Theoretical Computer Science 451 (2012) 38–69

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

PNL to HOL: From the logic of nominal sets to the logic of higher-order functions

Gilles Dowek[a], Murdoch J. Gabbay[b,*]

[a] *INRIA, 23 avenue d'Italie, CS 81321, 75214 Paris Cedex 13, France*

[b] *School of Mathematical and Computer Sciences, Heriot–Watt University, Riccarton Edinburgh, EH14 4AS, Great Britain, United Kingdom*

A R T I C L E I N F O

Article history:
Received 11 July 2011
Received in revised form 24 February 2012
Accepted 2 June 2012
Communicated by P. L. Curien

Keywords:
Permissive-nominal logic
Higher-order logic
Nominal sets
Nominal renaming sets
Mathematical foundations of programming

A B S T R A C T

Permissive-Nominal Logic (PNL) extends first-order predicate logic with term-formers that can bind names in their arguments. It takes a semantics in (permissive-)nominal sets. In PNL, the ∀-quantifier or λ-binder are just term-formers satisfying axioms, and their denotation is functions on nominal atoms-abstraction.

Then we have higher-order logic (HOL) and its models in ordinary (i.e. Zermelo–Fraenkel) sets; the denotation of ∀ or λ is functions on full or partial function spaces.

This raises the following question: how are these two models of binding connected? What translation is possible between PNL and HOL, and between nominal sets and functions?

We exhibit a translation of PNL into HOL, and from models of PNL to certain models of HOL. It is natural, but also partial: we translate a restricted subsystem of full PNL to HOL. The extra part which does not translate is the symmetry properties of nominal sets with respect to permutations. To use a little nominal jargon: we can translate names and binding, but not their nominal equivariance properties. This seems reasonable since HOL – and ordinary sets – are not equivariant.

Thus viewed through this translation, PNL and HOL and their models do different things, but they enjoy non-trivial and rich subsystems which are isomorphic.

1. Introduction

Permissive-Nominal Logic (PNL) extends first-order predicate logic with name-binding term-formers. For instance first-order logic, set theory, and the untyped λ-calculus axiomatise in PNL; their binders ∀, comprehension, and λ are just modelled as binding PNL term-formers. The canonical semantics of PNL is in nominal sets, and it is first-order.

Higher-order logic (HOL) also has binding [46,17]. This has been used to encode other binders, e.g. the Church encoding of quantifiers as constants of higher type such as $\forall : (\iota\to o)\to o$ [2,6], higher-order abstract syntax (HOAS) encoding term-formers of an encoded syntax with binders as constants of higher type such as $\forall : (\iota\to\rho)\to\rho$ or $\forall : (\nu\to\rho)\to\rho$ (strong vs. weak HOAS)[1] [14,50], and higher-order rewrite systems [48].

* Corresponding author.
E-mail address: gilles.dowek@inria.fr (G. Dowek).
URLs: http://www-roc.inria.fr/who/Gilles.Dowek (G. Dowek), http://www.gabbay.org.uk (M.J. Gabbay).

[1] A word of clarification here: we take o to be a type of truth-values, ι to be a type of terms, and ρ to be a type of predicates. ∀-the-quantifier generates truth-values, whence the type headed by o, namely $\forall : (\iota\to o)\to o$. ∀-the-syntax-building-constant in HOAS generates *terms*, whence the types headed by ρ, namely $\forall : (\iota\to\rho)\to\rho$ or $\forall : (\nu\to\rho)\to\rho$. Do not confuse a HOL constant for a HOAS-style binder (a way to give meaning to building syntax with binding) with a HOL constant for the corresponding quantifier (a way to give meaning to what that syntax is intended to denote; namely, actual quantification).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.06.007

Since PNL is first-order and has a sound and complete semantics (so expressivity and models are fairly 'small'), whereas HOL is higher-order (so expressivity and models are fairly 'large'), the natural direction for a translation is from nominal sets and PNL, to functions and HOL (a *shallow embedding* of PNL into HOL).[2]

In this paper we translate a subsystem of PNL into HOL and prove it sound and complete using arguments on nominal sets and nominal renaming sets models [29]. The proof of completeness involves giving a functional semantics to nominal terms, and a nominal semantics to λ-terms in the spirit of Henkin models [2,3]. This involves a construction on nominal sets models corresponding to a free extension to *nominal renaming sets*, as previously considered by the second author with Hofmann [29].

The partiality of the translation of PNL seems to be inherent and reflects natural differences in structure between nominal and 'ordinary' sets; nominal sets are subject to the action of a *symmetry group* of atoms-permutations, which cannot be naturally represented in HOL or its 'ordinary' sets semantics. That is, it is not the case that nominal techniques are 'just' a concise presentation of HOL with a weakened β-equivalence (e.g. higher-order patterns [45]). There is that, but there is also more. The nominal and functional models of binding are distinct, but they do have non-trivial and rich subsystems which are isomorphic in a sense made precise in this paper.

1.1. Some background on PNL

We study PNL for its own sake in this paper, but the interested reader can find example nominal theories in the literature: for substitution, β-equivalence, and first-order logic [32,35,39,33,36].

These axiomatisations are in nominal algebra (which can be viewed as the equality fragment of PNL) and are accompanied by proofs of correctness in the respective papers.

Not all PNL theories are expressed in the equality fragment. For instance, in the papers which introduced PNL [9,10] we included theories of first-order logic and arithmetic which put universal quantification to the left of an implication.

To give some idea of what this family of logics looks like in practice, assume a name-sort ν and a base sort ι and term-formers $\mathsf{lam} : ([\nu]\iota)\iota$, $\mathsf{app} : (\iota, \iota)\iota$, and $\mathsf{var} : (\nu)\iota$. (Full definitions are in the body of the paper.) We sugar $\mathsf{lam}([a]r)$ to $\lambda a.r$ and $\mathsf{app}(r', r)$ to $r'r$ and $\mathsf{var}(a)$ to a. Atoms in PNL are a form of data and populate their own sort ν; so var serves to map them into the sort ι, where they represent object-level variables.

Here is η-equivalence, written out as it would be informally:

$$\lambda x.(tx) = t \quad \text{if } x \text{ is not free in } t$$

Here is a PNL axiom for η-equivalence, written out formally:

$$\forall Z.(\lambda a.(Za) = Z) \quad (a \notin pms(Z))$$

(See [39] for a detailed study of this axiom in a nominal context.)

a is an *atom* and corresponds to the *object-level variable* x; a is not a PNL variable but it *represents* a variable of the object level system being axiomatised. Z is an *unknown* and correspond to the *meta-level variable* t; Z is a variable in PNL and may be instantiated.

The reader can see how similar the two axioms look. Their status is different in the following sense: whereas t is typically taken to range over terms, Z ranges over elements of nominal sets (via a valuation; see Definition 6.3). This is possible because nominal sets have a notion of *supporting set of atoms* which mirrors the free variables of a term.

The condition $a \notin pms(Z)$ is a *typing condition* in PNL. The types, or *permission sets* as we call them, restrict the support of denotations associated to Z by a valuation. They correspond to freshness side-conditions in nominal terms from [55] and to informal freshness conditions of the form 'x not free in t' in informal practice. To see this intuition made formal see a translation from nominal terms to permissive-nominal terms in [13].

There is no requirement to axiomatise α-equivalence because this is done automatically by the PNL system.

For instance, axioms for β-equivalence [39] are:

$$\begin{array}{llll}
\forall Y. & (\lambda a.a)Y & = Y & \\
\forall Z, X. & (\lambda a.Z)X & = Z & (a \notin pms(Z)) \\
\forall X', X, Y. & (\lambda a.(X'X))Y & = ((\lambda a.X')Y)((\lambda a.X)Y) & \\
\forall X, Z. & (\lambda b.(\lambda a.X))Z & = \lambda a.((\lambda b.X)Z) & (a \notin pms(Z)) \\
\forall X. & (\lambda a.X)a & = X &
\end{array}$$

See [39] for a proof that these axioms really *do* axiomatise the λ-calculus.[3]

It is important to appreciate that most models of the axioms above are abstract and algebraic, not concrete and syntactic. So for instance the final axiom is not admissible because the objects X ranges over do not necessarily have inductive structure and we cannot necessarily push the β-redex down to the atoms.

2 A *deep embedding* e.g. of HOL in PNL is an answer to a different question; for more on this direction, see [38].

3 These axioms first appeared in [32,33] where a slightly different version of the final axiom was used. They are equivalent; see part 5 of Example 2.20 in this paper.

In particular, this is not a paper about representing syntax-with-binding, unlike the first applications of nominal techniques to syntax-with-binding [41]. True, we wrote above that atoms represent object-level variables. But indeed, they represent *variables*, not variable *symbols*. Nominal sets admit open elements and we can write axioms about names and binding in PNL, so that the full behaviour of variables is in PNL susceptible to algebraic axiomatisation. For instance the equality axioms above give atoms the behaviour of β-convertible λ-calculus variables; see [39] for a proof. Most models of the theory above are not built out of syntax.

Several nominal algebraic theories have been developed, with proofs of correctness: see [35] (substitution), [39] (λ-calculus) or [9,10] (first-order logic and arithmetic). For examples of natural non-syntactic models of nominal-style theories see [10, Subsection 5.2] and [24].

Thus, the design philosophy of PNL is that axioms should look like what we would write informally anyway, where variables map to atoms, meta-variables to unknowns, binding to atoms-abstraction, and capture-avoidance conditions to choice of permission sets. In particular, open terms and predicates map to elements and subsets of nominal sets with nonempty support.

1.2. *On symmetry*

The thing that goes missing in the translation from PNL to HOL is *name-symmetry*. Nominal sets are sets with an action of permutations of atoms; that is, nominal sets are sets with a symmetry action.

It turns out that this symmetry is key. For instance, we can define atoms-abstraction as a symmetric equivalence class (Definition 5.29, in this paper). This class is 'first-order' in flavour and does not involve the construction of a full function-space.

In PNL syntax permutative symmetry is reflected in the use of atoms and atoms-abstraction and by the permutations in terms (the interested reader could look at the 'symmetry-based' definition of α-equivalence in Definition 2.18).

In PNL *derivation* this is reflected in the axiom rule (rule (**Ax**) of Fig. 1 in this paper). This rule gives that $\phi \Rightarrow \pi\cdot\phi$ for all PNL predicates; so PNL predicates are fully symmetric up to permuting atoms. This is impossible to model in HOL because functional abstraction is asymmetric: obviously, $\lambda x.\lambda y.x$ is not equal to $\lambda y.\lambda x.x$.

Truth in *restricted PNL* (Fig. 2) is also asymmetric; restricted PNL is therefore a weaker system. It still has nominal semantics, but its predicates are not symmetric. This is the logic that we translate to HOL.

The symmetry of full PNL might seem counterintuitive—especially if the reader is used to modelling names as functional arguments (or indeed as numbers). We do not expect $\lambda x, y.P(x, y)$ to be symmetric with $\lambda y, x.P(x, y)$, or (for numbers) $x \leq y$ to be symmetric with $y \leq x$.

Consider the predicate $a = b$ in PNL (assume an equality, for the sake of argument). This is *false* because a is a distinct atom from b. If the reader is used to thinking of variables as things that 'vary' then $x = y$ might be either false or true depending the values associated to x and y. Not so in PNL: at the level of PNL syntax atoms are not variables; a and b are distinct; and their distinctness is symmetric up to permuting atoms so that $b = a$, $c' = a$, and $d = e$ are also false. If the reader is used to thinking of variables as numbers then there might exist some predicate $\leq$ that puts them in order. Not so in PNL: such a predicate is forbidden.[4] The unknowns X and Y in PNL do vary, and they are variables. More on this in Remark 2.22 and Section 2.6.

1.3. *Map of the paper*

This paper has a lot of technical ground to cover. This is unavoidable, because we need to deal with two logics (restricted PNL and HOL) and two semantics (nominal sets, and the hand-crafted Henkin models in nominal renaming sets used in the completeness proof), as well as two translations (from logic to logic, and from models to models).

For the reader's convenience, we provide an overview of the main technical points with brief justifications for their design:

- Section 2 introduces permissive-nominal logic. This comes from previous work into 'nominal' axiomatisations of systems with binding [9,10].[5]

  In fact, we need to introduce two logics: full PNL and also a *restricted* version which has a weaker non-equivariant axiom rule. We write the entailment relations $\vdash$ and $\stackrel{\pi}{\vdash}$ respectively. It is the restricted version that we will eventually translate to HOL.
- Section 3 introduces higher-order logic as a theory over the syntax of the simply-typed λ-calculus. We write the entailment relation $\stackrel{\lambda}{\vdash}$.

4 This is because we used all finite permutations of atoms as our symmetry group. Generalisations of this as suggested e.g. in [4, Subsection 3.1] are possible. See also Remark 2.1.8 of [27].

5 Note that PNL is not only about nominal abstract syntax as considered in e.g. [41,23]. Nominal abstract syntax is a denotation for syntax with binding. PNL and its models are a (more general) syntax and semantics for denotations with binding in general, which are not all necessarily datatypes of abstract syntax.

- Section 4 defines the translation from restricted PNL to HOL, and proves it sound using arguments on syntax. In order to do the translation, we need to introduce a *capture typing* $D \vdash r : A$ which is a measure of how many functional abstractions are required to translate a given nominal term without losing information; that is, of the functional complexity of a nominal term.
- Our goal is then to prove completeness of the translation. We do this by transforming models of PNL into models of HOL. So Section 5 introduces two categories: PmsPrm of permissive-nominal sets and PmsRen of permissive-nominal renaming sets. We also give a *free* construction, transforming a permissive-nominal set into a permissive-nominal renaming set.
- In Section 6 we interpret full and restricted PNL in PmsPrm. In Section 7 we interpret HOL in PmsRen.
- Finally, in Section 8 we use the free construction of Section 5 to map a model of PNL in PmsPrm to a model in PmsRen, and because the free construction does not 'make anything equal' this is sufficient to prove completeness.
- As one further mathematical note, the results in the literature concern full PNL and not restricted PNL. So in Appendix we sketch proofs of soundness, cut-elimination, and completeness of restricted PNL with respect to non-equivariant models in PmsPrm. These are modest, if not entirely direct, modifications of the existing definitions and proofs for full PNL and equivariant models in PmsPrm.

Quite a number of new ideas are required to make this all work. The highlights are: permissive-nominal renaming sets and their application to give non-standard 'nominal' Henkin models for higher-order logic; restricted PNL and its semantics; the free construction; and the technical arguments as discussed in Section 8.

1.4. Review of motivation

Given that the proofs and constructions in this paper are non-trivial and involve an effort to extend existing machinery, we should pause to ask again why doing this is justified, even necessary.

Nominal techniques were designed originally to reason on syntax-with-binding (see the original journal paper [41] or a recent survey paper [23]). But since then this remit has expanded to reasoning about denotations with binding more generally (an overview of which is in [27]). In doing this, we have created a whole new syntax and semantics for meta-mathematics.

We will not argue for or against either the nominal foundation or the higher-order foundation for mathematics.[6] Our question is: given that these two foundations exist, how do they relate?

In fact, questions have been asked about how nominal names and binding are related to functions, ever since nominal techniques were conceived in the second author's thesis. Since then, the development of PNL [10] and nominal renaming sets [29] has given us two powerful new tools with which to address these questions: a proof-theory for a logic in which nominal reasoning so far can be formalised, and a visibly nominal semantics which is not based on permutations but on possibly non-bijective renamings on atoms, so that atoms-abstraction can be considered as a function in that semantics.

