \\ &\qquad b, c \notin \mathbb{A}^{<}\setminus\{0\} \\ &= \mathbb{A}^{<} \end{aligned}$$

■

Recall from Definition 2.1.10 that each permission set S has the form $\pi{\cdot}\mathbb{A}^<$ for some permutation π.

DEFINITION 3.6.8. For each S make some choice of permutation π_S such that $S = \pi_S^{\text{-}1}{\cdot}\mathbb{A}^<$.[8]

DEFINITION 3.6.9. Suppose S is a permission set and $a \in S$ and $b \notin S$. Then we define:

$$\boxed{\delta_{S\text{-}a} = \pi_S^{\text{-}1} \circ \delta^{\text{-}\pi_S(a)} \circ \pi_S \qquad \delta_{S\text{+}b} = \pi_S^{\text{-}1} \circ \delta^{\text{+}\pi_S(b)} \circ \pi_S}$$

The concrete details of the construction are only interesting insofar as they give us Lemma 3.6.10. Other permutations are possible, but we only need that one exists.

LEMMA 3.6.10.

1. $\delta_{S\text{-}a}$ bijects S with $S\setminus\{a\}$.
2. $\delta_{S\text{+}b}$ bijects S with $S\cup\{b\}$.

Proof. From Lemma 3.6.7. ■

DEFINITION 3.6.11. Suppose S is a permission set and $supp(X) = S$. Suppose D is a finite list of atoms $d_1, \ldots, d_n$ and E is a finite list of atoms $e_1, \ldots, e_n$. Suppose $\{d_1, \ldots, d_n\} \subseteq S$ and $\{e_1, \ldots, e_n\} \cap S = \varnothing$. Then define $\delta_{S\text{-}D}$ and $\delta_{S\text{+}E}$, and $X\text{-}D$ and $X\text{+}E$ by:

$$\begin{aligned}
\delta_{S\text{-}[]} &= id & \delta_{S\text{+}[]} &= id \\
\delta_{S\text{-}[d]} &= \delta_{S\text{-}d} & \delta_{S\text{+}[e]} &= \delta_{S\text{+}e} \\
\delta_{S\text{-}d,D} &= \delta_{(S\setminus\{d\})\text{-}D} \circ \delta_{S\text{-}d} & \delta_{S\text{+}e,E} &= \delta_{(S\cup\{e\})\text{+}E} \circ \delta_{S\text{+}e} \\
X\text{-}D &= \delta_{supp(X)\text{-}D}{\cdot}X & X\text{+}E &= \delta_{supp(X)\text{+}E}{\cdot}X
\end{aligned}$$

LEMMA 3.6.12. Suppose S is a permission set. Suppose D and E are finite lists of atoms $d_1, \ldots, d_n$ and $e_1, \ldots, e_n$. Suppose $\{d_1, \ldots, d_n\} \subseteq S$ and $\{e_1, \ldots, e_n\} \cap S = \varnothing$.

Then $\delta_{S\text{-}D}$ bijects S with $S\setminus\{d_1, \ldots, d_n\}$ and $\delta_{S\text{+}E}$ bijects S with $S\cup\{e_1, \ldots, e_n\}$.

Proof. Using Lemma 3.6.10. ■

COROLLARY 3.6.13. S is a permission set if and only if $S = (\mathbb{A}^< \setminus A) \cup B$ for some finite $A \subseteq \mathbb{A}^<$ and $B \subseteq \mathbb{A}^>$.

Proof. If S is a permission set then by Definition 2.1.10 $S = \pi{\cdot}\mathbb{A}^<$ for some π and the result follows by a routine induction on the generators of π (swapping and δ; see part 2 of Example 2.1.7).

Conversely consider $S = (\mathbb{A}^< \setminus A) \cup B$. Let D be the atoms in A in some order, and E be the atoms in B in some order. Then we apply $\delta_{S\text{-}D}$ and then $\delta_{(S\setminus A)\text{+}E}$ and use Lemma 3.6.12. ■

[8]Taking the inverse here saves writing $^{\text{-}1}$ quite so many times in Definition 3.6.9, and is harmless since permutations are invertible.

REMARK 3.6.14. The reader may be familiar with the de Bruijn *shift* function $\uparrow$ [ACCL91, Subsection 2.2]. This maps $\mathbb{N}$ to $\mathbb{N}\setminus\{0\}$ by mapping $j \in \mathbb{N}$ to $j+1 \in \mathbb{N}$, and in doing so it 'creates a fresh number' 0. The reader familiar with presheaf techniques may know of a functor δ and arrow up, which work the same way, as exemplified in [FPT99, Section 1].

δ_i from part 2 of Example 2.1.7 is in the same spirit. It shifts 'down' instead of 'up', but δ_i^{-1} shifts 'up'.

Note that δ is *invertible* ($\uparrow$ and up are not). This is consistent with the general preference of nominal techniques for using permutations where possible.

3.7 Occurrences

REMARK 3.7.1. As discussed in Remark 3.3.1 we have to be careful if we wish to say 'X appears in r'; this might not quite mean what we think it does.

For example if 'X appears in $[a]X$' where $a \in supp(X)$ then also '$(b\ a)\cdot X$ appears in $[a]X$' for any $b \notin supp(X)$. We dealt with this in Definition 3.3.2 by quotienting out all permutations.

But this is a little drastic. For instance, '$(b\ a)\cdot X$ appears in $[a]X$' is not true for $b \in supp(X)$; it is not the case that if 'X appears in r' then '$\pi\cdot X$ appears in r' for any π.

We did not need to quotient out *all* permutations—only some of them—and so returning $orb(X)$ in Definition 3.3.2 throws out more information than necessary.

Definitions 3.7.2 and 3.7.3 develop a more refined notion of occurrence, based on an intuition of 'X appears in r under a list of abstractions D'. This will be useful later.

DEFINITION 3.7.2. D will range over finite lists of distinct atoms. A **(level 2) occurrence** is a term of the form $[D]X$ where $[]X$ is X and $[a, D]X$ is $[a][D]X$.

DEFINITION 3.7.3. Define the **occurrences in** r inductively by:

$$\boxed{\begin{array}{rlrl} occ(a) &= \varnothing & occ(\mathsf{f}(r)) &= occ(r) \\ occ(C) &= \varnothing & occ((r_1,\ldots,r_n)) &= \bigcup occ(r_i) \\ occ(X) &= X & occ([a]r) &= \{[a]x \mid x{\in}occ(r)\} \end{array}}$$

EXAMPLE 3.7.4.

- X occurs in X.
- $[a]X$ occurs in $[a]X$ and also in $[a](X, Y)$; so does $[a]Y$. X does not occur in $[a]X$ or $[a](X, Y)$.
- $[a][b]X$ and $[a][a]X$ occur in $[a]([b]X, [a]X)$.

We write occurrences as $[D]X$ for D a finite list of distinct atoms. Note that $[a][a]X$ is an occurrence since it is equal to $[a][b](b\ a)\cdot X$ where $b \notin supp(X)$. This is an equality, not an equivalence imposed on terms after they are constructed, because of our use of atoms-abstraction (Definition 2.4.8) in syntax (Definition 3.2.1).

4 UNIFICATION

We want to write rewrite rules and equality axioms using nominal terms. In order to do this, we have to unify nominal terms (answer the question: "given r and s what substitutions θ make them equal?"). Unification makes unknowns 'come alive' and represent unknown terms.

Therefore, we now create a nominal unification algorithm. One notable property of nominal unification is that it has most general (principal) unifiers Theorem 4.4.6. Contrast this with higher-order unification, which does not [Dow01, Section 4]. This is one reason we say that the nominal approach to names and binding has a 'first-order' flavour.

The algorithm we use follows the spirit of [UPG04] but the design is different. In [UPG04] a solution to $[a]\mathtt{X} \overset{?}{=} [b]\mathtt{Y}$ would be $(b\#\mathtt{X}, [\mathtt{Y}{:=}(b\ a)\cdot\mathtt{X}])$; that is, the unification algorithm returns a pair of some freshness side-conditions and some equalities.[9]

Here, solutions are equalities only, without freshness conditions. The extra power resides in the notion of an *shift*-permutation (Definition 3.6.1).

A solution to $[a]X = [b]Y$ where $b \in \mathit{supp}(X) = \mathit{supp}(Y)$ would be

$$[X{:=}\delta'\cdot X,\ Y{:=}((b\ a)\circ\delta')\cdot X]$$

where δ' bijects $\mathit{supp}(X)$ with $\mathit{supp}(X) \setminus \{b\}$ (and by this bijection 'internally freshens' X with respect to b).

In another design [DGM10, Section 5] we use permission sets and fresh unknowns; a solution to $[a]X = [b]Y$ where $b \in \mathit{supp}(X) = \mathit{supp}(Y)$ is $[X{:=}Z,\ Y{:=}(b\ a)\cdot Z]$ where $\mathit{supp}(Z) = \mathit{supp}(X)\setminus\{b\}$. Generating Z fresh requires us to solve problems in a context of 'known unknowns' $\mathcal{V}$. This introduces a notion of state and sequentiality into the algorithm of [DGM10] which we avoid here.

Nothing forces us to feed the unification algorithm syntax with *shift*-permutations, even if the solutions it returns might mention them; similarly in [UPG04] we may obtain a solution with freshness side-conditions to a unification problem with only equalities. So use of *shift*-permutation in Definition 4.0.5 should not be read as a commitment to using them everywhere (though we do note empirically that *shift* seems to be useful elsewhere too).

The main definition of this section is Definition 4.1.7. The main result is Theorem 4.4.6.

DEFINITION 4.0.5. Throughout this Section we fix some signature Σ and we work with syntax over Σ. We assume a nominal permutation group $\mathbb{P}$ with *shift*-permutations and a set of unknowns $\mathcal{X}$ such that every unknown is supported by a permission set (see e.g. part 2 of Example 3.1.7).

[9]We write typewriter font to avoid confusion between the symbols used in [UPG04] (which have no support) and the elements $X \in \mathcal{X}$ used in this paper (which do have support). To see how to travel between these two worlds see part 2 of Example 3.1.7, or [DGM10].

4.1 The unification algorithm

DEFINITION 4.1.1. A **(unification) equality** is a unordered pair $r \stackrel{?}{=} s$ (so $r \stackrel{?}{=} s$ is identical to $s \stackrel{?}{=} r$) such that:

1. $sort(r) = sort(s)$.
2. If $[D]X$ and $[D']\pi{\cdot}X$ are both in $occ(r) \cup occ(s)$ then π is finite.
   So we exclude an equality like $X \stackrel{?}{=} \delta{\cdot}X$, where δ is a shift permutation and $nontriv(\delta) \cap supp(X)$ is not finite.

A **(unification) freshness** is an ordered pair $a\#_?r$.

Let ef range over equalities or freshnesses and define $ef\theta$ by:

- $(r \stackrel{?}{=} s)\theta = (r\theta \stackrel{?}{=} s\theta)$.
- $(a\#_?r)\theta = (a\#_?(r\theta))$.

A **nominal unification problem** Pr is a finite list $ef_1, \ldots, ef_n$.

We (ab)use standard sets notation and write $ef \in Pr$ as shorthand for 'ef appears in the list Pr'.

REMARK 4.1.2. Condition 2 in Definition 4.1.1 protects ($\stackrel{?}{=}$**X**) in Figure 2 from an 'infinite freshness explosion', if $nontriv(\pi) \cap supp(X)$ is not finite. This condition exists implicitly in [UPG04], in the sense that all permutations there are finite. However, condition 2 is not only computationally motivated. Given constants C and D with $supp(C) = \varnothing = supp(D)$, $X \stackrel{?}{=} \delta{\cdot}X$ may have solutions C and D but have no principal solution. We discuss the implications of this condition to nominal rewriting, at the end of Section 6.

DEFINITION 4.1.3. If $Pr = ef_1, \ldots, ef_n$ is a problem then define $Pr\theta$ by:

$$Pr\theta = ef_1\theta, \ldots, ef_n\theta$$

Say θ **solves** Pr and call θ a **solution** to Pr when

$$\boxed{\begin{array}{lll} r\theta = s\theta & \text{for every} & r \stackrel{?}{=} s \in Pr, \quad \text{and} \\ a \notin fa(r\theta) & \text{for every} & a\#_?r \in Pr. \end{array}}$$

Write $Sol(Pr)$ for the set of solutions to Pr and call Pr **solvable** when $Sol(Pr)$ is non-empty.

Recall the definition of $\theta \circ \theta'$ from Definition 3.5.1.

LEMMA 4.1.4. $\theta \circ \theta' \in Sol(Pr)$ if and only if $\theta' \in Sol(Pr\theta)$.

Proof. By unpacking Definition 4.1.3 and using Lemma 3.5.2. ■

DEFINITION 4.1.5. Define a **simplification** rewrite relation $Pr \Longrightarrow Pr'$ on unification problems by the rules in Figure 2.

We call rules (**IF**) and (**IE**) **instantiating rules**. We call all the other rules **non-instantiating rules**.

$$\begin{array}{llcl}
(\overset{?}{=}\mathbf{a}) & a \overset{?}{=} a,\ Pr & \Longrightarrow & Pr \\
(\overset{?}{=}\mathbf{C}) & C \overset{?}{=} C,\ Pr & \Longrightarrow & Pr \\
(\overset{?}{=}\mathsf{f}) & \mathsf{f}(r) \overset{?}{=} \mathsf{f}(s),\ Pr & \Longrightarrow & r \overset{?}{=} s,\ Pr \\
(\overset{?}{=}()) & (r_1,\dots,r_n) \overset{?}{=} (s_1,\dots,s_n),\ Pr & \Longrightarrow & r_1 \overset{?}{=} s_1,\dots,r_n \overset{?}{=} s_n, Pr \\
(\overset{?}{=}[]) & [a]r \overset{?}{=} [a]s,\ Pr & \Longrightarrow & r \overset{?}{=} s,\ Pr \\
(\overset{?}{=}\mathbf{X}) & X \overset{?}{=} \pi{\cdot}X,\ Pr & \Longrightarrow & a_1\#_?X,\dots,a_n\#_?X, Pr \\
 & & \multicolumn{2}{r}{(\{a_1,\dots,a_n\} = \mathit{nontriv}(\pi)\cap \mathit{supp}(X))} \\
(\mathbf{F}) & r \overset{?}{=} X,\ Pr & \Longrightarrow & a\#_?r,\ r\overset{?}{=}X,\ Pr \\
 & & & \qquad (a\in \mathit{fa}(r)\backslash \mathit{supp}(X)) \\
(\mathbf{F}\#) & a\#_?r,\ Pr & \Longrightarrow & Pr \qquad\qquad (a\notin \mathit{fa}(r)) \\
(\mathbf{F}\mathsf{f}) & a\#_?\mathsf{f}(r),\ Pr & \Longrightarrow & a\#_?r, Pr \\
(\mathbf{F}()) & a\#_?(r_1,\dots,r_n),\ Pr & \Longrightarrow & a\#_?r_1,\dots,a\#_?r_n, Pr \\
(\mathbf{F}[]) & a\#_?[b]r, Pr & \Longrightarrow & a\#_?r,\ Pr \\
(\mathbf{IE}) & r \overset{?}{=} X,\ Pr & \overset{[X{:=}r]}{\Longrightarrow} & Pr[X{:=}r] \\
 & & & (X\notin \mathit{fU}(r), \mathit{fa}(r)\subseteq \mathit{supp}(X)) \\
(\mathbf{IF}) & a\#_?X, Pr & \overset{[X{:=}\delta_{X\text{-}a}\cdot X]}{\Longrightarrow} & Pr[X{:=}\delta_{X\text{-}a}\cdot X]
\end{array}$$

Figure 2: Simplification rules for problems

In $(\mathbf{IF})$ $\delta_{X\text{-}a}$ is some permutation bijecting $\mathit{supp}(X)$ with $\mathit{supp}(X)\setminus\{a\}$. We can do this because we assumed *shift*-permutations in Definition 4.0.5.[10]

Write $\Longrightarrow^*$ for the transitive and reflexive closure of $\Longrightarrow$.

REMARK 4.1.6. Compare Figure 2 with Figure 3 of [UPG04]. Note of $(\overset{?}{=}[])$ that we do not consider the case $[a]r \overset{?}{=} [b]s$. This is because α-equivalence is handled automatically by nominal abstract syntax, specifically by Definition 2.4.8. So α-renaming is pushed into the background (just as is usually the case for first-order syntax) and these rules are somewhat higher-level than those of [UPG04].

We also do not require a rule $a\#_?[a]r,\ Pr \Longrightarrow Pr$ because the abstracted atom in $[a]r$ is α-convertible; more formally, $[a]r = [b](b\ a){\cdot}r$ for some/any fresh b (so $b\notin \mathit{fa}(r)$).

Finally, in $(\overset{?}{=}\mathbf{X})$ we do not need to write $\pi{\cdot}X \overset{?}{=} \pi'{\cdot}X$ (though we could) because unknowns are just a strongly-supported nominal set. We know that $\mathit{nontriv}(\pi)\cap \mathit{supp}(X)$ is finite by a routine argument based on condition 2 of Definition 4.1.1. It is not hard to check that the instantiating rules $(\mathbf{IF})$ and $(\mathbf{IE})$ do indeed preserve these conditions—$(\mathbf{IF})$ involves a *shift* permutation, but in a manner that is applied uniformly to the whole problem.

[10]The specific choice does not matter. Intuitively this is because permutations are invertible so any one choice and be undone and redone at will. A more formal statement of this is Theorem 4.3.6. For an example of a *shift*-permutation concretely constructed, see Definition 3.6.9.

This algorithm generates *shifts* just like in [UPG04] we generated freshness conditions, and for the same reason.

DEFINITION 4.1.7. If Pr is a problem, define a **unification algorithm** by:

1. Rewrite Pr using the rules of Definition 4.1.5 where possible, with top-down precedence (so apply $(\overset{?}{=}\mathbf{a})$ before $(\overset{?}{=}\mathsf{f})$, and so on).
2. If we reduce to $\varnothing$ then we succeed and return θ where θ is the composition of all the substitutions labelling rewrites (we take $\theta = id$ if there are none). Otherwise, we fail.

REMARK 4.1.8. Note in Definition 4.1.7 that we apply each rule to the head of the list Pr. This is to prevent 'unfair' looping, e.g. repeatedly applying $(\mathbf{F})$ to some equality $r \overset{?}{=} X$ wherever it appears in Pr.

Note also that the rule $(\mathbf{F}\#)$ is equivalent—in the presence of the other rules—to three rules as follows:

$$\begin{array}{lll} (\mathbf{Fa}) & a\#_?b,\ Pr \Longrightarrow Pr & \\ (\mathbf{FC}) & a\#_?C,\ Pr \Longrightarrow Pr & (a \notin supp(C)) \\ (\mathbf{FX}) & a\#_?X, Pr \Longrightarrow Pr & (a \notin supp(X)) \end{array}$$

PROPOSITION 4.1.9. The algorithm of Definition 4.1.7 always terminates.

Proof. It is not hard to generate an inductive quantity which is reduced by the reductions in Figure 2. ■

4.2 Examples of the algorithm

We assume the permutation group from part 2 of Example 2.1.7 and we recall the definition of X-D from Definition 3.6.9.

Example one (succeeds).

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

$$\begin{array}{lcl} \mathsf{g}([a]X, [a]a) \overset{?}{=} \mathsf{g}([d]c, [d]d) & \Longrightarrow & (\overset{?}{=}\mathsf{g}), (\overset{?}{=}()) \\ [a]X \overset{?}{=} [d]c\ , [a]a \overset{?}{=} [d]d & \Longrightarrow & (\overset{?}{=}[]), [a]X = [d](d\ a)\cdot X \\ (d\ a)\cdot X \overset{?}{=} c\ , [a]a \overset{?}{=} [d]d & \overset{[X:=c]}{\Longrightarrow} & (\mathbf{IE}) \\ [a]a \overset{?}{=} [d]d & \Longrightarrow & (\overset{?}{=}[]), [a]a = [d]d \\ d \overset{?}{=} d & \Longrightarrow & (\overset{?}{=}\mathbf{a}) \\ \varnothing \quad \text{Success, with } [X{:=}c] & & \end{array}$$

Example two (succeeds).

Suppose $a, c \in \mathbb{A}^<$ and $b, d \notin \mathbb{A}^<$. Take $supp(X) = \mathbb{A}^< \cup \{b, d\}$, $supp(Y) = \mathbb{A}^< \cup \{f\}$, and $supp(Z) = \mathbb{A}^<$. Suppose a term-former f.

We apply the algorithm to $\{\mathsf{f}([a]b, Z, X) \overset{?}{=} \mathsf{f}([d]b, [a]a, Y)\}$:

$$\begin{array}{lcl}
\mathsf{f}([a]b, Z, X) \stackrel{?}{=} \mathsf{f}([d]b, [a]a, Y) & \Longrightarrow & (\stackrel{?}{=}\mathsf{f}), (\stackrel{?}{=}()) \\
{[a]b} \stackrel{?}{=} [d]b\ ,\ Z \stackrel{?}{=} [a]a,\ X \stackrel{?}{=} Y & \Longrightarrow & (\stackrel{?}{=}[]), [a]b = [d]b \\
b \stackrel{?}{=} b\ ,\ Z \stackrel{?}{=} [a]a,\ X \stackrel{?}{=} Y & \Longrightarrow & (\stackrel{?}{=}\mathbf{a}) \\
Z \stackrel{?}{=} [a]a\ ,\ X \stackrel{?}{=} Y & \stackrel{[Z:=[a]a]}{\Longrightarrow} & (\mathbf{IE}) \\
X \stackrel{?}{=} Y & \Longrightarrow & (\mathbf{F}) \\
b\#_? X\ ,\ X \stackrel{?}{=} Y & \stackrel{[X:=X\text{-}b]}{\Longrightarrow} & (\mathbf{IF}) \\
X\text{-}b \stackrel{?}{=} Y & \Longrightarrow & (\mathbf{F}) \\
d\#_? X\text{-}b\ ,\ X\text{-}b \stackrel{?}{=} Y & \stackrel{[X\text{-}b:=X\text{-}b,d]}{\Longrightarrow} & (\mathbf{IF}) \\
X\text{-}b, d \stackrel{?}{=} Y & \Longrightarrow & (\mathbf{F}) \\
f\#_? Y\ ,\ X\text{-}b, d \stackrel{?}{=} Y & \stackrel{[Y:=Y\text{-}f]}{\Longrightarrow} & (\mathbf{IF}) \\
X\text{-}b, d \stackrel{?}{=} Y\text{-}f & \stackrel{[Y\text{-}f:=X\text{-}b,d]}{\Longrightarrow} & (\mathbf{IE}) \\
\varnothing \quad \text{Success, with } [X{:=}X\text{-}b, d,\ Y{:=}X\text{-}b, d,\ Z{:=}[a]a] & &
\end{array}$$

Example three (fails).

Take $supp(X) = \mathbb{A}^<$. We run the algorithm on $\{[a][b]X \stackrel{?}{=} [a]X\}$:

$$\begin{array}{lcl}
{[a][b]X} \stackrel{?}{=} [a]X & \Longrightarrow & (\stackrel{?}{=}[]) \\
{[b]X} \stackrel{?}{=} X \quad \text{Failure} & &
\end{array}$$

The algorithm fails because the precondition of rule $(\mathbf{IE})$, $X \notin fU([b]X)$ is not satisfied.

Example four (succeeds).

Take $supp(X) = \mathbb{A}^<$ and take $a, b \in \mathbb{A}^<$. We run the algorithm on $\{X \stackrel{?}{=} (a\ b)\cdot X\}$:

$$\begin{array}{lcl}
X \stackrel{?}{=} (a\ b)\cdot X & \Longrightarrow & (\stackrel{?}{=}\mathbf{X}) \\
a\#_? X\ ,\ b\#_? X & \stackrel{[X:=X\text{-}a]}{\Longrightarrow} & (\mathbf{IF}) \\
b\#_? X\text{-}a & \stackrel{[X\text{-}a:=(X\text{-}a)\text{-}b]}{\Longrightarrow} & \\
\varnothing \quad \text{Success, with } [X{:=}(X\text{-}a)\text{-}b] & &
\end{array}$$

Later we will prove Theorem 4.4.6, which tells us that failure here implies that no solution to the unification problem exists.

4.3 Preservation of solutions

. . . under non-instantiating rules

LEMMA 4.3.1. If $Pr \Longrightarrow Pr'$ by a non-instantiating rule (Definition 4.1.5) then $Sol(Pr) = Sol(Pr')$.

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

- *The case* $(\overset{?}{=}[])$. Suppose $Pr = [a]r \overset{?}{=} [a]s, Pr'$ and $[a]r \overset{?}{=} [a]s, Pr' \Longrightarrow r \overset{?}{=} s, Pr'$ by $(\overset{?}{=}[])$. By Definition 3.4.8 and properties of equality, $[a](r\theta) = [a](s\theta)$ if and only if $r\theta = s\theta$.
- *The case* $(\mathbf{F}())$. Suppose $Pr = a\#_?(r_1, \dots, r_n), Pr'$ and suppose that $a\#_?(r_1, \dots, r_n), Pr' \Longrightarrow a\#_?r_1, \dots, a\#_?r_n, Pr'$ by $(\mathbf{F}())$. By Definition 3.4.8 and Lemma 3.2.5, $a \notin fa((r_1, \dots, r_n)\theta)$ if and only if $a \notin fa(r_1\theta)$, $\dots$, $a \notin fa(r_n\theta)$.

■

LEMMA 4.3.2. Suppose $\theta(X) = \theta'(X)$ for all $X \in fU(Pr)$. Then $\theta \in Sol(Pr)$ if and only if $\theta' \in Sol(Pr)$.

Proof. From Definition 4.1.3 it suffices to show that $r\theta = s\theta$ if and only if $r\theta' = s\theta'$, for every $(r \overset{?}{=} s) \in Pr$, and $a \notin fa(r\theta)$ if and only if $a \notin fa(r\theta')$, for every $(a\#_?r) \in Pr$. This is immediate using Lemma 3.4.12. ■

...under $(\mathbf{IE})$

Recall from Remark 3.4.7 the discussion of why we write $\pi{\cdot}X$ when we have chosen a representative element X of an equivalence class of unknowns under permutations.

DEFINITION 4.3.3. Write $\theta{-}X$ for the substitution such that

$$\boxed{\begin{aligned}(\theta{-}X)(\pi{\cdot}X) &= \pi{\cdot}X \\ (\theta{-}X)(Y) &= \theta(Y) \quad \text{for all other } Y.\end{aligned}}$$

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

THEOREM 4.3.4. If $X\theta = s\theta$ and $X \notin fU(s)$ then

$$\theta = [X{:=}s] \circ (\theta{-}X).$$

That is:

$$\begin{aligned}\theta(X) &= X([X{:=}s] \circ (\theta{-}X)) \quad \text{and} \\ \theta(Y) &= Y([X{:=}s] \circ (\theta{-}X)).\end{aligned}$$

Proof. We reason as follows:

$$\begin{aligned}(\pi{\cdot}X)([X{:=}s] \circ (\theta{-}X)) &= (\pi{\cdot}s)(\theta{-}X) && \text{Definition 3.4.8, Lemma 3.5.2} \\ &= (\pi{\cdot}s)\theta && X \notin fU(s),\ \text{Lemma 3.4.12} \\ &= (\pi{\cdot}X)\theta && \text{Assumption}\end{aligned}$$

$$\begin{aligned}Y([X{:=}s] \circ (\theta{-}X)) &= Y(\theta{-}X) && \text{Definition 3.4.8, Lemma 3.5.2} \\ &= Y\theta && \text{Definition 4.3.3}\end{aligned}$$

■

… under (**IF**)

DEFINITION 4.3.5. Suppose θ is a substitution. Suppose $a \in \mathit{supp}(X)$ and $a \notin \mathit{fa}(\theta(X))$. Let $\delta_{X\text{-}a}$ be a *shift* permutation bijecting $\mathit{supp}(X)$ with $\mathit{supp}(X) \setminus \{a\}$.

Define a substitution $\theta_{[X\text{-}a:=X]}(X)$ by:

- $(\theta_{[X\text{-}a:=X]})(\pi\cdot X) = (\pi \circ \delta^{-1}_{X\text{-}a})\cdot\theta(X)$.
- $(\theta_{[X\text{-}a:=X]})(Y) = \theta(Y)$ for all other Y.

It is routine to verify that Definition 4.3.5 is well-defined and a substitution.

THEOREM 4.3.6. Suppose $a \in \mathit{supp}(X)$ and $a \notin \mathit{fa}(\theta(X))$. Then

$$\theta = [X{:=}X\text{-}a]\circ(\theta_{[X\text{-}a:=X]}).$$

That is:

$$\begin{aligned}\theta(\pi\cdot X) &= ([X{:=}X\text{-}a]\circ\theta_{[X\text{-}a:=X]})(\pi\cdot X) \quad \text{and}\\ \theta(Y) &= ([X{:=}X\text{-}a]\circ\theta_{[X\text{-}a:=X]})(Y).\end{aligned}$$

Proof. We unpack definitions:

$$\begin{aligned}([X{:=}X\text{-}a]\circ(\theta_{[X\text{-}a:=X]}))(\pi\cdot X) &= (\pi\cdot(X\text{-}a))\theta_{[X\text{-}a:=X]} && \text{Definition 3.5.1}\\ &= ((\pi\circ\delta_{X\text{-}a})\cdot X)\theta_{[X\text{-}a:=X]} && \text{Def. } X\text{-}a\\ &= (\pi\circ\delta_{X\text{-}a}\circ\delta^{-1}_{X\text{-}a})\cdot X && \text{Definition 4.3.5}\\ &= \pi\cdot X && \text{Group action}\end{aligned}$$

The result follows. ■

4.4 *Simplification rewrites calculate principal solutions*

DEFINITION 4.4.1. Write $\theta_1 \leq \theta_2$ when there exists some θ' such that $X\theta_2 = X(\theta_1 \circ \theta')$ always. Call $\leq$ the **instantiation ordering**.

DEFINITION 4.4.2. A **principal** (or **most general**) solution to a problem Pr is a solution $\theta \in \mathit{Sol}(Pr)$ such that $\theta \leq \theta'$ for all other $\theta' \in \mathit{Sol}(Pr)$.

Our main result is Theorem 4.4.5: the unification algorithm from Definition 4.1.7 calculates a principal solution.

LEMMA 4.4.3. If $\theta_1 \leq \theta_2$ then $\theta\circ\theta_1 \leq \theta\circ\theta_2$.

Proof. By Definition 4.4.1, θ' exists such that $X\theta_2 = X(\theta_1\circ\theta')$ always. Then:

$$\begin{aligned}X(\theta\circ\theta_2) &= (X\theta)\theta_2 && \text{Lemma 3.5.2}\\ &= (X\theta)(\theta_1\circ\theta') && \text{Lemma 3.4.12}\\ &= X((\theta\circ\theta_1)\circ\theta') && \text{Lemma 3.5.2}\end{aligned}$$

■

LEMMA 4.4.4.