In this paper, we leverage this to give a precise, concrete, and mathematically detailed account of how these two worlds really stand in relation to one another—and how they differ. In conclusion we speculate that there is some potential (not explored in this paper) that our translations might be used to piggyback nominal techniques on the substantial implementational efforts that have gone into developing HOL over the past seventy years.

2. Permissive-nominal logic

Permissive-nominal logic is a first-order logic for nominal terms quotiented by α-equivalence. Doing this is not entirely trivial; the interested reader can find more on this elsewhere [55,9,10,27].

2.1. Syntax

Definition 2.1. A **sort-signature** is a pair $(\mathcal{A}, \mathcal{B})$ of **name** and **base sorts**. ν will range over name sorts; τ will range over base sorts. A **sort language** is then defined by

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

Remark 2.2. Examples of base sorts are: 'λ-terms', 'formulae', 'π-calculus processes', and 'program environments', 'functions', 'truth-values', 'behaviours', and 'valuations'.

Examples of name sorts are 'variable symbols', 'channel names', or 'memory locations'.

$[\nu]\alpha$ is an *abstraction sort*. This does a similar job to function-types in higher-order logic but note that ν must always be a name-sort. The behaviour of a term of sort $[\nu]\alpha$ corresponds to 'bind a name of sort ν in a term of sort α'. Such a term does not denote a function, though later on in our completeness proof we will deliberately undermine that intuition to obtain our completeness result.

6 There has been more than enough of that already, and anyway, because truth is free, proving theorems is never a zero sum game.

Definition 2.3. For each ν fix a disjoint countably infinite set of **atoms** $\mathbb{A}_\nu$, and an arbitrary bijection f_ν between $\mathbb{A}_\nu$ and the integers $\mathbb{Z} = \{0, -1, 1, -2, 2, \ldots\}$. Write

$$\mathbb{A}_\nu^< = \{f_\nu(i) \mid i < 0\} \qquad \mathbb{A}_\nu^> = \{f_\nu(i) \mid i \geq 0\}$$

Finally, write

$$\mathbb{A}^< = \bigcup \mathbb{A}_\nu^< \qquad \mathbb{A}^> = \bigcup \mathbb{A}_\nu^> \qquad \mathbb{A} = \bigcup \mathbb{A}_\nu$$

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

A **permission set** has the form $(\mathbb{A}^< \cup A) \setminus B$ where $A \subseteq \mathbb{A}^>$ and $B \subseteq \mathbb{A}^<$ are finite (and a permission set may be finitely represented by the pair (A, B)). S, T, and U will range over permissions sets.

The use of $\mathbb{A}^<$ and $\mathbb{A}^>$ ensures that permission sets are infinite and also coinfinite (their complement is also infinite).

Remark 2.4. Permission sets are simple, but surprisingly subtle.

$\mathbb{A}^<$ and $\mathbb{A}^>$ are 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 [28, Section 1], and the *individual variables* and *parameters* of Prawitz [52, Section 1], or Smullyan [53, Chapter IV, Section 1].

However, for any given atom there is no *fixed* sense in which it is either capturable or not capturable. Each permission set defines a world of capturable/non-capturable atoms. Furthermore, even once a permission set is fixed, we shall see that permutations can shift an atom from one to the other. See Example 2.20 and Remark 2.28 for examples of permission sets in action, controlling α-equivalence and substitution respectively.

We make $\mathbb{A}^<$ and $\mathbb{A}^>$ both infinite so that we have inexhaustible supplies of both kinds of atom.[7] Further technical discussion of the advantages of permissive-nominal techniques is in [13].

Definition 2.5. A **term-signature** over a sort-signature $(\mathcal{A}, \mathcal{B})$ is a tuple $(\mathcal{F}, \mathcal{P}, ar, \mathcal{X})$ where:

- $\mathcal{F}$ and $\mathcal{P}$ are disjoint sets of **term-** and **proposition-formers**.
  f will range over term-formers. P will range over proposition-formers.
- ar assigns 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 $(\mathcal{A}, \mathcal{B})$.
  We will write $((\alpha_1, \ldots, \alpha_n))\tau$ just as $(\alpha_1, \ldots, \alpha_n)\tau$.
- $\mathcal{X}$ is a set of **unknowns** X, each of which has a sort $sort(X)$ and a permission set $pms(X)$, such that for each sort α and permission set S the set $\{X \in \mathcal{X} \mid sort(X) = \alpha, pms(X) = S\}$ is countably infinite. X, Y, Z will range over distinct unknowns.

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

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 2.6. The signature for the λ-calculus from the Introduction has a name-sort for λ-calculus object-level variables ν, a base sort ι for λ-terms, and appropriate term-formers:

- $\mathsf{var} : (\nu)\iota$ to form λ-calculus variables in ι out of names in ν,
- $\mathsf{app} : (\iota, \iota)\iota$ for application, and
- $\mathsf{lam} : ([\nu]\iota)\iota$ taking an abstraction in $[\nu]\iota$ and forming from it a λ-abstraction term in ι.

Definition 2.7. A **permutation** is a bijection π on $\mathbb{A}$ such that $a \in \mathbb{A}_\nu \Leftrightarrow \pi(a) \in \mathbb{A}_\nu$ and $nontriv(\pi) = \{a \mid \pi(a) \neq a\}$ is finite. Write $\mathbb{P}$ for the set of permutations.
Given $a, b \in \mathbb{A}_\nu$ let a **swapping** $(a\ b)$ be the bijection on atoms that maps a to b, b to a, and all other c to themselves.

Notation 2.8. We use the following notation:

- Write $\pi \circ \pi'$ for **functional composition**, so $(\pi \circ \pi')(a) = \pi(\pi'(a))$.
- Write id for the **identity permutation**, so $id(a) = a$ always.
- Write π^{-1} for **inverse**, so $\pi \circ \pi^{-1} = id$.

[7] For comparison, nominal terms have only a finite supply of fresh atoms. The effect of this is that the nominal terms of [55] cannot be quotiented by α-equivalence as primitive, and the freshness context may need to be extended dynamically with fresh names. This introduces an 'impure' flavour of state and sequentiality into the theory of nominal terms which is absent from the permissive-nominal version. In short, making permission sets infinite and coinfinite makes the whole theory noticeably more 'pure'.

Definition 2.9. For each signature $\mathcal{S}$, define **terms** and **propositions** over $\mathcal{S}$ by:

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

$$\dfrac{r : \alpha \quad (a \in \mathbb{A}_\nu)}{[a]r : [\nu]\alpha} \qquad \dfrac{(sort(X) = \alpha)}{\pi{\cdot}X : \alpha}$$

$$\dfrac{}{\bot \text{ prop.}} \qquad \dfrac{\phi \text{ prop.} \ \ \psi \text{ prop.}}{\phi \Rightarrow \psi \text{ prop.}} \qquad \dfrac{r : \alpha \ \ (ar(\mathsf{P}) = \alpha)}{\mathsf{P}(r) \text{ prop.}}$$

$$\dfrac{\phi \text{ prop.}}{\forall X.\phi \text{ prop.}}$$

Example 2.10. Continuing Example 2.6, we have the following terms and propositions:

- $\mathsf{var}(a) : \iota$ where $a \in \mathbb{A}_\nu$.
- $[a]X : [\nu]\iota$ where $a \in \mathbb{A}_\nu$ and $sort(X) = \iota$.
- $\mathsf{lam}([a]X) : \iota$ since $[a]X : [\nu]\iota$ and $\mathsf{lam} : ([\nu]\iota)\iota$.
- $\forall X.\mathsf{P}(\mathsf{lam}([a]X), X)$ is a proposition if P is a proposition-former and $\mathsf{P} : (\iota, \iota)$.

2.2. Permutation, substitution, and so on

These definitions are all needed for the rest of the paper, starting with α-equivalence in Section 2.3. We need them at both levels; both for atoms and for unknowns.

Definition 2.11. Define a (level 1) **permutation action** on syntax by:

$$\begin{array}{rlrl}
\pi{\cdot}a &= \pi(a) & \pi{\cdot}(r_1, \ldots, r_n) &= (\pi{\cdot}r_1, \ldots, \pi{\cdot}r_n) \\
\pi{\cdot}[a]r &= [\pi(a)]\pi{\cdot}r & \pi{\cdot}(\pi'{\cdot}X) &= (\pi \circ \pi'){\cdot}X \\
\pi{\cdot}\mathsf{f}(r) &= \mathsf{f}(\pi{\cdot}r) & & \\
\pi{\cdot}\bot &= \bot & \pi{\cdot}(\phi \Rightarrow \psi) &= (\pi{\cdot}\phi) \Rightarrow (\pi{\cdot}\psi) \\
\pi{\cdot}\mathsf{P}(r) &= \mathsf{P}(\pi{\cdot}r) & \pi{\cdot}(\forall X.\phi) &= \forall X.\pi{\cdot}\phi
\end{array}$$

Definition 2.12. Let Π range over sort- and permission-set-preserving bijections on unknowns (so $sort(\Pi(X)){=}sort(X)$ and $pms(\Pi(X)){=}pms(X)$) such that $\{X \mid \Pi(X) \neq X\}$ is finite.

Write $\Pi \circ \Pi'$ for functional composition, Id for the identity permutation, and Π^{-1} for inverse, much as in Notation 2.8.

Define a (level 2) **permutation action** by:

$$\begin{array}{rlrl}
\Pi{\cdot}a &= a & \Pi{\cdot}(r_1, \ldots, r_n) &= (\Pi{\cdot}r_1, \ldots, \Pi{\cdot}r_n) \\
\Pi{\cdot}[a]r &= [a]\Pi{\cdot}r & \Pi{\cdot}(\pi{\cdot}X) &= \pi{\cdot}(\Pi(X)) \\
\Pi{\cdot}\mathsf{f}(r) &= \mathsf{f}(\Pi{\cdot}r) & & \\
\Pi{\cdot}\bot &= \bot & \Pi{\cdot}(\phi \Rightarrow \psi) &= (\Pi{\cdot}\phi) \Rightarrow (\Pi{\cdot}\psi) \\
\Pi{\cdot}\mathsf{P}(r) &= \mathsf{P}(\Pi{\cdot}r) & \Pi{\cdot}(\forall X.\phi) &= \forall \Pi(X).\Pi{\cdot}\phi
\end{array}$$

Remark 2.13. A curious asymmetry between Definitions 2.11 and 2.12 is that $\pi{\cdot}(\forall X.\phi) = \forall X.\pi{\cdot}\phi$ but $\Pi{\cdot}(\forall X.\phi) = \forall \Pi(X).\Pi{\cdot}\phi$. Note that π not applied to the binding occurrence of X, but Π is.

In fact, we *could* take $\pi{\cdot}(\forall X.\phi) = \forall \pi{\cdot}X.\pi{\cdot}\phi$. We would have to complicate Definition 2.9 by introducing $\forall \pi{\cdot}X.\phi$ as well-formed syntax, but we could do this. However, it turns out that this would make no difference. It turns out that $\forall \pi{\cdot}X.\phi$ and $\forall X.\phi$ are logically equivalent.

To gain a quick intuitive understanding of why this is so, bear in mind that the substitution 'X maps to r' maps $\pi{\cdot}X$ to $\pi{\cdot}r$ (this is made formal later, in Definition 2.26) and in fact '$\pi{\cdot}X$ maps to $\pi{\cdot}r$' would map X to r (the interested reader can find a wider discussion of this in and around [27, Remark 3.4.7]).

So, rather curiously, $\forall \pi{\cdot}X.\phi$ means the same thing and would receive the same denotation, regardless of π: this would be the denotation of the PNL proposition $\forall X.\phi$. We can therefore simplify our syntax and take $\pi{\cdot}(\forall X.\phi) = \forall X.\pi{\cdot}\phi$.

Something similar happens in the much more abstract semantic context of two-level nominal sets; see [25, Lemma 2.25]. Further comments on asymmetries between atoms and unknowns in PNL will follow in Remark 2.22.

Definition 2.14. Suppose f is a function on a set X and $U \subseteq X$. Define $f{\cdot}U$ by

$$f{\cdot}U = \{f(x) \mid x \in U\}$$

This is the standard **pointwise** action of a function on a set. We use this for π acting on sets of atoms, Π acting on sets of unknowns, and (from Definition 5.1 onwards) ρ acting on sets of atoms.

Definition 2.15. Define **free atoms** $fa(r)$ and $fa(\phi)$ by:

$$fa(\pi\cdot X) = \pi\cdot pms(X) \qquad fa([a]r) = fa(r)\setminus\{a\} \qquad fa(a) = \{a\}$$
$$fa(\mathsf{f}(r)) = fa(r) \qquad fa((r_1,\dots,r_n)) = \bigcup fa(r_i)$$

$$fa(\bot) = \varnothing \qquad fa(\phi \Rightarrow \psi) = fa(\phi)\cup fa(\psi)$$
$$fa(\mathsf{P}(r)) = fa(r) \qquad fa(\forall X.\phi) = fa(\phi)$$

Define **free unknowns** $fU(r)$ and $fU(\phi)$ by:

$$fU(a) = \varnothing \qquad fU(\pi\cdot X) = \{X\} \qquad fU(\mathsf{f}(r)) = fU(r)$$
$$fU([a]r) = fU(r) \quad fU((r_1,\dots,r_n)) = \bigcup fU(r_i)$$

$$fU(\bot) = \varnothing \qquad fU(\phi \Rightarrow \psi) = fU(\phi)\cup fU(\psi)$$
$$fU(\mathsf{P}(r)) = fU(r) \qquad fU(\forall X.\phi) = fU(\phi)\setminus\{X\}$$

Lemma 2.16. *$fa(\pi\cdot r) = \pi\cdot fa(r)$ and $fa(\pi\cdot\phi) = \pi\cdot fa(\phi)$.*
Also, $fU(\Pi\cdot r) = \Pi\cdot fU(r)$ and $fU(\Pi\cdot\phi) = \Pi\cdot fU(\phi)$.

Proof. By routine inductions on r. □

2.3. *α-equivalence*

The use of permissive-nominal terms allows us to 'just quotient' syntax by α-equivalence. We can do this for both level 1 variable symbols (atoms) and level 2 variable symbols (unknowns).

Definition 2.17. Call a relation $\mathcal{R}$ on terms and on propositions a **congruence** when it is closed under the following rules[8]:

$$\frac{r_i \mathrel{\mathcal{R}} s_i \quad 1\le i\le n}{(r_1,\dots,r_n) \mathrel{\mathcal{R}} (s_1,\dots,s_n)} \qquad \frac{r \mathrel{\mathcal{R}} s \quad (\mathsf{f} : (\alpha)\tau,\ r,s:\alpha)}{\mathsf{f}(r) \mathrel{\mathcal{R}} \mathsf{f}(s)}$$

$$\frac{r \mathrel{\mathcal{R}} s}{[a]r \mathrel{\mathcal{R}} [a]s} \qquad \frac{\phi \mathrel{\mathcal{R}} \phi' \quad \psi \mathrel{\mathcal{R}} \psi'}{\phi\Rightarrow\psi \mathrel{\mathcal{R}} \phi'\Rightarrow\psi'}$$

$$\frac{r \mathrel{\mathcal{R}} s \quad (\mathsf{P} : \alpha,\ r,s:\alpha)}{\mathsf{P}(r) \mathrel{\mathcal{R}} \mathsf{P}(s)} \qquad \frac{\phi \mathrel{\mathcal{R}} \phi'}{\forall X.\phi \mathrel{\mathcal{R}} \forall X.\phi'}$$

Definition 2.18. Write $(a\ b)$ for the **(level 1) swapping** permutation which maps a to b and b to a and all other c to themselves. Similarly, provided $sort(X) = sort(Y)$ and $pms(X) = pms(Y)$, write $(X\ Y)$ for the **(level 2) swapping**.

Define α-**equivalence** $=_\alpha$ on terms and propositions to be the least equivalence relation that is a congruence and is such that:

$$\boxed{\quad \frac{(a, b \notin fa(r))}{(b\ a)\cdot r =_\alpha r} \qquad \frac{(X, Y \notin fU(\phi))}{(Y\ X)\cdot\phi =_\alpha \phi} \quad}$$

Remark 2.19. Definition 2.18 is inductive, but the reader familiar with nominal terms from e.g. [55,13,10] might be familiar with a more syntax-directed inductive characterisation whose characteristic rules for atoms would look like this in our current notation:

$$\frac{(b \notin fa(r))}{[a]r =_\alpha [b](b\ a)\cdot r} \qquad \frac{(\forall a.\pi(a) \neq \pi'(a) \Rightarrow a \notin pms(X))}{\pi\cdot X =_\alpha \pi'\cdot X}$$

The form of Definition 2.18 is more compact and more abstract. It was introduced in [32,34] (in particular see [34, Lemma 3.2] and the discussion surrounding it); a detailed proof of an equivalence of the two presentations is in [35, Theorem 2.31]

Example 2.20. We illustrate Definition 2.18. Suppose $a, b, c, d : \nu$ and $X, Y : \tau$ for some name sort ν and base sort τ. Also suppose $a, b, c, d \notin pms(Y)$ and suppose $b \notin pms(X)$.

1. *We α-convert a and b in $[a][b]a$.*
   First we note that $b, d \notin fa([b]a)$ so by symmetry $[b]a =_\alpha (b\ d)\cdot[b]a = [d]a$, and by congruence $[a][b]a =_\alpha [a][d]a$. Next we note that $a, c \notin fa([a][d]a)$ so by symmetry $[a][d]a =_\alpha (c\ a)\cdot[a][d]a = [c][d]c$. We use transitivity.

[8] We do not assume a congruence is an equivalence relation. We prefer to keep the notion of 'being preserved by all term-formers' orthogonal to the notion of 'being transitive, reflexive, and symmetric'. For instance, we want a rewrite relation to have the first property but not the second. In [15] the second author considers combining sequents with rewriting. This is *deduction modulo*; a nominal version of deduction modulo is future work and is one of the background motivations for the creation of PNL.

2. *We α-convert $[a][a]b$ to $[c][d]b$.*
   We reason as follows: $[a][a]b =_\alpha [a][d]b =_\alpha [c][d]b$.
3. *We α-convert $((a\ b) \circ (c\ d))\cdot Y$ to Y.* First we note that $a, b \notin fa((c\ d)\cdot Y) = (c\ d)\cdot pms(Y)$, so $((a\ b) \circ (c\ d))\cdot Y = (a\ b)\cdot((c\ d)\cdot Y) =_\alpha (c\ d)\cdot Y$. Then we note that $c, d \notin fa(Y)$ so $(c\ d)\cdot Y =_\alpha Y$. We use transitivity.
4. *We α-convert X and a in $\forall X.\mathsf{P}([a]X)$.*
   Using $(a\ b)$ and $(X\ Y)$ we deduce:
   $$\forall X.\mathsf{P}([a]X) \overset{(a\ b)}{=_\alpha} \forall X.\mathsf{P}([b](b\ a)\cdot X) \overset{(X\ Y)}{=_\alpha} \forall Y.\mathsf{P}([b](b\ a)\cdot Y)$$
   It is routine to convert this sketch into a full derivation-tree.
5. In [32,33] an axiom of substitution/β-equivalence was stated which, translated to a permissive-nominal syntax, is expressed thus:
   $$(\mapsto\mathbf{ren}) \qquad (\lambda([a]X))b = (b\ a)\cdot X \qquad (b \notin pms(X))$$
   Since $a, b \notin fa([a]X)$ we have $[a]X =_\alpha [b](b\ a)\cdot X$. Writing X' for an unknown with $pms(X') = (b\ a)\cdot pms(X)$ we can therefore rephrase this axiom as follows:
   $$(\mapsto\mathbf{id}) \qquad (\lambda([b]X'))b = X' \qquad (b \notin pms(X'))$$
   This is the form that the same axiom took in [39]; that paper used the slightly less flexible nominal terms framework, so the equivalence noted above had to be the subject of a proof [39, Lemma 2.16].

It is not hard to prove by inductive arguments that the definition of α-equivalence here gives the same result as that found for instance in [10, Subsection 2.4] or [9, Definition 2.16].[9]

Definition 2.21. For each signature $\mathcal{S}$, we take terms and propositions quotiented by α-equivalence.

Remark 2.22. Atoms and unknowns both have permutation actions and both participate in α-conversion and both 'look like' variable symbols. Indeed, in the translation to HOL of Section 4.1 atoms and unknowns *both* are translated to variables. However, in PNL syntax atoms and unknowns are very distinct:

- Atoms populate only their own special name-sorts ν. Unknowns populate any sort.
- Unknowns depend on atoms in the sense that $pms(X)$ is a set of atoms. Atoms do not depend on anything.
- Atoms and unknowns both get permuted in syntax but only unknowns also have a substitution action (defined next in Section 2.4). Conversely, only atoms-permutation is explicit in the syntax, as $\pi\cdot X$ (there is no Π in any r, only an action of Π *on* r).
- If ϕ is proposition then $\forall X.\phi$ is a proposition, but $\forall a.\phi$ is never well-formed syntax.
- Later on when we build the denotation in Section 6.2, atoms are interpreted as themselves (so $[\![a]\!]^\imath_\varsigma = a$) whereas unknowns are interpreted via a valuation, in typical Tarski style (so $[\![X]\!]^\imath_\varsigma = \varsigma(X)$).

Why this asymmetry?

- PNL is a first-order logic. Therefore it has a syntactic class of variables which we call *unknowns* X, with an unknowns-substitution action $[X{:=}r]$ (Definition 2.27), a universal quantifier $\forall X$, α-conversion, and a denotation using valuations for unknowns $[\![X]\!]^\imath_\varsigma = \varsigma(X)$ (Definition 6.4).
- PNL treats atoms (names) as a special kind of symmetric data (i.e. data with a group action). It has sorts for atoms ν which are populated by atoms-as-terms, a nominal-terms style explicit permutation $\pi\cdot X$ to permute atoms in terms, atoms-abstraction $[a]r$ to bind atoms in terms, and a nominal sets style semantics for atoms as themselves; so that $[\![a]\!]^\imath_\varsigma = a$ and $[\![[a]r]\!]^\imath_\varsigma = [a][\![r]\!]^\imath_\varsigma$ (Definition 6.4).

PNL is expressive enough to axiomatise a substitution action for atoms, and even a universal quantifier for atoms. Thus we can make atoms behave like first-order logic variables, if we want to do this. For more on these and other nominal theories see [35] or Figs. 4 and 5 of [9,10].

The asymmetry noted above reflects the standard design of first-order logic: unknowns 'live in' ϕ in the sense that they are used to make universal assertions $\forall X.\phi$, and atoms 'live in' r in the sense that they interact with the term- and sort-system and we can form $[a]r$ and $\mathsf{f}(a)$.

What can be a little confusing is that can mimic variables, and indeed, we can and do use PNL to axiomatise logic. In short: PNL is a first-order logic for axiomatising logics.

Remark 2.23. We can still ask to what extent it might be possible to go beyond PNL and to 'fold' unknowns back down into the syntax, possibly recursively, to obtain some language and/or semantics in which atoms and unknowns are just two aspects of a single well-founded structure.

In fact this idea predates PNL: the Lambda Context Calculus (LamCC) [30,31] features just such an infinite hierarchy, but it has no semantic theory, no primitive notion of proposition, and has a weak notion of α-equivalence.[10] Concurrently with

[9] In fact, this was the characterisation used to *design* the 'exotic' multi-level α-equivalences of permissive-nominal logic or two-level nominal sets [25].

[10] In the LamCC, atoms are level 1 variables, unknowns are level 2 variables and behave much like 'holes' in a program context, and there are level 3 variables, and so on. Many interesting program constructs can be expressed in this language.

PNL, in two recent papers [25,26] the first author has investigated these questions from a semantic perspective. In [25] we develop an abstract notion of two-level nominal set that can directly interpret unknowns X and atoms a just as the nominal sets of this paper directly interpret atoms a. In [26] we develop a concrete model of unknowns as infinite lists of distinct atoms. It would be future work to integrate these semantics into a new logic.[11] A PNL-like logic with more than two levels of variable and a perfect symmetry across levels, is certainly imaginable.

2.4. Substitution

Definition 2.24. A (level 2) **substitution** θ is a function from unknowns to terms such that:

- For all X, $\theta(X) : sort(X)$ and $fa(\theta(X)) \subseteq pms(X)$.
- $\theta(X) = id{\cdot}X$ for all but finitely many X.

θ will range over substitutions.

Definition 2.25. Define $nontriv(\theta)$ by:

$$nontriv(\theta) = \{X \mid \theta(X) \neq id{\cdot}X \text{ or } X \in fU(\theta(Y)) \text{ for some } Y\}$$

$nontriv(\theta)$ is unknowns that can be produced or consumed by θ, other than in the trivial manner that $\theta(X) = id{\cdot}X$.

Definition 2.26. Define a **substitution action** by:

$$\begin{array}{rlrl} a\theta &= a & (r_1,\ldots,r_n)\theta &= (r_1\theta,\ldots,r_n\theta) \\ ([a]r)\theta &= [a](r\theta) & (\pi{\cdot}X)\theta &= \pi{\cdot}\theta(X) \\ \mathsf{f}(r)\theta &= \mathsf{f}(r\theta) & & \\ \bot\theta &= \bot & (\phi \Rightarrow \psi)\theta &= (\phi\theta) \Rightarrow \psi\theta \\ (\mathsf{P}(r))\theta &= \mathsf{P}(r\theta) & (\forall X.\phi)\theta &= \forall X.(\phi\theta) \quad (X \notin nontriv(\theta)) \end{array}$$

One kind of substitution will be particularly useful, starting with ($\forall$**L**) in Fig. 1:

Definition 2.27. Suppose $X : \alpha$ and $r : \alpha$ and $fa(r) \subseteq pms(X)$. Define $[X{:=}r]$ by:

$$[X{:=}r](X) = r \qquad [X{:=}r](Y) = Y \quad \text{all other } Y$$

It is easy to verify that $[X{:=}r]$ is indeed a substitution.

Remark 2.28. Famously, nominal terms substitution is capturing [55, Definition 2.13]. We spell out how this works in our permissive-nominal context: Suppose $a \in pms(X)$ and $b \notin pms(X)$ (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 a substitution, since $b \notin pms(X)$. So $b \notin pms(X)$ cannot be captured by a substitution $[X{:=}b]$, because that substitution does not exist.
- Also, $[b](b\ a){\cdot}X = [a]X$. By construction,

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

  So the choice of representative of $[a]X$ does not matter for capture to occur.

In [26] we propose a view of X as a well-ordering on its permission set; that is, we identify X literally with an infinite list of atoms. Viewed from this perspective, the nominal substitution action is not capturing at all: it is simply a compact way to present an 'infinite raising' or 'infinite Skolemisation' (cf. Remark 4.5), or a de Bruijn index [8]. This idea underlies also the translation to HOL which we construct later in Definition 4.3.

2.5. Sequents and derivability

Definition 2.29. Φ and Ψ will range over sets of propositions. We may write ϕ, Φ and Φ, ϕ as shorthand for $\{\phi\} \cup \Phi$ (where we do not insist that $\phi \notin \Phi$, that is, the union need not be disjoint).

- A **sequent** of restricted PNL is a pair $\Phi \overset{\not\pi}{\vdash} \Psi$.
- A **sequent** of full PNL is a pair $\Phi \vdash \Psi$.

Write $fU(\Phi, \Psi) = \bigcup\{fU(\phi) \mid \phi \in \Phi\} \cup \bigcup\{fU(\psi) \mid \psi \in \Psi\}$.

[11] The second author is most interested in PNL as a first-order basic for specifying logic and computation in theorem-proving and also in undergraduate teaching. Fancier logics and semantics than PNL can certainly be imagined, and perhaps created, but this does not undermine the interest of a 'simple' logic, like PNL.

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

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

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

Fig. 1. Sequent derivation rules of full Permissive-Nominal Logic.

$$\frac{}{\Phi,\ \phi \vDash^{\not\pi} \phi,\ \Psi}\,(\mathbf{Ax}^{\not\pi}) \qquad \frac{}{\Phi,\ \bot \vDash^{\not\pi} \Psi}\,(\bot\mathbf{L})$$

$$\frac{\Phi \vDash^{\not\pi} \phi,\ \Psi \quad \Phi,\ \psi \vDash^{\not\pi} \Psi}{\Phi,\ \phi \Rightarrow \psi \vDash^{\not\pi} \Psi}\,(\Rightarrow\mathbf{L}) \qquad \frac{\Phi,\ \phi \vDash^{\not\pi} \psi,\ \Psi}{\Phi \vDash^{\not\pi} \phi \Rightarrow \psi,\ \Psi}\,(\Rightarrow\mathbf{R})$$

$$\frac{\begin{array}{c}\Phi,\ \phi[X{:=}r] \vDash^{\not\pi} \Psi \\ (fa(r) \subseteq pms(X),\ r{:}sort(X))\end{array}}{\Phi,\ \forall X.\phi \vDash^{\not\pi} \Psi}\,(\forall\mathbf{L}) \qquad \frac{\Phi \vDash^{\not\pi} \phi,\ \Psi \quad (X \notin fU(\Phi,\Psi))}{\Phi \vDash^{\not\pi} \forall X.\phi,\ \Psi}\,(\forall\mathbf{R})$$

Fig. 2. Sequent derivation rules of restricted Permissive-Nominal Logic.

Definition 2.30 (*Derivable Sequents*)**.** Define the **derivable sequents** of full PNL and restricted PNL by the rules in Figs. 1 and 2 respectively.

The sole difference between Figs. 1 and 2 is in the axiom rule, and is highlighted with a light blue rectangle.

Notation 2.31. We may write $\Phi \vDash^{\not\pi} \Psi$ as shorthand for '$\Phi \vDash^{\not\pi} \Psi$ is a derivable sequent'. We may write $\Phi \not\vDash^{\not\pi} \Psi$ as shorthand for '$\Phi \vDash^{\not\pi} \Psi$ is not a derivable sequent'.

Similarly for $\Phi \vdash \Psi$ and $\Phi \nvdash \Psi$.

2.6. Discussion of the PNL axiom rule

Fig. 1 is the logic of [10,27]. Fig. 2 is the logic we translate to HOL in this paper. The only difference is the 'π' in the axiom rule: full PNL has it (see (**Ax**)), and restricted PNL does not (see ($\mathbf{Ax}^{\not\pi}$)). Restricted PNL is a subset of full PNL, in the sense that (obviously) $\Phi \vDash^{\not\pi} \Psi$ implies $\Phi \vdash \Psi$ (this suggests that the models of restricted PNL should be a superset of those of full PNL, which will indeed turn out to be the case; see Appendix).

Why the difference? Because the translation to HOL identifies atoms with functional arguments. Atoms are symmetric up to permutation in full PNL; this is built into (**Ax**) in Fig. 1. Functional arguments are typically not symmetric.

We might try to translate full PNL to HOL by translating $n!$ permutation instances of each r or ϕ, where n is some notion of the number of atoms in r or ϕ (cf. *capture typings* in Definition 4.7); but that would be 'cheating' in the sense that most of the syntax would then be generated by a meta-level 'macro' which does $n!$ amount of work. The issue here is not whether PNL can be encoded in HOL; the issue is whether it can be cleanly translated into HOL. These are related but distinct questions.