1. Suppose $fa(s) \subseteq supp(X)$ and $X \notin fU(s)$. Write χ=$[X$:=$s]$. If $Pr \overset{\chi}{\Longrightarrow} Pr'$ with (**IE**) then $\theta \in Sol(Pr)$ implies $\theta - X \in Sol(Pr')$.

2. Suppose $a \in supp(X)$. Write ρ=$[X$:=X-$a]$. If $Pr \overset{\rho}{\Longrightarrow} Pr'$ with (**IF**) then $\theta \in Sol(Pr)$ implies $\theta_{[X\text{-}a:=X]} \in Sol(Pr')$.

Proof.

1. Suppose $Pr = X \overset{?}{=} s,\ Pr''$ so that $X \overset{?}{=} s,\ Pr'' \overset{\chi}{\Longrightarrow} Pr''\chi$. Now suppose $\theta \in Sol(Pr)$. By Theorem 4.3.4 $\chi \circ (\theta - X) \in Sol(Pr)$. By Lemma 4.1.4, $\theta - X \in Sol(Pr\chi)$. It follows that $\theta - X \in Sol(Pr''\chi)$ as required.

2. Suppose $Pr = a\#_? X,\ Pr''$ and $a \in supp(X)$ so that $Pr \overset{\rho}{\Longrightarrow} Pr\rho$. Now suppose $\theta \in Sol(Pr)$. By Theorem 4.3.6 $\rho \circ \theta_{[X\text{-}a:=X]} \in Sol(Pr)$. By Lemma 4.1.4, $\theta_{[X\text{-}a:=X]} \in Sol(Pr\rho)$ as required.

■

THEOREM 4.4.5. If $Pr \overset{\theta}{\Longrightarrow}{}^* \varnothing$ then θ is a principal solution to Pr (Definition 4.4.2).

Proof. By induction on the path of $Pr \overset{\theta}{\Longrightarrow}{}^* \varnothing$.

- *The empty path.* So $Pr = \varnothing$ and $\theta = id$. By Definition 4.4.1, $id \leq \theta'$.
- *The non-instantiating case.* Suppose
$$Pr \Longrightarrow Pr' \overset{\theta}{\Longrightarrow}{}^* \varnothing$$
where $Pr \Longrightarrow Pr'$ by a non-instantiating rule. By inductive hypothesis θ is a principal solution of Pr'. It follows from Lemma 4.3.1 that θ is also a principal solution of Pr.
- *The case* (**IE**). Suppose $fa(r) \subseteq supp(X)$ and $X \notin fU(r)$. Write $\chi = [X$:=$r]$. Suppose $Pr = r \overset{?}{=} X, Pr''$ so that
$$r \overset{?}{=} X,\ Pr'' \overset{\chi}{\Longrightarrow} Pr''\chi \overset{\theta''}{\Longrightarrow}{}^* \varnothing.$$
Further, consider any other $\theta' \in Sol(Pr)$.
By Lemma 4.4.4 $(\theta' - X) \in Sol(Pr''\chi)$ and by inductive hypothesis $\theta'' \in Sol(Pr''\chi)$ and $\theta'' \leq \theta' - X$. By Lemma 4.4.3, $\chi \circ \theta'' \leq \chi \circ (\theta' - X)$. By Theorem 4.3.4 $\chi \circ (\theta' - X) = \theta'$.
- *The case* (**IF**). Suppose $a \in supp(X)$. Write $\rho = [X$:=X-$a]$, so that
$$Pr \overset{\rho}{\Longrightarrow} Pr\rho \overset{\theta''}{\Longrightarrow}{}^* \varnothing,$$
Further, consider any other $\theta' \in Sol(Pr)$.
By Lemma 4.4.4, $\theta'_{[X\text{-}a:=X]} \in Sol(Pr\rho)$ and by inductive hypothesis $\theta'' \in Sol(Pr\rho)$ and $\theta'' \leq \theta'_{[X\text{-}a:=X]}$. By Lemma 4.4.3, $\rho \circ \theta'' \leq \rho \circ \theta'_{[X\text{-}a:=X]}$. By Theorem 4.3.6 $\rho \circ \theta'_{[X\text{-}a:=X]} = \theta'$.

■

THEOREM 4.4.6 (Correctness of algorithm). Given a problem Pr, if the algorithm of Definition 4.1.7 succeeds then it returns a principal solution; if it fails then there is no solution.

Proof. If the algorithm succeeds we use Theorem 4.4.5. Otherwise, the algorithm generates an element of the form $\mathsf{f}(r) \overset{?}{=} \mathsf{g}(s)$, $a \overset{?}{=} b$, $a\#_? a$, $a\#_? C$ where $a \in supp(C)$, or $X \overset{?}{=} s$ where $X \in fU(s)$ and s is not of the form $\pi{\cdot}X$. By arguments on syntax and size of syntax, no solution to the reduced problem exists. It follows by Lemma 4.4.4 that no solution to Pr exists. ■

DEFINITION 4.4.7. Fix terms r and s.

- Call **nominal unification** the problem of finding a θ to make $r\theta = s\theta$.
- Call **nominal matching** the problem of finding a θ to make $r\theta = s$.

COROLLARY 4.4.8. Providing that equality of $\mathcal{C}$ (constants), $\mathcal{X}$ (unknowns), and $\mathbb{P}$ (permutations) are decidable, nominal unification and nominal matching over signatures using them are also decidable.

Proof. An algorithm for unification is sketched in Definition 4.1.7; furthermore by Theorem 4.4.6 it calculates a most general θ which represents all other solutions.

For matching, we substitute unknowns in s with fresh (non-equivariant) constants of the same sorts and permission sets—we extend the signature if we need to—and run the unification algorithm. We then replace the constants by the original unknowns.[11] It is not hard to see that this calculates a most general matching solution. ■

REMARK 4.4.9. The matching and unification algorithms might generate solutions with *shift*-permutations. If we prefer to eliminate them then—provided that $\mathcal{X}$ has enough unknowns (Definition 3.1.10)—we may do so by appending an invertible substitution (Definition 3.5.3) mapping each shifted $\delta{\cdot}X$ in the solution to a fresh unknown Y such that $supp(Y) = \delta{\cdot}supp(X)$.

5 REWRITING

Nominal rewriting was the first logical system designed to study theories (sets of axioms, i.e. *rewrite rules*) over nominal terms. It was introduced by Fernández and the author in [FGM04; FG07]. Nominal terms allow us to express rewrite rules involving binding, like substitution and the λ-calculus (see Example 5.1.3).

The presentation of nominal rewriting here differs from that in [FG07], and is more concise. Partly this is optimisation, but this is also due to the permissive-nominal approach. We compare and contrast nominal rewriting from [FG07] with nominal rewriting here, in Subsection 5.6.

[11] We do not make this formal, but since constants are structurally just like unknowns the definitions can easily be constructed by proceeding exactly as we did when we defined substitution for unknowns.

5.1 Rewrite rules

DEFINITION 5.1.1. A **rewrite rule** in a signature $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{F}, ar)$ is a pair of terms $l \to m$ in Σ such that $sort(l) = sort(m) \in \mathcal{B}$ and $fU(m) \subseteq fU(l)$.
R will range over rewrite rules.
A **rewrite theory** $\mathsf{R} = (\Sigma, \mathit{Rew})$ is a pair of a signature Σ (Definition 3.1.5) and a (possibly infinite) set of rewrite rules Rew in Σ.

NOTATION 5.1.2. Write $(l \to m) \in \mathsf{R}$ to mean 'l and m are terms in Σ and $(l \to m) \in \mathit{Rew}$'.

The notion of rewrite rule and rewrite theory in Definition 5.1.1 is much like the first-order case, but because of the 'nominal' aspects of our syntax we can handle names and binding.

EXAMPLE 5.1.3. Here are some example rewrite theories:

- nrSUB expresses the usual capture-avoiding substitution action on λ-calculus terms.

  Let Σ have a base sort τ and the following term-formers:

  $$\mathsf{sub} : ([\nu]\tau, \tau)\tau \quad \mathsf{lam} : ([\nu]\tau)\tau \quad \mathsf{app} : (\tau, \tau)\tau \quad \mathsf{var} : (\nu)\tau$$

  Rewrite rules are as follows:

$$\begin{array}{llll}
(\mathbf{var}\!\to) & \mathsf{var}(a)[a\mapsto X] & \to X & \\
(\mathbf{var}\!\to') & \mathsf{var}(b)[a\mapsto X] & \to \mathsf{var}(b) & \\
(\mathsf{lam}\!\to) & \mathsf{lam}([a]X)[b\mapsto Y] & \to \mathsf{lam}([a](X[b\mapsto Y])) & (a\notin supp(Y)) \\
(\mathsf{app}\!\to) & \mathsf{app}(X, X')[b\mapsto Y] & \to \mathsf{app}(X[b\mapsto Y], X'[b\mapsto Y]) &
\end{array}$$

  Here and in the next example we sugar $\mathsf{sub}([a]r, t)$ to $r[a\mapsto t]$. Every permission set contains b and every permission set contains a except for $supp(Y)$, as indicated above.

- nrLAM extends the previous theory with two more rewrites:

$$\begin{array}{llll}
(\beta\!\to) & (\lambda[a]Z)X & \to Z[a\mapsto X] & \\
(\eta\!\to) & \lambda[a](Ya) & \to Y & (a\notin supp(Y))
\end{array}$$

Sugar $\mathsf{lam}(r)$ to λr, $\mathsf{app}(r, s)$ to rs, and $\mathsf{var}(a)$ to a. We anticipate Subsection 5.2 and sketch how one might rewrite $(\lambda[b](\lambda[a]ab))a$ to $\lambda[a']a'a$:

$$\begin{aligned}
(\lambda[b](\lambda[a]ab))a &\to (\lambda[a]ab)[b\mapsto a] \\
&= (\lambda[a']a'b)[b\mapsto a] \\
&\to \lambda[a']((a'b)[b\mapsto a]) \\
&\to^* \lambda[a']a'a
\end{aligned}$$

5.2 Rewrite steps

DEFINITION 5.2.1. Define the terms s in which X occurs **only once** by:

$$s ::= \pi{\cdot}X \mid [a]s \mid \mathsf{f}(r_1,\dots,r_{i-1},s,r_{i+1},\dots,r_n)$$
$$(X \notin fU(r_1),\dots,fU(r_{i-1}),fU(r_{i+1}),\dots,fU(r_n))$$

A **position** P is a pair (s,X) of a nominal term and an unknown X which occurs only once in s.

Our notion of *position* is also sometimes called a **context**; the idea goes back to at least [FH92].

In Definition 5.2.1, $\pi{\cdot}X$ denotes an unknown in the same permutation orbit as X.

NOTATION 5.2.2. If $P = (s,X)$ is a position write $supp(P)$ for $supp(X)$ and $sort(P)$ for $sort(X)$.

If $sort(r) = sort(P)$ and $fa(r) \subseteq supp(P)$ (so that $[X{:=}r]$ is a substitution) write $P[r]$ for $s[X{:=}r]$.

DEFINITION 5.2.3. The **one-step rewrite relation** $r \xrightarrow{\mathsf{R}} s$ is the least relation such that for every $(l \to m) \in \mathsf{R}$, position P, and substitution θ, if $sort(r) = sort(P)$ and $fa(l\theta) \cup fa(m\theta) \subseteq supp(P)$ (so that $P[l\theta]$ and $P[m\theta]$ are well-defined) then

$$\boxed{P[l\theta] \xrightarrow{R} P[m\theta].}$$

The **multi-step rewrite relation** $r \xrightarrow{\mathsf{R}}{}^{*} s$ is the reflexive transitive closure of the one-step rewrite relation.

We consider decidability and complexity of the rewrite relation in Section 6.

EXAMPLE 5.2.4. Let T have one name sort ν, one base sort τ, one term-former triv and one axiom $\mathsf{triv}(a) \to \mathsf{triv}(b)$.

Then $\mathsf{triv}(a) \to \mathsf{triv}(b)$ but also (using positions $(\pi{\cdot}X, X)$ for any π) $\mathsf{triv}(b) \to \mathsf{triv}(a)$ and $\mathsf{triv}(a') \to \mathsf{triv}(b')$ for any pair of distinct atoms a' and b'.

Thus atoms in rewrite rules range over 'any atom' analogously to how unknowns in rewrite rules range over 'any term'.

EXAMPLE 5.2.5. Recall the rule $(\eta{\to}) = (\lambda[a](Ya) \to Y)$ where $a \notin supp(Y)$ from Example 5.1.3. Suppose also $b \notin supp(Y)$.

1. To deduce $\lambda[a](ba) \to b$ we take $P = ((b\ c){\cdot}Y, Y)$ for some $c \in supp(Y)$ and we take $\theta = [Y{:=}c]$.
2. To deduce $\lambda[a'](ba') \to b$ for any other a' we *also* take $P = ((b\ c){\cdot}Y, Y)$ and $\theta = [Y{:=}c]$. This is because $\lambda[a'](ba')$ and $\lambda[a](ba)$ are the same term (Lemma 2.4.9).
3. To deduce $\lambda[a](Ya) \to Y$ we take $P = (Y,Y)$ and $\theta = id$.
4. Suppose $supp(Y') = supp(Y) \cup \{a\}$.
   Suppose we have *shift*-permutations so there exists a permutation, write it $\delta_{Y'\text{-}a}$, bijecting $supp(Y')$ with $supp(Y)$. To deduce $\lambda[a](Y'a) \to Y'$ we take $P = ((\delta_{Y'\text{-}a})^{-1}{\cdot}Y, Y)$ and $\theta = [Y{:=}\delta_{Y'\text{-}a}{\cdot}Y']$.
   Without *shift* we cannot deduce $\lambda[a](Y'a) \to Y'$; we can still deduce $\lambda[a](Ya) \to Y$.

5. We cannot deduce $\lambda[a](aa) \to a$, because $[Y{:=}a]$ is not a substitution: no function mapping Y to a can be equivariant, since $(b\ a)\cdot Y = Y$ but $(b\ a)\cdot a = b \neq a$ (also $a \notin \mathit{supp}(Y)$: see Proposition 3.4.3).
6. A rewrite $X \to X$ only entails rewrites for t with $\mathit{fa}(t) \subseteq \pi\cdot\mathit{supp}(X)$ for some π. With *shift*, the effect of this may be that we can deduce $t \to t$ from $X \to X$ for any t. We make no claim to there being a 'right' or 'wrong' answer here: the issue is purely a design question of how much expressivity we want permutations to have. Our results are parameterised over this choice.

DEFINITION 5.2.6.

- Call R **locally confluent** when $r \xrightarrow{\mathsf{R}} s_1$ and $r \xrightarrow{\mathsf{R}} s_2$ implies there exists some s' such that $s_1 \xrightarrow{\mathsf{R}}{}^{*} s'$ and $s_2 \xrightarrow{\mathsf{R}}{}^{*} s'$.
- Call R **confluent** when $r \xrightarrow{\mathsf{R}}{}^{*} s_1$ and $r \xrightarrow{\mathsf{R}}{}^{*} s_2$ implies there exists some s' such that $s_1 \xrightarrow{\mathsf{R}}{}^{*} s'$ and $s_2 \xrightarrow{\mathsf{R}}{}^{*} s'$.

5.3 Peaks, critical pairs, joinability

We now begin to investigate criteria for deducing confluence of nominal rewrite systems. Our first observation is that things are not quite as simple as in first-order rewriting [BN98, Section 6.2]: by Lemma 5.3.5, trivial critical pairs are not always joinable.

DEFINITION 5.3.1. Write $r \to s_1, s_2$ when $r \to s_1$ and $r \to s_2$ and call this a **peak**. Call this peak **joinable** when there exists a t such that $s_1 \to^{*} t$ and $s_2 \to^{*} t$.

So R is locally confluent when every peak is joinable.

DEFINITION 5.3.2. Consider two rewrite rules $R_1 = (l_1 \to m_1)$ and $R_2 = (l_2 \to m_2)$. Call R_1 a **copy** of R_2 when there exists an invertible substitution θ such that $(l_2\theta \to m_2\theta) = R_1$.

Clearly, if R_1 is a copy of R_2 then R_2 is also a copy of R_1. Furthermore:

LEMMA 5.3.3. If R_1 and R_2 are copies of the same rule then $l \xrightarrow{R_1} m$ if and only if $l \xrightarrow{R_2} m$.

Proof. Unpacking Definition 5.2.3 and exploiting the existence of an inverse θ^{-1}. ■

DEFINITION 5.3.4. Suppose that $R_i = (l_i \to m_i)$ for $i = 1, 2$ and $\mathit{fU}(R_1) \cap \mathit{fU}(R_2) = \varnothing$. Suppose $l_1 = P[l_1']$ for some l_1', and suppose $l_1' \overset{?}{=} l_2$ has a principal solution θ. Call the pair $(m_1\theta, P[m_2]\theta)$ a **critical pair**.

Call $(m_1\theta, P[m_2]\theta)$ **trivial** when at least one of the following hold:

1. $P = (\pi\cdot X, X)$ and R_1 and R_2 are copies of the same rule.
2. $l_1' = X$ for some unknown X.

LEMMA 5.3.5. Peaks that are instances of trivial critical pairs, are not always joinable.

Proof. It suffices to provide a counterexample. Fix term-formers $\mathsf{0}$ and f and take $R_1 = (\mathsf{0} \to a)$ and $R_2 = (X \to \mathsf{f}(a))$ where $a \notin \mathit{supp}(X)$.

There is a critical pair $(a, \mathsf{f}(a))$ between R_1 and R_2.

Also, $\mathsf{0} \xrightarrow{R_1} a$ and $\mathsf{0} \xrightarrow{R_2} \mathsf{f}(a)$ and it is a fact that this peak cannot be joined—we 'want' to close this peak by rewriting a to $\mathsf{f}(a)$ using R_2, but the fact that $a \notin \mathit{supp}(X)$ blocks this. ■

5.4 Uniform rewriting

The proof of Lemma 5.3.5 suggests a simple cure:

DEFINITION 5.4.1. Call a rule $R = (l \to m)$ **uniform** when

$$\mathit{fa}(m) \subseteq \mathit{fa}(l).$$

Call a rewrite theory R **uniform** when every $R \in \mathsf{R}$ is uniform.

Definition 5.4.1 mirrors the condition in Definition 5.1.1 that $\mathit{fU}(m) \subseteq \mathit{fU}(l)$, but for atoms instead of unknowns. This condition is sufficient to obtain Theorem 5.4.7, which is a nominal rewriting version of the well-known critical pair lemma from first-order rewriting [BN98, Theorem 6.2.4].

EXAMPLE 5.4.2. Let R have one name sort ν, one base sort τ, two term-formers $\mathsf{triv} : (\nu)\tau$ and $\mathsf{abs} : ([\nu]\tau)\tau$, and rewrite rules

$$\mathsf{triv}(a) \to \mathsf{triv}(a) \qquad \mathsf{triv}(a) \to \mathsf{triv}(b) \qquad \mathsf{abs}([a]X) \to X.$$

$\mathit{fa}(\mathsf{triv}(a)) \subseteq \mathit{fa}(\mathsf{triv}(a))$ and $\mathit{fa}(\mathsf{triv}(b)) \not\subseteq \mathit{fa}(\mathsf{triv}(a))$. Also $\mathit{fa}(X) \not\subseteq \mathit{fa}(\mathsf{abs}([a]X))$ if and only if $a \notin \mathit{supp}(X)$.

So the first rule is uniform, the second is not, and the third is uniform if and only if $a \notin \mathit{supp}(X)$.

The rewrite rules of nrSUB and nrLAM in Example 5.1.3 are uniform.[12]

LEMMA 5.4.3. If $\mathit{fa}(m) \subseteq \mathit{fa}(l)$ then $\mathit{fa}(P[m]) \subseteq \mathit{fa}(P[l])$.

Proof. Routine induction using Lemmas 3.2.8 and 3.2.5. ■

COROLLARY 5.4.4. $R = (l \to m)$ is uniform if and only if $\forall r,s.\big(r \xrightarrow{R} s \Rightarrow \mathit{fa}(s) \subseteq \mathit{fa}(r)\big)$.

Proof. From Lemmas 3.2.8 and 3.4.11. ■

LEMMA 5.4.5. Suppose $R = (l \to m)$ is uniform and $X \notin \mathit{fU}(R)$. Suppose $\theta(X) = l\theta$. Specify θ' by $\theta'(\pi{\cdot}X) = \pi{\cdot}(m\theta)$ and $\theta'(Y) = \theta(Y)$. Then $r\theta \to^* r\theta'$ for any r.

Proof. θ' is a substitution by Lemmas 3.4.11 and 3.2.8. The result follows by a routine induction on r. ■

[12]There is a deeper reason for this: they are also closed. See Example 6.2.2 and Theorem 6.2.3.

Because of Lemma 5.3.3, we can be relaxed about the particular (orbits of) unknowns that are used in a rewrite rule, if we only care about the rewrites that they generate. We do this in Theorems 5.4.6 and 5.4.7. This can always be made formal by inserting invertible 'freshening' substitutions as appropriate.

THEOREM 5.4.6. If a rewrite theory R (Definition 5.1.1) is uniform then peaks that are instances of trivial critical pairs, are joinable.

Proof. Consider two rules $R_i = (l_i \to m_i) \in \mathsf{R}$ for $i = 1, 2$. Taking copies if necessary, suppose $fU(R_1) \cap fU(R_2)$. Suppose they have a critical pair $(m_1\theta, P[m_2]\theta)$. That is, there exists l'_1 such that $l_1 = P[l'_1]$ and θ is a principal solution to $l'_1 \overset{?}{=} l_2$.

There are two cases:

- *The case $P = (\pi{\cdot}X, X)$ and R_1 and R_2 are copies of the same rule $l \to m$.* The peak we want to join is $l_1\theta = \pi{\cdot}l_2\theta \to m_1\theta,\ \pi{\cdot}m_2\theta$, where the rules $l_1 \to m_1$ and $l_2 \to m_2$ are identical aside from their free unknowns which are renamed disjoint. We use Lemma 3.4.12 and the assumption in Definition 5.1.1 that $fU(m) \subseteq fU(l)$.
- *The case of $(m_1\theta, P[m_2]\theta)$ where $l_1 = P[X]$ and $\theta(X) = l_2$.* Specify θ' by $\theta'(\pi{\cdot}X) = \pi{\cdot}m_2$ and $\theta'(Y) = \theta(Y)$ for all other Y; note that θ' *is* a substitution since $fa(m_2) \subseteq fa(l_2)$ by uniformity and $fa(l_2) \subseteq supp(X)$ by our assumption that θ is a substitution.

  By Lemma 5.4.5 $m_1\theta \to^* m_1\theta'$. By definition $P[m_2]\theta = l_1\theta' \xrightarrow{R_1} m_1\theta'$, so we have joined the peak.

■

THEOREM 5.4.7. Suppose all non-trivial critical pairs of R are joinable *and* suppose R is uniform. Then R is locally confluent.

Proof. Suppose $r \xrightarrow{R_1} s_1$ and $r \xrightarrow{R_2} s_2$. Write P_1 and P_2 for the positions at which the two rewrites occur. Taking copies if necessary, suppose $fU(R_1) \cap fU(R_2) = \varnothing$.

If P_1 and P_2 identify distinct subterms of r then local confluence holds by a standard diagrammatic argument (see for instance [BN98]).

Otherwise it must be that $P_2 = (P_1[P], X)$ for some position P; that is, P_2 identifies a point in r beneath the point identified by P_1 (or the symmetric case that $P_1 = (P_2[P], X)$, which is similar and we elide). There are now three possibilities:

1. X in P_2 replaces an unknown in r. This is an instance of a trivial critical pair; we use Theorem 5.4.6.
2. $P = (\pi{\cdot}X, X)$ and R_1 and R_2 are copies of the same rule. Then again this is an instance of a trivial critical pair and we use Theorem 5.4.6.
3. Otherwise, this is an instance of a non-trivial critical pair at it may be joined using our assumption that non-trivial critical pairs are joinable.

■

DEFINITION 5.4.8. Call a rewrite system R **terminating** when all rewrite sequences are finite. Call a term r a **normal form** (with respect to a rewrite system R) when $\forall s.\neg(r \xrightarrow{\mathsf{R}} s)$, that is, when r does not R-rewrite to anything.

EXAMPLE 5.4.9. It can be proved that nrSUB in Example 5.1.3 is terminating. nrLAM (famously) is not terminating, because of $(\beta\mapsto)$.

COROLLARY 5.4.10. Suppose R is terminating, uniform, and suppose non-trivial critical pairs in R are joinable. Then:

1. R is confluent.
2. If $r \to^* s$ and $r \to^* s'$ and s and s' are normal forms, then $s = s'$.

5.5 Orthogonal rewrite systems

We now treat another standard criterion in rewriting: orthogonality [DJ89; BN98]. By Theorem 5.5.7 orthogonality implies not only local confluence, but the stronger property of confluence (Definition 5.2.6). The proof is not direct: it turns out that it is easier to consider an auxilliary *parallel reduction* relation $\Rightarrow$ (Definition 5.5.4). The reflexive transitive closure of $\Rightarrow$ is equal to that of $\to$ (Lemma 5.5.5), but $\Rightarrow$ allows (intuitively) multiple reductions provided that they do not occur 'one after the other, in the same position'. This is the kind of multiple reduction generated in the second case of the proof of Theorem 5.4.6, when we rewrite $m_1\theta$ to $m_1\theta'$.

DEFINITION 5.5.1. Call $R = (l \to m)$ **left-linear** when each unknown occurring in l occurs only once (Definition 5.2.1).

For example $\mathsf{f}(X) \to \mathsf{g}(X, X)$ is left-linear but $\mathsf{g}(X, X) \to \mathsf{f}(X)$ and $\mathsf{g}(\pi\cdot X, x) \to \mathsf{f}(X)$ are not. Note that $(a, a) \to a$ is left-linear.

DEFINITION 5.5.2. Call R **orthogonal** when every $R \in$ R is uniform and left-linear, and all critical pairs are trivial.

(Note that we insist that R is uniform, as well as the standard condition that it be left-linear.)

DEFINITION 5.5.3. Suppose $R = (l \to m)$. Write $r \overset{R}{\to}_\epsilon s$ when $r \xrightarrow{R} s$ and the rewrite occurs at a position $P = (\pi\cdot X, X)$. We say that the rewrite with R occurs at **root position**.

Expanding Definition 5.5.3, $r \overset{R}{\to}_\epsilon s$ when there exists θ and π such that $r = \pi\cdot(l\theta)$ and $s = \pi\cdot(m\theta)$. For example: if $R = (a \to a)$ then $a \overset{R}{\to}_\epsilon a$ but not $[a]a \overset{R}{\to}_\epsilon [a]a$.

DEFINITION 5.5.4. We define a **parallel reduction** relation $\Rightarrow$ by the rules in Figure 3.

LEMMA 5.5.5. $r \to^* s$ if and only if $r \Rightarrow^* s$.

Proof. By routine inductions. ■

LEMMA 5.5.6. If R is orthogonal then $\Rightarrow$ is confluent.

Proof. We prove by induction on the derivation of $r \Rightarrow s$ that a stronger property holds, often called the **diamond property**: for all s' if $r \Rightarrow s'$ then there exists some

$$\frac{r_1 \Rightarrow s_1 \ \cdots \ r_n \Rightarrow s_n}{\mathsf{f}(r_1,\dots,r_n) \Rightarrow \mathsf{f}(s_1,\dots,s_n)}\,(\Rightarrow\mathsf{f})$$

$$\frac{r_1 \Rightarrow s_1 \ \cdots \ r_n \Rightarrow s_n \quad \mathsf{f}(s_1,\dots,s_n) \overset{R}{\to}_\epsilon s'}{\mathsf{f}(r_1,\dots,r_n) \Rightarrow s'}\,(\Rightarrow\mathsf{f}')$$

$$\frac{s \Rightarrow t}{[a]s \Rightarrow [a]t}\,(\Rightarrow\mathbf{abs}) \qquad \frac{r \Rightarrow s \quad [a]s \overset{R}{\to}_\epsilon s'}{[a]r \Rightarrow s'}\,(\Rightarrow\mathbf{abs}')$$

$$\frac{}{r \Rightarrow r}\,(\mathbf{refl}) \qquad \frac{a \overset{R}{\to}_\epsilon s'}{a \Rightarrow s'}\,(\Rightarrow\mathbf{a}') \qquad \frac{X \overset{R}{\to}_\epsilon s'}{X \Rightarrow s'}\,(\Rightarrow\mathbf{X}')$$

Figure 3: Parallel reduction relation

s'' such that $s \Rightarrow s''$ and $s' \Rightarrow s''$. From this, confluence easily follows by a standard diagrammatic argument.

We consider a selection of cases:

- *The derivations of $r \Rightarrow s$ and $r \Rightarrow s'$ both end in* $(\Rightarrow\mathsf{f})$. We use the inductive hypotheses and $(\Rightarrow\mathsf{f})$.
- *The derivation of $r \Rightarrow s$ ends in* $(\Rightarrow\mathsf{f})$ *and that of $r \Rightarrow s'$ ends in* $(\Rightarrow\mathsf{f}')$. So $r_i \Rightarrow s_i$ and $r_i \Rightarrow s'_i$ for $1 \leq i \leq n$, and $\mathsf{f}(s'_1,\dots,s'_n) = \pi\cdot(l\theta) \overset{R}{\to}_\epsilon \pi\cdot(m\theta)$ for some π and $R = (l \to m) \in \mathsf{R}$. By inductive hypothesis there exist s''_i such that $s_i \Rightarrow s''_i$ and $s'_i \Rightarrow s''_i$. We now proceed as illustrated and explained below:

$$\begin{tikzcd}
\mathsf{f}(r_1,\dots,r_n) \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \mathsf{f}(s'_1,\dots,s'_n) \arrow[d, Rightarrow] \arrow[r, phantom, "="] & \pi\cdot(l\theta) \arrow[r, "R_\epsilon"] & \pi\cdot(m\theta) \arrow[d, Rightarrow] \\
\mathsf{f}(s_1,\dots,s_n) \arrow[r, Rightarrow] & \mathsf{f}(s''_1,\dots,s''_n) \arrow[r, phantom, "="] & \pi\cdot(l\theta') \arrow[r, "R_\epsilon"] & \pi\cdot(m\theta')
\end{tikzcd}$$

Either l is an unknown X or the rewrite $\mathsf{f}(s'_1,\dots,s'_n) \Rightarrow \mathsf{f}(s''_1,\dots,s''_n)$ takes place in the substitution θ.
If l is an unknown then by uniformity we may rewrite $f(s''_1,\dots,s''_n)$ using R and close the diagram by rewriting corresponding instances of $\theta(X)$ in $\pi\cdot(m\theta)$.
Otherwise, by uniformity there is a substitution θ' such that $\theta(X) \Rightarrow \theta'(X)$ for every X and $\mathsf{f}(s''_1,\dots,s''_n) = \pi\cdot(l\theta')$. Rules are also left-linear so R still applies to $\pi\cdot(l\theta)$: $\mathsf{f}(s''_1,\dots,s''_n) \overset{R}{\to}_\epsilon \pi\cdot(m\theta')$ and therefore $\mathsf{f}(s_1,\dots,s_n) \Rightarrow s\theta'$ by $(\Rightarrow\mathsf{f}')$ for R.