To quickly see the difference in derivational power between full and restricted PNL, assume a name sort ν, a proposition-former $\mathsf{P} : \nu$, and two atoms $a, b : \nu$. Then the difference in the entailment relations of PNL and restricted PNL can be summed up as follows:

- $\mathsf{P}(a) \vdash \mathsf{P}(a)$ and $\mathsf{P}(a) \vdash \mathsf{P}(b)$.
- $\mathsf{P}(a) \vDash^{\not\pi} \mathsf{P}(a)$ but ~~$\mathsf{P}(a) \vDash^{\not\pi} \mathsf{P}(b)$~~.

Note that not even full PNL can derive that $Q(a, b)$ entails $Q(a, a)$; we can permute, but we have to permute in the *entire* proposition. So for instance if we axiomatise the syntax and derivability of first-order logic in PNL then we would have a predicate $\mathsf{entails}$ and we might prove $\mathsf{entails}(\mathsf{P}(a), \mathsf{P}(a))$. By equivariance of full PNL we could also derive $\mathsf{entails}(P(b), \mathsf{P}(b))$.[12] However, we still cannot derive ~~$\mathsf{entails}(\mathsf{P}(a), \mathsf{P}(b))$~~, and if we could then that would be wrong.

[12] This has actually been used in real proofs; starting with [20] we used what amounts to (**Ax**) to rename variable symbols in inductive proofs. See [23, Subsection 4.2] for this *equivariance* principle discussed as a practical tool to obtain one-line proofs at the rigorous but informal meta-level of papers.

In Appendix we see that this difference corresponds in models to proposition-formers being interpreted by equivariant functions (for full PNL) or not necessarily equivariant functions (for restricted PNL).

It has to be this way: Definition 4.3 translates PNL terms and predicates to HOL terms and predicates. In Lemma 4.17 we illustrate why only restricted PNL can be translated to HOL by our translation: the derivability of full PNL is too strong for HOL derivability and the translation would not be sound.

Note that this does not prove that other translations to HOL do not exist, but (as the discussion of $n!$ above suggests) we speculate that they would be significantly less natural.

3. HOL syntax and derivability

Higher-order logic (HOL) syntax and derivability should be familiar [46,17,2,6]. We give the basics.

3.1. Syntax

We present HOL as a derivation system over simply-typed λ-terms with constants and types for logical reasoning (like a type of truth-values and constant symbols like ⇒ and ∀). This is all standard.

Definition 3.1. A **HOL signature** is a set $\mathcal{D}$ of **base types**, which includes a distinguished base type of **truth-values** $o \in \mathcal{D}$. μ will range over base types. A **type-language** is defined by

$$\boxed{\beta ::= \mu \mid (\beta, \dots, \beta) \mid \beta \to \beta.}$$

It is not necessary to include products $(\beta_1, \dots, \beta_n)$, but for the purposes of translating PNL into HOL doing this is convenient.

Definition 3.2. A **term-signature** over a HOL signature $\mathcal{D}$ is a tuple $(\mathcal{G}, type)$ where:

- $\mathcal{G}$ is a set of **constants**, which must contain elements $\bot$, $\Rightarrow$, and $\forall_\beta$ for every type β.
- *type* assigns to each $\mathsf{g} \in \mathcal{G}$ a type β in the type-language determined by $\mathcal{D}$, such that $type(\bot) = o$, $type(\Rightarrow) = o \to o \to o$, and $type(\forall_\beta) = (\beta \to o) \to o$.

A **signature** $\mathcal{T}$ is then a tuple $(\mathcal{D}, \mathcal{G}, type)$.

Notation 3.3. We write $\mathsf{g} : \beta$ for $type(\mathsf{g}) = \beta$.

Remark 3.4. We strongly deprecate referring to the constant $\forall_\beta : (\beta \to o) \to o$ as ‘higher-order abstract syntax’ (HOAS), as sometimes happens. That term should refer to inductive types with binding constructed using constants of higher type like $(\Lambda \to \Lambda) \to \Lambda$ (strong HOAS) or $(\nu \to \Lambda) \to \Lambda$ (weak HOAS) [14,50] (the study of inductive syntax with binding is not the topic of this paper).

A term $\forall_\beta : (\beta \to o) \to o$ (plus axioms) expresses the *meaning* of ∀ [6, Section 2] and would still have meaning if our syntax was, e.g. combinators. In contrast, the *syntax* of combinators could be represented without any need for higher-order syntax, since it does not have binders [42, Section 2].

Definition 3.5. For each signature $\mathcal{T} = (\mathcal{D}, \mathcal{G}, type)$ and each type β over $\mathcal{D}$ fix a countably infinite set of **variables** of that type.

X, Y, Z will range over distinct HOL variables.[13] Write $type(X)$ for the type of X.

Definition 3.6. For each signature $\mathcal{T}$ define **HOL terms** over $\mathcal{T}$ by

$$t ::= X \mid \lambda X.t \mid tt \mid (t, \dots, t) \mid \mathsf{g}$$

and a **typing** relation by:

$$\boxed{\frac{t : \beta \ \ (type(X){=}\beta')}{\lambda X.t : \beta'{\to}\beta} \qquad \frac{t' : \beta'{\to}\beta \quad t : \beta'}{t't : \beta} \qquad \frac{t_1 : \beta_1 \ \dots \ t_n : \beta_n}{(t_1, \dots, t_n) : (\beta_1, \dots, \beta_n)} \qquad \frac{(type(\mathsf{g}){=}\mu)}{\mathsf{g} : \mu}}$$

We now define α-equivalence. We would not normally be so detailed about this, but when we map PNL terms and propositions to HOL later, it will be useful to have been precise here:

Definition 3.7. A **permutation** of HOL variables is a bijection ϖ such that $nontriv(\varpi) = \{X \mid \varpi(X) \neq X\}$ is finite. Give HOL terms a permutation action $\varpi \cdot t$ defined by:

[13] This means that if the reader sees ‘X’ this could refer either to a HOL variable or—recalling Definition 2.5—to a PNL unknown. We will make sure that it is always clear from context which is meant.

$$\frac{}{\Xi,\ \xi \vdash^{\lambda} \xi,\ \chi}\ (\mathbf{hAx}) \qquad \frac{}{\Xi,\ \bot \vdash^{\lambda} \chi}\ (\mathbf{h\bot L})$$

$$\frac{\Xi \vdash^{\lambda} \xi,\ \chi \quad \Xi,\ \chi \vdash^{\lambda} \chi}{\Xi,\ \xi \Rightarrow \chi \vdash^{\lambda} \chi}\ (\mathbf{h{\Rightarrow}L}) \qquad \frac{\Xi,\ \xi \vdash^{\lambda} \chi,\ \chi}{\Xi \vdash^{\lambda} \xi \Rightarrow \chi,\ \chi}\ (\mathbf{h{\Rightarrow}R})$$

$$\frac{\Xi,\ \xi[X{:=}t] \vdash^{\lambda} \chi \quad (t{:}type(X))}{\Xi,\ \forall X.\xi \vdash^{\lambda} \chi}\ (\mathbf{h\forall L}) \qquad \frac{\Xi \vdash^{\lambda} \xi,\ \chi \quad (X \notin fv(\Xi,\ \chi))}{\Xi \vdash^{\lambda} \forall X.\xi,\ \chi}\ (\mathbf{h\forall R})$$

Fig. 3. Sequent derivation rules of Higher-Order Logic.

$$\varpi{\cdot}X{=}\varpi(X) \qquad \varpi{\cdot}\lambda X.t{=}\lambda\varpi(X).\varpi{\cdot}t \qquad \varpi{\cdot}(t't){=}(\varpi{\cdot}t')(\varpi{\cdot}t)$$
$$\varpi{\cdot}(t_1,\ldots,t_n)={(\varpi{\cdot}t_1,\ldots,\varpi{\cdot}t_n)} \quad \varpi{\cdot}\mathsf{g}{=}\mathsf{g}$$

Free variables are defined by:

$$fv(X)=\{X\} \qquad fv(\lambda X.t){=}fv(t)\setminus\{X\} \qquad fv(t't){=}fv(t')\cup fv(t)$$
$$fv((t_1,\ldots,t_n)){=}\bigcup_i fv(t_i) \qquad fv(\mathsf{g}){=}\varnothing$$

Call a relation $\mathcal{R}$ on HOL terms a **congruence** when it is closed under the following rules:

$$\frac{t\ \mathcal{R}\ u}{\lambda X.t\ \mathcal{R}\ \lambda X.u} \qquad \frac{t'\ \mathcal{R}\ u' \quad t\ \mathcal{R}\ u}{t't\ \mathcal{R}\ u'u} \qquad \frac{t_i\ \mathcal{R}\ u_i \quad (1\leq i\leq n)}{(t_1,\ldots,t_n)\ \mathcal{R}\ (u_1,\ldots,u_n)}$$

Define α-equivalence to be the least congruence that is an equivalence relation and is such that:

$$\frac{(X,Y\notin fv(t))}{(Y\ X){\cdot}t =_{\alpha} t}$$

We quotient terms by α-equivalence and define **capture-avoiding substitution** $t[X{:=}u]$ as usual.

Definition 3.8. We write $t:\beta$ for *t is a term and has type β*. We call t **typable** when $t\,:\,\beta$ for some type β.

We call a term a **HOL proposition** when it has type o. ξ and χ will range over HOL propositions. We may write $\forall_\beta\, \lambda X.\xi$ as $\forall X.\xi$.

Definition 3.9. Ξ and $\mathcal{X}$ will range over sets of HOL propositions. We may write $\xi,\ \Xi$ and $\Xi,\ \xi$ as shorthand for $\{\xi\}\cup\Xi$.

Write $fv(\Xi,\ \mathcal{X}) = \bigcup\{fv(\xi)\mid \xi\in\Xi\}\cup\bigcup\{fv(\chi)\mid \chi\in\mathcal{X}\}$.

A **sequent** is a pair $\Xi \vdash^{\lambda} \mathcal{X}$.

Definition 3.10 (*Derivable Sequents*)**.** The **derivable sequents** are defined in Fig. 3.

4. The translation from nominal to functional syntax, and its soundness

4.1. Translation from PNL to higher-order logic

In this subsection we show how to translate a PNL signature $\mathcal{S}$ and propositions and terms in that signature, to a higher-order logic (HOL) signature and propositions and terms in that signature. We start by translating a PNL signature $\mathcal{S}$ to a HOL signature $\mathcal{T}_{\mathcal{S}}$. First, we set up some notation:

Notation 4.1. Let D range over finite lists of distinct atoms.

- Write $a\in D$ when a occurs in D.
- Write $D'\subseteq D$ when every element in D' occurs in D (disregarding order). Similarly if S is a set of atoms write $D\subseteq S$ when every element in D occurs in S.
- If S is a set of atoms write $D\cap S$ for the list obtained by removing from D just those atoms not in S.
- Write $\pi{\cdot}D$ for the list obtained by applying π pointwise to the elements of D in order.
- Write D, a for the list obtained by appending a; when we write this we include an assumption that $a\notin D$.
- Write $\lambda D.t$ for $\lambda d_1.\ldots.\lambda d_n.t$ where $D=[d_1,\ldots,d_n]$.

Definition 4.2. From a PNL signature $\mathcal{S}$ determine a HOL signature $\mathcal{T}_{\mathcal{S}}$ by the following specification:

1. For every atoms-sort ν in $\mathcal{S}$ assume a HOL base type μ_ν.
2. For every base sort τ assume a HOL type μ_τ.

$$\begin{array}{ccc}
\lfloor a\rfloor^D = a & \lfloor(r_1,\dots,r_n)\rfloor^D = (\lfloor r_1\rfloor^D,\dots,\lfloor r_n\rfloor^D) & \lfloor \mathsf{f}(r)\rfloor^D = \mathsf{g}_{\mathsf{f}}\,\lfloor r\rfloor^D \\
\lfloor [a]r\rfloor^D = \lambda a.\lfloor r\rfloor^D & \lfloor \pi\cdot X\rfloor^D = X_D\pi\cdot(D\cap pms(X)) & \\
\lfloor\bot\rfloor^D = \bot & \lfloor \phi\Rightarrow\psi\rfloor^D = \Rightarrow\lfloor\phi\rfloor^D\lfloor\psi\rfloor^D & \lfloor \mathsf{P}(r)\rfloor^D = \mathsf{g}_{\mathsf{P}}\,\lfloor r\rfloor^D \\
\lfloor \forall X.\phi\rfloor^D = \forall\,\lambda X_D.\lfloor\phi\rfloor^D & &
\end{array}$$

Fig. 4. Translation from PNL to HOL.

Translate sorts in $\mathcal{S}$ to types in $\mathcal{T}_{\mathcal{S}}$ as follows:

$$\begin{array}{ccc}
\lfloor\nu\rfloor = \mu_\nu & \lfloor\tau\rfloor = \mu_\tau & \lfloor(\alpha_1,\dots,\alpha_n)\rfloor = (\lfloor\alpha_1\rfloor,\dots,\lfloor\alpha_n\rfloor) \\
\lfloor[\nu]\alpha\rfloor = \mu_\nu\to\lfloor\alpha\rfloor & &
\end{array}$$

3. For every term-former $\mathsf{f} : (\alpha)\tau$ assume a HOL constant $\mathsf{g}_{\mathsf{f}} : \lfloor\alpha\rfloor\to\tau$.
4. For every proposition-former $\mathsf{P} : \alpha$ assume a HOL constant $\mathsf{g}_{\mathsf{P}} : \lfloor\alpha\rfloor\to o$.
5. For every atom $a : \nu$ assume a HOL variable $a : \mu_\nu$.

   It is convenient to assume this correspondence is a literal identity; i.e. that $\mathbb{A}_\nu$ is actually a subset of the set of HOL variables of type μ_ν, and that there are countably infinitely many HOL variables of type μ_ν that are not atoms.

   In particular, this means that every permutation π in the sense of Definition 2.7 is also a permutation ϖ in the sense of Definition 3.7.
6. For every unknown $X : \alpha$ and list D assume a distinct HOL variable X_D that is not an atom[14] of type $\mu_{\nu_1}\to\dots\to\mu_{\nu_n}\to\lfloor\alpha\rfloor$ where ν_i is the sort of the ith atom in $D\cap pms(X)$ (by convention and as standard, if $D\cap pms(X)$ is empty we take this to be $\lfloor\alpha\rfloor$).

Definition 4.3. Given a list D translate PNL terms and propositions in $\mathcal{S}$ to HOL terms and propositions in $\mathcal{T}_{\mathcal{S}}$ (Definition 4.2) by the rules in Fig. 4.

(The notation $\pi\cdot(D\cap pms(X))$ is defined in Notation 4.1.)