The other cases are no harder. ■

THEOREM 5.5.7. If a theory R is orthogonal (Definition 5.5.2) then R is confluent (Definition 5.2.6).

Proof. If the uniform rewrite system has only left-linear rules and only trivial critical pairs, then $\Rrightarrow$ is confluent by Lemma 5.5.6. It follows that $\Rrightarrow^*$ is confluent. By Lemma 5.5.5 the result follows. ■

5.6 *Nominal rewriting with freshness contexts versus permissive-nominal rewriting*

As mentioned in the introduction to this Section, the presentation of this paper differs from that of [FG07] in being permissive-nominal.

For clarity, let us call the nominal rewrite framework from [FG07] 'System ∇' and the nominal rewrite framework here 'System S'.

In system ∇ a rewrite rule takes the form $\nabla \vdash t \to u$ where ∇ is a set of assumptions $a\#\mathtt{X}$ called a *freshness context*. $\mathtt{X}$ is an *unknown*. This is not typed by a permission set; freshness information is given by ∇.

Here are $(\lambda{\to})$ from Example 5.1.3, and how it would look in System ∇:

$$\begin{array}{lrlll} \text{System S} & & \mathsf{lam}([a]X)[b{\mapsto}Y] \to & \mathsf{lam}([a](X[b{\mapsto}Y])) & (a{\notin}supp(Y)) \\ \text{System } \nabla & a\#\mathtt{Y} \vdash & \mathsf{lam}([a]\mathtt{X})[b{\mapsto}\mathtt{Y}] \to & \mathsf{lam}([a](\mathtt{X}[b{\mapsto}\mathtt{Y}])) & \end{array}$$

$a \notin supp(Y)$ is a fact (we must choose Y so that this is true). It does not matter *which* permission set we give Y because using δ and swappings we can build a π to map $supp(Y)$ to every other permission set $\pi{\cdot}supp(Y)$—which will contain $\pi(a)$.

Conversely $a\#\mathtt{Y}$ is a freshness condition. It directly controls the terms to which we may instantiate $\mathtt{Y}$; they must not contain a free. Here we attain this effect using Proposition 3.4.3.

Freshness conditions are elementary: they mean what they say and what they mean be quickly understood. Permission sets are still finitely representable, but somewhat harder to understand. So from the point of view of keeping a gentle learning curve, System ∇ may be preferable to System S.

However, System S rewards us with some advantages: we can use nominal abstract syntax and the freshness conditions which must be explicitly stated (repeatedly) in [FG07] are handled in the background by equivariance of substitutions (as Proposition 3.4.3 makes formal).

This also has some effects on mathematical properties. In System ∇ from [FG07] it was not in general the case that if $\nabla \vdash r \approx_\alpha r'$ and $\nabla \vdash r \xrightarrow{R} s$ then $\nabla \vdash r' \xrightarrow{R} s$ (see the end of Subsection 5.2 in [FG07]). It was also not in general the case that nominal rewriting coincides with nominal algebra (Section 7), essentially because any fixed freshness contexts might not be 'big enough'. Fernández and the author wrote a paper on how to adjust for this [FG10]. In a permissive-nominal context, these issues do not arise in the first place.

This author's feeling is that nominal-terms-with-freshness-contexts and permissive-nominal terms can be considered as essentially the same thing. However, if our goal is to prove theorems then we get closer to what is 'really going on' via the permissive-nominal presentation.

6 CLOSED TERMS

Equivariant unification—the problem of finding θ *and* π such that $\pi{\cdot}(r\theta) = s\theta$—is NP complete [Che04; Che10]. The same applies to corresponding matching problems. This matters to us because the rewrite relation in Definition 5.2.3 is equivariant; to determine whether r rewrites with a rule $(l \to r)$, we must solve an equivariant matching problem.

Fernández and the author introduced a notion of *closed term* such that for closed terms, equivariant matching/unification coincides with 'ordinary' matching/unification [FG07]. That is, for closed terms we can throw away the π.

We now develop corresponding definitions and results. The definitions and proofs in this paper are significantly different from those in [FG07].[13]

6.1 The definition

DEFINITION 6.1.1. Define **explicit atoms** $ea(r)$ inductively by:

$$\boxed{\begin{array}{ccc} ea(a) = \{a\} & ea(C) = supp(C) & ea(X) = \varnothing \\ ea(\mathsf{f}(r)) = ea(r) & ea((r_1,\dots,r_n)) = \bigcup ea(r_i) & ea([a]r) = ea(r)\backslash\{a\} \end{array}}$$

REMARK 6.1.2. Intuitions for $ea(r)$ versus $fa(r)$ are as follows:

- The explicit atoms of r are the atoms that actually appear in r (unbound). That is, we can read '$a \in ea(r)$' as 'a appears in r'.
- The free atoms of r are the atoms that can appear in $r\theta$ for some θ.

For instance, $ea(X) = \varnothing \neq supp(X) = fa(X)$.

This is an intuition, not a fact. $fa(r) = \bigcup_\theta ea(r\theta)$ is not true in general (but see Lemma 6.1.5). For instance in a signature with one base sort τ and no term formers, terms containing atoms simply do not populate the sort τ.

Recall the notion of *occurrences* $occ(r)$ from Definition 3.7.3.

NOTATION 6.1.3. Write $\pi{\cdot}occ(r) = \{\pi{\cdot}x \mid x \in occ(r)\}$. Also if $D = [d_1,\dots,d_n]$ and S is a permission set define $S \setminus D = S \setminus \{d_1,\dots,d_n\}$.

LEMMA 6.1.4. $ea(\pi{\cdot}r) = \pi{\cdot}ea(r)$ and $occ(\pi{\cdot}r) = \pi{\cdot}occ(r)$. In addition, $ea(r) \subseteq ea(r\theta)$.

Proof. By routine inductions on r. ■

LEMMA 6.1.5. $fa(r) = ea(r) \cup \bigcup\{supp(x) \mid x \in occ(r)\}$.

As an easy corollary using Lemma 3.2.5, $fa(r) = ea(r) \cup \bigcup\{supp(X) \setminus D \mid [D]X \in occ(r)\}$.

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

[13]The interested reader can begin by comparing our notion of closed terms in Definition 6.1.7, based on two simpler inductive definitions, with that used in [FG07, Definition 68], based on a renamed variant of a term and an equality derivable in an extended freshness context. See also an inductive characterisation of closed terms in unpublished notes [Clo07].

- *The case* $[a]r$. Suppose $\mathit{fa}(r) = \mathit{ea}(r) \cup \bigcup\{\mathit{supp}(x) \mid x \in \mathit{occ}(r)\}$. By definition $\mathit{fa}([a]r) = \mathit{fa}(r) \setminus \{a\}$, and $\mathit{ea}([a]r) = \mathit{ea}(r) \setminus \{a\}$ and $\mathit{occ}([a]r) = \{[a]x \mid x \in \mathit{occ}(r)\}$. The result follows by an easy sets calculation.

∎

DEFINITION 6.1.6. Call r fa**-functional** when if $[D_1]X \in \mathit{occ}(r)$ and $[D_2]X \in \mathit{occ}(r)$ then $\mathit{fa}([D_1]X) = \mathit{fa}([D_2]X)$ (equivalently, when D_1 and D_2 contain the same atoms but not necessarily in the same order).

DEFINITION 6.1.7. Call r **closed** when r is fa-functional and $\mathit{ea}(r) = \varnothing$.

EXAMPLE 6.1.8.

- a is not closed (ea is non-empty).
- X is closed, so note that 'closed' does not mean '$fU(r) = \varnothing$'. Our terminology is consistent with [FG07] and the subsequent literature.
- $([a]X, X)$ is not closed (occ is not fa-functional).
- $[a](X, a)$ is closed.

LEMMA 6.1.9. Suppose $\mathit{ea}(r) = \varnothing$. Then $\pi\cdot(r\theta) = r\theta'$ if and only if $\pi\cdot(([D]X)\theta) = ([D]X)\theta'$ for every $[D]X \in \mathit{occ}(r)$.

Proof. By a routine induction on r. ∎

THEOREM 6.1.10. r is closed if and only if

$$\exists S.\mathit{fa}(r) \subseteq S \wedge \forall \pi, \theta. \pi\cdot \mathit{fa}(r\theta) \subseteq S \Rightarrow \exists \theta'.\pi\cdot(r\theta) = r\theta'.$$

Proof. Suppose there is a permission set $S \supseteq \mathit{fa}(r)$ such that if $\pi\cdot\mathit{fa}(r\theta) \subseteq S$ then there exists θ' such that $\pi\cdot(r\theta) = r\theta'$. There are two things to prove:

- *$\mathit{ea}(r)$ is empty.* Suppose there exists $a \in \mathit{ea}(r)$. Pick $b \in S \setminus \mathit{ea}(r)$. By assumption taking $\theta = \mathit{id}$ there exists θ' such that $(b\ a)\cdot(r\theta) = r\theta'$. By Lemma 6.1.4 $\mathit{ea}((b\ a)\cdot r) = (b\ a)\cdot \mathit{ea}(r) \not\ni a$ and $a \in \mathit{ea}(r) \subseteq \mathit{ea}(r\theta')$, a contradiction.
- *$\mathit{occ}(r)$ is fa-functional.* Consider $[D_1]X$ and $[D_2]X$ in $\mathit{occ}(r)$; choose D_i such that $D_i \cap \mathit{fa}(r) = \varnothing$ for $i = 1, 2$. Suppose there exists $a \in \mathit{fa}([D_2]X) \setminus \mathit{fa}([D_1]X)$, and choose any $b \in \mathit{fa}([D_1]X)$ (since $\mathit{supp}(X)$ is infinite and D_1 is finite, such a b exists).
  By Lemma 6.1.5 $a, b \in \mathit{fa}(r)$ so by assumption taking $\theta = \mathit{id}$ there exists θ' such that $(b\ a)\cdot r = r\theta'$. We proved above that $\mathit{ea}(r) = \varnothing$, so by Lemma 6.1.9 $(b\ a)\cdot[D_1]X = ([D_1]X)\theta$. By Lemma 3.2.8 a is free in the left-hand side, and by Lemma 3.4.11 a is not free in the right-hand side; a contradiction.

Suppose $occ(r)$ is fa-functional and $ea(r) = \varnothing$ and choose some permutation π and substitution θ.

If $occ(r) = \varnothing$ then by Lemma 6.1.5 $fa(r) = \varnothing$ so by Lemmas 3.2.9 and 3.4.12 $\pi\cdot(r\theta) = r$ and $r\theta' = r$, so there is nothing to prove.

Otherwise take $S = fa(r)$. For every element of in $occ(r)$ make a fixed but arbitrary choice of representation as $[D]X$ where the atoms in D are disjoint from the atoms in $nontriv(\pi)$. We take θ' to equivariantly extend this choice (Definition 2.5.4), so we map $\pi'\cdot X$ to $(\pi'\circ\pi)\cdot\theta(X)$ for the choice of representing X above, and otherwise to map Y to Y. Using Proposition 2.5.5 this is a substitution and $\pi\cdot(([D]X)\theta) = ([D]X)\theta'$ for every $[D]X \in occ(r)$. We use Lemma 6.1.9. ■

6.2 *Closed rewrite rules*

DEFINITION 6.2.1. Call a rewrite rule $l \to m$ **closed** when (l, m) is closed.

EXAMPLE 6.2.2. Let R have one name sort ν, one base sort τ, two term-formers triv : $(\nu)\tau$ and abs : $([\nu]\tau)\tau$, and rewrite rules

$$\mathsf{triv}(a) \to \mathsf{triv}(a) \qquad \mathsf{triv}(a) \to \mathsf{triv}(b) \qquad \mathsf{abs}([a]X) \to X.$$

The terms $\mathsf{triv}(a)$ and $\mathsf{triv}(b)$ are not closed; the terms $\mathsf{abs}([a]X)$ and X are closed. The terms $(\mathsf{triv}(a), \mathsf{triv}(a))$ and $(\mathsf{triv}(a), \mathsf{triv}(b))$ are not closed. The term $(\mathsf{abs}([a]X), X)$ is closed if and only if $a \notin supp(X)$. So the first two rules are not closed and the third is closed if and only if $a \notin supp(X)$.

The rewrite rules of nrSUB and nrLAM in Example 5.1.3 are closed.

Recall that uniform rules have good properties like Theorems 5.4.7 and 5.5.7. Closed rules inherit these good properties, because:

THEOREM 6.2.3. If $R = (l \to m)$ is closed then it is uniform.

Proof. By assumption $fU(m) \subseteq fU(l)$. Also (l, m) is fa-functional; it follows that $occ(m) \subseteq occ(l)$. The result follows from Lemma 6.1.5. ■

LEMMA 6.2.4. Suppose r and l are terms and l is closed. Then

1. $\exists\pi, \theta. r = \pi\cdot(l\theta)$ implies
2. $\forall\pi. fa(r) \subseteq \pi\cdot fa(l) \Rightarrow \exists\theta. r = \pi\cdot(l\theta)$

Proof. Suppose $fa(r) \subseteq \pi\cdot fa(l)$ and $fa(r) \subseteq \pi'\cdot fa(l)$ and $r = \pi\cdot(l\theta)$.

We need a θ' such that $r = \pi'\cdot(l\theta')$. It follows from the above that $(\pi'^{-1} \circ \pi)\cdot fa(l\theta) \subseteq fa(l)$. We use Theorem 6.1.10. ■

THEOREM 6.2.5. If R is closed then $\xrightarrow{\mathsf{R}}$ can be checked as follows, where for simplicity we suppose $\mathsf{R} = \{(l \to m)\}$:

1. We try to match r against $\pi\cdot l$ for some π such that $fa(r) \subseteq \pi\cdot fa(l)$, if such a π exists.

2. If we fail then by Lemma 6.2.4 we must fail for instantiating for any $\pi{\cdot}l$. We descend into subterms of r and repeat the previous step.

Whether step 1 of the algorithm above is decidable depends on the decidability of $\mathbb{P}$, $\mathcal{X}$, and $\mathcal{C}$; obviously, if equality of the syntax is undecidable then matching will also be undecidable. So assuming that we have not been *silly*, closed rules are useful because we only need to compute one π and consider matching, rather than consider an equivariant matching problem.

To use the matching algorithm of Section 4, we need terms to satisfy condition 2 of Definition 4.1.1. So, we could forbid *shift* permutations altogether. The algorithm might reintroduce them but as noted in Remark 4.4.9, *shift* can be eliminated once a solution is found. Thus, if we care about decidability and not so much about infinite permutations—which was the case e.g. in [FGM04; FG07]—then *shift* can be viewed as an internal mechanism of our unification/matching algorithm. However we have designed the mathematics to allow the possibility of exploring other, more liberal (and perhaps still decidable) choices, if we wish. More on this in [Gab12a].

7 EQUALITY: (PERMISSIVE-)NOMINAL ALGEBRA

Permissive-nominal algebra has one judgement form: an equality $r = s$. This is just an unoriented nominal rewriting rule, so what makes algebra different from rewriting is not so much the judgement form as the properties we care about: instead of confluence and decidability, we primarily care about soundness and completeness. These are Theorems 7.4.6 and Corollary 7.5.12.

This different emphasis affects the axioms we write. The rewrites in Example 5.1.3 are designed to work on λ-terms without unknowns (since we expect to 'evaluate' closed terms using rewrites). The analogous axioms in Example 7.1.3 are designed to work also on open terms (since we expect to reason about arbitrary denotations).

Permissive-nominal algebra simplifies and streamlines the nominal algebra logic of [GM09a] (which was based on nominal terms). Essentially, these two logics do the same thing, but there are significant differences which we discuss in Subsection 7.7. Nominal Algebra (NA) was presented in [Gab05; GM06b]; see also [GM07; GM09a]. It was first used to axiomatise substitution, first-order logic, and the λ-calculus [GM06a; GM08a; GM06c; GM08c; GM08b; GM10]. The interest of these papers was not merely to write down the axioms—which all take advantage of atoms-abstraction to axiomatise various binding operators—but also to prove these axioms sound and complete. These proofs are not included here; see the presentations in [GM08a; GM08c; GM10]. Or, to see a much more sophisticated instance of the same general idea, the reader can examine the permissive-nominal logic axiomatisation of arithmetic which is proved correct in the case study of Section 10.

7.1 Judgement form, axioms, theories

DEFINITION 7.1.1. A (nominal algebra) **equality judgement** is a pair $r = s$.

DEFINITION 7.1.2. A **theory** $\mathsf{T} = (\Sigma, Ax)$ is a pair of a signature Σ and a possibly infinite set of *equality* judgements Ax in that signature; we call them the **axioms**.

EXAMPLE 7.1.3. Here are some example nominal algebra theories:

- naSUB axiomatises capture-avoiding substitution (on the λ-calculus).

  Let Σ have a base sort τ and the following term-formers:

  $$\mathsf{sub} : ([\nu]\tau, \tau)\tau \quad \mathsf{lam} : ([\nu]\tau)\tau \quad \mathsf{app} : (\tau, \tau)\tau \quad \mathsf{var} : (\nu)\tau$$

  Axioms are as follows:

$(\mathbf{var}{\mapsto})$	$\mathsf{var}(a)[a{\mapsto}X]$	$= X$	
$(\#{\mapsto})$	$Y[a{\mapsto}X]$	$= Y$	$(a{\notin}supp(Y))$
$(\mathsf{f}{\mapsto})$	$\mathsf{f}(Y)[a{\mapsto}X]$	$= \mathsf{f}(Y[a{\mapsto}X])$	$(\mathsf{f}{\in}\{\mathsf{lam}, \mathsf{app}, \mathsf{var}, \mathsf{sub}\})$
$(\mathbf{tup}{\mapsto})$	$(X_1, \ldots, X_n)[a{\mapsto}X]$	$= (X_1[a{\mapsto}X], \ldots, X_n[a{\mapsto}X])$	
$(\mathbf{abs}{\mapsto})$	$([b]Y)[a{\mapsto}X]$	$= [b](Y[a{\mapsto}X])$	$(a{\notin}supp(Y))$
$(\mathbf{id}{\mapsto})$	$Y[b{\mapsto}\mathsf{var}(b)]$	$= Y$	
$(\eta{\mapsto})$	$[a]\mathsf{sub}(Y, \mathsf{var}(a))$	$= Y$	$(a{\notin}supp(Y))$

  Here and in the next example we sugar $\mathsf{sub}([a]r, t)$ to $r[a{\mapsto}t]$. Every permission set contains b and every permission set contains a except for $supp(Y)$, as indicated above. Sorts are filled in as appropriate.

  naSUB is based on the nominal algebra axioms of [GM06a; GM08a] (which were parameterised over the signature Σ).

  There, we proved the axioms sound and complete for a specific syntactic model in which $Z[a{:=}X]$ really is interpreted as capture-avoiding substitution. The completeness result from Corollary 7.5.12 remains valid but is weaker because it holds not for the specific syntactic model, but the class of all nominal algebra models of the axioms.

- naLAM extends the previous theory with two more axioms:

(β)	$(\lambda[a]Y)X$	$= Y[a{\mapsto}X]$	
(η)	$\lambda[a](Xa)$	$= X$	$(a{\notin}supp(X))$

  This theory is studied in [GM08b; GM10]. Analogously to naSUB, we prove the axioms sound and complete for a syntactic model where substitution *is* substitution and β- and η-conversion *are* β- and η-conversion.

REMARK 7.1.4. Compare and contrast Example 7.1.3 with Example 5.1.3. Clearly, one is an equality theory and another a rewrite theory, but we obtain a nominal algebra theory from Example 5.1.3 by replacing $\to$ by $=$, and conversely we can replace $=$ with $\to$ in Example 7.1.3.

So why are they different? They demonstrate different design priorities.

The rewrites in Example 5.1.3 are designed to operate on ground terms ($fU(r) = \varnothing$), following an intuition that rewriting is about ‘executing programs’. The equalities in Example 7.1.3 are designed to operate on possibly open terms, following an intuition that algebra is about models, not all of whose elements need be referenced by ground terms.

What we gain in deductive power we lose in computational properties. For instance, nrSUB is terminating whereas (an oriented version of) naSUB is not terminating, because explicit substitutions can ‘churn’ by distributing repeatedly over one another

$$\frac{}{r = r}\,(\mathbf{Refl}) \qquad \frac{r = s \quad s = t}{r = t}\,(\mathbf{Trans})$$

$$\frac{r = s}{s = r}\,(\mathbf{Symm}) \qquad \frac{r_1 = r_1' \ \ldots\ r_n = r_n'}{(r_1,\ldots,r_n) = (r_1',\ldots,r_n')}\,(\mathbf{Cong1})$$

$$\frac{r = r'}{\mathsf{f}(r) = \mathsf{f}(r')}\,(\mathbf{Cong2}) \qquad \frac{r = r'}{[a]r = [a]r'}\,(\mathbf{Cong3})$$

$$\frac{((r{=}s){\in}\mathsf{T})}{\pi{\cdot}(r\theta) = \pi{\cdot}(s\theta)}\,(\mathbf{Ax_{r=s}})$$

Figure 4: Derivable entailment in Permissive-Nominal Algebra (PNA)

(this is essentially the idea behind Melliès's counterexample in [Mel95]). On the other hand while the effect of $(\#{\mapsto})$ from naSUB can be obtained on ground terms using the rules in nrSUB, by pushing the substitution down to the atoms, the rules of nrSUB are not deductively powerful enough to do this for open terms (or arbitrary models). More on this in [GM08a; GM10].

7.2 Derivable equality

DEFINITION 7.2.1. Suppose T is a theory. **Derivable equality** $\mathsf{T} \vdash r = s$ is the least transitive reflexive symmetric relation such that for every $(r = s) \in \mathsf{T}$, position P, and substitution θ, if $sort(r) = sort(P)$ and $fa(r\theta) \cup fa(s\theta) \subseteq supp(P)$ (so that $P[l\theta]$ and $P[s\theta]$ are well-defined) then

$$\boxed{\mathsf{T} \vdash P[r\theta] = P[s\theta].}$$

REMARK 7.2.2. Definition 7.2.1 is rather compact; it might be useful to expand it a little. This is Figure 4, given in natural deduction style.

The reader familiar with nominal terms (see for instance Figure 2 of [UPG04]) should note of $(\mathbf{Cong3})$ that we do not need to consider the case $[a]r = [b]s$, because α-equivalence is handled automatically for us by nominal abstract syntax. It is built in by Definition 2.4.8. In other words, thanks to how we set up our permissive-nominal terms syntax, we can always rename abstracted atoms so that they are equal. We noted an analogous point earlier on, in Remark 4.1.6.

LEMMA 7.2.3. Suppose $\mathsf{T} = (\Sigma, Ax)$ is a theory. Then:

- $\mathsf{T} \vdash a = b$ is impossible.
- $\mathsf{T} \vdash [a]r = [b]s$ if and only if $b \notin fa(r)$ and $(b\ a){\cdot}r = s$.
- $\mathsf{T} \vdash (r_1,\ldots,r_n) = (s_1,\ldots,s_n)$ if and only if $\mathsf{T} \vdash r_i = s_i$ for $1 \leq i \leq n$.

Proof. In axiom $(r = s) \in Ax$, r and s must have base sort τ; thus it is not possible to assert equalitities between atoms, abstractions, or tuples (unless wrapped in a term-former and so injected into a base sort). The second part additionally uses Lemma 2.4.9. ■

LEMMA 7.2.4. Suppose $\mathsf{T} \vdash r = s$. Then:

1. $\mathsf{T} \vdash \pi{\cdot}r = \pi{\cdot}s$.
2. $\mathsf{T} \vdash r\theta = s\theta$.

Proof. Both parts are by a routine argument on derivations. We consider one case:

- *The case* $(r' = s') \in \mathsf{T}$ *and* $r = P[r'\theta']$ *and* $s = P[s'\theta']$ *and* $P = (t, X)$. For the first part we use a position $(\pi{\cdot}t, X)$.
  For the second part we consider a position $P' = (t(\theta{-}X), X)$ and consider $P'[r'\theta'\theta]$ and $S'[r'\theta'\theta]$ ($\theta{-}X$ defined in Definition 4.3.3). It is not hard to check that $P'[r'\theta'\theta] = P[r'\theta']\theta$ and $P'[s'\theta'\theta] = P[s'\theta']\theta$, and the result follows.

■

7.3 Interpretation of signatures and terms

DEFINITION 7.3.1. Suppose $(\mathcal{A}, \mathcal{B})$ is a sort-signature (Definition 3.1.1).

An **interpretation** $\mathfrak{I}$ for $(\mathcal{A}, \mathcal{B})$ consists of an assignment of a nonempty permissive-nominal set $[\![\alpha]\!]^{\mathfrak{I}}$ to each sort α in $(\mathcal{A}, \mathcal{B})$, along with equivariant maps

- for each $\nu \in \mathcal{A}$ an equivariant and injective map $\mathbb{A}_\nu \to [\![\nu]\!]^{\mathfrak{I}}$ which we write $a_{\mathfrak{I}}$,
- for each $\nu \in \mathcal{A}$ and α an equivariant and injective map $[\mathbb{A}_\nu][\![\alpha]\!]^{\mathfrak{I}} \to [\![[\nu]\alpha]\!]^{\mathfrak{I}}$ which we write $[a]^{\mathfrak{I}}x$, and
- for each α_i for $1 \leq i \leq n$ an equivariant and injective map $\Pi_i[\![\alpha_i]\!]^{\mathfrak{I}} \to [\![(\alpha_1, \ldots, \alpha_n)]\!]^{\mathfrak{I}}$ which we write $(x_1, \ldots, x_n)^{\mathfrak{I}}$.

DEFINITION 7.3.2. Suppose $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{F}, ar)$ is a signature (Definition 3.1.5).

An **interpretation** $\mathfrak{I}$ for Σ, or Σ**-algebra**, consists of the following data:

- An interpretation for the sort-signature $(\mathcal{A}, \mathcal{B})$ (Definition 7.3.1).
- For every $\mathsf{f} \in \mathcal{F}$ with $ar(\mathsf{f}) = (\alpha)\tau$ an equivariant function $\mathsf{f}^{\mathfrak{I}}$ from $[\![\alpha]\!]^{\mathfrak{I}}$ to $[\![\tau]\!]^{\mathfrak{I}}$.
- An equivariant assignment from $C \in \mathcal{C}$ to $C^{\mathfrak{I}} \in [\![sort(C)]\!]^{\mathfrak{I}}$. (That is, $(\pi{\cdot}C)^{\mathfrak{I}} = \pi{\cdot}(C^{\mathfrak{I}})$.)

DEFINITION 7.3.3. Suppose $\mathfrak{I}$ is a Σ-algebra. A **valuation** ς to $\mathfrak{I}$ is an equivariant function on unknowns $\mathcal{X}$ such that for each unknown X, $\varsigma(X) \in [\![sort(X)]\!]^{\mathfrak{I}}$.

ς will range over valuations.

DEFINITION 7.3.4. Suppose $\mathfrak{I}$ is a Σ-algebra. Suppose ς is a valuation to $\mathfrak{I}$.

Extend $\mathcal{I}$ to an **interpretation** on terms $[\![r]\!]^{\mathcal{I}}_{\varsigma}$ (where of course r is a term in the signature Σ) by:

$$\begin{array}{ll}
[\![a]\!]^{\mathcal{I}}_{\varsigma} = a^{\mathcal{I}} & [\![\mathsf{f}(r)]\!]^{\mathcal{I}}_{\varsigma} = \mathsf{f}^{\mathcal{I}}([\![r]\!]^{\mathcal{I}}_{\varsigma}) \\
[\![C]\!]^{\mathcal{I}}_{\varsigma} = C^{\mathcal{I}} & [\![(r_1,\dots,r_n)]\!]^{\mathcal{I}}_{\varsigma} = ([\![r_1]\!]^{\mathcal{I}}_{\varsigma},\dots,[\![r_n]\!]^{\mathcal{I}}_{\varsigma})^{\mathcal{I}} \\
[\![X]\!]^{\mathcal{I}}_{\varsigma} = \varsigma(X) & [\![[a]r]\!]^{\mathcal{I}}_{\varsigma} = [a]^{\mathcal{I}}[\![r]\!]^{\mathcal{I}}_{\varsigma}
\end{array}$$

Lemma 7.3.5 is a basic sanity check and an important soundness result:

LEMMA 7.3.5. If $r : \alpha$ then $[\![r]\!]^{\mathcal{I}}_{\varsigma} \in [\![\alpha]\!]^{\mathcal{I}}$.

Proof. By a routine induction on r. ■

LEMMA 7.3.6. $\pi\cdot[\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![\pi\cdot r]\!]^{\mathcal{I}}_{\varsigma}$.

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

- *The case X.* By Definition 7.3.4 $[\![X]\!]^{\mathcal{I}}_{\varsigma} = \varsigma(X)$. Therefore $\pi\cdot[\![X]\!]^{\mathcal{I}}_{\varsigma} = \pi\cdot\varsigma(X)$. By assumption $\pi\cdot\varsigma(X) = \varsigma(\pi\cdot X) = [\![\pi\cdot X]\!]^{\mathcal{I}}_{\varsigma}$.

■

LEMMA 7.3.7. $\mathit{supp}([\![r]\!]^{\mathcal{I}}_{\varsigma}) \subseteq \mathit{fa}(r)$.

Proof. From Lemmas 2.3.3 and 7.3.6. ■

7.4 Models and soundness

DEFINITION 7.4.1. For a theory $\mathsf{T} = (\Sigma, Ax)$ and interpretation $\mathcal{I}$ of T call $(r = s)$ **valid** in $\mathcal{I}$ when $[\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![s]\!]^{\mathcal{I}}_{\varsigma}$ for every valuation ς to $\mathcal{I}$.
Call $\mathcal{I}$ a **model** of T when every axiom $(r = s) \in Ax$ is valid in $\mathcal{I}$.
Write $\mathsf{T} \models r = s$ when $(r = s)$ is valid in every model of T.

LEMMA 7.4.2. If $\varsigma(X) = \varsigma'(X)$ for all $X \in \mathit{fU}(r)$ then $[\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![r]\!]^{\mathcal{I}}_{\varsigma'}$.

Proof. By a routine induction on r. ■

DEFINITION 7.4.3. Suppose ς is a valuation to $\mathcal{I}$. Suppose X is an unknown and $x \in [\![\mathit{sort}(X)]\!]^{\mathcal{I}}$ is such that $\mathit{supp}(x) \subseteq \mathit{supp}(X)$. Define a function $\varsigma[X{:=}x]$ by

$$(\varsigma[X{:=}x])(\pi\cdot X) = \pi\cdot x \quad \text{and} \quad (\varsigma[X{:=}x])(Y) = \varsigma(Y) \quad \text{all other } Y$$

LEMMA 7.4.4. $\varsigma[X{:=}x]$ in Definition 7.4.3 is well-defined and a valuation to $\mathcal{I}$.

Proof. As that of Proposition 2.5.5. ■

LEMMA 7.4.5. $[\![r]\!]^{\mathcal{I}}_{\varsigma[X:=[\![t]\!]^{\mathcal{I}}_{\varsigma}]} = [\![r[X{:=}t]]\!]^{\mathcal{I}}_{\varsigma}$. As corollaries we have:

1. If $[\![r]\!]^{\mathfrak{I}}_{\varsigma} = [\![s]\!]^{\mathfrak{I}}_{\varsigma}$ then $[\![P[r]]\!]^{\mathfrak{I}}_{\varsigma} = [\![P[s]]\!]^{\mathfrak{I}}_{\varsigma}$.
2. If $[\![r]\!]^{\mathfrak{I}}_{\varsigma} = [\![s]\!]^{\mathfrak{I}}_{\varsigma}$ then $[\![r\theta]\!]^{\mathfrak{I}}_{\varsigma} = [\![s\theta]\!]^{\mathfrak{I}}_{\varsigma}$.

Proof. By a routine induction on the definition of $[\![r]\!]^{\mathfrak{I}}_{\varsigma}$. We consider one case:

- *The case of* $[\![\pi{\cdot}X]\!]^{\mathfrak{I}}_{\varsigma[X:=t]}$. We reason as follows:

$$\begin{aligned} [\![\pi{\cdot}X]\!]^{\mathfrak{I}}_{\varsigma[X:=[\![t]\!]^{\mathfrak{I}}_{\varsigma}]} &= \pi{\cdot}[\![t]\!]^{\mathfrak{I}}_{\varsigma} && \text{Definition 7.3.4} \\ &= [\![\pi{\cdot}t]\!]^{\mathfrak{I}}_{\varsigma} && \text{Lemma 7.3.6} \\ &= [\![(\pi{\cdot}X)[X{:=}t]]\!]^{\mathfrak{I}}_{\varsigma} && \text{Definition 3.4.8.} \end{aligned}$$

For the two corollaries we reason as follows:

1. By definition where $P = (t, X)$, $P[r] = t[X{:=}r]$ and $P[s] = t[X{:=}s]$. Using the assumptions,

$$[\![t[X{:=}r]]\!]^{\mathfrak{I}}_{\varsigma} = [\![t]\!]^{\mathfrak{I}}_{\varsigma[X:=[\![r]\!]^{\mathfrak{I}}_{\varsigma}]} = [\![t]\!]^{\mathfrak{I}}_{\varsigma[X:=[\![s]\!]^{\mathfrak{I}}_{\varsigma}]} = [\![t[X{:=}s]]\!]^{\mathfrak{I}}_{\varsigma}.$$

2. It is a fact of syntax that $fU(r)$ and $fU(s)$ are finite. Using Lemma 3.4.12 we may represent the effect of θ on r and s as a sequence of atomic substitutions (Definition 3.4.5). The result follows.

■

THEOREM 7.4.6 (Soundness). For any $\mathsf{T} = (\Sigma, Ax)$ if $\mathsf{T} \vdash r = s$ then $\mathsf{T} \models r = s$.

Proof. Let $\mathfrak{I}$ be a model of T and ς be a valuation to $\mathfrak{I}$.

Identity in the denotation is reflexive, transitive, and symmetric so it suffices to check the theorem for axioms. That is, suppose $(r = s) \in Ax$ and assume a position P and substitution θ such that $sort(r) = sort(P)$ and $fa(r\theta) \cup fa(s\theta) \subseteq supp(P)$. We must show that $[\![P[r\theta]]\!]^{\mathfrak{I}}_{\varsigma} = [\![P[s\theta]]\!]^{\mathfrak{I}}_{\varsigma}$.

$\mathfrak{I}$ is a model so $[\![r]\!]^{\mathfrak{I}}_{\varsigma} = [\![s]\!]^{\mathfrak{I}}_{\varsigma}$. We use parts 1 and 2 of Lemma 7.4.5. ■

7.5 Free term models and completeness

In this subsection fix a signature Σ and a theory $\mathsf{T} = (\Sigma, Ax)$.

The proof of completeness follows a standard method: we construct a model out of syntax in which by construction two terms denote equal elements if and only if they are derivably equal.

The subtlety occurs in Lemma 7.5.9. We want to eliminate ς in $[\![r]\!]^{\mathcal{F}(\mathsf{T})}_{\varsigma}$ by converting it into a substitution θ. This 'should' be easy, since for each X, $\varsigma(X)$ is a provably equivalent class of terms. We need only choose some representative term in $\varsigma(X)$ for each X and set $\theta(X)$ to be that representative.

If we are naive in our construction then this could be impossible, as outlined in Example 7.5.10: there might be 'too many atoms' in the available representatives. We enrich our syntax with 'enough' extra constant symbols, to guarantee 'enough' representatives of every element of the model. Nominal algebra without the constant symbols is complete for the same semantics, but the proof would be more complex.

DEFINITION 7.5.1. For each sort α in Σ define $[r]_{\mathsf{T}}$ and $\mathcal{F}(\mathsf{T})_\alpha$ by

$$\begin{aligned} [r]_{\mathsf{T}} &= \{r' : \alpha \mid \mathsf{T} \vdash r = r'\} \qquad (r : \alpha) \\ \mathcal{F}(\mathsf{T})_\alpha &= \{[r]_{\mathsf{T}} \mid r : \alpha\}. \end{aligned}$$

Make each $\mathcal{F}(\mathsf{T})_\alpha$ into a permissive-nominal set by giving it a permutation action

$$\pi\cdot[r]_{\mathsf{T}} = [\pi\cdot r]_{\mathsf{T}}.$$

$\mathcal{F}(\mathsf{T})$ stands for '$\mathcal{F}$ree terms in the signature of T, up to derivable equality in T'. Lemmas 7.5.2 and 7.5.3 relate permutation and support to the natural notions from nominal sets:

LEMMA 7.5.2. The permutation action on $[r]_{\mathsf{T}}$ is pointwise on $[r]_{\mathsf{T}}$ as a set: that is, $\pi\cdot[r]_{\mathsf{T}} = \{\pi\cdot r' \mid r' \in [r]_{\mathsf{T}}\}$.

Proof. From Definition 7.5.1 and Lemma 7.2.4. ■

LEMMA 7.5.3. $\mathit{supp}([r]_{\mathsf{T}}) \subseteq \mathit{fa}(r)$.

Proof. From Definition 7.5.1 and Lemma 2.3.3. ■

DEFINITION 7.5.4. We construct the **free term interpretation** $\mathcal{F}(\mathsf{T})$ of T as follows:

- Take $\mathcal{F}(\mathsf{T})_\alpha$ as in Definition 7.5.1.
- $a^{\mathcal{F}(\mathsf{T})} = [a]_{\mathsf{T}}$, $[a]^{\mathcal{F}(\mathsf{T})}[r]_{\mathsf{T}} = [[a]r]_{\mathsf{T}}$, and $([r_1]_{\mathsf{T}}, \dots, [r_n]_{\mathsf{T}})^{\mathcal{F}(\mathsf{T})} = [(r_1, \dots, r_n)]_{\mathsf{T}}$.
- $\mathsf{f}^{\mathcal{F}(\mathsf{T})}([r]_{\mathsf{T}}) = [\mathsf{f}(r)]_{\mathsf{T}}$ for each term-former $\mathsf{f} : (\alpha)\tau$ in Σ and each $r : \alpha$.
- $C^{\mathcal{F}(\mathsf{T})} = [C]_{\mathsf{T}}$ for each constant in Σ.

LEMMA 7.5.5. Definition 7.5.4 is well-defined and is an interpretation. That is:

- The choice of representative of $[r]_{\mathsf{T}}$ does not matter in any of the clauses.
- The choice of abstracted atom in the clause for $[a]^{\mathcal{F}(\mathsf{T})}[r]_{\mathsf{T}}$ does not matter.
- The maps $a^{\mathcal{F}(\mathsf{T})}$, $[a]^{\mathcal{F}(\mathsf{T})}[r]_{\mathsf{T}}$, and $([r_1]_{\mathsf{T}}, \dots, [r_n]_{\mathsf{T}})^{\mathcal{F}(\mathsf{T})}$ are injective.

Proof. The first part follows by congruence properties of derivable equality. The second part additionally uses Lemmas 2.4.10 and 2.4.11. The third part uses Lemma 7.2.3. ■

DEFINITION 7.5.6. Define a theory $\mathsf{T}^+ = (\Sigma^+, \mathit{Ax}^+)$ to be equal to T except that we adjoin $\bigcup_\tau \mathcal{F}(\mathsf{T})_\tau$ to the set of constants in Σ, and we add axioms equating r with $[r]_{\mathsf{T}}$ in Ax.

That is, for every $r : \tau$ there is a constant $C_r = [r]_{\mathsf{T}} \in \Sigma^+$, and an axiom $(C_r = r) \in \mathcal{F}(\mathsf{T})^+$.

LEMMA 7.5.7. $\mathcal{F}(\mathsf{T})$ extends to an interpretation $\mathcal{F}(\mathsf{T})^+$ of T^+, where for each $r : \tau$ we take $C_r^{\mathcal{F}(\mathsf{T})^+} = [r]_\mathsf{T}$. Furthermore, $\mathcal{F}(\mathsf{T})^+$ is a model of T^+.

DEFINITION 7.5.8. Write ς_{id} for the valuation to $\mathcal{F}(\mathsf{T})$ mapping each X to $C_X = [X]_\mathsf{T}$.

LEMMA 7.5.9. For every valuation ς to $\mathcal{F}(\mathsf{T})$ there exists a substitution θ in T^+ such that $[\![r]\!]_\varsigma^{\mathcal{F}(\mathsf{T})} = [\![r\theta]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})^+}$.

Proof. For each orbit $x \in |orb(\mathcal{X})|$ choose a representative $X \in x$. Define θ by $\theta(\pi{\cdot}X) = \pi{\cdot}C_X$. Recall that $C_X = [X]_\mathsf{T}$ and by Lemma 7.5.3 $supp([X]_\mathsf{T}) \subseteq supp(X)$. By Proposition 2.5.5 θ is well-defined and is a substitution

It is not hard to check by induction on r that $[\![r]\!]_\varsigma^{\mathcal{F}(\mathsf{T})} = [\![r\theta]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})^+}$. ∎

EXAMPLE 7.5.10. To see why Lemma 7.5.9 is non-trivial and how T^+ helps, suppose T has one name sort ν, two base sorts τ and τ', one term-former $\mathsf{abs} : (\nu, \tau)\tau'$, and one axiom $\mathsf{abs}(b, (b\ a){\cdot}X) = \mathsf{abs}(a, X)$ where $a \in supp(X)$ and $b \notin supp(X)$.

Then it is a fact that there is no $r \in [\mathsf{abs}(a, X)]_\mathsf{T}$ such that $fa(r) \subseteq supp([\mathsf{abs}(a, X)]_\mathsf{T})$ and it follows that there is no θ such that $[\![X']\!]^{\mathcal{F}(\mathsf{T})}_{[X' := [\mathsf{abs}(a,X)]_\mathsf{T}]} = [\![X'\theta]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})^+}$ (recall that substitutions must be equivariant).

THEOREM 7.5.11. $\mathcal{F}(\mathsf{T})$ is a model of T.

Proof. We must show that $\mathcal{F}(\mathsf{T})$ validates the axioms.

Suppose $(r = s) \in Ax$. Suppose ς is a valuation to $\mathcal{F}(\mathsf{T})$. We must show that $[\![r]\!]_\varsigma^{\mathcal{F}(\mathsf{T})} = [\![s]\!]_\varsigma^{\mathcal{F}(\mathsf{T})}$.

By Lemma 7.5.9 there exists θ to T^+ such that $[\![r]\!]_\varsigma^{\mathcal{F}(\mathsf{T})} = [\![r\theta]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})^+}$ and $[\![s]\!]_\varsigma^{\mathcal{F}(\mathsf{T})} = [\![s\theta]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})^+}$.

By assumption $\mathsf{T}^+ \vdash r\theta = s\theta$. By Lemma 7.5.7, $[\![r\theta]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})^+} = [\![s\theta]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})^+}$. The result follows. ∎

COROLLARY 7.5.12 (Completeness). If $\mathsf{T} \models r = s$ then $\mathsf{T} \vdash r = s$.

Proof. Suppose $\mathsf{T} \models r = s$. By Theorem 7.5.11 $[\![r]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})} = [\![s]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})}$ (ς_{id} is defined in Definition 7.5.8).

It is not hard to prove by induction that $[\![r]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})} = [r]_\mathsf{T}$ and $[\![s]\!]_{\varsigma_{id}}^{\mathcal{F}(\mathsf{T})} = [s]_\mathsf{T}$. It follows that $\mathsf{T} \vdash r = s$ as required. ∎

7.6 Freshness

Nominal terms freshness conditions $a\#X$ and $a\#r$ from [UPG04] correspond in this paper to 'free atoms of' $a \notin supp(X)$ and $a \notin fa(r)$. See Notation 3.2.7 and Lemma 3.2.5. Call this **syntactic** freshness.

Nominal sets freshness $a \notin supp([\![r]\!])$ is a distinct notion which can be expressed using equality; call this **semantic** freshness. The two are not identical, but they are connected in various ways which we briefly explore.

Proposition 7.6.1 corresponds to Theorem 5.5 from [GM07] and Lemma 4.51 from [GM09a]:

PROPOSITION 7.6.1. Suppose $b \notin fa(r)$.

Then $\mathsf{T} \vdash (b\ a)\cdot r = r$ if and only if for every model $\mathcal{I}$ of T and valuation ς to $\mathcal{I}$, $a \notin \mathit{supp}([\![r]\!]^{\mathcal{I}}_{\varsigma})$.

Proof. By Theorem 7.4.6 and Corollary 7.5.12 $\mathsf{T} \vdash (b\ a)\cdot r = r$ if and only if $\mathsf{T} \models (b\ a)\cdot r = r$, which unpacking definitions means that for every $\mathcal{I}$ and ς, $[\![(b\ a)\cdot r]\!]^{\mathcal{I}}_{\varsigma} = [\![r]\!]^{\mathcal{I}}_{\varsigma}$. By Lemma 7.3.6 $[\![(b\ a)\cdot r]\!]^{\mathcal{I}}_{\varsigma} = (b\ a)\cdot[\![r]\!]^{\mathcal{I}}_{\varsigma}$, and by Lemma 7.3.7 $b \notin \mathit{supp}([\![r]\!]^{\mathcal{I}}_{\varsigma})$. The result follows by Corollary 2.2.7. ■

Lemmas 7.5.3 (and also Lemma 7.3.7) express that syntactic freshness implies semantic freshness. A partial converse is Proposition 7.6.3, which is based on a technical property of nominal sets:

LEMMA 7.6.2. Suppose X is a nominal set and $U \subseteq |\mathsf{X}|$ is finitely-supported (so $U \in |\mathit{pow}(\mathsf{X})|$ from Example 2.2.5) and nonempty.
Then if $a\#U$ then there exists some $x \in U$ with $a\#x$.

Proof. U is nonempty so choose any $x' \in U$. Choose fresh b (so $b \notin \mathit{supp}(U) \cup \mathit{supp}(x')$) and set $x = (b\ a)\cdot x'$. By the definition of support $(b\ a)\cdot U = U$. By the pointwise action (Example 2.2.5) $x \in U$. By Lemma 2.2.6 $a \notin \mathit{supp}(x)$. ■

PROPOSITION 7.6.3. $a\#[r]_{\mathsf{T}}$ implies there exists some r' such that $\mathsf{T} \vdash r = r'$ and $a \notin \mathit{fa}(r')$.

Proof. By Lemmas 7.5.2 and Lemma 7.6.2. ■

7.7 Design of nominal algebra

We designed nominal algebra originally to axiomatise substitution, first-order logic, and the λ-calculus [GM06a; GM06c; GM08a; GM08c; GM09a].

We encountered two design decisions: whether to include freshness axioms, and whether to include atoms-abstraction as primitive.

We disallowed freshness axioms because they are a definitional extension of the system without them, and we chose to include atoms-abstraction as primitive because—even though they too are a definitional extension (see next paragraph)—they make for more compact derivations and proofs and we knew that the reader would expect to see them in a 'nominal' paper. These decisions do not matter for expressivity because of the following two equalities from [GP01], here written in the language of FM sets:

$$\begin{array}{ll} (\mathbf{Freshness}) & a\#x \Leftrightarrow \reflectbox{N}b.(b\ a)\cdot x = x \\ (\mathbf{Abstraction}) & \reflectbox{N}b.([b](b\ a)\cdot x = [a]x) \end{array}$$

In Subsection 11.1 we express the equalities above in PNL. In [Gab12b], we do the same in nominal algebra, showing how to compile nominal algebra with semantic freshness judgements and atoms-abstraction down to the core logic without it.

$\reflectbox{N}$ above is the *new*-quantifier meaning 'for some/any fresh atom' [GP01; Gab11b].[14] $\reflectbox{N}$ does not care which fresh atom we choose (the some/any property [Gab11b, Theorem 6.5]). So, we do not have to be exact about $\mathit{supp}(x)$ when we choose fresh b;

[14]In words: 'a is fresh for x if for some/any fresh b, $(b\ a)\cdot x = x$' and 'for some/any fresh b, $[b](b\ a)\cdot x = [a]x$'.

any will do, and for instance Proposition 7.6.1 is an 'if and only if' even though we chose $b \notin \mathit{fa}(r)$ (syntactic freshness) instead of $b \notin \mathit{supp}([\![r]\!])$ (semantic freshness), and it may be that $\mathit{supp}([\![r]\!]) \subsetneq \mathit{fa}(r)$. More on this Subsection 11.1.

Note that including atoms-abstraction is orthogonal to the rest of the logic in the sense that it is isolated by the sort system: if we provide no term-formers injecting atoms-abstraction into base sorts, then it cannot interact with the rest of the logic.

The permissive-nominal algebra of this paper differs from the nominal algebra of [GM09a] in the following respects:

- The system here is sorted, the system in [GM09a] is not.
- We use permissive-nominal terms and semantics here, and 'vanilla' nominal terms and nominal sets in [GM09a]. That is, the logic here is *permissive*-nominal algebra. Freshness conditions $a\#X$ and $a\#r$ translate to $a \notin \mathit{supp}(X)$ and $a \notin \mathit{fa}(r)$ here.
- Axioms are exactly equalities, with no freshness contexts: permission sets play this role instead.
- The syntax here admits non-equivariant constant symbols, that of [GM09a] does not. That does not matter if we are using finitely-supported models (as is the case in [GM09a]) because finite non-equivariance can be emulated using term-formers applied to finitely many atoms. Here, elements can have infinite support, which cannot be emulated using (finite) equivariant term-formers.
- The syntax here admits the possibility of unknowns with empty support ranging over closed elements (so it includes the $\bullet t$ freshness constraint of [FG07, Subsection 9.2]), unknowns with finite support ranging over finitely-supported elements, unknowns with support equal to a permission set, and whatever else we can imagine in-between.
- The development is parameterised over the set of unknowns $\mathcal{X}$ and *also* the group of permutations $\mathbb{P}$. In particular we admit (but do not insist on) the possibility of infinite permutations, including the *shift*-permutations considered in Subsection 3.6.
- Substitutions and valuations are—rather elegantly—treated as equivariant functions on $\mathcal{X}$ the set of unknowns.

In spite of these many differences, the spirit of the proofs remains the same. The details become simpler, and in particular the non-equivariant constants make construction of the free term model easier.[15]

8 THE NOMINAL HSP THEOREM

The HSP theorem states that a class of Σ-algebras is equational if and only if it is closed under Homomorphism, Subobject, and Product. Definitions follow below, and the main result is Theorem 8.7.3.

The result was first proved for the case of 'ordinary' algebra (using first-order terms and not over nominal sets) by Birkhoff [Bir35]. It is also called *Birkhoff's*

[15] In [GM09a] to build the free term model we enriched syntax with n-ary term-formers applied to atoms. This idea goes back to a completeness proof in [Gab07a].

theorem [BS81, Theorem 11.12]. We prefer 'HSP' since this is more descriptive and Birkhoff's name is attached to several other results.

The result was first proved for nominal algebra by the author [Gab09], and an alternative proof was provided by Kurz and Petrişan [KP10]. The new proof presented here is also rather short.

HSP was interesting for two reasons: first, it is not obvious that nominal algebra is a true logic of equality, because of the freshness side-conditions which give the nominal algebra as presented e.g. in [GM09a] or in Mathijssen's thesis [Mat07] a *prima facie* flavour of conditional equalities. The HSP result holding for nominal algebra was a way of making formal that this *is* a logic of equality.

The use of permission sets to phrase the logic entirely in terms of equality (freshness migrates to the types, as permission sets) is a step forward from this point of view: the nominal algebra of this paper is more *visibly* an equational logic. Still, HSP along with soundness and completeness (Theorem 7.4.6 and Corollary 7.5.12) form a triumvirate of results of interest for an algebraic reasoning framework.

The proofs here are much shorter and clearer than those of [Gab09]—and the final result is strictly stronger than [Gab09; KP10], which actually proved an HSPA theorem that a class of Σ-algebras is equational if and only if it is closed under Homomorphism, Subobject, Product, and Atoms-abstraction.

That is, we have dropped the 'atoms-abstraction' from the closure conditions. How can this be? The use of permission sets gives us finer control over the support of valuations; we needed atoms-abstraction in the proof of [Gab09, Theorem 9.8] to eliminate 'extra' atoms introduced by a valuation ς—'extra' relative to the freshness information in a freshness context Δ. Here, because freshness contexts/permission sets are fixed, this cannot happen.

8.1 Algebra homomorphisms

DEFINITION 8.1.1. Suppose $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$ is a signature and suppose $\mathfrak{X}$ and $\mathfrak{Y}$ are interpretations of Σ. A **Σ-homomorphism** Θ from $\mathfrak{X}$ to $\mathfrak{Y}$ is a family of equivariant functions Θ_α from $[\![\alpha]\!]^{\mathfrak{X}}$ to $[\![\alpha]\!]^{\mathfrak{Y}}$ for each sort α in the sort-signature $(\mathcal{A}, \mathcal{B})$ such that:

- $\Theta_\nu(a^{\mathfrak{X}}) = a^{\mathfrak{Y}}$.
- $\Theta_{(\alpha_1,\dots,\alpha_n)}(x_1,\dots,x_n)^{\mathfrak{X}} = (\Theta_{\alpha_1}(x_1),\dots,\Theta_{\alpha_n}(x_n))^{\mathfrak{Y}}$.
- $\Theta_{[\nu]\alpha}([a]^{\mathfrak{X}}x) = [a]^{\mathfrak{Y}}\Theta_\alpha(x)$.
- $\Theta_\tau(\mathsf{f}^{\mathfrak{X}}(x)) = \mathsf{f}^{\mathfrak{Y}}(\Theta_\alpha(x))$ where $\mathsf{f} : (\alpha)\tau$ is in $\mathcal{F}$.

DEFINITION 8.1.2. Call $\mathfrak{Y}$ a **homomorphic image** of $\mathfrak{X}$ when there is a Σ-homomorphism Θ from $\mathfrak{X}$ to $\mathfrak{Y}$ such that Θ_α is surjective for every sort α in $(\mathcal{A}, \mathcal{B})$.

Call Θ **injective** when Θ_α is injective for every sort α in $(\mathcal{A}, \mathcal{B})$.

LEMMA 8.1.3. Suppose Σ is a signature and $\mathfrak{X}$ and $\mathfrak{Y}$ are Σ-algebras. Suppose Θ is a Σ-algebra homomorphism from $\mathfrak{X}$ to $\mathfrak{Y}$.

Suppose that ς is a valuation to $\mathfrak{X}$. Define $\Theta(\varsigma)$ a valuation to $\mathfrak{Y}$ by $\Theta(\varsigma)(X) = \Theta_{sort(X)}(\varsigma(X))$ for every $X \in \mathcal{X}$.

Then for every $r : \alpha,\ \Theta_\alpha([\![r]\!]^{\mathfrak{X}}_{\varsigma}) = [\![r]\!]^{\mathfrak{Y}}_{\Theta(\varsigma)}$.

Proof. By an easy induction on r. ■

LEMMA 8.1.4. Suppose Σ is a signature and $\mathsf{T} = (\Sigma, Ax)$ is a theory. Suppose $\mathcal{X}$ and $\mathcal{Y}$ are Σ-algebras and $\mathcal{Y}$ is a homomorphic image of $\mathcal{X}$ under Θ.

Then if $\mathcal{X}$ is a model of T, then so is $\mathcal{Y}$.

Proof. Choose $(r = s) \in Ax$ and a valuation ς to $\mathcal{Y}$. It suffices to show that $[\![r]\!]^{\mathcal{Y}}_{\varsigma} = [\![s]\!]^{\mathcal{Y}}_{\varsigma}$.

We construct a valuation ς' to $\mathcal{X}$ as an equivariant extension (Definition 2.5.4) of the following data. For each unknown $X : \alpha$ let $\mathcal{X}_X = \{x \in |\mathcal{X}_\alpha| \mid \Theta(x) = \varsigma(X)\}$. We construct a valuation ς' to $\mathcal{X}$ by for each orbit and representative $X \in orb(X) \in \mathcal{X}$ setting $\varsigma'(X) = x$ for some choice of $x \in \mathcal{X}_X$.

By construction $\Theta\varsigma' = \varsigma$. By assumption $[\![r]\!]^{\mathcal{X}}_{\varsigma'} = [\![s]\!]^{\mathcal{X}}_{\varsigma'}$. We apply Θ to both sides and use Lemma 8.1.3. ■

8.2 Subalgebras

DEFINITION 8.2.1. For Σ-algebras $\mathcal{X}$ and $\mathcal{Y}$, call $\mathcal{X}$ a **subalgebra** of $\mathcal{Y}$ when:

- $|\tau^{\mathcal{X}}| \subseteq |\tau^{\mathcal{Y}}|$ for every $\tau \in \mathcal{B}$.
- The subset inclusion maps form a Σ-algebra homomorphism (Definition 8.1.1).[16]

LEMMA 8.2.2. For Σ-algebras $\mathcal{X}$, $\mathcal{Y}$ and a theory $\mathsf{T} = (\Sigma, Ax)$, if $\mathcal{Y}$ is a model of T and $\mathcal{X}$ is a subalgebra of $\mathcal{Y}$ then $\mathcal{X}$ is a model of T.

8.3 Products

DEFINITION 8.3.1. Let I be a (possibly countably infinite) indexing set and $(\mathcal{X}_i)_{i \in I}$ be an I-indexed collection of Σ-algebras. The **product algebra** $\Pi_{i \in I}\mathcal{X}_i$ is the Σ-algebra such that:

- For each α in Σ, $\alpha^{\Pi_{i\in I}\mathcal{X}_i} = \Pi_{i \in I}\alpha^{\mathcal{X}_i}$ as defined in Definition 2.4.5.
- The ith projection map to $\mathcal{X}_i$ is a Σ-algebra homomorphism for every $i \in I$.

LEMMA 8.3.2. For any I-indexed collection of Σ-algebras $(\mathcal{X}_i)_{i \in I}$, if $\mathcal{X}_i$ is a model of $\mathsf{T} = (\Sigma, Ax)$ for every $i \in I$ then so is $\Pi_{i \in I}\mathcal{X}_i$.

Proof. Suppose $(r = s) \in Ax$. Suppose ς is a valuation to $\Pi_{i \in I}\mathcal{X}_i$. For each $i \in I$ we obtain a valuation ς_i to $\mathcal{X}_i$ by projecting to the ith component. It follows that $[\![r]\!]^{\mathcal{X}_i}_{\varsigma_i} = [\![s]\!]^{\mathcal{X}_i}_{\varsigma_i}$, and thus $[\![r]\!]^{\Pi_{i\in I}\mathcal{X}_i}_{\varsigma} = [\![s]\!]^{\Pi_{i\in I}\mathcal{Y}_i}_{\varsigma}$. ■

[16]That is:

- $a^{\mathcal{X}} = a^{\mathcal{Y}}$ for every atom a.
- $(x_1, \dots, x_n)^{\mathcal{X}} = (x_1, \dots, x_n)^{\mathcal{Y}}$ for every $x_1 \in |[\![\alpha_1]\!]^{\mathcal{X}}|, \dots, x_n \in |[\![\alpha_n]\!]^{\mathcal{Y}}|$.
- $[a]^{\mathcal{X}}x = [a]^{\mathcal{Y}}x$ for every $x \in |[\![\alpha]\!]^{\mathcal{X}}|$.
- For every term-former f in $\mathcal{F}$, $\mathsf{f}^{\mathcal{X}}(x) = \mathsf{f}^{\mathcal{Y}}(x)$ for every $x \in |[\![\alpha]\!]^{\mathcal{X}}|$ where $ar(\mathsf{f}) = (\alpha)\tau$.

8.4 Ground term models and extending a signature

DEFINITION 8.4.1. Call r **ground** when $fU(r) = \varnothing$.

Definition 8.4.2 exactly follows Definition 7.5.1 (cf. Remark 8.4.6):

DEFINITION 8.4.2. Suppose $\mathsf{T} = (\Sigma, Ax)$ is a theory. For each sort α in Σ define $[r]^{gnd}_{\mathsf{T}}$ and $\mathcal{G}(\mathsf{T})_\alpha$ by

$$\boxed{\begin{aligned} [r]^{gnd}_{\mathsf{T}} &= \{r' : \alpha \mid \mathsf{T} \vdash r = r'\} \qquad (r : \alpha, r \text{ ground}) \\ \mathcal{G}(\mathsf{T})_\alpha &= \{[r]^{gnd}_{\mathsf{T}} \mid r : \alpha,\ r \text{ ground}\}. \end{aligned}}$$

Make each $\mathcal{G}(\mathsf{T})_\alpha$ into a permissive-nominal set by giving it a permutation action

$$\pi \cdot [r]^{gnd}_{\mathsf{T}} = [\pi \cdot r]^{gnd}_{\mathsf{T}}.$$

LEMMA 8.4.3. $supp([r]^{gnd}_{\mathsf{T}}) \subseteq fa(r)$.