Example 4.4. Suppose $D\cap pms(X)$ (Notation 4.1) is the list $[a]$. Assume a proposition-former equal of appropriate arity. Then:

$$\lfloor id{\cdot}X\rfloor^D = X_D\,a \quad \lfloor(b\ a){\cdot}X\rfloor^D = X_D\,b \quad \lfloor[a]id{\cdot}X\rfloor^D = \lambda a.(X_D\,a) \quad \lfloor[b](b\ a){\cdot}X\rfloor^D = \lambda b.(X_D\,b)$$
$$\lfloor\forall X.\mathsf{equal}([a]X,[b](b\ a){\cdot}X)\rfloor^D = \forall_{\lfloor type(X_D)\rfloor}\,\lambda X.(\mathsf{equal}(\lambda a.(X_D\,a))(\lambda b.(X_D\,b)))$$

Assuming appropriate axioms for equal, we would expect this to be true. Now assume $D\cap pms(Y)$ is the list $[a,b]$. Then:

$$\lfloor id{\cdot}Y\rfloor^D = Y_Dab \quad \lfloor(b\ a){\cdot}Y\rfloor^D = Y_Dba \quad \lfloor[a]id{\cdot}Y\rfloor^D = \lambda a.(Y_Dab) \quad \lfloor[b](b\ a){\cdot}Y\rfloor^D = \lambda b.(Y_Dba)$$
$$\lfloor\forall Y.\mathsf{equal}([a]Y,[b](b\ a){\cdot}Y)\rfloor^D = \forall_{\lfloor type(Y_D)\rfloor}\,\lambda Y_D.(\mathsf{equal}(\lambda a.(Y_Dab))(\lambda b.(Y_Dba)))$$

We would expect this to be false. What has changed with respect to the previous case, is that b is fresh for X but not for Y.

Remark 4.5. The translation of $\pi\cdot X$ to $X_D\pi\cdot(D\cap pms(X))$ can be seen as a *Skolemisation* or *raising*, where D is a finite collection of atoms/variables that are considered 'relevant'.

See for instance Sections 4.2 and 4.3 of Chapter 1 of [11] or Definition 53 of Chapter 3 of [11], where Skolemisation is discussed in the context of tableau methods. There, just as here, there is a notion of the 'relevant' variables. See also Section 5 of [47], where *raising* is discussed in the context of unification in the presence of mixed quantifiers.

In fact, Definition 4.3 is a modified version of [13, Definition 8.3] which translated between nominal unification and unification over a generalisation of the same notion of *pattern* used by Miller in [45,47]. See also [38], where similar ideas are applied to algebraic reasoning. The idea of these 'nominal' translations is that X translates to a variable X_D of higher order. In Proposition 4.10 we state a formal sense in which, provided D contains all 'relevant' atoms, this translation loses no information (the result itself is from [13]). What does 'relevant' mean? We examine Fig. 5 and see that it means intuitively 'is permuted by some π acting on $pms(X)$'. In Section 4.3 we apply this to PNL.

Lemma 4.6. • *Suppose a is an atom. Then if $a\in fv(\lfloor r\rfloor^D)$ then $a\in fa(r)$.*
- *$\lfloor\pi\cdot r\rfloor^D = \pi\cdot\lfloor r\rfloor^D$ (for π on the right-hand side considered as a permutation of HOL variables).*

As a corollary, the translation $\lfloor r\rfloor^D$ is well-defined. That is, if r and s are α-equivalent then $\lfloor r\rfloor^D = \lfloor s\rfloor^D$.

Proof. By routine inductions on r. The proof that $fv(\lfloor\pi\cdot X\rfloor^D)\subseteq fa(\pi\cdot X)$ uses the fact that $D\cap pms(X)\subseteq pms(X)$. The corollary follows; for more details see [13, Section 8]. $\square$

[14] So X is one of the countably infinitely many HOL variables that are not atoms.

$$\frac{}{D \vdash a : A} \qquad \frac{D \vdash r : A}{D \vdash \mathsf{f}(r) : A} \qquad \frac{D \vdash r : A, a}{D \vdash [a]r : A}$$

$$\frac{D \vdash r_i : A \quad (1{\le}i{\le}n)}{D \vdash (r_1, \dots, r_n) : A} \qquad \frac{((nontriv(\pi) \cup A) \cap pms(X) \subseteq D)}{D \vdash \pi{\cdot}X : A}$$

$$\frac{D \vdash r : A}{D \vdash \mathsf{P}(r) : A} \qquad \frac{D \vdash \phi : A \quad D \vdash \psi : A}{D \vdash \phi \Rightarrow \psi : A} \qquad \frac{}{D \vdash \bot : A} \qquad \frac{D \vdash \phi : A}{D \vdash \forall X.\phi : A}$$

Fig. 5. Capture typing.

4.2. Capture typing

We mentioned in Remark 4.5 that the translation of Definition 4.3 requires us to declare a finite list D of 'relevant' atoms. How large must D be in order to capture all the important information in some r or ϕ? This is calculated by a *capture typing*, an idea going back to [12,13].

Definition 4.7. Define **capture typings** $D \vdash r : A$ and $D \vdash \phi : A$ inductively by the rules in Fig. 5. Here D ranges over finite lists of distinct atoms as described in Notation 4.1, and A ranges over finite sets of atoms.

If $A = \varnothing$ then we may omit the '$:A$' and write just $D \vdash r$ and $D \vdash \phi$. Write $D \vdash \Psi$ when $D \vdash \psi$ for every $\psi \in \Psi$.

Remark 4.8. We are only interested in the case $A = \varnothing$, but we need to consider nonempty A just as part of the inductive definition. Intuitively, $D \vdash r : A$ can be read as 'D is relevant to r in the context of abstractions over the atoms in A'.

Remark 4.9. As we mentioned in Remark 4.5, the intuitive reading of $D \vdash r$ is 'the atoms in D get permuted in some X occurring in r'.

Thus, the interesting case in Fig. 5 is the rule for $\pi{\cdot}X$. This ensures that D is large enough to record all the important atoms in π or abstracted further up in the term – that is, those permitted in X – so that we do not lose information when we form $\lfloor \pi{\cdot}X \rfloor^D = X_D\pi{\cdot}(D \cap pms(X))$. This is made formal in Proposition 4.10, which is Theorems 8.12 and 8.14 of [13]:

Proposition 4.10. • *If $D \vdash r$ and $D \vdash s$ then $\lfloor r \rfloor^D = \lfloor s \rfloor^D$ implies $r = s$ (note that $=$ denotes α-equality, because we quotiented terms by this relation), and similarly for ϕ and ψ.*

- *If $D \nvdash r$ then there exists s such that $\lfloor r \rfloor^D = \lfloor s \rfloor^D$ yet $r \neq s$, and similarly for ϕ.*

Definition 4.3 maps PNL terms and predicates to typable HOL terms:

Proposition 4.11. *If $r : \alpha$ then for any D, $\lfloor r \rfloor^D : \lfloor \alpha \rfloor$, and $\lfloor \phi \rfloor^D : o$.*

Proof. By inductions on r and ϕ.

- *The case $a \in \mathbb{A}_\nu$.* By Definition 4.2 $a : \mu_\nu$.
- *The case $[a]r$ where $a \in \mathbb{A}_\nu$.* By inductive hypothesis $\lfloor r \rfloor^D : \beta$ for some type β. It follows that $\lfloor [a]r \rfloor^D = \lambda a.\lfloor r \rfloor^D : \mu_\nu \to \beta$.
- *The case $\pi{\cdot}X$.* Suppose $D \vdash \pi{\cdot}X$. It is routine to check that $X_D\pi{\cdot}(D \cap pms(X)) : \lfloor sort(X) \rfloor$. □

4.3. Soundness of the translation

Recall that HOL terms have a permutation action $\pi{\cdot}t$ given by considering π as a permutation on HOL variables and using Definition 3.7. Then:

Lemma 4.12. *If $nontriv(\pi) \cap fv(t) \subseteq D$ then $(\lambda D.t)\pi{\cdot}D =_{\alpha\beta} \pi{\cdot}t$ (see Notation 4.1).*

Proof. A fact of $\alpha\beta$-conversion [13, Lemma 9.2]. □

Lemma 4.13. *Suppose $D \vdash r$ and $D \vdash \phi$. Suppose $r' : sort(X)$ and $fa(r') \subseteq pms(X)$. Then:*

- $\lfloor r[X{:=}r'] \rfloor^D =_{\alpha\beta} \lfloor r \rfloor^D[X_D{:=}\lambda(D \cap pms(X)).\lfloor r' \rfloor^D]$.
- $\lfloor \phi[X{:=}r'] \rfloor^D =_{\alpha\beta} \lfloor \phi \rfloor^D[X_D{:=}\lambda(D \cap pms(X)).\lfloor r' \rfloor^D]$.

Proof. By routine inductions on r and ϕ. We sketch two cases:

- *The case $(\pi{\cdot}X)[X{:=}r']$.* We must prove that

$$\lfloor \pi{\cdot}r' \rfloor^D =_{\alpha\beta} \big(\lambda(D \cap pms(X)).\lfloor r' \rfloor^D\big)\pi{\cdot}(D \cap pms(X))$$

This follows by Lemmas 4.6 and 4.12.

- *The case* $\mathsf{P}(r)[X{:=}r']$. We must prove that

$$\lfloor \mathsf{P}(r[X{:=}r'])\rfloor^{\scriptscriptstyle D} =_{\alpha\beta} \mathsf{g}_{\mathsf{P}}(\lfloor r\rfloor^{\scriptscriptstyle D})[X{:=}\lambda(D\cap pms(X)).\lfloor r'\rfloor^{\scriptscriptstyle D}]$$

This follows directly from the first part. □

Proposition 4.14. *Suppose $D \vdash \phi$ and $D \vdash r'$ and $r' : sort(X)$ and $fa(r') \subseteq pms(X)$. Then $\lfloor \forall X.\phi\rfloor^{\scriptscriptstyle D} \stackrel{\lambda}{\vDash} \lfloor \phi[X{:=}r']\rfloor^{\scriptscriptstyle D}$.*

Proof. Using Lemma 4.13 and (**h∀L**) from Fig. 3. □

Lemma 4.15. *For any finite set of predicates $\{\phi_i \mid i \in I\}$ there exists some D such that $D \vdash \phi_i$ for every $i \in I$.*

Proof. We calculate $\bigcup nontriv(\pi)$ for every π occurring in every ϕ_i, in some order, and take this to be D. It is not hard to verify that this sufficies. □

Theorem 4.16. *The interpretation is sound: if $\Phi \stackrel{\pi}{\vDash} \Psi$ with a derivation $\mathfrak{B}$ and $D \vdash \Phi'$ and $D \vdash \Psi'$ for every sequent $\Phi' \stackrel{\pi}{\vDash} \Psi'$ appearing in $\mathfrak{B}$, then $\lfloor\Phi\rfloor^{\scriptscriptstyle D} \stackrel{\lambda}{\vDash} \lfloor\Psi\rfloor^{\scriptscriptstyle D}$ (by Lemma 4.15, some such D always exists).*

Proof. By induction on the derivation $\mathfrak{B}$ of $\Phi \stackrel{\pi}{\vDash} \Psi$. It is routine to verify by induction on $\mathfrak{B}$ that $\lfloor\Phi'\rfloor^{\scriptscriptstyle D} \stackrel{\lambda}{\vDash} \lfloor\Psi'\rfloor^{\scriptscriptstyle D}$ is derivable for each $\Phi' \stackrel{\pi}{\vDash} \Psi'$ appearing in $\mathfrak{B}$; the case of (∀**R**) uses Proposition 4.14. So in particular $\lfloor\Phi\rfloor^{\scriptscriptstyle D} \stackrel{\lambda}{\vDash} \lfloor\Psi\rfloor^{\scriptscriptstyle D}$. □

Lemma 4.17. *The interpretation for full PNL (Fig. 1, with the stronger axiom rule) would not be sound. That is, there exist Φ and Ψ and D such that $D \vdash \Phi$, $D \vdash \Psi$, and $\Phi \vdash \Psi$, but $\lfloor\Phi\rfloor^{\scriptscriptstyle D} \not\stackrel{\lambda}{\vDash} \lfloor\Psi\rfloor^{\scriptscriptstyle D}$.*

Proof. Consider a name sort ν and a unary predicate $\mathsf{P} : \nu$. Then $\mathsf{P}(a) \vdash \mathsf{P}(b)$ in full PNL, but it is not the case that $\mathsf{g}_{\mathsf{P}}\, a \vdash \mathsf{g}_{\mathsf{P}}\, b$ in HOL. □

5. Semantics

For the reader's convenience we will clarify one aspect of the coming notation now: if the reader sees X° this is a set with a permutation action; if the reader sees $\mathsf{X}^{\rightrightarrows}$ this is a set with a renaming action. There is no particular connection between X° and $\mathsf{X}^{\rightrightarrows}$.

A typical renaming is $[a{:=}b]$ (instead of a typical permutation $(a\ b)$). Formal definitions are in Definitions 2.7 and 5.1.

The reader may not be surprised by the use of sets with a permutation action—nominal techniques are based on these [41]. But why the renaming action? We need renamings to make a function out of an atoms-abstraction, mirroring the clause $\lfloor [a]r\rfloor^{\scriptscriptstyle D} = \lambda a.\lfloor r\rfloor^{\scriptscriptstyle D}$ in Definition 4.3.

In PNL models, an abstraction $[a]r$ is modelled as Gabbay–Pitts atoms-abstraction $[a]x$, a sets-based construction from [41] (Definition 5.29, in this paper). This is constructed like a pair, forming $[a]x$ from a and x, but destructed like a *partial function* the graph of which is evident in Definition 5.29. It is defined on $[a]x$ for fresh b but not for $b \in supp(x) \setminus \{a\}$.

When we translate $[a]r$ to HOL we interpret $[a]r$ as a function using λ-abstraction. This suggests of our models that we translate a *partial* function $[a]x$ to a total function. But then we have to give meaning to $[a]x$ applied to b where b is not fresh. This is where renaming sets are used.

We can then conclude by noting that every model of PNL can be transformed into a model of HOL, and in a compositional manner (Lemma 8.9). Completeness quickly follows.

5.1. Categories of supported permutation and renaming sets

5.1.1. Permutation and renaming sets

Definition 5.1. Suppose ρ is a map from $\mathbb{A}$ to $\mathbb{A}$. Define $dom(\rho)$ and $img(\rho)$ by

$$dom(\rho) = \{a \mid \rho(a) \neq a\} \quad \text{and} \quad img(\rho) = \{\rho(a) \mid a \in dom(\rho)\}$$

Echoing Definition 2.7, a **renaming** is a map ρ from $\mathbb{A}$ to $\mathbb{A}$ such that $a \in \mathbb{A}_\nu \Leftrightarrow \rho(a) \in \mathbb{A}_\nu$ and $nontriv(\rho) = dom(\rho)\cup img(\rho)$ is finite. Write $\mathbb{R}$ for the set of renamings.

For $a, b \in \mathbb{A}_\nu$ let an **atomic renaming** $[a{:=}b]$ map a to b, b to b, and other c to themselves.

ρ will range over renamings.

Notation 5.2. We use the following notation (analogously to Notation 2.8):

- Write $\rho \circ \rho'$ for **functional composition**, so $(\rho \circ \rho')(a) = \rho(\rho'(a))$.
- Write id for the **identity renaming**, so $id(a) = a$ always.

Unlike in Notation 2.8, we have no notation for inverse, because for renamings inverses need not exist.

Definition 5.3.
- A **permutation set** is a pair $\mathsf{X}^{\circ} = (|\mathsf{X}^{\circ}|, \cdot)$ of an **underlying set** $|\mathsf{X}^{\circ}|$ and a **permutation action** $(\mathbb{P} \times |\mathsf{X}^{\circ}|) \to |\mathsf{X}^{\circ}|$ which is a group action; write it infix.
  (So $id{\cdot}x = x$ and $\pi\cdot(\pi'\cdot x) = (\pi \circ \pi')\cdot x$.)
- A **renaming set** is a pair $\mathsf{X}^{\rightrightarrows} = (|\mathsf{X}^{\rightrightarrows}|, \cdot)$ of an **underlying set** $|\mathsf{X}^{\rightrightarrows}|$ and a **renaming action** $(\mathbb{R} \times |\mathsf{X}^{\rightrightarrows}|) \to |\mathsf{X}^{\rightrightarrows}|$ which is a monoid action; write it infix.
  (So $id{\cdot}x = x$ and $\rho{\bullet}(\rho'{\bullet}x) = (\rho \circ \rho'){\bullet}x$.)