Proof. From Definition 7.5.1 and Lemma 2.3.3. ■

DEFINITION 8.4.4. We construct the **ground free term interpretation** $\mathcal{G}(\mathsf{T})$ of T as follows:

We take $\mathcal{G}(\mathsf{T})_\alpha$ as in Definition 8.4.2. We define:

$$\begin{aligned} a^{\mathcal{G}(\mathsf{T})} &= [a]^{gnd}_{\mathsf{T}} \\ [a]^{\mathcal{G}(\mathsf{T})}[r]^{gnd}_{\mathsf{T}} &= [[a]r]^{gnd}_{\mathsf{T}} \\ ([r_1]^{gnd}_{\mathsf{T}}, \dots, [r_n]^{gnd}_{\mathsf{T}})^{\mathcal{G}(\mathsf{T})} &= [(r_1, \dots, r_n)]^{gnd}_{\mathsf{T}} \\ \mathsf{f}^{\mathcal{G}(\mathsf{T})}([r]^{gnd}_{\mathsf{T}}) &= [\mathsf{f}(r)]^{gnd}_{\mathsf{T}} \\ C^{\mathcal{G}(\mathsf{T})} &= [C]^{gnd}_{\mathsf{T}} \end{aligned}$$

Above, f ranges over each term-former $\mathsf{f} : (\alpha)\tau$ in Σ and C ranges over each constant in Σ.

LEMMA 8.4.5. Definition 8.4.4 is well-defined and is an interpretation.

Proof. As the proof of Lemma 7.5.5. ■

REMARK 8.4.6. Definition 7.5.1 is a special case of Definition 7.5.1. We obtain $\mathcal{F}(\mathsf{T})$ as $\mathcal{G}(\mathsf{T}')$ where T' is obtained from T by extending its signature with a copy of $\mathcal{X}$ as constants (the construction is made formal in Definition 8.5.2 below).

Doing this in Definition 7.5.1 would have complicated the presentation for no immediate gain, so it seemed kinder on the reader to build the special case first by hand.

Note that we *need* to use ground terms now, for the proof of Theorem 8.5.3 to work. The reason is that $\mathcal{F}(\mathsf{T})$ has elements in each sort given by the elements $[X]_{\mathsf{T}}$, whereas $\mathcal{G}(\mathsf{T})$ lacks these elements.

8.5 Surjective maps onto algebras

Fix a signature Σ and any collection of Σ-algebras $\mathcal{V}$.

DEFINITION 8.5.1. Suppose $\mathsf{T} = (\Sigma, Ax)$ and suppose $\mathcal{X}$ and $\mathcal{Y}_i$ for $i \in I$ are models of T. Suppose $\theta_i \in \mathcal{X} \to \mathcal{Y}_i$ is a family of homomorphisms.

Write $\Pi_i\theta_i$ for the natural map from $\mathcal{X}$ to $\Pi_i\mathcal{Y}_i$, mapping $x \in |\mathcal{X}_\alpha|$ to $(\theta_i(x))_i \in |\Pi_i\mathcal{Y}_i|$.

It is easy to verify that $\Pi_i\theta_i$ is a Σ-algebra homomorphism.

DEFINITION 8.5.2. Suppose Σ and Σ' are signatures. Say Σ' **extends** Σ **with fresh constants** when $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar')$ and $\Sigma' = (\mathcal{A}, \mathcal{B}, \mathcal{C} \cup \mathcal{D}, \mathcal{X}, \mathcal{F}, ar')$ where $\mathcal{D} \cap \mathcal{C} = \varnothing$ and $ar'(C) = ar(C)$ for every $C \in \mathcal{C}$.

THEOREM 8.5.3. Suppose $\mathsf{T} = (\Sigma, Ax)$ is a theory and $\mathcal{V}$ is a model of T. Then there exists a theory $\mathsf{T}' = (\Sigma', Ax)$ where Σ' extends Σ with some fresh constants $\mathcal{D}$ such that $\mathcal{V}$ is a homomorphic image of $\mathcal{G}(\mathsf{T}')$.

Proof. We take $\mathcal{D} = \bigcup_\alpha |\mathcal{V}_\alpha|$ and construct a homomorphism based on mapping $x \in \mathcal{V}_\alpha$ (as a constant in $\mathcal{D}$) to itself (as an element of $|\mathcal{V}_\alpha|$). ■

8.6 Injections out of free algebras

DEFINITION 8.6.1. Suppose Σ is a signature and $\mathcal{V}$ is a set of Σ-algebras. Let $\mathsf{T} = (\Sigma, Ax)$ where Ax is the collection of judgements valid in all $\mathcal{V} \in \mathcal{V}$ for all valuations. Call T the (Σ-)theory **generated by** $\mathcal{V}$.

REMARK 8.6.2. So $(r = s) \in Ax$ in Definition 8.6.1 when for every $\mathcal{V} \in \mathcal{V}$ and every valuation ς to $\mathcal{V}$, it is the case that $[\![r]\!]^{\mathcal{V}}_{\varsigma} = [\![s]\!]^{\mathcal{V}}_{\varsigma}$.

DEFINITION 8.6.3. Define the **constants** of a term $consts(r)$ just as Definition 3.3.2 except that we take $consts(C) = \{orb(C)\}$ and $consts(X) = \varnothing$.

LEMMA 8.6.4. Suppose Σ is a signature and Σ' extends Σ with some fresh constants $\mathcal{D}$. Suppose Σ has enough unknowns (Definition 3.1.10).

If g is a ground term in Σ' then there exists a term g^{-1} in Σ and substitution θ such that $g^{-1}\theta = g$.

Proof. For each orbit in $consts(r)$ choose a representative $C \in orb(C) \in consts(r)$, and some distinct unknown X_C with $sort(X_C) = sort(C)$ and $supp(C) \subseteq supp(X_C)$—we can do this because we have assumed enough unknowns and it is a fact that $consts(r)$ is finite. Define θ to be the equivariant extension of this choice, so $\theta(\pi \cdot X_C) = \pi \cdot C$ and (for all the other unknowns) $\theta(Y) = Y$. This is well-defined by Proposition 2.5.5.

It is now easy to generate g^{-1} by replacing each C in g with X_C (modulo some permutations). ■

THEOREM 8.6.5. Suppose $\mathcal{V}$ is a collection of Σ-algebras and Σ has enough unknowns. Let $\mathsf{T} = (\Sigma, Ax)$ be the Σ-theory generated by $\mathcal{V}$. Suppose Σ' extends Σ with some fresh constants $\mathcal{D}$ and write $\mathsf{T}' = (\Sigma', Ax)$.

Then there exists some indexing set I, set of algebras $\{\mathcal{V}_i \in \mathcal{V} \mid i \in I\}$, and an injective Σ-algebra homomorphism Θ from $\mathcal{G}(\Sigma')$ to $\Pi_{i \in I}\mathcal{V}_i$.

Proof. Take I to be the set of all pairs (g,h) of ground terms in Σ' such that $\mathsf{T}' \not\vdash g = h$.

Consider some $i = (g,h) \in I$. By Lemma 8.6.4 there exist $g^{\text{-}1}$, $h^{\text{-}1}$, and θ_i such that $g^{\text{-}1}\theta_i = g$ and $h^{\text{-}1}\theta_i = h$. We assumed that $\mathsf{T}' \not\vdash g = h$ and it follows using Lemma 7.2.4 that $\mathsf{T} \not\vdash g^{\text{-}1} = h^{\text{-}1}$. Since T is the theory generated by $\mathcal{V}$ there exists a model $\mathcal{V}_i \in \mathcal{V}$ and a valuation ς such that $[\![g^{\text{-}1}]\!]^{\mathcal{V}_i}_{\varsigma} \neq [\![h^{\text{-}1}]\!]^{\mathcal{V}_i}_{\varsigma}$. We define a Σ-homomorphism Θ_i from $\mathcal{G}(\mathsf{T}')$ to $\mathcal{V}_i$ as an equivariant extension of mapping $C \in \mathcal{D}$ to $\varsigma(X_C)$, where C and X_C are as chosen in the proof of Lemma 8.6.4.

It follows by the choice of $\mathcal{V}_i$ that $\Pi_{i\in I}\theta_i$ from $\mathcal{G}(\mathsf{T}')$ to $\Pi_{i\in I}\mathcal{V}_i$ is injective as a map on underlying sets. ■

8.7 Proof of the HSP theorem

We can now prove Theorem 8.7.3; a similar result for nominal algebra is proved in [Gab09].

DEFINITION 8.7.1. Suppose Σ is a signature. Suppose $\mathcal{V}$ is a collection of Σ-algebras. Then:

- Call $\mathcal{V}$ a **(Σ-)variety** when it is closed under Homomorphic image (Definition 8.1.1), Subalgebra (Definition 8.2.1), and countable Product (Definition 8.3.1).
- Call $\mathcal{V}$ **(Σ-)equational** when it is the collection of Σ-algebras that are models of $\mathsf{T} = (\Sigma, Ax)$ for some set of axioms Ax.

LEMMA 8.7.2. Suppose Σ is a signature with enough unknowns. Suppose $\mathcal{V}$ is a Σ-variety and let $\mathsf{T} = (\Sigma, Ax)$ be the Σ-theory generated by $\mathcal{V}$. Suppose Σ' extends Σ with some fresh constants $\mathcal{D}$ and write $\mathsf{T}' = (\Sigma', Ax)$. Then $\mathcal{G}(\mathsf{T}') \in \mathcal{V}$.

Proof. By Theorem 8.6.5 there is some indexing set I, set of Σ-algebras $\{\mathcal{V}_i \in \mathcal{V} \mid i \in I\}$, and injective Σ-algebra homomorphism Θ from $\mathcal{G}(\mathsf{T}')$ to $\Pi_{i\in I}\mathcal{V}_i$. $\mathcal{V}$ is closed under products so $\Pi_{i\in I}\mathcal{V}_i \in \mathcal{V}$. The image of $|\mathcal{G}(\mathsf{T}')|$ is a subalgebra of $\Pi_{i\in I}\mathcal{V}_i$, and is a homomorphic image of that subalgebra (by inverting Θ). $\mathcal{V}$ is closed under subalgebras and homomorphic images, so the result follows. ■

THEOREM 8.7.3. Suppose Σ is a signature with enough unknowns. A collection of Σ-algebras $\mathcal{V}$ is equational if and only if it is a variety

Proof. Suppose $\mathcal{V}$ is equational. By Lemma 8.3.2 $\mathcal{V}$ is closed under products. By Lemma 8.1.4 $\mathcal{V}$ is closed under homomorphic images. By Lemma 8.2.2 $\mathcal{V}$ is closed under subalgebras. Therefore $\mathcal{V}$ is a variety.

Conversely, suppose $\mathcal{V}$ is a variety. Let T be the theory on Σ generated by $\mathcal{V}$ as described in Definition 8.6.1. Let $\mathcal{V}$ be any model of T. By Theorem 8.5.3 there exists a signature Σ' extending Σ with some fresh constants $\mathcal{D}$ such that $\mathcal{V}$ is a homomorphic image of $\mathcal{G}(\mathsf{T}')$. By Lemma 8.7.2 $\mathcal{G}(\mathsf{T}') \in \mathcal{V}$. Since $\mathcal{V}$ is closed under homomorphisms, $\mathcal{V} \in \mathcal{V}$ as required. Therefore $\mathcal{V}$ is equational. ■

PERMISSIVE-NOMINAL LOGIC: $\forall X$

9 PERMISSIVE-NOMINAL LOGIC

We now add quantification over unknowns—that is, $\forall X$—to permissive-nominal terms. Permissive nominal techniques makes the theory of α-equivalence easy here: if we used ‘vanilla’ nominal terms, then the development below might not be impossible, but we believe that it would be harder. We obtain a first-order logic which we call *permissive-nominal logic*.

Permissive-nominal logic (PNL) is just first-order logic, enriched with nominal-style names. Thus, the derivation rules in Figure 5 are (virtually) identical to those of first-order logic. Only the term language is really changed.

In this section we set up PNL as a sequent derivation system (Figure 5), interpret it in permissive-nominal sets (Definition 9.3.2), and prove soundness and completeness (Theorems 9.3.6 and 9.4.16).

In Section 10 we undertake as an extended case study a sound and complete finite axiomatisation of arithmetic inside PNL.

9.1 Syntax

The notions of sort-signature $(\mathcal{A}, \mathcal{B})$ and sort language are as in Definition 3.1.1. An interpretation $\mathcal{I}$ for $(\mathcal{A}, \mathcal{B})$ consists of an assignment of a permissive-nominal set $\tau^{\mathcal{I}}$ to each $\tau \in \mathcal{B}$, and we extend $\mathcal{I}$ to sorts as in Definition 7.3.1.

DEFINITION 9.1.1. For this section it is convenient to take $\mathcal{X}$ to be specifically example 2 of Example 3.1.7.

REMARK 9.1.2. So an unknown X takes the form

$$\pi\cdot\mathtt{X}_\alpha = \{(\pi', \mathtt{X}_\alpha) \mid \forall a{\in}\mathbb{A}^{<}.\pi(a) = \pi'(a)\}.$$

In this case, in the light of Remark 3.1.8, we may take $fU(r)$ to be equal to the set of $\mathtt{X}_\alpha$ occurring in r.

It is now easy to define binding of unknowns (level 2 variables) in terms. A more abstract account of level 2 binding is also available [Gab11c].

DEFINITION 9.1.3. A **PNL signature** over a sort-signature $(\mathcal{A}, \mathcal{B})$ is a tuple $(\mathcal{C}, \mathcal{F}, \mathcal{P}, ar)$ where:

- $\mathcal{C}$ is a permissive-nominal set of **constants**.
- $\mathcal{F}$ is a set of equivariant **term-formers**.
- $\mathcal{P}$ is a set of equivariant **predicate-formers**.
- ar assigns
  - to each constant $C \in \mathcal{C}$ an arity τ,
  - to each $\mathsf{f} \in \mathcal{F}$ a **term-former arity** $(\alpha)\tau$, and

– to each $\mathsf{P} \in \mathcal{P}$ a **proposition-former arity** α, where

α and τ are in the sort-language determined by $(\mathcal{A}, \mathcal{B})$.

A **(PNL) signature** $\mathcal{S}$ is then a tuple $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, ar)$.

DEFINITION 9.1.4. Suppose $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, ar)$ is a PNL signature.

Terms are defined as in Definition 3.2.1 for the signature $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$.[17]

Propositions are defined as follows:

$$\frac{}{\bot \text{ proposition}} \qquad \frac{\phi \text{ proposition} \quad \psi \text{ proposition}}{\phi \Rightarrow \psi \text{ proposition}}$$

$$\frac{r : \alpha \quad (ar(\mathsf{P}) = \alpha)}{\mathsf{P}(r) \text{ proposition}} \qquad \frac{\phi \text{ proposition}}{\forall \mathtt{X}_\alpha.\phi \text{ proposition}}$$

Here $\forall \mathtt{X}_\alpha$ binds $\mathtt{X}_\alpha$ in ϕ. We can use nominal abstract syntax to do this.

NOTATION 9.1.5. Write $\forall X.\phi$ as shorthand for $\forall \mathtt{X}_\alpha.\phi$ where $X = \{(\pi', \mathtt{X}_\alpha) \mid \forall a{\in}\mathbb{A}^<.\pi'(a) = \pi(a)\}$ for some π.

LEMMA 9.1.6. Support and the permutation action as characterised for terms on Lemma 3.2.5 extend to propositions as follows:

$$\begin{array}{cc} supp(\bot) = \varnothing & supp(\mathsf{P}(r)) = supp(r) \\ supp(\phi \Rightarrow \psi) = supp(\phi) \cup supp(\psi) & supp(\forall X.\phi) = supp(\phi) \end{array}$$

$$\begin{array}{cc} \pi{\cdot}\bot = \bot & \pi{\cdot}\mathsf{P}(r) = \mathsf{P}(\pi{\cdot}r) \\ \pi{\cdot}(\phi \Rightarrow \psi) = (\pi{\cdot}\phi) \Rightarrow \pi{\cdot}\psi & \pi{\cdot}\forall X.\phi = \forall X.\pi{\cdot}\phi \end{array}$$

NOTATION 9.1.7. We may write $fa(\phi)$ for $supp(\phi)$.

9.2 Derivability

DEFINITION 9.2.1. Φ and Ψ will range over sets of propositions. We may write ϕ, Φ and Φ, ϕ as shorthand for $\{\phi\} \cup \Phi$. We may write Φ, Ψ as shorthand for $\Phi \cup \Psi$.

Write $fU(\Phi) = \bigcup\{fU(\phi) \mid \phi \in \Phi\}$.

A **sequent** is a pair $\Phi \vdash \Psi$.

DEFINITION 9.2.2 (Derivable sequents). The **derivable sequents** are defined in Figure 5.

REMARK 9.2.3. As standard, the intuition of $\Phi \vdash \Psi$ being derivable is "the conjunction (logical and) of the propositions in Φ entails the disjunction (logical or) of

[17]The reader who would answer 'Can you pass the salt?' with 'Yes.' should note that we have to adjust ar to remove $\mathcal{P}$ and add $\mathcal{X}$ mapping X to α where $(\pi, \mathtt{X}_\alpha) \in X$.

$$\frac{}{\Phi,\ \phi \vdash \pi{\cdot}\phi,\ \Psi}\ (\mathbf{Ax}) \qquad \frac{}{\Phi,\ \bot \vdash \Psi}\ (\bot\mathbf{L})$$

$$\frac{\Phi \vdash \phi,\ \Psi \quad \Phi,\ \psi \vdash \Psi}{\Phi,\ \phi \Rightarrow \psi \vdash \Psi}\ (\Rightarrow\mathbf{L}) \qquad \frac{\Phi,\ \phi \vdash \psi,\ \Psi}{\Phi \vdash \phi \Rightarrow \psi,\ \Psi}\ (\Rightarrow\mathbf{R})$$

$$\frac{\begin{array}{c}\Phi,\ \phi[X{:=}r] \vdash \Psi \\ (\mathit{fa}(r){\subseteq}\mathit{supp}(X),\ r{:}\mathit{sort}(X))\end{array}}{\Phi,\ \forall X.\phi \vdash \Psi}\ (\forall\mathbf{L}) \qquad \frac{\Phi \vdash \phi,\ \Psi \quad (X \notin \mathit{fU}(\Phi,\Psi))}{\Phi \vdash \forall X.\phi,\ \Psi}\ (\forall\mathbf{R})$$

$$\frac{\Phi \vdash \phi,\ \Psi \qquad \Phi,\ \phi \vdash \Psi}{\Phi \vdash \Psi}\ (\mathbf{Cut})$$

Figure 5: Sequent derivation rules of Permissive-Nominal Logic

the propositions in Ψ". So for instance, intuitively the axiom rule $(\mathbf{Ax})$ expresses that ϕ if and only if $\pi{\cdot}\phi$.

The permutation π in $(\mathbf{Ax})$ is deliberate and represents equivariance (preservation of truth under permuting atoms) within permissive-nominal logic. Because of this permutation π, free atoms can behave like variables ranging over distinct atoms.

Thus in PNL we can express a theory of atoms-inequality as follows: Suppose a name sort Atm and a proposition-former $\mathsf{neq} : (\mathsf{Atm}, \mathsf{Atm})$, along with a single proposition $\mathsf{neq}(a, b)$ for two distinct atoms in Atm—and, if we wish, another proposition $\mathsf{neq}(a, a) \Rightarrow \bot$. The permutation π in $(\mathbf{Ax})$ ensures that a and b represent *any* two distinct atoms.

REMARK 9.2.4. The condition $\mathit{fa}(r) \subseteq \mathit{supp}(X)$ in $(\forall\mathbf{L})$ might suggest that $\forall X.\phi$ means "$\phi[X{:=}r]$ for every r such that $\mathit{fa}(r) \subseteq \mathit{supp}(X)$". This is so, but the π in $(\mathbf{Ax})$ means that what $\mathit{supp}(X)$ in $\forall X.\phi$ really restricts is *capture*, as we now discuss.

- Suppose a name sort Atm and suppose $X : \mathsf{Atm}$ and $\mathsf{P} : \mathsf{Atm}$. Suppose $b \in \mathit{pmss}(X)$. By considering the swapping $(b\ a)$ and $(\mathbf{Ax})$, and $(\forall\mathbf{L})$, $\forall X.\mathsf{P}(X) \vdash \mathsf{P}(b)$ for *all* a, even if $a \notin \mathit{supp}(X)$, as follows:

$$\frac{\dfrac{}{\mathsf{P}(b) \vdash \mathsf{P}(a)}\ (\mathbf{Ax}) \quad \pi = (b\ a)}{\forall X.\mathsf{P}(X) \vdash \mathsf{P}(a)}\ (\forall\mathbf{L}) \quad [X{:=}b]$$

  So we can derive $\mathsf{P}(a)$ from $\forall X.\mathsf{P}(X)$, even if a is not permitted in X.

- This may not work if we have to 'shift' infinitely many atoms; e.g. there is no finite π such that $\mathit{fa}(\pi{\cdot}(X, a)) \subseteq \mathit{supp}(X)$ where $a \notin \mathit{supp}(X)$. If $\mathbb{P}$ has *shift*-permutations (Definition 3.6.1), then we can use them.

  Consider any sort α and suppose $X : \alpha$ and $\mathit{supp}(X) = S$. Suppose $\mathsf{Q} : \alpha$. Consider any other $Y : \alpha$ with $\mathit{supp}(Y) = S \cup \{a\}$ where $a \notin S$. We will show that given *shift*-permutations, $\forall X.\mathsf{Q}(X) \vdash \mathsf{Q}(Y)$ is derivable.

Suppose $S \cup \{a\} = \pi{\cdot}S$. We derive as follows:

$$\dfrac{\dfrac{}{\mathsf{Q}(\pi{\cdot}Y) \vdash \mathsf{Q}(Y)}\,(\mathbf{Ax})}{\forall X.\mathsf{Q}(X) \vdash \mathsf{Q}(Y)}\,(\forall\mathbf{L}) \quad [X{:=}\pi{\cdot}Y]$$

- Nevertheless, $\forall X.\phi$ does not mean "$\phi[X{:=}r]$ for every r". This is because permutations are bijective. For example, suppose $X : \mathsf{Atm}$, $a \notin \mathit{supp}(X)$, and $\mathsf{P} : ([\mathsf{Atm}]\mathsf{Atm})$. Then $\forall X.\mathsf{P}([a]X) \vdash P([a]r)$ for all r such that $a \notin \mathit{fa}(r)$, and also $\forall X.\mathsf{P}([b]X) \vdash P([b]r)$ for all r and all b such that $b \notin \mathit{fa}(r)$. However,

$$\forall X.\mathsf{P}([a]X) \not\vdash P([a]a), \quad \text{and for all } b,\ \forall X.\mathsf{P}([a]X) \not\vdash P([b]b).$$

The fact that $a \notin \mathit{supp}(X)$ forbids capture by an instantiation, in a suitable sense.

9.3 Interpretation and soundness

DEFINITION 9.3.1. Suppose $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, \mathit{ar})$ is a signature.
A **(PNL) interpretation** $\mathcal{J}$ for $\mathcal{S}$ consists of the following data:

- An interpretation for the sort-signature $(\mathcal{A}, \mathcal{B})$ (Definition 7.3.1).
- For every $\mathsf{f} \in \mathcal{F}$ with $\mathit{ar}(\mathsf{f}) = (\alpha')\alpha$ an equivariant function $\mathsf{f}^{\mathcal{J}}$ from $[\![\alpha']\!]^{\mathcal{J}}$ to $[\![\alpha]\!]^{\mathcal{J}}$ (Definition 2.3.1).
- For every $\mathsf{P} \in \mathcal{P}$ with $\mathit{ar}(\mathsf{P}) = \alpha$ an equivariant function $\mathsf{P}^{\mathcal{J}}$ from $[\![\alpha]\!]^{\mathcal{J}}$ to $\{0, 1\}$ (Definition 2.4.1).

DEFINITION 9.3.2. Suppose that $\mathcal{J}$ is an interpretation. Define an **interpretation of propositions** by:

$$\boxed{\begin{aligned}
[\![\mathsf{P}(r)]\!]^{\mathcal{J}}_{\varsigma} &= \mathsf{P}^{\mathcal{J}}([\![r]\!]^{\mathcal{J}}_{\varsigma})\\
[\![\bot]\!]^{\mathcal{J}}_{\varsigma} &= 0\\
[\![\phi \Rightarrow \psi]\!]^{\mathcal{J}}_{\varsigma} &= \mathit{max}\{1{-}[\![\phi]\!]^{\mathcal{J}}_{\varsigma}, [\![\psi]\!]^{\mathcal{J}}_{\varsigma}\}\\
[\![\forall X.\phi]\!]^{\mathcal{J}}_{\varsigma} &= \mathit{min}\{[\![\phi]\!]^{\mathcal{J}}_{\varsigma[X:=x]} \mid x{\in}[\![\mathit{sort}(X)]\!]^{\mathcal{J}},\ \mathit{supp}(x){\subseteq}\mathit{supp}(X)\}
\end{aligned}}$$

LEMMA 9.3.3. $[\![\phi]\!]^{\mathcal{J}}_{\varsigma} = [\![\pi{\cdot}\phi]\!]^{\mathcal{J}}_{\varsigma}$ always.

Proof. By induction on ϕ. We consider two cases:

- The case $\forall X.\phi$. Suppose $[\![\forall X.\phi]\!]^{\mathcal{J}}_{\varsigma} = 1$. This means that $[\![\phi]\!]^{\mathcal{J}}_{\varsigma[X:=x]} = 1$ for all $x \in [\![\alpha]\!]^{\mathcal{J}}$ such that $\mathit{supp}(x) \subseteq \mathit{supp}(X)$. By inductive hypothesis $[\![\pi{\cdot}\phi]\!]^{\mathcal{J}}_{\varsigma[X:=x]} = 1$ for all $x \in [\![\alpha]\!]^{\mathcal{J}}$ such that $\mathit{supp}(x) \subseteq \mathit{supp}(X)$. Therefore $[\![\forall X.\pi{\cdot}\phi]\!]^{\mathcal{J}}_{\varsigma} = 1$. The result follows, since $\pi{\cdot}(\forall X.\phi) = \forall X.\pi{\cdot}\phi$.

- The case $\mathsf{P}(r)$. We have $[\![\mathsf{P}(r)]\!]^{\mathcal{J}}_{\varsigma} = \mathsf{P}^{\mathcal{J}}([\![r]\!]^{\mathcal{J}}_{\varsigma})$. As $\mathsf{P}^{\mathcal{J}}$ is equivariant, we get $[\![\mathsf{P}(r)]\!]^{\mathcal{J}}_{\varsigma} = \mathsf{P}^{\mathcal{J}}(\pi{\cdot}[\![r]\!]^{\mathcal{J}}_{\varsigma})$. By Lemma 7.3.6 $\pi{\cdot}[\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![\pi{\cdot}r]\!]^{\mathcal{J}}_{\varsigma}$. Thus $[\![\mathsf{P}(r)]\!]^{\mathcal{J}}_{\varsigma} = \mathsf{P}^{\mathcal{J}}([\![\pi{\cdot}r]\!]^{\mathcal{J}}_{\varsigma}) = [\![\pi{\cdot}\mathsf{P}(r)]\!]^{\mathcal{J}}_{\varsigma}$.

■

LEMMA 9.3.4. $[\![\phi]\!]^{\mathfrak{I}}_{\varsigma[X:=[\![t]\!]^{\mathfrak{I}}_{\varsigma}]} = [\![\phi[X:=t]]\!]^{\mathfrak{I}}_{\varsigma}$.

Proof. By a routine induction on the definition of $[\![\phi]\!]^{\mathfrak{I}}_{\varsigma}$ in Definition 9.3.2. We consider one case:

- The case of $[\![\mathsf{P}(r)]\!]^{\mathfrak{I}}_{\varsigma[X:=t]}$. We reason as follows:

$$\begin{aligned} [\![\mathsf{P}(r)]\!]^{\mathfrak{I}}_{\varsigma[X:=[\![t]\!]^{\mathfrak{I}}_{\varsigma}]} &= \mathsf{P}^{\mathfrak{I}}([\![r]\!]^{\mathfrak{I}}_{\varsigma[X:=[\![t]\!]^{\mathfrak{I}}_{\varsigma}]}) && \text{Definition 9.3.2} \\ &= \mathsf{P}^{\mathfrak{I}}([\![r[X:=t]]\!]^{\mathfrak{I}}_{\varsigma}) && \text{Lemma 7.4.5} \\ &= [\![\mathsf{P}(r)[X:=t]]\!]^{\mathfrak{I}}_{\varsigma} && \text{Definition 9.3.2.} \end{aligned}$$

■

Validity and soundness

DEFINITION 9.3.5 (Validity). Call the proposition ϕ **valid** in $\mathfrak{I}$ when $[\![\phi]\!]^{\mathfrak{I}}_{\varsigma} = 1$ for all ς.

Call the sequent $\phi_1, \dots, \phi_n \vdash \psi_1, \dots, \psi_p$ **valid** in $\mathfrak{I}$ when $(\phi_1 \wedge \cdots \wedge \phi_n) \Rightarrow (\psi_1 \vee \cdots \vee \psi_p)$ is valid.

THEOREM 9.3.6 (Soundness). If $\Phi \vdash \Psi$ is derivable, then it is valid in all interpretations.

Proof. By induction on derivations (Figure 5). The case of $(\mathbf{Ax})$ uses Lemma 9.3.3. The case of $(\forall\mathbf{L})$ uses Lemma 9.3.4. The case of $(\forall\mathbf{R})$ uses Lemma 7.4.2. Other rules are routine by unpacking definitions. ■

9.4 Completeness

In this subsection we prove Theorem 9.4.16: a converse to Theorem 9.3.6, that if ϕ is valid in all interpretations, then ϕ it is derivable.

For this subsection fix the following data:

- A signature $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, ar)$.
- A formula ϕ such that $\not\vdash \phi$.

We will construct an interpretation $\mathfrak{I}$ and a valuation ς (Definition 7.3.1) such that $[\![\phi]\!]^{\mathfrak{I}}_{\varsigma} = 0$. This suffices to prove the result.

Maximally consistent set of propositions

DEFINITION 9.4.1. Make a fixed but arbitrary order on propositions $\xi_1, \xi_2, \xi_3, \dots$

Define $\Phi_1 = \{\neg\phi\}$ (where ϕ was fixed above). For each $i \geq 1$ we define Φ_{i+1} as follows:

- If $\Phi_i \vdash \xi_i$ then write $\xi = \xi_i$.
- If $\Phi_i \vdash \neg\xi_i$ then write $\xi = \neg\xi_i$.
- If $\Phi_i \not\vdash \xi_i$ and $\Phi_i \not\vdash \neg\xi_i$ then write $\xi = \xi_i$.