Definition 5.4.
- Suppose X° is a permutation set. Say that $A \subseteq \mathbb{A}$ **supports** $x \in |\mathsf{X}^{\circ}|$ when for all $\pi, \pi' \in \mathbb{P}$, if $\forall a \in A.\pi(a) = \pi'(a)$ then $\pi\cdot x = \pi'\cdot x$.
- Suppose $\mathsf{X}^{\rightrightarrows}$ is a renaming set. Say that $A \subseteq \mathbb{A}$ **supports** $x \in |\mathsf{X}^{\rightrightarrows}|$ when for all $\rho, \rho' \in \mathbb{R}$, if $\forall a \in A.\rho(a) = \rho'(a)$ then $\rho{\bullet}x = \rho'{\bullet}x$.

Lemma 5.5. *If $x \in |\mathsf{X}^{\circ}|/|\mathsf{X}^{\rightrightarrows}|$ has a supporting permission set (Definition 2.3) then it has a unique least supporting set which is equal to the intersection of all permission sets supporting x. We call this the* **support** *of x when it exists, and write it $supp(x)$.*

Definition 5.6.
- Call $x \in |\mathsf{X}^{\circ}|/|\mathsf{X}^{\rightrightarrows}|$ **supported** when $supp(x)$ exists.
- Call $\mathsf{X}^{\circ}/\mathsf{X}^{\rightrightarrows}$ **supported** when every element $x \in |\mathsf{X}^{\circ}|/|\mathsf{X}^{\rightrightarrows}|$ is supported.

Recall from Definition 2.14 the *pointwise* actions of π and ρ on sets of atoms.

Lemma 5.7. *1. If $x \in |\mathsf{X}^{\circ}|$ then $supp(\pi\cdot x) = \pi\cdot supp(x)$.*
2. If $x \in |\mathsf{X}^{\rightrightarrows}|$ then $supp(\rho{\bullet}x) \subseteq \rho{\cdot}supp(x)$.
As a corollary, if ρ is injective on $supp(x)$ then $supp(\rho{\bullet}x) = \rho{\cdot}supp(x)$.

Proof. By routine calculations using the group/monoid action. □

The reverse subset inclusion in part 2 of Lemma 5.7 is not true in general:

Lemma 5.8. *There exists a renaming set $\mathsf{X}^{\rightrightarrows}$, an element $x \in |\mathsf{X}^{\rightrightarrows}|$, and a renaming ρ such that $\rho{\cdot}supp(x) \not\subseteq supp(\rho{\bullet}x)$.*

Proof. Consider $(\mathbb{A} \times \mathbb{A}) \cup \{*\}$ with the non-standard **'exploding'** renaming action such that:

$$\begin{array}{lll} \rho(*) = * & \rho{\bullet}(a, b) = (\rho(a), \rho(b)) \text{ if } \rho(a) \neq \rho(b) & \\ \rho{\bullet}(a, a) = (\rho(a), \rho(a)) & \qquad\rho{\bullet}(a, b) = * & \text{if } \rho(a) = \rho(b) \end{array}$$

(Recall from Definition 2.3 that by our permutative convention, a and b are distinct.) Then $supp([a{:=}b]{\bullet}(a, b)) = \varnothing \subsetneq \{b\} = [a{:=}b]{\bullet}supp((a, b))$. □

5.1.2. Equivariant elements and maps

Definition 5.9. Call an element x in $|\mathsf{X}^{\circ}|/|\mathsf{X}^{\rightrightarrows}|$ **equivariant** when $supp(x) = \varnothing$.

x is equivariant when $\pi\cdot x = x$ for all π, or $\rho{\bullet}x = x$ for all ρ, respectively.

Definition 5.10.
- Call a function $F \in |\mathsf{X}^{\circ}| \to |\mathsf{Y}^{\circ}|$ **equivariant** when

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

- Call a function $G \in |\mathsf{X}^{\rightrightarrows}| \to |\mathsf{Y}^{\rightrightarrows}|$ **equivariant** when

$$\forall\rho{\in}\mathbb{R}.\forall x{\in}|\mathsf{X}^{\rightrightarrows}|.G(\rho{\bullet}x) = \rho{\bullet}G(x)$$

F and G will range over equivariant functions between pairs of permutation and renaming sets respectively.

Remark 5.11. Equivariance is a characteristic feature of nominal techniques. Equivariance means in words 'symmetric under permuting atoms'; in this paper we are also interested in 'symmetric under renaming atoms'. Whichever meaning is appropriate, equivariance is a symmetry property.

For an element $x \in |\mathsf{X}^{\circ}|$ or $x \in |\mathsf{X}^{\rightrightarrows}|$ equivariance means that x has empty support (in fact, $supp(x)$ is a measure of asymmetry in x). For a function $F \in |\mathsf{X}^{\circ}| \to |\mathsf{Y}^{\circ}|$ equivariance means that F commutes with π. For a function $F \in |\mathsf{X}^{\rightrightarrows}| \to |\mathsf{Y}^{\rightrightarrows}|$ equivariance means that F commutes with ρ.

Equivariance and support are abstract mathematical concepts, but they were originally derived from the study of syntax in [41]. For syntax, equivariance corresponds to 'closed'; in usual informal usage the syntax $\lambda x.x$ is closed and is symmetric under changing x to y.[15] The reader might therefore find it useful to read 'equivariant' as 'closed'.

[15] For the rest of this remark λ is a term-former. So $\lambda x.x$ refers to the term, not the function.

However, this is just an analogy and it can be quite treacherous. For instance, $\lambda x.\lambda x.x$ is closed as a term but it is not *permutatively* symmetric with $\lambda x.\lambda y.y$. (If we build our syntax using nominal abstract syntax then $\lambda[a]\lambda[a]a$ is equal to $\lambda[a]\lambda[b]b$, but the reason for this is that $[a]a$ is equal, via symmetry up to permutation, with $[b]b$.)

So, while an analogy of 'equivariance' with 'closure' is historically reasonable and may be helpful, bear in mind that what it really means is 'symmetric'.

Lemma 5.12. *1. Suppose $F \in |\mathsf{X}^{\circ}| \to |\mathsf{Y}^{\circ}|$ is equivariant. Then $supp(F(x)) \subseteq supp(x)$ for every $x \in |\mathsf{X}^{\circ}|$.*
2. Suppose $G \in |\mathsf{X}^{\rightrightarrows}| \to |\mathsf{Y}^{\rightrightarrows}|$ is equivariant. Then $supp(G(x)) \subseteq supp(x)$ for every $x \in |\mathsf{X}^{\rightrightarrows}|$.

Proof. We consider only the second part. Suppose S supports x so that for all ρ and ρ', if $\forall a \in S.\rho(a) = \rho'(a)$ then $\rho{\bullet}x = \rho'{\bullet}x$. The result follows if we note that $\rho{\bullet}G(x) = G(\rho{\bullet}x)$ and $\rho'{\bullet}G(x) = G(\rho'{\bullet}x)$. □

Definition 5.13. • Write PmsPrm for the category with objects supported permutation sets and arrows equivariant functions between them.
Henceforth, X° and Y° will range over objects in PmsPrm.
• Write PmsRen for the category with objects supported renaming sets and arrows equivariant functions between them.
Henceforth, $\mathsf{X}^{\rightrightarrows}$ and $\mathsf{Y}^{\rightrightarrows}$ will range over objects in PmsPrm.

Remark 5.14. Both PmsPrm and PmsRen are categories of sets with a monoid action (in the case of PmsPrm that monoid happens to be a group).

PmsPrm can be thought of as the category of pullback-preserving presheaves on the category $\mathbb{I}'$ of permission sets and finite injections between them (so an object $S \in \mathbb{I}'$ is a permission set, and an arrow from S to T is a permutation π such that $\pi\cdot S \subseteq T$). PmsRen can be thought of as the category of those presheaves on the category $\mathbb{F}'$ of permission sets and finite renamings between them (so an object $S \in \mathbb{F}'$ is a permission set, and an arrow from S to T is a renaming ρ such that $\rho{\bullet}S \subseteq T$) that preserve pullbacks of monos.

For details on this see [29], and for a more wide-ranging survey of the applications of sets with an action and presheaves see [40]. See also the discussion of presheaves in Section 9.2.

5.2. *The exponential in* PmsRen

PmsPrm and PmsRen are both cartesian closed, but we only discuss exponentials for PmsRen in this paper. The reader can find the constructions for PmsPrm e.g. in [23, Section 9].

PmsPrm is used to give denotation to PNL only, while PrmRen is used to give a denotation to PNL and also to HOL. For this reason, the exponentials of PmsRen are of specific and immediate importance to us, but not those of PmsPrm.

5.2.1. *Functions*

Recall the definitions of *dom* and *img* from Definition 5.1.

Definition 5.15. • Suppose $\mathsf{X}^{\circ}, \mathsf{Y}^{\circ} \in$ PmsPrm. Suppose $f \in |\mathsf{X}^{\circ}| \to |\mathsf{Y}^{\circ}|$ (f is not necessarily equivariant).
Call f **supported** when there exists a permission set $S_f \subseteq \mathbb{A}$ such that for every $x \in |\mathsf{X}^{\circ}|$ and permutation $\pi \in \mathbb{P}$, if $nontriv(\pi) \cap S_f = \varnothing$ then

$$\pi{\bullet}(f(x)) = f(\pi{\bullet}x).$$

• Suppose $\mathsf{X}^{\rightrightarrows}, \mathsf{Y}^{\rightrightarrows} \in$ PmsRen. Suppose $f \in |\mathsf{X}^{\rightrightarrows}| \to |\mathsf{Y}^{\rightrightarrows}|$ (f is not necessarily equivariant).
Call f **supported** when there exists a permission set $S_f \subseteq \mathbb{A}$ such that for every $x \in |\mathsf{X}^{\rightrightarrows}|$ and renaming $\rho \in \mathbb{R}$, if $dom(\rho) \cap S_f = \varnothing$ then

$$\rho{\bullet}(f(x)) = f(\rho{\bullet}x).$$

Remark 5.16. Definition 5.15 uses a word 'supported' for f, suggestive of Definition 5.4, even though f has no permutation/renaming action. It *will* have a permutation/renaming action (Remark 5.17 and Definition 5.20), and then the terminologies will coincide (see Lemma 5.24).

Remark 5.17. It is a fact that PmsPrm is cartesian closed and functions have the *conjugation action*

$$(\pi\cdot f)(x) = \pi\cdot(f(\pi^{-1}\cdot x))$$

and f is supported in the sense of Definition 5.15 if and only if it is supported as an element of $|\mathsf{X}^{\circ}| \to |\mathsf{Y}^{\circ}|$ with the conjugation action. For more on this see [23,41].

Renamings ρ are not invertible, so we must work a little harder to define a renaming action. This is Definition 5.20. However, the end result is similar to the conjugation action, in a sense made formal in Lemma 5.22 which is similar to an immediate corollary of the conjugation action that $\pi\cdot(f(x)) = (\pi\cdot f)(\pi\cdot x)$.

Lemma 5.18. *If f is supported then $supp(f(x)) \subseteq S_f \cup supp(x)$ for every $x \in |\mathsf{X}^{\rightrightarrows}|$.*

Proof. By contradiction. Suppose there exists $a \in supp(f(x)) \setminus (S_f \cup supp(x))$. Choose b fresh (so $b \notin supp(f(x)) \cup S_f \cup supp(x)$). Then $(b\ a)\cdot(f(x)) = f((b\ a)\cdot x)$ since $a, b \notin S_f$ and $f((b\ a)\cdot x) = f(x)$ since $b, a \notin supp(x)$. It follows by Lemma 5.7 that $(b\ a)\cdot supp(f(x)) = supp(f(x))$, which is impossible. □

Definition 5.19. Suppose $S \subseteq \mathbb{A}$ is a permission set and $A \subseteq \mathbb{A}$ is finite. Call ρ_1 and ρ_2 a **freshening pair** of renamings for A with respect to S when:

- $dom(\rho_1) = A$ and $dom(\rho_2) = img(\rho_1)$.
- $(\rho_2 \circ \rho_1)(a) = a$ for all $a \in A$.
- $dom(\rho_2) \cap (S \cup A) = \varnothing$.

In words, ρ_1 maps the atoms in A to be outside S (and A), and ρ_2 is an 'inverse' to ρ_1 that puts them back.

5.2.2. *Renaming action*

Definition 5.20. (We continue the notation of Definition 5.15.) If f is supported then define $\rho\cdot f$ by

$$\boxed{(\rho\cdot f)(x) = (\rho_2 \circ \rho)\cdot f(\rho_1\cdot x)}$$

for some/any freshening pair of renamings ρ_1 and ρ_2 for $nontriv(\rho)$ (which is finite), with respect to $supp(x) \cup S_f$.

Lemma 5.21. *Definition 5.20 is well-defined. That is, it does not matter which freshening pair of renamings we choose.*

Proof. Consider two freshening pairs of renamings ρ_1, ρ_2 and ρ_1', ρ_2'. These exist because $nontriv(\rho)$ is finite and $\mathbb{A} \setminus (supp(x) \cup S_f)$ is infinite.

Let ρ_1'' map $img(\rho_1)$ to $img(\rho_1')$ and ρ_2'' map $dom(\rho_2') = img(\rho_1')$ to $dom(\rho_2) = img(\rho_1)$ in such a way that

- $\rho_1'(a) = (\rho_1'' \circ \rho_1)(a)$ for all $a \in dom(\rho_1')$,
- $\rho_2'(a) = (\rho_2 \circ \rho_2'')(a)$ for all $a \in dom(\rho_2')$, and
- $nontriv(\rho_1'') = img(\rho_1) \cup img(\rho_1')$ and $nontriv(\rho_2'') = dom(\rho_2') \cup dom(\rho_2)$.

We reason as follows:

$$\begin{aligned}
(\rho_2' \circ \rho)\cdot f((\rho_1' \circ \rho)\cdot x) &= (\rho_2 \circ \rho_2'' \circ \rho)\cdot f((\rho_1'' \circ \rho_1 \circ \rho)\cdot x) && \text{Lemmas 5.7 and 5.18, Definition 5.4}\\
&= (\rho_2 \circ \rho_2'' \circ \rho \circ \rho_1'')\cdot f((\rho_1 \circ \rho)\cdot x) && dom(\rho_1'') \cap S_f = \varnothing\\
&= (\rho_2 \circ \rho_2'' \circ \rho_1'' \circ \rho)\cdot f((\rho_1 \circ \rho)\cdot x) && nontriv(\rho_1'') \cap nontriv(\rho) = \varnothing\\
&= (\rho_2 \circ \rho)\cdot f((\rho_1 \circ \rho)\cdot x) && \text{Lemmas 5.18 and 5.7, Definition 5.4} \quad \square
\end{aligned}$$

Lemma 5.22. *Suppose $x \in |\mathsf{X}^{\rightrightarrows}|$ and ρ is a renaming. Suppose $f \in |\mathsf{X}^{\rightrightarrows}| \to |\mathsf{Y}^{\rightrightarrows}|$ is supported. Then $\rho\cdot(f(x)) = (\rho\cdot f)(\rho\cdot x)$.*

Proof. Let ρ_1 and ρ_2 be a freshening pair of renamings of $nontriv(\rho)$ with respect to $S_f \cup supp(x)$.

Let ρ' be a renaming with $nontriv(\rho') = img(\rho_1)$ such that $\rho_1 \circ \rho = \rho' \circ \rho_1$; this exists since ρ_1 is injective on $nontriv(\rho)$ and 'freshens' this set to some fresh set of atoms.