There are now two cases:

- If ξ has the form $\neg\forall X.\xi'$ then we define $\Phi_{i+1} = \Phi_i \cup \{\xi, \neg\xi'[X{:=}Z]\}$ where Z is some fixed but arbitrary choice of unknown that is not free in any proposition in Φ_i and is such that $\mathit{supp}(Z) = \mathit{supp}(X)$ and $\mathit{sort}(Z) = \mathit{sort}(X)$.
- Otherwise, we define $\Phi_{i+1} = \Phi_i \cup \{\xi\}$.

Finally, we define Φ_ω by $\Phi_\omega = \bigcup_i \Phi_i$.

LEMMA 9.4.2. For every i, $\Phi_i \nvdash \bot$.

Proof. By induction on i:

- By definition $\Phi_1 = \{\neg\phi\}$. As $\nvdash \phi$, we have $\neg\phi \nvdash \bot$
- Suppose $\Phi_i \nvdash \bot$.
  Either $\Phi_{i+1} = \Phi_i \cup \{\neg\xi\}$ or $\Phi_{i+1} = \Phi_i \cup \{\neg\xi, \neg\xi'[X{:=}Z]\}$—we consider the first, simpler case; the second case is similar. Suppose $\Phi_i, \xi \vdash \bot$. It follows that $\Phi_i \vdash \neg\xi$. So we are not in the third case of Definition 9.4.1 and we are either in the first or the second. So $\Phi_i \vdash \xi$ and thus $\Phi_i \vdash \bot$; a contradiction.

■

LEMMA 9.4.3. $\Phi_\omega \nvdash \bot$.

Proof. Assume $\Phi_\omega \vdash \bot$. So there exists a finite subset Γ of Φ_ω such that $\Gamma \vdash \bot$. As Γ is finite it is included in some Φ_i, and $\Phi_i \vdash \bot$, contradicting Proposition 9.4.2. ■

REMARK 9.4.4. It is well-known that in nominal sets, least upper bounds can sometimes not exist if there are 'too many' atoms; so sometimes we have to be careful to not make too many arbitrary choices.[18]

The reader familiar with nominal techniques will be alert to the possibility that Φ_ω might fail to have a supporting permission set, and that this could cause problems. In fact, in this particular case this is a non-issue: $(\mathbf{Ax})$ from Figure 5 ensures that Φ_ω is not only supported, but in fact equivariant.

LEMMA 9.4.5. For every ξ, at least one of $\xi \in \Phi_\omega$ and $\neg\xi \in \Phi_\omega$ holds.

Proof. We check of Definition 9.4.1 that for every i, either $\xi_i \in \Phi_{i+1}$ or $\neg\xi_i \in \Phi_{i+1}$. The result follows. ■

LEMMA 9.4.6. For every ξ, if $\neg\forall X.\xi \in \Phi_\omega$ then there exists a Z such that $\neg\xi[X{:=}Z] \in \Phi_\omega$.

Proof. There exists an i such that $\xi_i = \neg\forall X.\xi$. Since $\Phi_\omega \vdash \xi_i$ and $\Phi_\omega \nvdash \bot$, we have that $\Phi_\omega \nvdash \neg\xi_i$, and so $\Phi_i \nvdash \neg\xi_i$. Thus $\Phi_{i+1} = \Phi_i \cup \{\neg\forall X.\xi,\ \neg\xi[X{:=}Z]\}$. The result follows. ■

LEMMA 9.4.7. If $\Phi_\omega \vdash \phi$ then $\phi \in \Phi_\omega$.

Proof. As, by Lemma 9.4.3, $\Phi_\omega \nvdash \bot$, if $\Phi_\omega \vdash \phi$ then $\neg\phi \notin \Phi_\omega$. Thus by Lemma 9.4.5, $\phi \in \Phi_\omega$. ■

[18]For instance, in permissive-nominal sets it is possible represent a well-order of each permission set, but not to represent a well-ordering of the set of all atoms (which is a limit of permission sets). This is also the feature which Fraenkel and Mostowksi used to prove the independence of the axiom of choice from the other axioms of set theory.

COROLLARY 9.4.8.

1. $(\phi \Rightarrow \psi) \in \Phi_\omega$ if and only if $(\phi \notin \Phi_\omega$ or $\psi \in \Phi_\omega)$.
2. $\forall X.\phi \in \Phi_\omega$ if and only if
(for every r such that $r : \mathit{sort}(X)$ and $\mathit{fa}(r) \subseteq \mathit{supp}(X)$, $\phi[X{:=}r] \in \Phi_\omega$).

Proof.

1. Suppose $(\phi \Rightarrow \psi) \in \Phi_\omega$ and $\phi \in \Phi_\omega$. Then $\Phi_\omega \vdash \psi$ and so by Lemma 9.4.7 $\psi \in \Phi_\omega$.

   Suppose $\phi \notin \Phi_\omega$. By Lemma 9.4.5 $\neg\phi \in \Phi_\omega$. So $\Phi_\omega \vdash \neg\phi$ and therefore $\Phi_\omega \vdash \phi \Rightarrow \psi$. By Lemma 9.4.7 $(\phi \Rightarrow \psi) \in \Phi_\omega$.

   Suppose $\psi \in \Phi_\omega$. Then $\Phi_\omega \vdash \psi$ and so $\Phi_\omega \vdash \phi \Rightarrow \psi$. By Lemma 9.4.7 $(\phi \Rightarrow \psi) \in \Phi_\omega$.

2. Suppose $\forall X.\phi \in \Phi_\omega$. By Lemma 9.4.7, if $r : \mathit{sort}(X)$ and $\mathit{fa}(r) \subseteq \mathit{supp}(X)$ then $\phi[X{:=}r] \in \Phi_\omega$.

   Conversely, suppose $\phi[X{:=}r] \in \Phi_\omega$ for every r such that $r : \mathit{sort}(X)$ and $\mathit{fa}(r) \subseteq \mathit{supp}(X)$. We proceed by contradiction: suppose $\forall X.\phi \notin \Phi_\omega$. By Lemma 9.4.5 $\neg\forall X.\phi \in \Phi_\omega$ and by Lemma 9.4.6, there exists a Z such that $\neg\phi[X{:=}Z] \in \Phi_\omega$. So $\Phi_\omega \vdash \neg\phi[X{:=}Z]$, and so $\Phi_\omega \vdash \phi[X{:=}Z]$, and so $\Phi_\omega \vdash \bot$, contradicting Lemma 9.4.3.

■

The term interpretation

DEFINITION 9.4.9. Define the **term interpretation** $\mathcal{I}$ by:

- $[\![\alpha]\!]^{\mathcal{I}} = \{r \mid r : \alpha\}$.
- $\mathsf{f}^{\mathcal{I}}$ maps r to $\mathsf{f}(r)$.
- $\mathsf{P}^{\mathcal{I}}$ maps $r_1, \dots, r_n$ to 1 if $\mathsf{P}(r_1, \dots, r_n) \in \Phi_\omega$ and to 0 otherwise.

Define ς by $\varsigma(X) = X$ for all $X \in \mathcal{X}$ and endow $[\![\alpha]\!]^{\mathcal{I}}$ with a permutation action given by the action on terms.

REMARK 9.4.10. In Definition 7.5.4 we built an interpretation to prove completeness of nominal algebra (Corollary 7.5.12). There, we built our interpretation out of terms quotiented by derivable equality; here we just use terms. Why the difference?

In nominal algebra the judgement-form of the logic *is* equality—so it makes sense to build an interpretation such that equality maps to denotational identity.

LEMMA 9.4.11.

1. If $ar(\mathsf{f}) = (\alpha)\tau$ then $\mathsf{f}^{\mathcal{J}}$ is well-defined, equivariant, and maps $\llbracket\alpha\rrbracket^{\mathcal{J}}$ to $\llbracket\tau\rrbracket^{\mathcal{J}}$.
2. If $ar(\mathsf{P}) = \alpha$ then $\mathsf{P}^{\mathcal{J}}$ is well-defined, equivariant, and maps $\llbracket\alpha\rrbracket^{\mathcal{J}}$ to $\{0, 1\}$.

PROPOSITION 9.4.12. $\mathcal{J}$ is an interpretation of the signature $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{P}, ar)$ which we fixed at the beginning of this subsection. In addition, ς is a valuation to $\mathcal{J}$.

Proof. By Lemma 9.4.11 for each $\mathsf{f} : (\alpha')\alpha \in \mathcal{F}$, $\mathsf{f}^{\mathcal{J}}$ is an equivariant map from $\llbracket\alpha'\rrbracket^{\mathcal{J}}$ to $\llbracket\alpha\rrbracket^{\mathcal{J}}$ and for each $\mathsf{P} : \alpha \in \mathcal{P}$, $\mathsf{P}^{\mathcal{J}}$ is an equivariant function from $\llbracket\alpha\rrbracket^{\mathcal{J}}$ to $\{0, 1\}$.

By construction $\varsigma(X) \in \llbracket sort(X)\rrbracket^{\mathcal{J}}$ always. Equivariance is easy. ∎

LEMMA 9.4.13. $\llbracket r\rrbracket^{\mathcal{J}}_{\varsigma} = r$.

LEMMA 9.4.14. $\llbracket\xi\rrbracket^{\mathcal{J}}_{\varsigma} = 1$ if and only if $\xi \in \Phi_\omega$.

Proof. By induction on the definition of $\llbracket\xi\rrbracket^{\mathcal{J}}_{\varsigma}$ (Definition 9.3.2):

- The case of $\llbracket\mathsf{P}(r)\rrbracket^{\mathcal{J}}_{\varsigma}$. We reason as follows:

$$\begin{array}{rcll}
\llbracket\mathsf{P}(r)\rrbracket^{\mathcal{J}}_{\varsigma} = 1 & \Leftrightarrow & \mathsf{P}^{\mathcal{J}}(\llbracket r\rrbracket^{\mathcal{J}}_{\varsigma}) = 1 & \text{Definition 9.3.2} \\
& \Leftrightarrow & \mathsf{P}^{\mathcal{J}}(r) = 1 & \text{Lemma 9.4.13} \\
& \Leftrightarrow & \mathsf{P}(r) \in \Phi_\omega & \text{Definition 9.4.9}
\end{array}$$

- The case of $\llbracket\bot\rrbracket^{\mathcal{J}}_{\varsigma}$. By definition $\llbracket\bot\rrbracket^{\mathcal{J}}_{\varsigma} = 0$. By part 1 of Corollary 9.4.8, $\bot \notin \Phi_\omega$.
- The case of $\llbracket\phi \Rightarrow \psi\rrbracket^{\mathcal{J}}_{\varsigma}$. We reason as follows:

$$\begin{array}{rcll}
\llbracket\phi \Rightarrow \psi\rrbracket^{\mathcal{J}}_{\varsigma} = 1 & \Leftrightarrow & \llbracket\phi\rrbracket^{\mathcal{J}}_{\varsigma} = 0 \text{ or } \llbracket\psi\rrbracket^{\mathcal{J}}_{\varsigma} = 1 & \text{Definition 9.3.2} \\
& \Leftrightarrow & \phi \notin \Phi_\omega \text{ or } \psi \in \Phi_\omega & \text{ind. hyp.} \\
& \Leftrightarrow & \phi \Rightarrow \psi \in \Phi_\omega & \text{Cor. 9.4.8, part 2}
\end{array}$$

- The case of $\llbracket\forall X.\phi\rrbracket^{\mathcal{J}}_{\varsigma}$, where $\alpha = sort(X)$ and $S = supp(X)$.

$$\begin{array}{rcll}
\llbracket\forall X.\phi\rrbracket^{\mathcal{J}}_{\varsigma}{=}1 & \Leftrightarrow & \forall t{\in}\llbracket\alpha\rrbracket^{\mathcal{J}}.supp(t){\subseteq}S \Rightarrow \llbracket\phi\rrbracket^{\mathcal{J}}_{\varsigma[X:=t]}{=}1 & \text{Definition 9.3.2} \\
& \Leftrightarrow & \forall t{\in}\llbracket\alpha\rrbracket^{\mathcal{J}}.supp(t){\subseteq}S \Rightarrow \llbracket\phi[X{:=}t]\rrbracket^{\mathcal{J}}_{\varsigma} = 1 & \text{Lems. 7.4.2, 9.4.13} \\
& \Leftrightarrow & \llbracket\phi[X{:=}t]\rrbracket^{\mathcal{J}}_{\varsigma} = 1 \text{ every } t{:}\alpha \text{ s.t. } fa(t){\subseteq}S & supp(t) = fa(t) \\
& \Leftrightarrow & \phi[X{:=}t]{\in}\Phi_\omega \text{ every } t{:}\alpha \text{ s.t. } fa(t){\subseteq}S & \text{ind. hyp.} \\
& \Leftrightarrow & \forall X.\phi{\in}\Phi_\omega & \text{Cor. 9.4.8, part 4}
\end{array}$$

∎

LEMMA 9.4.15. If $\nvdash \phi$, then there exists an interpretation $\mathcal{J}$ and a valuation ς such that $\llbracket\phi\rrbracket^{\mathcal{J}}_{\varsigma} = 0$.

Proof. As $\neg\phi{\in}\Phi_0{\subseteq}\Phi_\omega$ and $\Phi_\omega{\nvdash}\bot$, we have $\phi{\notin}\Phi_\omega$. By Lemma 9.4.14, we get $\llbracket\phi\rrbracket^{\mathcal{J}}_{\varsigma} = 0$. ∎

As a corollary we get Theorem 9.4.16:

THEOREM 9.4.16 (Completeness). If ϕ is valid in all interpretations, then ϕ is derivable.

10 CASE STUDY: ARITHMETIC IN PERMISSIVE-NOMINAL LOGIC

Because term-formers in PNL can bind, we can axiomatise first-order logic. Thus assume a sort o whose terms reflect formulas of first-order logic. Then PNL quantification $\forall Z$ where $Z : o$ has the quality of an *axiom schema*, and we can use those terms to axiomatise arithmetic (a theory which in first-order logic famously involves an axiom schema).

So, we should be able to use PNL to give a finite, first-order axiomatisation of arithmetic. Writing down some plausible-looking axioms is one thing—proving they do what we expect them to do, is another. In this section, as a case study of an application of PNL, we do just that.

We proceed as as follows, starting with the following PNL definitions:

- Figure 6 gives $\dot{\mathcal{L}}$ a signature for a shallow embedding of terms and formulas of first-order logic as PNL terms of sort ι and o respectively.
- Figure 7 gives equality axioms, as a transitive reflexive symmetric congruence for the term-formers in $\dot{\mathcal{L}}$.
- Figure 8 axiomatises substitution. We can have some confidence in this axiomatisation because it was already considered for nominal algebra in [GM08a] and proven correct.
- Figure 9 gives axioms for first-order logic.
- Finally, Figure 10 gives axioms for arithmetic. As discussed above, the induction axiom schema is captured using a universal quantification (the $\forall Z$ in $(\mathbf{PInd})$).

Subsection 10.4 briefly recalls the syntax and derivability relation of 'real' first-order logic. Then Subsection 10.5 maps this into the PNL theory we just constructed. Subsection 10.6 briefly recalls Peano arithmetic in the 'real' first-order logic.

Finally, in Subsection 10.7 by arguments on models we show our main result of this section: Theorem 10.7.7. A formula is derivable in 'real' Peano arithmetic if and only if its translation in PNL is derivable in the PNL theory for arithmetic.

The permissive-nominal terms, PNL, permission-sets, and permissive-nominal sets semantics, all work together, and at the end of it all we really *can* embed a non-trivial theory with binding in PNL, and know it is correct.

10.1 The signature $\dot{\mathcal{L}}$ and the axioms

DEFINITION 10.1.1. A signature $\dot{\mathcal{L}}$ suitable for writing out a PNL theory of first-order logic is given in Figure 6.

NOTATION 10.1.2. We introduce the following syntactic sugar:

- We may write $\mathsf{sub}_o([a]r, t)$ as $r[a\mapsto t]$.
- We may write $\mathsf{sub}_\iota([a]r, t)$ as $r[a\mapsto t]$.
- We may write both $\approx_\iota$ and $\approx_o$ just as $\approx$.

Examples of this in use, follow immediately below.

We assume one atomic sort ν and two base sorts ι and o.
We assume term-formers and proposition-formers as follows:

$\dot{0} : \iota$	$\mathsf{s\dot{u}cc} : (\iota)\iota$	$\dot{+} : (\iota,\iota)\iota$	$\dot{*} : (\iota,\iota)\iota$
$\dot{\bot} : o$	$\dot{\Rightarrow} : (o,o)o$	$\dot{\forall} : ([\nu]o)o$	$\dot{\approx} : (\iota,\iota)o$
$\mathsf{var} : (\nu)\iota$	$\mathsf{sub}_\iota : ([\nu]\iota,\iota)\iota$	$\mathsf{sub}_o : ([\nu]o,\iota)o$	
$\approx_\iota : (\iota,\iota)$	$\approx_o : (o,o)$	$\epsilon : (o)$	

Figure 6: Signature $\dot{\mathcal{L}}$ suitable for a PNL specification of arithmetic

$$\begin{array}{llcl}
(\approx\mathbf{2}) & \forall X',X,Y',Y.(X'{\approx}X \wedge Y'{\approx}Y) & \Rightarrow & \big(\, X' \dot{+} Y' \approx X \dot{+} Y \;\wedge \\
 & & & X' \dot{*} Y' \approx X \dot{*} Y \;\wedge \\
 & & & X' \dot{\Rightarrow} Y' \approx X \dot{\Rightarrow} Y \;\wedge \\
 & & & X' \dot{\approx} Y' \approx X \dot{\approx} Y \big) \\
(\approx\mathbf{1}) & \forall X',X. \qquad X'{\approx}X & \Rightarrow & \mathsf{s\dot{u}cc}(X') \approx \mathsf{s\dot{u}cc}(X) \\
(\approx\mathbf{0}) & \forall X. & & X \approx X \\
(\approx\dot{\forall}) & \forall Z',Z. \qquad Z'{\approx}Z & \Rightarrow & \dot{\forall}([a]Z') \approx \dot{\forall}([a]Z) \\
(\approx\mathsf{sub}) & \forall X',X,Y',Y.(X'{\approx}X \wedge Y'{\approx}Y) & \Rightarrow & \big(\, \mathsf{sub}_\iota([a]X',Y') \approx \mathsf{sub}_\iota([a]X,Y) \;\wedge \\
 & & & \mathsf{sub}_o([a]X',Y') \approx \mathsf{sub}_o([a]X,Y)\big) \\
(\approx o) & \forall Z',Z. \qquad Z'{\approx}Z & \Rightarrow & (\epsilon(Z') \Leftrightarrow \epsilon(Z)) \\
(\approx\iota) & \forall X',X. \qquad X'{\approx}X & \Rightarrow & \epsilon(X' \dot{\approx} X)
\end{array}$$

We fill in sorts as appropriate. Thus, $\dot{\bot} \approx_o \dot{\bot}$ whereas $0 \approx_\iota 0$, and so on. The permission sets of all unknowns are equal to $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Figure 7: EQU: axioms for equality as a PNL theory

10.2 *The axioms: equality, substitution, first-order logic, and arithmetic*

Equality

Axioms for equality $\approx : (\iota,\iota)$ and equality $\approx : (o,o)$ are given in Figure 7.

Substitution

Axioms for substitution sub_ι and sub_o are given in Figure 8.

We arguably abuse notation in Figure 8 when we use unknowns of sort ι and o as appropriate not necessarily giving them distinct names (e.g. in $(\mathbf{sub}{*})$ X has sort ι, whereas in $(\mathbf{sub}\dot{\Rightarrow})$ we use another unknown also written X with sort o).

First-order logic

Axioms for (a shallow reflection of) first-order formulas as terms in PNL (the $\dot{\bot}$, $\dot{\Rightarrow}$, and $\dot{\forall}$) are given in Figure 9.

$$\begin{array}{lrl}
(\mathbf{sub}\mathsf{var}) & \forall X.\ \mathsf{var}(a)[a{\mapsto}X] & \approx X \\
(\mathbf{sub}\#) & \forall X, Z.\ Z[a{\mapsto}X] & \approx Z \\
(\mathbf{sub}\mathsf{s\dot{u}cc}) & \forall X', X.\ \mathsf{s\dot{u}cc}(X')[a{\mapsto}X] & \approx \mathsf{s\dot{u}cc}(X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{+}) & \forall X'', X', X.\ (X'' \mathbin{\dot{+}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathbin{\dot{+}} X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{*}) & \forall X'', X', X.\ (X'' \mathbin{\dot{*}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathbin{\dot{*}} X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{\Rightarrow}) & \forall X'', X', X.\ (X'' \mathbin{\dot{\Rightarrow}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathbin{\dot{\Rightarrow}} X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{\approx}) & \forall X'', X', X.\ (X'' \mathbin{\dot{\approx}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathbin{\dot{\approx}} X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{\forall}) & \forall X, Z.\ (\dot{\forall}([b]Z))[a{\mapsto}X] & \approx \dot{\forall}([b](Z[a{\mapsto}X])) \\
(\mathbf{subid}) & \forall X.\ X[a{\mapsto}\mathsf{var}(a)] & \approx X
\end{array}$$

$a \in \mathbb{A}^<$ and $b \notin \mathbb{A}^<$. The permission set of X'', X', and X is equal to $\mathbb{A}^<$. The permission set of Z is equal to $(b\ a)\cdot\mathbb{A}^<$.

Figure 8: SUB: axioms for substitution as a PNL theory

$$\begin{array}{lrl}
(\dot{\Rightarrow}) & \forall Z', Z.\ \epsilon(Z' \mathbin{\dot{\Rightarrow}} Z) & \Leftrightarrow (\epsilon(Z') \Rightarrow \epsilon(Z)) \\
(\dot{\forall}) & \forall Z.\ \big(\epsilon(\dot{\forall}([a]Z)) & \Leftrightarrow \forall X.\epsilon(Z[a{\mapsto}X])\big) \\
(\dot{\bot}) & \epsilon(\dot{\bot}) & \Rightarrow \bot
\end{array}$$

Here Z' and Z have sort o, permission set $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Figure 9: FOL: axioms for first-order formulas as a PNL theory

$$\begin{array}{lll}
(\mathbf{PS0}) & \forall X. & \mathsf{s\dot{u}cc}(X) \approx \dot{0} \Rightarrow \bot \\
(\mathbf{PSS}) & \forall X', X. & \mathsf{s\dot{u}cc}(X') \approx \mathsf{s\dot{u}cc}(X) \Rightarrow X' \approx X \\
(\mathbf{P{+}0}) & \forall X. & X \dot{+} \dot{0} \approx X \\
(\mathbf{P{+}succ}) & \forall X', X. & X' \dot{+} \mathsf{s\dot{u}cc}(X) \approx \mathsf{s\dot{u}cc}(X') \dot{+} X \\
(\mathbf{P{*}0}) & \forall X. & X \dot{*} \dot{0} \approx \dot{0} \\
(\mathbf{P{*}succ}) & \forall X', X. & X' \dot{*} \mathsf{s\dot{u}cc}(X) \approx (X' \dot{*} X) \dot{+} X \\
(\mathbf{PInd}) & \multicolumn{2}{l}{\forall Z.\epsilon(Z[a{\mapsto}\dot{0}]) \Rightarrow} \\
 & \multicolumn{2}{l}{\quad\big(\forall X.(\epsilon(Z[a{\mapsto}X]) \Rightarrow \epsilon(Z[a{\mapsto}\mathsf{s\dot{u}cc}(X)]))\big) \Rightarrow} \\
 & \multicolumn{2}{l}{\qquad \forall X.\epsilon(Z[a{\mapsto}X])}
\end{array}$$

The permission set of X, X', and Z is $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Figure 10: ARITH: axioms for arithmetic as a PNL theory

Arithmetic

Given EQU, SUB, and FOL, it is not hard to write axioms for arithmetic in PNL. This is in Figure 10. Later on in Theorem 10.7.7 we prove that this *is* an axiomatisation of arithmetic in PNL.

10.3 Comments on the axioms

REMARK 10.3.1. SUB is a PNL rendering of the nominal algebra theory naSUB from Example 7.1.3; the universal quantifiers which are implicit in the nominal algebraic judgement-form are made explicit here. This is essentially the same axiomatisation as studied in [GM06a; GM08a]. Soundness and completeness are proved, so providing some formal sense in which the axioms of SUB are ‘right’.

In [GM08c] first-order logic is equationally axiomatised using nominal algebra (so the axioms involve only equality). Because PNL is already a first-order logic, we can use $\bot$, $\Rightarrow$, and $\forall$ directly to capture the behaviour of $\dot{\bot}$, $\dot{\Rightarrow}$, and $\dot{\forall}$. So note that FOL here is *not* the axiomatisation of [GM08c]; there we had to work a little harder because the ambient logic, nominal algebra, was purely equational.

REMARK 10.3.2. Instead of the axioms for equality EQU, we could directly extend PNL by adding derivation rules Figure 5 as follows:

$$\frac{\Phi,\, r \approx s,\, \phi[X{:=}r],\, \phi[X{:=}s] \vdash \Psi \qquad (\mathit{fa}(r) \cup \mathit{fa}(s) \subseteq \mathit{supp}(X))}{\Phi,\, r \approx s,\, \phi[X{:=}r] \vdash \Psi}\,(\approx\mathbf{S})$$

$$\frac{\Phi,\, r \approx r \vdash \Psi}{\Phi \vdash \Psi}\,(\approx\mathbf{R})$$

REMARK 10.3.3. Every unknown has a sort, and a permission set.

Different choices of permission set may yield logically equivalent results. For example, in $(\mathbf{sub}\mathsf{lam})$ it is not vital that $\mathit{supp}(Z)$ is *exactly* $(b\ a)\cdot\mathbb{A}^{<}$. The important point is that $a \notin \mathit{supp}(Z)$.

Similarly, in $(\mathbf{sub}\mathsf{app})$ it is not vital that $\mathit{supp}(X'') = \mathit{supp}(X')$; when we use the axiom we can instantiate X'' and X' to r'' and r' such that $\mathit{fa}(r'') \neq \mathit{fa}(r')$, and conversely if we take $\mathit{supp}(X'') \neq \mathit{supp}(X')$ then we can still instantiate X'' and X' to r'' and r' such that $\mathit{fa}(r'') = \mathit{fa}(r') \subseteq \mathit{supp}(X'') \cap \mathit{supp}(X')$.

10.4 First-order logic $\mathcal{L}$

We will use the atoms $\mathbb{A}_\nu$ from $\dot{\mathcal{L}}$ in Section 10 as variables of our first-order logic (this is not necessary, but it is convenient). So for this section, $a, b, c, \ldots$ will range over distinct atoms in $\mathbb{A}_\nu$.

DEFINITION 10.4.1. Define **terms** and **formulas** of $\mathcal{L}$ by:

$$\boxed{\begin{array}{l} t ::= a \mid 0 \mid \mathit{succ}(t) \mid t + t \mid t * t \\ \xi ::= t \approx t \mid \bot \mid \xi \Rightarrow \xi \mid \forall a.\xi \end{array}}$$

Substitution $t'[a{:=}t]$ and $\xi[a{:=}t]$ is as usual for first-order logic. We write sequents $\Xi \vdash X$ where Ξ and X are sets of formulas. Derivability is as usual for first-order logic.

DEFINITION 10.4.2. Define a mapping $(\text{-})^{\bullet}$ from terms and formulas of $\mathcal{L}$ to terms of $\dot{\mathcal{L}}$ (Subsection 10.1) by:

$$\boxed{\begin{array}{rlrl} (a)^{\bullet} &= a & (0)^{\bullet} &= \dot{0} \\ (\mathit{succ}(t))^{\bullet} &= \mathsf{s\dot{u}cc}((t)^{\bullet}) & (t' + t)^{\bullet} &= (t')^{\bullet} \mathbin{\dot{+}} (t)^{\bullet} \\ (t' * t)^{\bullet} &= (t')^{\bullet} \mathbin{\dot{*}} (t)^{\bullet} & & \\ (t' \approx t)^{\bullet} &= (t')^{\bullet} \mathbin{\dot{\approx}} (t)^{\bullet} & (\bot)^{\bullet} &= \dot{\bot} \\ (\xi' \Rightarrow \xi)^{\bullet} &= (\xi')^{\bullet} \mathbin{\dot{\Rightarrow}} (\xi)^{\bullet} & (\forall a.\xi)^{\bullet} &= \dot{\forall}[a](\xi)^{\bullet} \end{array}}$$

DEFINITION 10.4.3. Extend $(\text{-})^\bullet$ to first-order logic sequents $\Xi \vdash \mathcal{X}$ as follows:

$$\boxed{(\Xi \vdash \mathcal{X})^\bullet = \epsilon(\dot{\forall}[a_1]\ldots\dot{\forall}[a_n]((\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l))^\bullet)}$$

Here, $\Xi = \{\xi_1, \ldots, \xi_k\}$, $\mathcal{X} = \{\chi_1, \ldots, \chi_l\}$, and the free variables of Ξ and $\mathcal{X}$ are $\{a_1, \ldots, a_n\}$ (in some order).

NOTATION 10.4.4. Write S for $\mathsf{EQU} \cup \mathsf{SUB} \cup \mathsf{FOL}$.

LEMMA 10.4.5. $\mathsf{S} \vdash (t'[a{:=}t])^\bullet \approx (t')^\bullet[a{\mapsto}(t)^\bullet]$ and
$\mathsf{S} \vdash (\xi[a{:=}t])^\bullet \approx (\xi)^\bullet[a{\mapsto}(t)^\bullet]$.

Proof. By routine inductions on t and ξ. ■

THEOREM 10.4.6 (Correctness). If $\Xi \vdash \mathcal{X}$ is derivable in first-order logic then $\mathsf{S} \vdash (\Xi \vdash \mathcal{X})^\bullet$ is derivable in PNL.

Proof. By a long but routine inspection we can check that EQU, SUB, and FOL allow us to model the behaviour of 'real' first-order logic. We use Lemma 10.4.5. ■

10.5 Interpretation of first-order logic

We recall the usual definition of interpretations in first-order logic:

DEFINITION 10.5.1. A nominal **(first-order logic) interpretation** $\mathcal{M}$ is a **carrier set** M, and elements:

$0^{\mathcal{M}} \in M$, $\quad succ^{\mathcal{M}} \in M \to M$,
$+^{\mathcal{M}} \in (M \times M) \to M$, and $\quad *^{\mathcal{M}} \in (M \times M) \to M$.

It is convenient to fix some $\mathcal{M}$ from here until Theorem 10.7.7.

DEFINITION 10.5.2. Define $Valu_{\mathbb{A}_\nu}(M)$ by:

$$Valu_{\mathbb{A}_\nu}(M) = \{\varepsilon \in \mathbb{A}_\nu \to M \mid \exists A \subseteq \mathbb{A}_\nu.A \text{ finite } \wedge\ \forall a, b \notin A.\varepsilon(a) = \varepsilon(b)\}$$

Call elements of $Valu_{\mathbb{A}_\nu}(M)$ $\mathbb{A}_\nu$**-valuations** (to M). ε will range over $\mathbb{A}_\nu$-valuations.

If $x \in M$ write $\varepsilon[a{:=}x]$ for the valuation mapping b to $\varepsilon(b)$ and mapping a to x:

$$\begin{aligned} \varepsilon[a{:=}x](a) &= x \\ \varepsilon[a{:=}x](b) &= \varepsilon(b) \end{aligned}$$

Give $\varepsilon \in Valu_{\mathbb{A}_\nu}(M)$ and $X \subseteq Valu_{\mathbb{A}_\nu}(M)$ a **pointwise** permutation action:

$$\begin{aligned} (\pi\cdot\varepsilon)(a) &= \varepsilon(\pi^{-1}(a)). \\ \pi\cdot X &= \{\pi\cdot\varepsilon \mid \varepsilon \in X\}. \end{aligned}$$

U, V will range over *finitely-supported* subsets of $Valu_{\mathbb{A}_\nu}(M)$—so there exists some finite $A \subseteq \mathbb{A}_\nu$ such that for all π, if $\pi(a) = a$ for all $a \in A$ then $\pi\cdot U = U$.

$$\begin{array}{lll}
(\mathbf{ps0}) & \forall a. & succ(a) \approx 0 \Rightarrow \bot \\
(\mathbf{pss}) & \forall a', a. & succ(a) \approx succ(a') \Rightarrow a \approx a' \\
(\mathbf{p{+}0}) & \forall a. & a{+}0 \approx a \\
(\mathbf{p{+}succ}) & \forall a', a. & a'{+}succ(a) \approx succ(a'){+}a \\
(\mathbf{p{*}0}) & \forall a. & a{*}0 \approx 0 \\
(\mathbf{p{*}succ}) & \forall a', a. & a'{*}succ(a) \approx (a'{*}a){+}a
\end{array}$$

$$(\mathbf{pind}) \qquad \xi[a{:=}0] \Rightarrow \big(\forall a.(\xi \Rightarrow \xi[a{:=}succ(a)])\big) \Rightarrow \forall a.\xi \qquad \text{(every } \xi\text{, every } a\text{)}$$

Figure 11: arithmetic: axioms for arithmetic in first-order logic

REMARK 10.5.3. $Valu_{\mathbb{A}_\nu}(M)$ would normally just be called ‘the set of valuations’. We are more specific since we separately also have valuations on unknowns X (Definition 7.3.3).

PNL atoms are serving as variable symbols of $\mathcal{L}$. To conveniently apply nominal techniques, it is useful to restrict to valuations that are finite in the sense given in Definition 10.5.2. In any case, any term or formula will only contain finitely many atoms.

DEFINITION 10.5.4. We extend the interpretation to first-order logic syntax as follows:

$$\begin{aligned}
[\![a]\!]^{\mathcal{M}}_{\varepsilon} &= \varepsilon(a) \\
[\![0]\!]^{\mathcal{M}}_{\varepsilon} &= 0^{\mathcal{M}} \\
[\![succ(t)]\!]^{\mathcal{M}}_{\varepsilon} &= succ^{\mathcal{M}}([\![t]\!]^{\mathcal{M}}_{\varepsilon}) \\
[\![t' + t]\!]^{\mathcal{M}}_{\varepsilon} &= +^{\mathcal{M}}([\![t']\!]^{\mathcal{M}}_{\varepsilon}, [\![t]\!]^{\mathcal{M}}_{\varepsilon}) \\
[\![t' * t]\!]^{\mathcal{M}}_{\varepsilon} &= *^{\mathcal{M}}([\![t']\!]^{\mathcal{M}}_{\varepsilon}, [\![t]\!]^{\mathcal{M}}_{\varepsilon}) \\
[\![\bot]\!]^{\mathcal{M}}_{\varepsilon} &= 0 \\
[\![\xi' \Rightarrow \xi]\!]^{\mathcal{M}}_{\varepsilon} &= max\{1{-}[\![\xi']\!]^{\mathcal{M}}_{\varepsilon}, [\![\xi]\!]^{\mathcal{M}}_{\varepsilon}\} \\
[\![\forall a.\xi]\!]^{\mathcal{M}}_{\varepsilon} &= min\{[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a:=x]} \mid x \in M\} \\
[\![t' \approx t]\!]^{\mathcal{M}}_{\varepsilon} &= 1 \text{ if } [\![t']\!]^{\mathcal{M}}_{\varepsilon} = [\![t]\!]^{\mathcal{M}}_{\varepsilon} \text{ and } 0 \text{ otherwise}
\end{aligned}$$

DEFINITION 10.5.5. Call the formula ξ **valid** in $\mathcal{M}$ when $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon} = 1$ for all ε.

Call $\xi_1, \dots, \xi_k \vdash \chi_1, \dots, \chi_l$ **valid** in $\mathcal{M}$ when $(\xi_1 \wedge \dots \wedge \xi_k) \Rightarrow (\chi_1 \vee \dots \vee \chi_l)$ is valid.

10.6 A theory of arithmetic in $\mathcal{L}$

DEFINITION 10.6.1. Define a first-order theory of **arithmetic** by the axioms in Figure 11.
An interpretation $\mathcal{M}$ is a **model** of arithmetic when $[\![\xi]\!]^{\mathcal{M}} = 1$ for ξ each of $(\mathbf{ps0})$, $(\mathbf{pss})$, $(\mathbf{p{+}0})$, $(\mathbf{p{+}succ})$, $(\mathbf{p{*}0})$, $(\mathbf{p{*}succ})$, and every instance of $(\mathbf{pind})$.

REMARK 10.6.2. $(\mathbf{pind})$ the induction axiom-scheme is of course of particular interest. We therefore unpack what its validity

$$[\![\xi[a{:=}0] \Rightarrow \forall a.(\xi \Rightarrow \xi[a{:=}succ(a)]) \Rightarrow \forall a.\xi]\!]^{\mathcal{M}} = 1 \qquad \text{(every } \xi\text{, every } a\text{)}$$

means, in a little more detail. For every a and ξ:

- If $[\![\xi[a{:=}0]]\!]^{\mathcal{M}}_{\varepsilon} = 1$, and
- if for every $x \in M$, $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a:=x]} = 1$ implies that $[\![\xi[a{:=}succ(a)]]\!]^{\mathcal{M}}_{\varepsilon[a:=x]} = 1$,
- then for every $x \in M$, $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a:=x]} = 1$.

In $(\mathbf{pind})$ we take 'every a', and in $(\mathbf{PInd})$ we do not. This is because in $(\mathbf{PInd})$, a is α-convertible,

10.7 Building an interpretation for $\dot{\mathcal{L}}$ from one for $\mathcal{L}$

Recall the PNL signature $\dot{\mathcal{L}}$ from Subsection 10.1. Suppose $\mathcal{M}$ is a model of arithmetic. We use it to build an interpretation $\mathcal{N}$ of $\dot{\mathcal{L}}$.

DEFINITION 10.7.1. Extend $\mathcal{L}$ to $\mathcal{L}{+}M$ where we add all elements of M as constants, and extend the interpretation to interpret these constants as themselves in M. (So if $x \in M$ then x is a constant symbol in $\mathcal{L}{+}M$ and $[\![x]\!]^{\mathcal{M}}_{\varepsilon} = x$.)

Define an $\mathbb{A}_{\nu}$-valuation $\varepsilon_0 \in \mathit{Valu}_{\mathbb{A}_{\nu}}(M)$ by

$$\varepsilon_0(a) = 0^{\mathcal{M}} \quad \text{always.}$$

If t is a term, we write $[\![t]\!]^{\mathcal{M}}$ for the function $\lambda\varepsilon.[\![t]\!]^{\mathcal{M}}_{\varepsilon}$. If ξ is a formula, we write $[\![\xi]\!]^{\mathcal{M}}$ for the function $\lambda\varepsilon.[\![\xi]\!]^{\mathcal{M}}_{\varepsilon}$.

We now define an interpretation $\mathcal{N}$ for $\dot{\mathcal{L}}$. We give a denotation to the base sorts ι and o of $\dot{\mathcal{L}}$, as follows:

$$\boxed{\begin{array}{c} \iota^{\mathcal{N}} = \{[\![t]\!]^{\mathcal{M}} \mid t \text{ a term of } \mathcal{L}{+}M\} \\ o^{\mathcal{N}} = \{[\![\xi]\!]^{\mathcal{M}} \mid \xi \text{ a formula of } \mathcal{L}{+}M\} \end{array}}$$

We give a denotation to the term-formers and proposition-formers of $\dot{\mathcal{L}}$, as follows:

$$\boxed{\begin{array}{rr} \mathsf{var}^{\mathcal{N}}\, a\,\varepsilon = \varepsilon(a) & \mathsf{sub}_o^{\mathcal{N}}\,([a]u, v)\,\varepsilon = u(\varepsilon[a{:=}v\varepsilon]) \\ \dot{0}^{\mathcal{N}}\,\varepsilon = 0^{\mathcal{M}} & \dot{\Rightarrow}^{\mathcal{N}}\,(U, V)\,\varepsilon = max\{1{-}U(\varepsilon), V(\varepsilon)\} \\ \mathsf{s\dot{u}cc}^{\mathcal{N}}\,u\,\varepsilon = succ^{\mathcal{M}}(u\varepsilon) & \dot{\forall}^{\mathcal{N}}\,[a]U\,\varepsilon = min\{U(\varepsilon[a{:=}x]) \mid x \in M\} \\ \dot{+}^{\mathcal{N}}\,(u, v)\,\varepsilon = +^{\mathcal{M}}(u\varepsilon, v\varepsilon) & \dot{\approx}^{\mathcal{N}}\,(u, v)\,\varepsilon = \approx^{\mathcal{M}}(u\varepsilon, v\varepsilon) \\ \dot{*}^{\mathcal{N}}\,(u, v)\,\varepsilon = *^{\mathcal{M}}(u\varepsilon, v\varepsilon) & \approx_{\iota}^{\mathcal{N}}\,(u, v) = 1 \text{ if } u{=}v \text{ and } 0 \text{ otherwise} \\ \mathsf{sub}_{\iota}^{\mathcal{N}}\,([a]u, v)\,\varepsilon = u(\varepsilon[a{:=}v\varepsilon]) & \approx_{o}^{\mathcal{N}}\,(U, V) = 1 \text{ if } U{=}V \text{ and } 0 \text{ otherwise} \\ \dot{\bot}^{\mathcal{N}}\,\varepsilon = 0 & \epsilon^{\mathcal{N}}\,U = U(\varepsilon_0) \end{array}}$$

Here, u and v range over $\iota^{\mathcal{N}}$ and U and V range over $o^{\mathcal{N}}$.

LEMMA 10.7.2.

1. $[\![t'[a{:=}t]]\!]^{\mathcal{M}}_{\varepsilon} = [\![t']\!]^{\mathcal{M}}_{\varepsilon[a:=[\![t]\!]^{\mathcal{M}}_{\varepsilon}]}$.
2. $[\![\xi[a{:=}t]]\!]^{\mathcal{M}}_{\varepsilon} = 1$ if and only if $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a:=[\![t]\!]^{\mathcal{M}}_{\varepsilon}]} = 1$.

LEMMA 10.7.3. The following equalities all hold:

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{var}^{\mathcal{N}}(a) &=& [\![a]\!]^{\mathcal{M}} & \mathsf{sub}^{\mathcal{N}}_{\iota}([a][\![t']\!]^{\mathcal{M}}, [\![t]\!]^{\mathcal{M}}) &=& [\![t'[a{:=}t]]\!]^{\mathcal{M}} \\
\dot{0}^{\mathcal{N}} &=& [\![0]\!]^{\mathcal{M}} & \mathsf{sub}^{\mathcal{N}}_{o}([a][\![\xi]\!]^{\mathcal{M}}, [\![s]\!]^{\mathcal{M}}) &=& [\![\xi[a{:=}s]]\!]^{\mathcal{M}} \\
\dot{\mathsf{succ}}^{\mathcal{N}}([\![t]\!]^{\mathcal{M}}) &=& [\![succ(t)]\!]^{\mathcal{M}} & \dot{\bot}^{\mathcal{N}} &=& [\![\bot]\!]^{\mathcal{M}} \\
\dot{+}^{\mathcal{N}}([\![t']\!]^{\mathcal{M}}, [\![t]\!]^{\mathcal{M}}) &=& [\![t' + t]\!]^{\mathcal{M}} & \dot{\Rightarrow}^{\mathcal{N}}([\![\xi']\!]^{\mathcal{M}}, [\![\xi]\!]^{\mathcal{M}}) &=& [\![\xi' \Rightarrow \xi]\!]^{\mathcal{M}} \\
\dot{*}^{\mathcal{N}}([\![t']\!]^{\mathcal{M}}, [\![t]\!]^{\mathcal{M}}) &=& [\![t' * t]\!]^{\mathcal{M}} & \dot{\forall}^{\mathcal{N}}([a][\![\xi]\!]^{\mathcal{M}}) &=& [\![\forall a.\xi]\!]^{\mathcal{M}} \\
 & & & \dot{\approx}^{\mathcal{N}}([\![r]\!]^{\mathcal{M}}, [\![s]\!]^{\mathcal{M}}) &=& [\![r \approx s]\!]^{\mathcal{M}}
\end{array}$$

Proof. We compare Definitions 10.7.1 and 10.5.4. Most cases are immediate; we consider only the slightly less trivial ones:

$$\begin{array}{rcl@{\qquad}l}
\mathsf{var}^{\mathcal{N}}(a) &=& (\lambda a.\lambda\varepsilon.\varepsilon(a))a & \text{Definition 10.7.1} \\
&=& (\lambda a.[\![a]\!]^{\mathcal{M}})a & \text{Definition 10.5.4} \\
&=& [\![a]\!]^{\mathcal{M}} & \text{fact} \\
\mathsf{sub}^{\mathcal{N}}_{\iota}([a][\![t']\!]^{\mathcal{M}}, [\![t]\!]^{\mathcal{M}}) &=& \lambda\varepsilon.[\![t']\!]^{\mathcal{M}}(\varepsilon[a{:=}[\![t]\!]^{\mathcal{M}}\varepsilon]) & \text{Definition 10.7.1} \\
&=& \lambda\varepsilon.[\![t'[a{:=}t]]\!]^{\mathcal{M}} & \text{Lemma 10.7.2}
\end{array}$$

Other cases are no harder. ■

LEMMA 10.7.4. $\mathcal{N}$ (Definition 10.7.1) is a PNL interpretation.

Proof. We must check that:

- *$\iota^{\mathcal{N}}$ and $o^{\mathcal{N}}$ are permissive-nominal sets.*
  By routine calculations. (In fact, $\iota^{\mathcal{N}}$ and $o^{\mathcal{N}}$ are *nominal* sets; that is, their elements all have finite support.)
- *The functions defined in Definition 10.7.1 map elements of $\iota^{\mathcal{N}}$, $o^{\mathcal{N}}$, $[\mathbb{A}]\iota^{\mathcal{N}}$, and $[\mathbb{A}]o^{\mathcal{N}}$ correctly to the appropriate sets.*
  By Lemma 10.7.3.
- *$\epsilon^{\mathcal{N}}$ is equivariant from $o^{\mathcal{N}}$ to $\{0, 1\}$.*
  By routine calculations using the fact that $(a\ b){\cdot}\varepsilon_0 = \varepsilon_0$.

■

LEMMA 10.7.5. If $(\Xi \vdash X)^{\bullet}$ is valid in $\mathcal{N}$, then $\Xi \vdash X$ is valid in $\mathcal{M}$.

Proof. We calculate that if $(\Xi \vdash X)^{\bullet}$ is valid in $\mathcal{N}$, then

$$[\![(\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l)]\!]^{\mathcal{M}}_{\varepsilon_0} = 1$$

But the proposition written out above is closed, so for all valuations ε, $[\![(\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l)]\!]^{\mathcal{M}}_{\varepsilon} = 1$. ■

Recall from Notation 10.4.4 that we write S for $\mathsf{EQU} \cup \mathsf{SUB} \cup \mathsf{FOL}$.

PROPOSITION 10.7.6. The axioms of $\mathsf{S} \cup \mathsf{ARITH}$ are valid in $\mathcal{N}$.

Proof. By a routine verification. We consider the axiom $(\dot{\forall})$ from Figure 9. We unpack definitions and see that we must prove that for every ξ in $\mathcal{L}{+}M$,

- $\forall x{\in}M.\varepsilon_0[a{:=}x] \in (\xi)^{\bullet}$ if and only if
- $\varepsilon_0[a{:=}(t)^{\bullet}] \in (\xi)^{\bullet}$ for every t a term of $\mathcal{L}{+}M$.

This follows, because $\mathcal{L}{+}M$ has a constant symbol for every $x \in M$. Validity of the other axioms is no harder. ■

THEOREM 10.7.7. $\mathsf{arithmetic}, \Xi \vdash \chi$ in first-order logic if and only if $\mathsf{S} \cup \mathsf{ARITH} \vdash (\Xi \vdash \chi)^{\bullet}$ in PNL.

Proof. We prove two implications. The top-to-bottom implication follows using Theorem 10.4.6.

For the bottom-to-top implication, we reason as follows: Suppose $\mathsf{S} \cup \mathsf{ARITH} \vdash (\Xi \vdash \chi)^{\bullet}$ in PNL. Choose an arbitrary interpretation $\mathcal{M}$ of first-order logic that is a model of arithmetic, with carrier set M. By Soundness (Theorem 9.3.6) and Proposition 10.7.6, $(\Xi \vdash \chi)^{\bullet}$ is valid in $\mathcal{N}$. By Lemma 10.7.5 $\Xi \vdash \chi$ is valid in $\mathcal{M}$. $\mathcal{M}$ was arbitrary, so by completeness of first-order logic [Sho67, §4.2] it follows that $\Xi \vdash \chi$ is derivable. ■

11 FURTHER PROPERTIES OF PNL

11.1 More PNL theories

We briefly mention on how to express some familiar 'nominal' constructs in PNL.

Inductive types

Permissive-nominal logic can express the principles of nominal abstract syntax developed in [GP01].

Suppose a base sort ι, a name sort ν, and term-formers

$$\mathsf{var} : (\nu)\iota, \quad \mathsf{app} : (\iota,\iota)\iota, \quad \text{and} \quad \mathsf{lam} : ([\nu]\iota)\iota.$$

Fix an unknown $U : \iota$ and for brevity write $\phi[U{:=}r]$ as $\phi(r)$ for every ϕ. Suppose an axiom-scheme, for every ϕ:

$$\begin{array}{l} \phi(\mathsf{var}(a)) \Rightarrow \\ \forall X.(\phi(X) \Rightarrow \phi(\mathsf{lam}([a]X))) \Rightarrow \\ \forall X,Y.(\phi(X) \Rightarrow \phi(Y) \Rightarrow \mathsf{app}(X,Y)) \Rightarrow \\ \qquad\qquad \forall X.(\phi(X)) \end{array}$$

Here X and Y have sort ι and we make a fixed but arbitrary choice of atom $a \in \mathit{supp}(X)$.

We can also express this finitely, if we axiomatise a sort for predicates (as we did for arithmetic). Here is the axiom-scheme above made finite by using the theories EQU, SUB, and FOL from Section 10:

$$\begin{array}{l} \forall Z.\epsilon(Z[a\mapsto\mathsf{var}(a)]) \Rightarrow \\ \quad \forall X.(\epsilon(Z[a\mapsto X]) \Rightarrow \epsilon(Z[a\mapsto\mathsf{lam}([a]X))) \Rightarrow \\ \quad \forall X,Y.(\epsilon(Z[a\mapsto X]) \Rightarrow \epsilon(Z[a\mapsto Y]) \Rightarrow \epsilon(Z[a\mapsto\mathsf{app}(X,Y)])) \Rightarrow \\ \qquad\qquad \forall X.\epsilon(Z[a\mapsto X]) \end{array}$$

The И *quantifier*

Nominal sets support the И-quantifier [GP01]. PNL also includes the И-quantifier; the way in which it does this is quite interesting, as we shall see in a moment.

И has some distinctive properties which are reflected in nominal logic (NL) and the logic of FM sets (FM):

$$\begin{array}{c} \forall x.(\mathsf{P}(x) \Rightarrow \text{И}a.\mathsf{Q}(a,x)) \\ \hline\hline \forall x.\text{И}a.(\mathsf{P}(x) \Rightarrow \mathsf{Q}(a,x)) \end{array} \qquad \begin{array}{c} \forall x.\text{И}a.\text{И}b.(b\,a)\cdot x\approx x \\ \hline\hline \text{И}a.\text{И}b.\forall x.(a\#x \Rightarrow b\#x \Rightarrow (b\,a)\cdot x\approx x) \end{array}$$

Here and below we write a double horizontal line for 'is provably equivalent to'. И appears absent from Permissive-Nominal Logic (PNL). It is 'hiding' in the permission sets. Corresponding propositions are, where $a, b \notin supp(X)$

$$\begin{array}{c} \forall X.(\mathsf{P}(X) \Rightarrow \mathsf{Q}(a,X)) \\ \hline\hline \forall X.(\mathsf{P}(X) \Rightarrow \mathsf{Q}(a,X)) \end{array} \qquad \begin{array}{c} \forall X.(b\ a)\cdot X \approx X \\ \hline\hline \forall X.(b\ a)\cdot X \approx X \end{array}$$

We see from these examples that two things are happening: first, freshness conditions are hard-coded into the syntax by permission sets—and second, so is the И-quantifier.

It is interesting to consider another example. In NL/FM:

$$\begin{array}{c} \text{И}a.\mathsf{P}(a) \wedge \text{И}a.\mathsf{Q}(b) \\ \hline\hline \text{И}a.\text{И}b.(\mathsf{P}(a) \wedge \mathsf{Q}(b)) \end{array} \qquad \begin{array}{c} \text{И}a.\mathsf{P}(a) \wedge \text{И}a.\mathsf{Q}(b) \\ \hline\hline \text{И}a.(\mathsf{P}(a) \wedge \mathsf{Q}(a)) \end{array}$$

Correspondingly in PNL we have:

$$\begin{array}{c} \mathsf{P}(a) \wedge \mathsf{Q}(b) \\ \hline\hline \mathsf{P}(a) \wedge \mathsf{Q}(b) \end{array} \qquad \begin{array}{c} \mathsf{P}(a) \wedge \mathsf{Q}(b) \\ \hline\hline \mathsf{P}(a) \wedge \mathsf{Q}(a) \end{array}$$

It is easy to use the rule $(\mathbf{Ax})$ from Figure 5 to construct a derivation proving that $\mathsf{P}(a) \wedge \mathsf{Q}(b)$ and $\mathsf{P}(a) \wedge \mathsf{Q}(a)$ are indeed logically equivalent in Permissive-Nominal Logic.

The π in $(\mathbf{Ax})$ expresses that truth is preserved by permutative renaming, or in symbols: $\vdash \phi \Leftrightarrow \pi\cdot\phi$ always.

A permission set S can be viewed in two ways: as giving permission to instantiate using free atoms in S—but also as a form of И for the atoms not in S.

Semantic freshness

To express in permissive-nominal algebra that a is fresh for the denotation of s it suffices to assert $(b\ a)\cdot s = s$ where $b \notin \mathit{supp}(s)$. Thus the theory of a semantic freshness predicate Fresh has one axiom $\mathsf{Fresh}(a, X) \Leftrightarrow (b\ a)\cdot X = X$ where $a \in \mathit{supp}(X)$ and $b \notin \mathit{supp}(X)$ (and we fill in sorts as appropriate). In PNL with equality, the axiom is $\forall X.\mathsf{Fresh}(a, X) \Leftrightarrow (b\ a)\cdot X = X$.

Atoms-abstraction

Atoms-abstraction can also be expressed as a theory in permissive-nominal algebra. For a base sort sort τ and name sort ν assume a base sort $[\nu]\tau$ and a term-former $\mathsf{abs} : (\nu, \tau)([\nu]\tau)$, along with a single axiom $\mathsf{abs}(a, X) = \mathsf{abs}(b, (b\ a)\cdot X)$ where $a \in \mathit{supp}(X)$ and $b \notin \mathit{supp}(X)$. In PNL with equality, the axiom is $\forall X.\mathsf{abs}(a, X) = \mathsf{abs}(b, (b\ a)\cdot X)$.

11.2 Admissibility of Cut

We indicate how $(\mathbf{Cut})$ is admissible in the presence of the other rules in Figure 5.

DEFINITION 11.2.1. Suppose $\mathit{fa}(r) \subseteq \mathit{supp}(X)$ and $r : \mathit{sort}(X)$. Define $\Phi[X{:=}r]$ by

$$\Phi[X{:=}r] = \{\phi[X{:=}r] \mid \phi \in \Phi\}.$$

Lemma 11.2.2 is proved by routine arguments like those in [DGM10; UPG04]:

LEMMA 11.2.2. Suppose $Y \notin \mathit{fV}(t)$. Then

$$r[Y{:=}u][X{:=}t] = r[X{:=}t][Y{:=}u[X{:=}t]].$$

LEMMA 11.2.3. Suppose $\mathit{fa}(r) \subseteq \mathit{supp}(X)$ and $r : \mathit{sort}(X)$. Then

$$\Phi \vdash \Psi \quad \text{implies} \quad \Phi[X{:=}r] \vdash \Psi[X{:=}r].$$

Proof. By a routine induction on derivations. The case of $(\mathbf{Ax})$ uses Lemmas 3.4.10 and 11.2.2. The case of $(\forall\mathbf{L})$ uses Lemma 11.2.2. ■

LEMMA 11.2.4.

1. If there exists a derivation Δ of $\Phi \vdash \psi, \Psi$ then there exists a derivation of $\Phi \vdash \pi\cdot\psi, \Psi$.
2. If there exists a derivation Δ of $\Phi, \phi \vdash \Psi$ then there exists a derivation of $\Phi, \pi\cdot\phi \vdash \Psi$.

Proof. By a simultaneous induction on Δ. The case of $(\forall\mathbf{L})$ uses Lemma 3.4.10. (We need the *simultaneous* induction for $(\Rightarrow\mathbf{L})$ and $(\Rightarrow\mathbf{R})$, since parts of the proposition move between left and right.) ■

NOTATION 11.2.5. An instance of $(\mathbf{Cut})$ rests on two sub-derivations. It is convenient to call them the **left branch** and **right branch** as illustrated:

$$\dfrac{\begin{array}{c}\vdots\ \textit{Left branch}\\ \Phi,\ \phi\vdash\Psi\end{array}\qquad\qquad \begin{array}{c}\vdots\ \textit{Right branch}\\ \Phi\vdash\phi,\Psi\end{array}}{\Phi\vdash\Psi}\,(\mathbf{Cut})$$

THEOREM 11.2.6 (Cut-elimination). If $\Phi \vdash \Psi$ is derivable with a derivation that uses $(\mathbf{Cut})$, then it is derivable with a derivation that does not use $(\mathbf{Cut})$.

Proof. The proof is as for first-order logic. The only differences are a π in $(\mathbf{Ax})$ and a side-condition $fa(r) \subseteq supp(X)$ in $(\forall\mathbf{L})$. These affect terms and have no effect on the structure of derivations; for the purposes of this proof they are irrelevant.

We commute instances of $(\mathbf{Cut})$ upwards, as usual, following the method of [Dum77, pages 139-145] or [Gab11a]. At each step, the following measure based on the depth of subderivations and the size of the cut formula, decreases:

- The size of the cut formula, and
- the longest path up the derivation the cut, that the formula persists,

lexicographically ordered.

- The commutation cases between rules for $\Rightarrow$ and $\forall$ are as standard for first-order logic.
- The essential case for $\Rightarrow$ is as standard.
- For the essential case for $\forall$, suppose the subderivation has the following form:

$$\dfrac{\dfrac{\Phi,\ \phi[X{:=}r]\vdash\Psi}{\Phi,\ \forall X.\phi\vdash\Psi}\,(\forall\mathbf{L})\qquad \dfrac{\begin{array}{c}\vdots\ \Delta\\ \Phi\vdash\phi,\ \Psi\end{array}}{\Phi\vdash\forall X.\phi,\ \Psi}\,(\forall\mathbf{R})}{\Phi\vdash\Psi}\,(\mathbf{Cut})$$

  By Lemma 11.2.3 there is a derivation $\Delta[X{:=}r]$ of $\Phi \vdash \phi[X{:=}r],\ \Psi$. We eliminate the essential case as follows:

$$\dfrac{\Phi,\ \phi[X{:=}r]\vdash\Psi\qquad \begin{array}{c}\vdots\ \Delta[X{:=}r]\\ \Phi\vdash\phi[X{:=}r],\ \Psi\end{array}}{\Phi\vdash\Psi}\,(\mathbf{Cut})$$

- Suppose the subderivation has the following form:

$$\dfrac{\dfrac{}{\Phi,\ \phi\vdash\pi\cdot\phi,\Psi}\,(\mathbf{Ax})\qquad \begin{array}{c}\vdots\ \Delta\\ \Phi,\ \pi\cdot\phi\vdash\Psi\end{array}}{\Phi,\ \phi\vdash\Psi}\,(\mathbf{Cut})$$

  We use Lemma 11.2.4 to obtain a derivation Δ' of $\Phi,\ \phi \vdash \Psi$ (the transformations involved in the proof of Lemma 11.2.4 do not increase the inductive measure).

■

11.3 Exhausting the available atoms

We conclude with a brief discussion on a subtle point in the PNL design. Suppose a name sort ν, a base sort τ, and a proposition former $\# : (\nu, \tau)$. Suppose an atom a and an unknown $X : \tau$ with $supp(X) = \mathbb{A}^<$. Suppose an unknown $Y : \nu$ with $supp(Y) = \mathbb{A}^<$. Consider an interpretation in which $\#(a, X)$ is interpreted as $a \notin supp(\varsigma(X))$ and τ is interpreted as $\mathbb{L}$ (Definition 2.4.4).

That is, $\#$ is interpreted as freshness and τ is interpreted as well-orderings of permission-sets.

In the PNL of this paper, the interpretation of the proposition $\phi = \forall X.\exists Y.\#(Y, X)$ is false: we take $\varsigma(X)$ to well-order $\mathbb{A}^<$ and there is no $a \in supp(Y)$ such that $a \notin supp(\varsigma(X))$.