We reason as follows:

$$\begin{aligned}
(\rho\cdot f)(\rho\cdot x) &= (\rho_2 \circ \rho)\cdot f((\rho_1 \circ \rho)\cdot x) && \text{Definition 5.20}\\
&= (\rho_2 \circ \rho)\cdot f((\rho' \circ \rho_1)\cdot x) && \text{Definition 5.4}\\
&= (\rho_2 \circ \rho \circ \rho')\cdot f(\rho_1\cdot x) && nontriv(\rho') \cap S_f = \varnothing\\
&= (\rho \circ \rho_2)\cdot f(\rho_1\cdot x) && \text{Lemma 5.18, Definition 5.4}\\
&= \rho\cdot f((\rho_2 \circ \rho_1)\cdot x) && dom(\rho_2) \cap S_f = \varnothing\\
&= \rho\cdot f(x) && \text{Definition 5.4} \quad \square
\end{aligned}$$

5.2.3. *Definition of the exponential*

Definition 5.23. Write $\mathsf{X}^{\rightrightarrows} \Rightarrow \mathsf{Y}^{\rightrightarrows}$ for the renaming set with underlying set those $f \in |\mathsf{X}^{\rightrightarrows}| \to |\mathsf{Y}^{\rightrightarrows}|$ that are supported in the sense of Definition 5.15, and renaming action as defined in Definition 5.20.

Lemma 5.24. *If f is supported in the sense of Definition 5.15 then it is supported by S_f in the sense of Definition 5.4. Thus, $\mathsf{X}^{\rightrightarrows} \Rightarrow \mathsf{Y}^{\rightrightarrows}$ is indeed a permissive-nominal renaming set.*

Proof. It suffices to show that if $a \notin S_f$ then $([a{:=}b]\cdot f)(x) = f(x)$. This follows by routine calculations. □

Lemma 5.25. PmsRen *(Definition 5.13) is cartesian closed:*

- *The exponential is $\mathsf{X}^{\rightrightarrows} \Rightarrow \mathsf{Y}^{\rightrightarrows}$ from Definition 5.23.*
- *Products are given pointwise as in Definition 5.36.*
- *The terminal object $1^{\rightrightarrows}$ is the singleton set $\{0\}$ with the trivial action $\rho\cdot 0 = 0$.*

Proof. The bijection between $(\mathsf{X}^{\leftrightharpoons} \times \mathsf{Y}^{\leftrightharpoons}) \to \mathsf{Z}^{\leftrightharpoons}$ and $\mathsf{X}^{\leftrightharpoons} \to (\mathsf{X}^{\leftrightharpoons} \Rightarrow \mathsf{Y}^{\leftrightharpoons})$ is given by currying and uncurrying as usual. Thus $G : (\mathsf{X}^{\leftrightharpoons} \times \mathsf{Y}^{\leftrightharpoons}) \to \mathsf{Z}^{\leftrightharpoons}$ maps to $x \mapsto \lambda y.G(x, y)$. It is not hard to verify that if $dom(\rho) \cap supp(x) = \varnothing$ then

$$(\rho\bullet\lambda y.F(x, y))(y) = \rho\bullet F(x, y) = F(x, \rho\bullet y) = (\lambda y.F(x, y))(\rho\bullet y)$$

Thus $\lambda y.G(x, y)$ is supported by $supp(x)$ and is in $\mathsf{Y}^{\leftrightharpoons} \Rightarrow \mathsf{Z}^{\leftrightharpoons}$. □

We take a moment to build a particular exponential which will be useful later.

Definition 5.26. Suppose $x \in |\mathsf{X}^{\leftrightharpoons}|$ and $a \in \mathbb{A}_\nu$. Write $\lambda a.x \in |\mathbb{A}_\nu| \to |\mathsf{X}^{\leftrightharpoons}|$ for the function mapping a to x and b to $[a{:=}b]\bullet x$.

Lemma 5.27. $\lambda a.x \in |\mathbb{A}_\nu \Rightarrow \mathsf{X}^{\leftrightharpoons}|$.

Proof. It suffices to show that $\lambda a.x$ is supported by $supp(x)$ (in fact, it is also supported by $supp(x)\setminus\{a\}$). Suppose $dom(\rho) \cap supp(x) = \varnothing$ and $z \in \mathbb{A}_\nu$ (z is not necessarily distinct from a). Write $\rho{-}a$ for the renaming such that $(\rho{-}a)(b) = \rho(b)$ and $(\rho{-}a)(a) = a$. We sketch the relevant reasoning:

$$\rho\bullet((\lambda a.x)z) = (\rho \circ [a{:=}z])\bullet x = ([a{:=}\rho(z)] \circ (\rho{-}a))\bullet x = [a{:=}\rho(z)]\bullet x = (\lambda a.x)(\rho\bullet z) \quad \square$$

5.3. Atoms, products, atoms-abstraction, and functions out of atoms

5.3.1. Atoms

Definition 5.28. Write $\mathbb{B}$ for the nominal set and the permutation/renaming set with underlying set $\{0, 1\}$ and the **trivial** permutation/renaming action such that $\pi\cdot x = x/\rho\bullet x = x$ always.

We will be lax and write $x \in \mathbb{B}$ for $x \in |\mathbb{B}|$.

Write $\mathbb{A}_\nu$ for the permutation set and the renaming set with underlying set $\mathbb{A}_\nu$ and the natural permutation/renaming action such that $\pi\cdot x = \pi(x)/\rho\bullet x = \rho(x)$ always.

We will be lax and write $x \in \mathbb{A}_\nu$ for $x \in |\mathbb{A}_\nu|$.

5.3.2. Atoms-abstraction in permutation and renaming sets

Definition 5.29. Suppose X° is a supported permutation set. Suppose $x \in |\mathsf{X}^{\circ}|$ and $a \in \mathbb{A}_\nu$. Define **atoms-abstraction** $[a]x$ and $[\mathbb{A}_\nu]\mathsf{X}^{\circ}$ by:

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

Lemma 5.30. *Suppose X° is a supported permutation set.*

1. $[\mathbb{A}_\nu]\mathsf{X}^{\circ}$ *is a supported permutation set.*
2. $[a]x{=}[a]x'$ *if and only if* $x{=}x'$, *for* $a{\in}\mathbb{A}_\nu$ *and* $x{\in}|\mathsf{X}^{\circ}|$.
3. $[a]x{=}[a']x'$ *if and only if* $a'{\notin}supp(x)$ *and* $(a'\ a)\cdot x{=}x'$, *for* $a, a'{\in}\mathbb{A}_\nu$ *and* $x, x'{\in}|\mathsf{X}^{\circ}|$.

We do not need Definition 5.31 for the completeness proof but we include it for the interested reader to compare and contrast with Definition 5.29.

Definition 5.31. Suppose $\mathsf{X}^{\leftrightharpoons}$ is a supported renaming set. Suppose $x \in |\mathsf{X}^{\leftrightharpoons}|$ and $a \in \mathbb{A}_\nu$. Define **atoms-abstraction** $[a]x$ and $[\mathbb{A}_\nu]\mathsf{X}^{\leftrightharpoons}$ by:

$$\boxed{\begin{aligned} [a]x &= \{(a, x)\} \cup \{(b, [a{:=}b]\bullet x) \mid b \in \mathbb{A}_\nu\setminus supp(x)\} \\ |[\mathbb{A}_\nu]\mathsf{X}^{\leftrightharpoons}| &= \{[a]x \mid a \in \mathbb{A}_\nu, x \in |\mathsf{X}^{\leftrightharpoons}|\} \\ \rho\bullet[a]x &= [a]\rho\bullet x \quad (a \notin nontriv(\rho)) \end{aligned}}$$

Remark 5.32. Definitions 5.29 and 5.31 look similar; both define graphs of partial functions defined on $supp(x) \setminus \{a\}$. However, the critical difference is that in renaming sets, this partial function can be extended to a total function in $\mathbb{A}_\nu \to \mathsf{X}^{\leftrightharpoons}$.

That is, $[a]x \in [\mathbb{A}_\nu]\mathsf{X}^{\leftrightharpoons}$ determines the total function $\lambda a.x$ from Definition 5.26, mapping a to x and any other b to $[a{:=}b]\bullet x$. We return to this in Lemma 7.3 where we show that the natural map from $[\mathbb{A}_\nu]\mathsf{X}^{\leftrightharpoons}$ to $\mathbb{A}_\nu \Rightarrow \mathsf{X}^{\leftrightharpoons}$ which we construct in a moment in Definition 5.34, is not surjective. So Definition 5.31 identifies a 'small' and 'well-behaved' subset of the function space.

A cognate of Lemma 5.30 also holds for $[\mathbb{A}_\nu]\mathsf{X}^{\leftrightharpoons}$:

Lemma 5.33. *Suppose $\mathsf{X}^{\leftrightharpoons}$ is a supported renaming set.*

1. $[\mathbb{A}_\nu]\mathsf{X}^{\leftrightharpoons}$ *is a supported renaming set.*

2. *$[a]x=[a]x'$ if and only if $x=x'$, for $a\in\mathbb{A}_\nu$ and $x\in|\mathsf{X}^{\rightrightarrows}|$.*
3. *$[a]x=[a']x'$ if and only if $a'\notin supp(x)$ and $(a'\ a){\cdot}x=x'$ (or equivalently $[a{:=}a']{\bullet}x=x'$), for $a, a'\in\mathbb{A}_\nu$ and $x, x'\in|\mathsf{X}^{\rightrightarrows}|$.*

It may be useful to organise Definitions 5.26 and 5.31 and Lemma 5.33 into two functors and a natural transformation:

Definition 5.34. Write $[\mathbb{A}_\nu]$- for the functor taking $\mathsf{X}^{\rightrightarrows}$ to $[\mathbb{A}_\nu]\mathsf{X}^{\rightrightarrows}$ and taking $G : \mathsf{X}^{\rightrightarrows} \longrightarrow \mathsf{Y}^{\rightrightarrows}$ to $[\mathbb{A}_\nu]G : \mathbb{A}_\nu{\Rightarrow}\mathsf{X}^{\rightrightarrows} \longrightarrow \mathbb{A}_\nu{\Rightarrow}\mathsf{Y}^{\rightrightarrows}$ which maps $[a]x$ to $[a]G(x)$.

Write $\mathbb{A}_\nu{\Rightarrow}$- for the functor taking $\mathsf{X}^{\rightrightarrows}$ to $\mathbb{A}_\nu{\Rightarrow}\mathsf{X}^{\rightrightarrows}$ and taking $G : \mathsf{X}^{\rightrightarrows} \longrightarrow \mathsf{Y}^{\rightrightarrows}$ to $\mathbb{A}_\nu{\Rightarrow}G : \mathbb{A}_\nu{\Rightarrow}\mathsf{X}^{\rightrightarrows} \longrightarrow \mathbb{A}_\nu{\Rightarrow}\mathsf{Y}^{\rightrightarrows}$ which maps f to $G\circ f = \lambda n{\in}\mathbb{A}_\nu.G(f(n))$.

Finally, write *AbsFun* for the natural transformation from $[\mathbb{A}_\nu]$- to $\mathbb{A}_\nu{\Rightarrow}$- such that $AbsFun(\mathsf{X}^{\rightrightarrows}) : [\mathbb{A}_\nu]\mathsf{X}^{\rightrightarrows} \longrightarrow \mathbb{A}_\nu{\Rightarrow}\mathsf{X}^{\rightrightarrows}$ maps $[a]x$ to $\lambda a.x$ (Definition 5.26).

Lemma 5.35. *The functors $[\mathbb{A}_\nu]$- and $\mathbb{A}_\nu{\Rightarrow}$- and the natural transformation AbsFun are indeed well-defined.*

Proof. By routine calculations using the fact that $G : \mathsf{X}^{\rightrightarrows} \longrightarrow \mathsf{Y}^{\rightrightarrows}$ is equivariant (Definition 5.10). We consider two of the relevant calculations:

- *If $G : \mathsf{X}^{\rightrightarrows} \longrightarrow \mathsf{Y}^{\rightrightarrows}$ then $[\mathbb{A}_\nu]G$ is well-defined.* Suppose $[a]x = [b]y \in |[\mathbb{A}_\nu]\mathsf{X}^{\rightrightarrows}|$. By Lemma 5.33 $b \notin supp(x)$ and $(b\ a){\cdot}x = y$. By equivariance $G((b\ a){\bullet}x) = (b\ a){\bullet}G(x)$. By part 2 of Lemma 5.12 $b \notin supp(G(x))$. By Lemma 5.33 $[a]G(x) = [b]G(y)$.
- *AbsFun is a natural transformation.* We need that $\lambda a.(G(x)) = \lambda n{\in}\mathbb{A}_\nu.G((\lambda a.x)n)$. Unpacking Definition 5.26 $(\lambda a.x)n = [a{:=}n]{\bullet}x$ (here $n{\in}\mathbb{A}_\nu$ is not necessarily distinct from a). By equivariance of G, $\lambda n{\in}\mathbb{A}_\nu.G((\lambda a.x)n) = \lambda n{\in}\mathbb{A}_\nu.[a{:=}n]{\bullet}G(x)$. This is exactly equal to $\lambda a.G(x)$ as required. □

In part 4 of Lemma 7.3 we prove that *AbsFun* does not have an inverse. This merely points out the obvious: there are more functions from $\mathbb{A}_\nu$ to $\mathsf{X}^{\rightrightarrows}$ than can be represented in the form $\lambda a.x = \lambda n{\in}\mathbb{A}_\nu.[a{:=}n]{\bullet}x$.

5.3.3. Product

Definition 5.36. If X_i° and $\mathsf{X}_i^{\rightrightarrows}$ are supported permutation and renaming sets respectively for $1 \leq i \leq n$ then define $\mathsf{X}_1^{\circ}\times\cdots\times\mathsf{X}_n^{\circ}$ and $\mathsf{X}_1^{\rightrightarrows}\times\cdots\times\mathsf{X}_n^{\rightrightarrows}$ by:

$$\begin{array}{ll} |\mathsf{X}_1^{\circ}\times\ldots\times\mathsf{X}_n^{\circ}| = |\mathsf{X}_1^{\circ}|\times\ldots\times|\mathsf{X}_n^{\circ}| & |\mathsf{X}_1^{\rightrightarrows}\times\ldots\times\mathsf{X}_n^{\rightrightarrows}| = |\mathsf{X}_1^{\rightrightarrows}|\times\ldots\times|\mathsf{X}_n^{\rightrightarrows}| \\ \pi\cdot(x_1,\ldots,x_n) = (\pi\cdot x_1,\ldots,\pi\cdot x_n) & \rho\bullet(x_1,\ldots,x_n) = (\rho\bullet x_1,\ldots,\rho\bullet x_n) \end{array}$$

Lemma 5.37.
- *$supp(a) = \{a\}$.*
- *$supp([a]x) = supp(x)\setminus\{a\}$.*
- *$supp((x_1,\ldots,x_n)) = \bigcup\{supp(x_i) \mid 1 \leq i \leq n\}$.*

Proof. By routine arguments like those in [41] or [23, Corollary 2.30 & Theorem 3.11]. □

5.4. The free extension of a permutation set to a renaming set

We now show how to construct a renaming set $ren(\mathsf{X}^{\circ})$ out of a nominal set. At the start of Section 5 we noted that an atoms-abstraction $[a]x \in |[\mathbb{A}_\nu]\mathsf{X}^{\circ}|$ can be viewed as a partial function and that in renaming sets atoms-abstraction also exists but can be completed to a total function (Remark 5.32). We can view our free construction as a canonical way to move from a world in which atoms-abstraction is partial, to a world in which it is total.