Suppose we decide that we want a version of PNL in which ϕ is true. In this case, we can consider denotations such that every element has support of the form $\pi\cdot\mathbb{A}^{\lll}$ where $\mathbb{A}^{\lll}$ is infinite and $\mathbb{A}^{\lll} \subseteq \mathbb{A}^<$ and $\mathbb{A}^< \setminus \mathbb{A}^{\lll}$ is also infinite. In this way, an unknown X cannot 'exhaust' $\mathbb{A}^<$.

The lesson we draw from this small example is that nominal semantics offer a host of interesting and inspiring design options. In this paper, we have cut one path through this design space which is expressive enough to get the results we want. Other paths are possible.

12 CONCLUSIONS

This paper reflects a research arc by the author in collaboration with others, roughly from 2005 to 2012. Thanks to improvements in presentation and the use of permissive-nominal techniques, definitions and proofs are simpler than in previous literature, and new properties emerge.

We have constructed permissive-nominal sets. We gave a nominal syntax for them and explored their computational properties in nominal unification and rewriting. We considered nominal algebra and proved soundness, completeness, and HSP over permissive-nominal sets. We gave nominal terms a $\forall$-quantifier over unknowns and used this to build a first-order logic *permissive-nominal logic*. Finally, in an extended case study we gave finite axiomatisations of first-order logic and arithmetic and proved correctness.

Mathematical foundations influence language, and (famously) language influences thought. Nominal sets are a foundation with a model of names which is different from what has been considered before, so the question is: what new languages, and new thoughts, can emerge? This chapter attempts to address that question by illustrating the broad sweeps of what a 'nominal' meta-mathematics might look like.

We are not and cannot be encyclopaedic or exhaustive. For other work we should mention αProlog, which allows Horn clauses [CU08] (this preceded PNL, and could be viewed as a subset of it). The author in collaboration has proved correctness for several non-trivial theories in nominal syntaxes, including equational treatments of substitution [GM06a; GM08a], λ-calculus [GM08b; GM10], and first-order logic [GM06c; GM08c], as well as the finite first-order nominal axiomatisation of arithmetic [DG10; DG12a] which we considered in Section 10. There are translations from nominal terms to λ-terms by Levy and Villaret and by Dowek, the author, and

Mulligan [LV08; DGM10], including a translation of algebraic reasoning (so, not just unification) [GM09b]; and there is a translation of permissive-nominal logic to higher-order logic in [DG12b] which illustrates the differences and similarities of the two logics, and exploits some unusual model-theoretic ideas.

We also mention a translation of nominal terms to many-sorted first-order syntax by Kurz and Petrişan [KP10], and a categorical treatment of nominal Lawvere theories in [Clo09]. It may also prove useful to consider nominal languages over nominal structures other than sets, for instance over nominal domains [Tur09].

See also the 'atlas of nominal languages' in Appendix A.

This research is developing a topic which this author believes could become an immense field; the informal meta-level having been relatively unformalised until now for want of a denotation with names, which is what nominal sets provide.

It is important to realise that this story is not just about nominal sets, nor is it just about semantics; there is also the issue of finding appropriate syntaxes for our semantics. The logic of FM sets, nominal logic, and the Nominal Isabelle package [GP01; Pit03; Urb08] are all first-order axiomatisations of nominal sets.[19] In all these cases, the syntax is that of 'ordinary' first- or higher-logic.[20] These are denotations for syntax-with-binding.

Nominal terms and permissive-nominal terms, and the syntaxes based on them such as nominal rewriting, algebra, and permissive-nominal logic, do not follow automatically from nominal sets. They are syntaxes for meta-mathematics of independent interest. Thus, this chapter has surveyed the author's attempts, via methods which are both syntactic and also semantic, to outline what meta-mathematics could look like if it were based on nominal foundations. The fact that—for instance—we were able to finitely axiomatise arithmetic in the nominal first-order that is PNL in Subsection 10.2, is one demonstration that this meta-mathematics is a new and different place from what the reader may be used to.

In a sense this paper is a sequel to the survey of [Gab11b] (written in 2008 and submitted in early 2009). But whereas [Gab11b] concentrated on applications of nominal sets to syntax with binding, this paper considers nominal sets as a basis for meta-mathematics. Hints of this appeared in nominal rewriting [FGM04; FG07], which allowed arbitrary oriented equality theories over nominal terms. Perhaps unwisely, we shall succumb to a wordplay: [Gab11b; GP01] explore *denotation of specification with binding*; whereas here we explore *specification of denotation with binding*.

Thus, in this document we have explored the consequences of taking FM-sets style names seriously in meta-mathematics. But even that does not exhaust the potential applications of nominal techniques. Mathematics and computer science are evolving in ways which increase the importance of names, and nominal techniques have arisen

[19]Essentially, [GP01] is the first third of the author's thesis; [Pit03] is the same but minus the cumulative sets hierarchy; and [Urb08] is an extensive implementation in higher-order logic, with a library of powerful macros. One reason this is non-trivial has to do with automatically deriving the *equivariance* properties described e.g. in [Gab11b, Subsection 4.2].

[20]Sometimes, authors write 'nominal logic' for that logic obtained by adding for each atom a constant symbol to the syntax of first-order logic, and adding infinitely many axioms reflecting nominal sets (equalities of swapping atoms, fresh atoms, and so on). This is nominal sets wearing a 'syntactic disguise': consider by analogy a theory of arithmetic with a constant symbol for each number and an axiom for every arithmetic equality.

from this; we can expect that evolution to continue.

This motivates us to revisit certain foundational design decisions; whether to admit atoms—to sound more mathematical, we say *urelemente* and to sound less mathematical, we say *names*—and what properties these should have. Linguists might well call these *referents*, and have been studying them for a long time.

Whatever we call them, they exist and we use them all the time. So we will conclude with two slogans:

- *Names are data.*
- *Names with additional properties are ubiquitous.*

This chapter has studied formal languages with which to specify some of the possible additional properties of names, such as 'having a substitution action' or 'being universally quantifiable'. But more generally, by this combination of a new point of view and a rigorous mathematics, nominal techniques have the potential to simplify, factor out common properties, and help control some of a modern mathematics of logic and computation.

Names are not just a technical issue, to be ignored or circumvented with 'tricks'. Names are a philosophical, foundational, linguistic, and computational issue. The mathematics of names is the mathematics of mathematics.

Dov Gabbay wrote in his preface to the second edition that

> the researcher ... is having more and more in common with the traditional philosopher who has been analysing such questions for centuries (unrestricted by the capabilities of any hardware). ... I believe the day is not far away in the future when the computer scientist will wake up one morning with the realisation that he is actually a kind of formal philosopher!

We would add "and philosophers, linguists—and some artists too—may wake up one morning with the realisation that they are actually a kind of abstract computer scientist". Amen.

BIBLIOGRAPHY

[ACCL91] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[Bir35] Garrett Birkhoff. On the structure of abstract algebras. *Proceedings of the Cambridge Philosophical Society*, 31:433–454, 1935.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, Great Britain, 1998.

[BS81] Stanley N. Burris and H. P. Sankappanavar. *A Course in Universal Algebra.* Graduate texts in mathematics. Springer, 1981.

[Cal10] Christophe Calvès. *Complexity and implementation of nominal algorithms.* PhD thesis, King's College London, 2010.

[Che04] James Cheney. The complexity of equivariant unification. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 332–344. Springer, 2004.

[Che05a] James Cheney. Relating nominal and higher-order pattern unification. In *Proceedings of the 19th International Workshop on Unification (UNIF 2005)*, pages 104–119. LORIA research report A05-R-022, 2005.

[Che05b] James Cheney. A simpler proof theory for nominal logic. In *FoSSaCS*, volume 3441 of *Lecture Notes in Computer Science*, pages 379–394. Springer, 2005.

[Che06] James Cheney. Completeness and Herbrand theorems for nominal logic. *Journal of Symbolic Logic*, 71:299–320, 2006.

[Che10] James Cheney. Equivariant unification. *Journal of Automated Reasoning*, 45(3):267–300, October 2010.

[Clo07] Ranald Clouston. Closed terms (unpublished notes). `http://users.cecs.anu.edu.au/\~rclouston/closedterms.pdf`, 2007.

[Clo09] Ranald Clouston. *Equational logic for names and binding*. PhD thesis, University of Cambridge, UK, 2009.

[Clo11] Ranald Clouston. Nominal Lawvere theories. In *Proceedings of the 18th International Workshop on Logic, Language, and Information (WoLLIC)*, volume 6642 of *Lecture Notes in Computer Science*. Springer, 2011.

[CP07] Ranald A. Clouston and Andrew M. Pitts. Nominal equational logic. In *Computation, Meaning and Logic: Articles dedicated to Gordon Plotkin*, volume 172 of *Electronic Notes in Theoretical Computer Science*, pages 223–257. Elsevier Science, 2007.

[CU03] James Cheney and Christian Urban. System description: Alpha-Prolog, a fresh approach to logic programming modulo alpha-equivalence. In *UNIF'03*, pages 15–19. Universidad Politécnica de Valencia, 2003.

[CU08] James Cheney and Christian Urban. Nominal logic programming. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 30(5):1–47, 2008.

[DG10] Gilles Dowek and Murdoch J. Gabbay. Permissive Nominal Logic. In *Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP 2010)*, pages 165–176, New York, 2010. ACM Press.

[DG12a] Gilles Dowek and Murdoch J. Gabbay. Permissive Nominal Logic (journal version). *Transactions on Computational Logic*, 13(3), 2012.

[DG12b] Gilles Dowek and Murdoch J. Gabbay. PNL to HOL: from the logic of nominal sets to the logic of higher-order functions. *Theoretical Computer Science*, 451:38–69, 2012.

[DGM09] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. In *Proceedings of the 24th Italian Conference on Computational Logic (CILC'09)*, 2009.

[DGM10] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification: an infinite, co-infinite approach to nominal techniques (journal version). *Logic Journal of the IGPL*, 18(6):769–822, 2010.

[DJ89] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Methods and Semantics*, volume B. North-Holland, 1989.

[Dow01] Gilles Dowek. Higher-order unification and matching. In *Handbook of automated reasoning*, pages 1009–1062. Elsevier, 2001.

[Dum77] Michael Dummett. *Elements of intuitionism*. Clarendon Press, 1 edition, 1977.

[FG07] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting (journal version). *Information and Computation*, 205(6):917–965, June 2007.

[FG10] Maribel Fernández and Murdoch J. Gabbay. Closed nominal rewriting and efficiently computable nominal algebra equality. In *Electronic Proceedings in Theoretical Computer Science*, volume 34, pages 37–51, 2010.

[FGM04] Maribel Fernández, Murdoch J. Gabbay, and Ian Mackie. Nominal Rewriting Systems. In *Proceedings of the 6th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming (PPDP 2004)*, pages 108–119. ACM Press, August 2004.

[FH92] Matthias Felleisen and Robert Hieb. The revised report on the syntactic theories of sequential control and state. *Theoretical computer science*, 103(2):235–271, 1992.

[FH10] Marcelo Fiore and Chung-Kil Hur. Second-order equational logic. In *Proceedings of the 19th EACSL Annual Conference on Computer Science Logic (CSL 2010)*, Lecture Notes in Computer Science. Springer, 2010.

[FPT99] Marcelo P. Fiore, Gordon D. Plotkin, and Daniele Turi. Abstract syntax and variable binding. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science (LICS 1999)*, pages 193–202. IEEE Computer Society Press, 1999.

[Gab01] Murdoch J. Gabbay. *A Theory of Inductive Definitions with alpha-Equivalence*. PhD thesis, University of Cambridge, UK, March 2001.

[Gab05] Murdoch J. Gabbay. Axiomatisation of first-order logic (talk). In *Second workshop on Computational Aspects of Nominal sets (CANS'05)*. King's College, London, December 2005.

[Gab07a] Murdoch J. Gabbay. Fresh Logic. *Journal of Applied Logic*, 5(2):356–387, June 2007.

[Gab07b] Murdoch J. Gabbay. A General Mathematics of Names. *Information and Computation*, 205(7):982–1011, July 2007.

[Gab09] Murdoch J. Gabbay. Nominal Algebra and the HSP Theorem. *Journal of Logic and Computation*, 19(2):341–367, April 2009.

[Gab11a] Michael Gabbay. A proof-theoretic treatment of lambda-reduction with cut-elimination: lambda calculus as a logic programming language. *Journal of Symbolic Logic*, 76(2):673–699, June 2011.

[Gab11b] Murdoch J. Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic*, 17(2):161–229, 2011.

[Gab11c] Murdoch J. Gabbay. Two-level nominal sets and semantic nominal terms: an extension of nominal set theory for handling meta-variables. *Mathematical Structures in Computer Science*, 21:997–1033, 2011.

[Gab12a] Murdoch J. Gabbay. Meta-variables as infinite lists in nominal terms unification and rewriting. *Logic Journal of the IGPL*, 2012.

[Gab12b] Murdoch J. Gabbay. Unity in nominal equational reasoning: The algebra of equality on nominal sets. *Journal of Applied Logic*, 10:199–217, June 2012.

[GC04] Murdoch J. Gabbay and James Cheney. A Sequent Calculus for Nominal Logic. In *Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS 2004)*, pages 139–148. IEEE Computer Society, July 2004.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen [Investigations into logical deduction]. *Mathematische Zeitschrift 39*, pages 176–210,405–431, 1935. Translated in [Sza69], pages 68–131.

[GH08] Murdoch J. Gabbay and Martin Hofmann. Nominal renaming sets. In *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008)*, pages 158–173. Springer, November 2008.

[GM06a] Murdoch J. Gabbay and Aad Mathijssen. Capture-avoiding Substitution as a Nominal Algebra. In *ICTAC 2006: Theoretical Aspects of Computing*, volume 4281 of *Lecture Notes in Computer Science*, pages 198–212, November 2006.

[GM06b] Murdoch J. Gabbay and Aad Mathijssen. Nominal Algebra. In *18th Nordic Workshop on Programming Theory*, October 2006.

[GM06c] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order logic. In *Proceedings of the 8th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP 2006)*, pages 189–200. ACM, July 2006.

[GM07] 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, July 2007.

[GM08a] Murdoch J. Gabbay and Aad Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing*, 20(4-5):451–479, June 2008.

[GM08b] Murdoch J. Gabbay and Aad Mathijssen. The lambda-calculus is nominal algebraic. In Christoph Benzmüller, Chad Brown, Jörg Siekmann, and Rick Statman, editors, *Reasoning in simple type theory: Festschrift in Honour of Peter B. Andrews on his 70th Birthday*, Studies in Logic and the Foundations of Mathematics. IFCoLog, December 2008.

[GM08c] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic. *Journal of Logic and Computation*, 18(4):521–562, August 2008.

[GM09a] Murdoch J. Gabbay and Aad Mathijssen. Nominal universal algebra: equational logic with names and binding. *Journal of Logic and Computation*, 19(6):1455–1508, December 2009.

[GM09b] Murdoch J. Gabbay and Dominic P. Mulligan. Universal algebra over lambda-terms and nominal terms: the connection in logic between nominal techniques and higher-order variables. In *Proceedings of the 4th International Workshop on Logical Frameworks and Meta-Languages (LFMTP 2009)*, pages 64–73. ACM, August 2009.

[GM10] Murdoch J. Gabbay and Aad Mathijssen. A nominal axiomatisation of the lambda-calculus. *Journal of Logic and Computation*, 20(2):501–531, April 2010.

[GP99] Murdoch J. Gabbay and Andrew M. Pitts. A New Approach to Abstract Syntax Involving Binders. In *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS 1999)*, pages 214–224. IEEE Computer Society Press, July 1999.

[GP01] 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, July 2001.

[KP10] Alexander Kurz and Daniela Petrişan. On universal algebra over nominal sets. *Mathematical Structures in Computer Science*, 20:285–318, 2010.

[LV08] Jordi Levy and Mateu Villaret. Nominal unification from a higher-order perspective. In *Rewriting Techniques and Applications, Proceedings of RTA 2008*, volume 5117 of *Lecture Notes in Computer Science*. Springer, 2008.

[LV10] Jordi Levy and Mateu Villaret. An efficient nominal unification algorithm. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA 2010)*, volume 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 209–226. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

[LV12] Jordi Levy and Mateu Villaret. Nominal unification from a higher-order perspective. *Transactions on Computational logic (TOCL)*, 13(2), 2012.

[Mat07] Aad Mathijssen. *Logical Calculi for Reasoning with Binding*. PhD thesis, Technische Universiteit Eindhoven, 2007.

[Mel95] Paul-André Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In Mariangiola Dezani-Ciancaglini and Gordon D. Plotkin, editors, *Proceedings of the 2nd International Conference on Typed Lambda Calculi and Applications, (TLCA 1995)*, volume 902 of *Lecture Notes in Computer Science*, pages 328–334. Springer, April 1995.

[MM92] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Universitext. Springer, 1992.

[MNPS89] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. Technical report, Durham, NC, USA, 1989.

[Pit01] Andrew M. Pitts. Nominal logic: A first order theory of names and binding. In N. Kobayashi and B. C. Pierce, editors, *Proc. 4th Int'l Symposium on Theoretical Aspects of Computer Software (TACS 2001)*, volume 2215 of *Lecture Notes in Computer Science*, pages 219–242. Springer, 2001.

[Pit03] Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003.

[Pit11] Andrew Pitts. Nominal sets and their applications. In *Midlands Graduate School (MGS 2011)*, 11-15 April 2011. available online at `cl.cam.ac.uk/\~amp12/talks/MGS2011_nominal_sets_slides.pdf`.

[Pra65] Dag Prawitz. *Natural deduction: a prooof-theoretical study*. Almquist and Wiksell, 1965. Reprinted by Dover, 2006.

[Sho67] Joseph Shoenfield. *Mathematical Logic*. Addison-Wesley, 1967.

[Smu68] Raymond Smullyan. *First-order logic*. Springer, 1968. Reprinted by Dover, 1995.

[Sza69] M. E. Szabo, editor. *Collected Papers of Gerhard Gentzen*. North Holland, 1969.

[Tur09] David C. Turner. *Nominal Domain Theory for Concurrency*. PhD thesis, University of Cambridge, 2009.

[Tze07] Nikos Tzevelekos. Full abstraction for nominal general references. In *Proceedings of the 22nd IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 399–410. IEEE Computer Society Press, 2007.

[UPG03] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. In *Proceedings of the 17th International Workshop on Computer Science Logic (CSL 2003)*, volume 2803 of *Lecture Notes in Computer Science*, pages 513–527. Springer, December 2003.

[UPG04] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, September 2004.

[Urb08] Christian Urban. Nominal reasoning techniques in Isabelle/HOL. *Journal of Automatic Reasoning*, 40(4):327–356, 2008.

Appendix

A AN ATLAS OF NOMINAL LANGUAGES

The reader coming to the nominal literature could be forgiven for finding it perplexing. What are 'Fraenkel-mostowski sets', 'nominal sets', 'nominal terms', 'nominal logic', 'nominal rewriting and algebra', 'αProlog', 'nominal equational logic', 'permissive-nominal algebra', 'permissive-nominal logic' (with/without *shift*-permutations)? In this Appendix we will give a brief annotated bibliography covering, loosely, the relevant publications. This list is not meant to be exhaustive.

Traditionally, nominal sets are understood as a tool for the mathematical analysis of syntax, as described for instance in the author's previous survey/research paper [Gab11b], or in slides of an excellent course of lectures by Pitts [Pit11]. This author takes a view of nominal sets not just as a foundation for syntax with binding, but as a foundation for mathematics itself—names and binding, after all, appear everywhere. The atlas below surveys relevant publications.

For each item in the list below, we reference where the idea was introduced to the 'nominal' literature, and any other relevant conference and journal papers.

FM set theory [GP99; GP01]. Fraenkel-Mostowski set theory (FM) and nominal sets (called 'equivariant FM sets' in that paper) are the foundational semantics for nominal techniques.

Fraenkel-Mostowski sets were already known and had been used for other purposes; see [Gab11b, Remark 2.22] for more detailed historical comments. Nominal sets were familiar as e.g. the Schanuel topos. So both semantics were known.

What was new to [GP01] was the observation by the author and Pitts of the notions of support, atoms-abstraction, the self-dual behaviour of the И quantifier, and the application to what is now called *nominal abstract syntax*.[21]

Nominal logic [Pit01; Pit03]. The constructions of [GP01] are repeated, but in a first-order axiomatisation of nominal sets rather than one of the FM cumulative hierarchy. Pitts also coined the catchy label 'nominal'.

Sometimes authors identify the nominal logic of [Pit03] with nominal techniques in general. This is limiting, and it gets the mathematical development the wrong way round. Nominal logic is a Hilbert-style axiomatisation in first-order logic. These axioms have meaning because of the underlying nominal sets models, and not the other way around; nor does the axiomatisation *per se* contribute to new syntax or proof-theory with which to study names.

[21] At the same time, Fiore Plotkin and Turi developed an approach to abstract syntax which was really exactly the same thing [FPT99]. The key difference turned out to be that nominal sets admit a relatively elementary sets-based interpretation of the presheaves. As argued in [GH08] there are 'fewer presheaves' in the nominal semantics, we feel that an elementary presentation of the mathematics—where this is possible—is a powerful advantage not just for the reader but also for the practicing theorist.

Fiore has continued this line of research in collaboration and produced logics which in some sense which has never been made formal, parallel the development here. For an example of this see [FH10].

In order to make progress, we needed new syntax that more explicitly represents atoms and their properties.

Thus for instance the *nominal logic programming* developed by Cheney and Urban [CU08] (also referenced below) is called logic-programming in nominal logic, but we also see from Figures 6, 7, and 8 of [CU08] that the syntax and axioms used are a variant of nominal terms.

Proof-theories for the И-quantifier [Gab07a; GC04; Che05b]. Some attempts have been made to give the distinctive И-quantifier of nominal techniques, a proof-theory. In arguably increasing order of elegance these are [Gab07a] (this was received by the journal in 2003 but took four years to get printed), [GC04] (written with Cheney to develop on [Gab07a]), and [Che05b].

The permissive-nominal logic (PNL) of this survey is another item on that list, and perhaps it is one of the nicest; certainly the PNL treatment of И is very different from what has come before, see Subsection 11.1.

Complete semantics for this family of logics are in [Gab07a], [Che06], and in [DG12a]. See also Subsection 9.4 of this survey.

Nominal terms [UPG03; UPG04]. This new syntax introduced the distinctive freshness side-conditions and the nominal terms syntax, with its separation of atoms a and unknowns X into two syntactic classes. [UPG04] is where the syntactic ideas of this survey were born, if not the specific 'permissive' implementation, which came later (*permissive-nominal terms* below).

There is now quite a substantial body of work devoted to computing efficiently on nominal terms; notably [Cal10; LV10]. There is also a body of work devoted to translating between nominal terms and *higher-order patterns* [MNPS89]. We are far from exhaustive, but good places to start reading are [Che05a], [LV08; LV12], and [GM09b; DGM10].

Nominal rewriting [FGM04; FG07] and αProlog [CU03; CU08]. These were the first logical languages using nominal terms as a general-purpose assertion language; nominal rewriting was designed explicitly to allow us to assert (directed) equalities between terms such as β or η-equivalence. αProlog was intended by its designers for reasoning on nominal abstract syntax, and explicitly presented as such—but in retrospect it can also be viewed as a general-purpose 'nominal' reasoning system in the same family as nominal rewriting and later work.[22]

Nominal algebra [Gab05; GM06a; GM07; GM09a]. Nominal algebra is simply the undirected version of nominal rewriting.[23] What makes nominal algebra interesting above and beyond nominal rewriting is the different theorems we prove about equality instead of rewriting; for instance the HSPA theorem of [Gab09] (much

[22]James Cheney, private communication.

[23]Actually, this is a simplification. There is a significant difference, which is described in [FG10]: nominal rewriting does not have an explicit rule to generate fresh atoms, whereas nominal algebra does. To the level of detail we wish to go into here, this does not matter. The permissive-nominal syntax of this survey makes the issue obsolete because fresh atoms are a structural fact of the permission sets.

simplified here in Section 8), and various correctness results for axiomatisations of e.g. substitution, λ-calculus, and first-order logic [GM06a; GM08a; GM06c; GM08b; GM08c; GM10].

The paper [GM06a] is where the *permutative convention* of Definition 2.1.2 was introduced, used by the author consistently since then. This comes from the author's work formalising nominal reasoning in Isabelle in [Gab01] and spares us from having to explicitly enumerate all inequalities between atoms. Thus, if pressed to be entirely formal, 'a and b' refers to two meta-variables ranging over *distinct* atoms.

Kurz and Petrişan proved an HSP theorem for nominal algebra by treating nominal algebra as a kind of many-sorted first-order logic [KP10]—the sorts are finite sets of atoms and come from the categorical view of nominal sets as presheaves. The effect of nominal theories can thus be attained in many-sorted first-order syntax. That syntax is just standard first-order syntax is potentially a big advantage, for instance if one wants to transfer results directly from universal algebra. This offers alternative and effective methods of semantic proof; e.g. [KP10] significantly simplifies the proofs of [Gab09]. We pay for this convenience with infinities; e.g. even the simplest theory is infinite since equalities are replicated at every sort. Of course, the theory may still be finitely presentable. Section 8 of the current paper contains another, further simplified, HSP proof.

Nominal equational logic [CP07; Clo11]. Call the judgement 'a is fresh for the syntax s' **syntactic freshness** and 'a is fresh for the denotation of s' **semantic freshness**. *Nominal Equational Logic* (**NEL**) closely resembles NA, but whereas both have a semantic equality judgement ($s = t$), NEL adds a semantic freshness judgement.

In [CP07] Clouston and Pitts claimed that NEL was significantly more complete than NA because of this, but they had missed that semantic freshness is expressible using equality and syntactic freshness (see for instance [GM07, Theorem 5.5] and [GM09a, Lemma 4.51]).[24]

Note that two distinct logics have been called NEL: one in [CP07], and one in [Clo11] which restricts semantic freshness to the left of the turnstile; compare Figure 5 of [CP07] with Figure 1 of [Clo11]. Both have syntactic freshness: see the side-condition $a\#(\bar{a}, t, t')$ in the Atm-Intro and Atm-Elim rules of Figure 5, and similar side-conditions in Figure 1. Thus, when Clouston writes in [Clo11] that "*[syntactic]* freshness in NA is sound, but not complete, for freshness in the underlying nominal sets interpretation *[semantic freshness]*", echoing similar comments in [CP07], this omits mention that NEL also has a syntactic freshness.

It is in any case a red herring. If we can choose fresh atoms and compare elements for semantic equality, then semantic freshness makes the logic 'do equality twice' and just adds complexity [Gab12b]—*without* equality, the story can be different; this was encountered in the first attempt at a nominal functional programming language, in which we included freshness information in the types [Gab01].

Permissive-nominal terms [GM09b; DGM09; DGM10]. These simplify and improve classical nominal terms in two ways: we give explicitly the (countably infinite)

[24]Syntactic freshness appears in this paper as $a \notin fa(r)$. We considered semantic freshness in Section 11. See also Proposition 7.6.1.

atoms that may be free / are guaranteed to be fresh in every unknown, and since freshness information is stored directly we eliminate the need for freshness contexts. Thus good properties emerge: permissive-nominal terms can be constructed as nominal abstract syntax, we can directly choose a name fresh for a term (which is not possible in nominal terms without expanding the freshness context), and properties and proofs can then be expressed for terms alone, rather than for terms-in-freshness-context.

For instance, in classical nominal terms a solution of a nominal unification problem is a pair of a substitution and a freshness context; a nominal rewrite rule is a left and a right-hand side term and a freshness context; the proof-theory of nominal algebra requires an explicit freshness rule to generate fresh atoms, and so on. In fact, manipulating nominal terms almost always requires us to manipulate an external structure representing freshness constraints.

In contrast permissive terms are 'self-sufficient', like ordinary syntax. Proofs and algorithms have more of the look and feel of ordinary syntax. We have seen how, in the body of this survey. A detailed treatment of permissive-nominal syntax, including a simple translation from the nominal terms of [UPG04] into permissive-nominal terms, is also in [DGM10].

Permissive-nominal algebra ([GM09b], and Section 7). The permissive-nominal algebra of Section 7 uses permissive-nominal terms and has a significantly different proof-theory.

The notable differences are, aside from being permissive-nominal, the inclusion (if we want them) of infinitely-supported constant symbols and of infinitely-supported permutations. So previous work is a special case of the general framework of this survey, but what we do here goes strictly beyond what was possible in previous work, also in some significant mathematical properties such as satisfying an HSP instead of an HSPA result; see the discussion opening Section 8.

Permissive-nominal logic ([DG10; DG12a] and Section 9). As we discuss in this survey, permissive-nominal logic (**PNL**) adds universal quantification over unknowns X. This is non-evident for nominal terms because of their freshness contexts; in nominal terms X behaves like an element with cofinite support so we lose α-equivalence whereas in permissive-nominal terms X has coinfinite support and we can always α-rename bound atoms. We get a proof-theory which is pleasingly close to that of first-order logic, a sound and complete semantics, and we can axiomatise and prove correct a non-trivial and mathematically relevant theory, such as arithmetic.

Name	**Intended model**	**Refs**	**Notes**
FM set theory	Cumulative hierarchy / Cat. of FM sets	[GP99; GP01]	Previously used to prove independence of AC
Nominal / FM sets	Themselves	[GP99; GP01]	Nominal sets called ‘equivariant FM sets’ in these papers
Nom. logic	Schanuel topos / Cat. of nom. sets	[Pit01; Pit03]	States axioms of nominal sets, used word ‘nominal’
Nom. terms	Nom. sets	[UPG03; UPG04]	Introduced a, X, $a\#X$, $[a]X$, $\pi{\cdot}X$
Nom. rewriting	Nom. terms	[FGM04; FG07]	First framework for asserting general theories on nominal terms
αProlog	Nom. terms	[CU03; CU08]	Intended as logic programming language for abstract syntax, but can be viewed more generally
Nom. algebra	Nom. sets	[Gab05; GM06a; GM07; GM09a]	Axiomatisation & models for binders like $[a{\mapsto}t]$, λa, $\forall a$
Nom. equational logic	Nom. sets / Nom. Lawvere Theories	[CP07; Clo09]	Also provides semantics for nominal algebra
Permissive-nom. terms	Nom. sets or permissive-nom. sets	[DGM09; GM09b; DGM10]	Eliminate freshness contexts; add *shift*-permutation; standardise α-equivalence
Permissive-nom. algebra	Permissive-nom. sets or nom. sets	[GM09b], this survey	More expressive, esp. in presence of *shift*-permutation
Permissive-nom. logic	Permissive-nom. sets or nom. sets	[DG10; DG12a], this survey	A first-order logic for nom. terms

Figure 12: Cheat-sheet of nominal languages

INDEX