Notation 5.38. If $\sim$ is an equivalence relation, $[\text{-}]_\sim$ will denote the equivalence class of - in $\sim$.

Definition 5.39. We define a functor *ren*(-) from PmsPrm to PmsRen as follows:

- *Action of ren(-) on objects.*
  X° maps to $ren(\mathsf{X}^{\circ}) = ((\mathbb{R}\times|\mathsf{X}^{\circ}|)/{\sim}, \bullet)$ where $\rho\bullet[(\rho', x)]_\sim = [(\rho\circ\rho', x)]_\sim$ and $\sim$ is the least equivalence relation such that:

  1. If $\rho(a) = \rho'(a)$ for every $a \in supp(x)$ then $(\rho, x) \sim (\rho', x)$.
  2. $(\rho\circ\pi, x) \sim (\rho, \pi\cdot x)$.

  For convenience we will write $\rho\bullet x$ as shorthand for $[(\rho, x)]_\sim$.[16]

[16] $\rho\bullet x$ is not 'ρ acting on x' and cannot be, since $x \in |\mathsf{X}^{\circ}|$ only has a permutation action. However this notation gives us cleaner-looking maths and the nice equality $\rho\bullet(id\bullet x) = \rho\bullet x$ (in long form: $\rho\bullet[(id, x)]_\sim = [(\rho, x)]_\sim$). The notation follows the nominal terms tradition of writing $\pi\cdot X$ both for 'π acting on the moderated unknown $id\cdot X$', and for 'the moderated unknown $\pi\cdot X$' [55].

- *Action of $ren(\text{-})$ on arrows.*
  An arrow $F : \mathsf{X}^{\circ} \longrightarrow \mathsf{Y}^{\circ}$ maps to $ren(F) : ren(\mathsf{X}^{\circ}) \longrightarrow ren(\mathsf{Y}^{\circ})$ given by:
  $$ren(F)(\rho \bullet x) = \rho \bullet F(x)$$

Lemma 5.40. *$ren(F)$ is well-defined; that is, that if $(\rho, x) \sim (\rho', x')$ then $ren(F)((\rho, x)) \sim ren(F)((\rho', x'))$.*

Proof. Induction on the derivation that $(\rho, x) \sim (\rho', x')$. We consider the two base cases:

- *The case $\rho(a) = \rho'(a)$ for every $a \in supp(x)$.* By part 2 of Lemma 5.12 also $\rho(a) = \rho'(a)$ for every $a \in supp(F(x))$.
- *The case $(\rho \circ \pi, x) \sim (\rho, \pi \cdot x)$.* Then also $(\rho \circ \pi, F(x)) \sim (\rho, \pi \cdot F(x))$ and by equivariance $\pi \cdot F(x) = F(\pi \cdot x)$. □

Remark 5.41. Rules 2 and 1 of Definition 5.39 can be viewed as α-conversion and garbage-collection respectively. Thus in $\rho \bullet x \in ren(\mathsf{X}^{\circ})$ we may without loss of generality (using rule 2) assume that $dom(\rho) \cap S = \varnothing$ for any permission set S, and we may also assume (using rule 1) that $dom(\rho) \subseteq supp(x)$.

Lemma 5.42. 1. *$ren(\mathbb{B})$ (for $\mathbb{B}$ considered a set with the trivial permutation action) is isomorphic to $\mathbb{B}$ (for $\mathbb{B}$ considered a set with a trivial renaming action).*
2. *$ren(\mathbb{A}_\nu)$ (for $\mathbb{A}_\nu$ with its natural permutation action) is isomorphic to $\mathbb{A}_\nu$ (for $\mathbb{A}_\nu$ with its natural renaming action).*

Proof. We consider only the second part. This follows if we note that according to the rules for $\sim$ in Definition 5.39,

$$(\rho, a) \overset{\textit{rule 1}}{\sim} ((\rho(a)\ a), a) \overset{\textit{rule 2}}{\sim} (id, \rho(a)) \quad \square$$

Where we are dealing with more than zero or one atoms at a time, isomorphisms like those in Lemma 5.42 may fail:

Lemma 5.43. *$ren(\mathbb{A}_\nu \times \mathbb{A}_\nu)$ is not isomorphic to $ren(\mathbb{A}_\nu) \times ren(\mathbb{A}_\nu)$ (which is isomorphic to $\mathbb{A}_\nu \times \mathbb{A}_\nu$).*

Proof. Consider the element $[a{:=}b] \bullet (a, b)$. □

6. Interpretation of permissive-nominal logic

6.1. Interpretation of signatures

Definition 6.1. Suppose $(\mathcal{A}, \mathcal{B})$ is a sort-signature (Definition 2.1).
A **PNL interpretation** $\mathcal{I}$ for $(\mathcal{A}, \mathcal{B})$ consists of an assignment of a nonempty supported permutation set $\tau^{\mathcal{I}}$ to each $\tau \in \mathcal{B}$.
We extend an interpretation $\mathcal{I}$ to sorts by:

$$\boxed{\begin{array}{ll} [\![\tau]\!]^{\mathcal{I}} = \tau^{\mathcal{I}} & [\![(\alpha_1, \ldots, \alpha_n)]\!]^{\mathcal{I}} = [\![\alpha_1]\!]^{\mathcal{I}} \times \ldots \times [\![\alpha_n]\!]^{\mathcal{I}} \\ [\![\nu]\!]^{\mathcal{I}} = \mathbb{A}_\nu & [\![[\nu]\alpha]\!]^{\mathcal{I}} = [\mathbb{A}_\nu][\![\alpha]\!]^{\mathcal{I}} \end{array}}$$

Definition 6.2. Suppose $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{P}, ar, \mathcal{X})$ is a signature (Definition 2.5).
A **(non-equivariant) PNL interpretation** $\mathcal{I}$ for $\mathcal{S}$ consists of the following data:

- An interpretation for the sort-signature $(\mathcal{A}, \mathcal{B})$ (Definition 6.1).
- For every $\mathsf{f} \in \mathcal{F}$ with $ar(\mathsf{f}) = (\alpha')\alpha$ an equivariant function $\mathsf{f}^{\mathcal{I}}$ from $[\![\alpha']\!]^{\mathcal{I}}$ to $[\![\alpha]\!]^{\mathcal{I}}$ (Definition 5.10).
- For every $\mathsf{P} \in \mathcal{P}$ with $ar(\mathsf{P}) = \alpha$ a supported function $\mathsf{P}^{\mathcal{I}}$ from $[\![\alpha]\!]^{\mathcal{I}}$ to $\{0, 1\}$.

If every $\mathsf{P}^{\mathcal{I}}$ is equivariant, then call $\mathcal{I}$ a **fully equivariant** interpretation.[17]

6.2. Interpretation of terms

Definition 6.3. Suppose $\mathcal{I}$ is an interpretation for $\mathcal{S}$. A **valuation** ς to $\mathcal{I}$ is a map on unknowns such that for each unknown X,

[17] A non-equivariant PNL interpretation still interprets term-formers equivariantly. Only the predicates might not be equivariant. We do this in order to completely model $(\mathbf{Ax}^{\pi})$ from Fig. 2, so that $\mathsf{P}(r) \not\Leftrightarrow \mathsf{P}(\pi \cdot r)$; see Theorem A.9. Of course it is possible to imagine a notion of non-equivariant interpretation where term-formers are interpreted as non-equivariant functions. This would correspond to something else: namely, to losing the property that $\pi \cdot \mathsf{f}(r) = \mathsf{f}(\pi \cdot r)$.

- $\varsigma(X) \in [\![sort(X)]\!]^{\mathcal{I}}$, and
- $supp(\varsigma(X)) \subseteq pms(X)$.

ς will range over valuations.

Having interpreted sorts α in Definition 6.1 as permissive-nominal sets $[\![\alpha]\!]^{\mathcal{I}}$, we now interpret nominal terms $r : \alpha$ as elements of those sets $[\![r]\!]^{\mathcal{I}}_{\varsigma}$:

Definition 6.4. Suppose $\mathcal{I}$ is an interpretation of a signature $\mathcal{S}$. Suppose ς is a valuation to $\mathcal{I}$.
Define an **interpretation** $[\![r]\!]^{\mathcal{I}}_{\varsigma}$ in $\mathcal{S}$ by:

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

Lemma 6.5. *If $r : \alpha$ then $[\![r]\!]^{\mathcal{I}}_{\varsigma} \in [\![\alpha]\!]^{\mathcal{I}}$.*

Proof. By a routine induction on r. □

Lemma 6.6. $\pi\cdot[\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![\pi\cdot r]\!]^{\mathcal{I}}_{\varsigma}$.

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

- *The case $\pi'\cdot X$.* By Definition 6.4 $[\![\pi'\cdot X]\!]^{\mathcal{I}}_{\varsigma} = \pi'\cdot\varsigma(X)$. Therefore $\pi\cdot[\![\pi'\cdot X]\!]^{\mathcal{I}}_{\varsigma} = \pi\cdot(\pi'\cdot\varsigma(X))$. It is a fact of the group action (Definition 5.3) that $\pi\cdot(\pi'\cdot\varsigma(X)) = (\pi\circ\pi')\cdot\varsigma(X)$, and of the permutation action (Definition 2.11) that $\pi\cdot(\pi'\cdot X) = (\pi\circ\pi')\cdot X$. The result follows. □

Lemma 6.7. *$supp([\![r]\!]^{\mathcal{I}}_{\varsigma}) \subseteq fa(r)$.*

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

- *The case $\pi\cdot X$.* $fa(\pi\cdot X) = \pi\cdot pms(X)$ by Definition 2.15. By assumption in Definition 6.3 $supp(\varsigma(X)) \subseteq pms(X)$.

The cases of a, $[a]r$, and $(r_1,\dots,r_n)$ use parts 1, 2, and 3 of Lemma 5.37. The case of f uses part 1 of Lemma 5.12. □

6.3. Interpretation of propositions

Definition 6.8. Suppose ς is a valuation to an interpretation $\mathcal{I}$. Suppose X is an unknown and $x \in [\![sort(X)]\!]^{\mathcal{I}}$ is such that $supp(x) \subseteq pms(X)$. Define $\varsigma[X{:=}x]$ by

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

It is easy to verify that $\varsigma[X{:=}x]$ is also a valuation to $\mathcal{I}$.

Definition 6.9. Suppose $\mathcal{I}$ is an interpretation. Define an **interpretation of propositions** by:

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

We may identify $[\![\phi]\!]^{\mathcal{I}}$ with a set of valuations $\{\varsigma \mid [\![\phi]\!]^{\mathcal{I}}_{\varsigma} = 1\}$. We discuss soundness and completeness in Appendix.

Lemma 6.10.
- $[\![r]\!]^{\mathcal{I}}_{\varsigma[X:=[\![r']\!]^{\mathcal{I}}_{\varsigma}]} = [\![r[X{:=}r']]\!]^{\mathcal{I}}_{\varsigma}$.
- $[\![\phi]\!]^{\mathcal{I}}_{\varsigma[X:=[\![r']\!]^{\mathcal{I}}_{\varsigma}]} = [\![\phi[X{:=}r']]\!]^{\mathcal{I}}_{\varsigma}$.

Proof. By routine inductions on the definitions of $[\![r]\!]^{\mathcal{I}}_{\varsigma}$ and $[\![\phi]\!]^{\mathcal{I}}_{\varsigma}$ in Definitions 6.4 and 6.9. We consider two cases:

- The case of $[\![\pi\cdot X]\!]^{\mathcal{I}}_{\varsigma[X:=r']}$ We reason as follows:

$$\begin{array}{rll} [\![\pi\cdot X]\!]^{\mathcal{I}}_{\varsigma[X:=[\![r']\!]^{\mathcal{I}}_{\varsigma}]} &= \pi\cdot[\![r']\!]^{\mathcal{I}}_{\varsigma} & \text{Definition 6.4} \\ &= [\![\pi\cdot r']\!]^{\mathcal{I}}_{\varsigma} & \text{Lemma 6.6} \\ &= [\![(\pi\cdot X)[X{:=}r']]\!]^{\mathcal{I}}_{\varsigma} & \text{Definition 2.26} \end{array}$$

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

$$\begin{array}{rll} [\![\mathsf{P}(r)]\!]^{\mathcal{I}}_{\varsigma[X:=[\![r']\!]^{\mathcal{I}}_{\varsigma}]} &= \mathsf{P}^{\mathcal{I}}([\![r]\!]^{\mathcal{I}}_{\varsigma[X:=[\![r']\!]^{\mathcal{I}}_{\varsigma}]}) & \text{Definition 6.9} \\ &= \mathsf{P}^{\mathcal{I}}([\![r[X{:=}r']]\!]^{\mathcal{I}}_{\varsigma}) & \text{Part 1 of this result} \\ &= [\![\mathsf{P}(r)[X{:=}r']]\!]^{\mathcal{I}}_{\varsigma} & \text{Definition 6.9.}\quad\Box \end{array}$$

Lemma 6.11. *If $\varsigma(X) = \varsigma'(X)$ for all $X \in fU(r)$ then $[\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![r]\!]^{\mathcal{I}}_{\varsigma'}$, and similarly for ϕ.*

Proof. By a routine induction on r and ϕ. □

7. Interpretation of HOL

For this section fix some PNL interpretation $\mathcal{I}$ of a PNL signature $\mathcal{S}$. Recall from Definition 4.2 the definition of the corresponding HOL signature $\mathcal{T}_{\mathcal{S}}$.

We have our interpretation of PNL and we have from Definition 4.3 a translation of PNL syntax to HOL syntax. We also have a functor from nominal sets to renaming sets (Definition 5.39). It remains to interpret HOL in renaming sets consistent with these interpretations and translations. This is Definitions 7.1 and 7.6, and the key technical result Lemma 8.9. Completeness follows quickly as a corollary (Theorem 8.11).

Note that in the interpretation (Definition 7.1) the type $\mu_\nu \to \beta$ is not necessarily interpreted as the set of all functions; it may be interpreted as a small subset of this function space. This is an old idea: since Henkin, models of HOL have been constructed to cut down on the full function-space (e.g. to create a complete semantics [2, Section 55]; see also [3] for a survey of non-standard semantics for HOL).

What we need to prove completeness of the syntactic translation $\lfloor\text{-}\rfloor^p$ is the existence of *some* interpretation of HOL with certain properties. This should not be mistaken as a commitment of nominal techniques to using this model of HOL always (unless we want to).

7.1. Interpretation of types

Recall the definition of a valuation ς (Definition 6.3) to an interpretation $\mathcal{I}$ for the PNL signature $\mathcal{S}$. Recall the definition of $\varsigma[X{:=}x]$ (Definition 6.8), and the interpretations of terms $[\![r]\!]^{\mathcal{I}}_{\varsigma}$ (Definition 6.4) and propositions $[\![\phi]\!]^{\mathcal{I}}_{\varsigma}$ (Definition 6.9).

We give similar definitions for HOL and renaming sets, culminating with Theorem 7.15 (soundness).

Definition 7.1. We provide an interpretation $\mathcal{H}$ of $\mathcal{T}_{\mathcal{S}}$ by:

$$\begin{array}{rcll} [\![\lfloor\alpha\rfloor]\!]^{\mathcal{H}} &=& ren([\![\alpha]\!]^{\mathcal{I}}) & \\ [\![o]\!]^{\mathcal{H}} &=& \mathbb{B} & \\ [\![(\beta_1,\ldots,\beta_n)]\!]^{\mathcal{H}} &=& [\![\beta_1]\!]^{\mathcal{H}} \times \ldots \times [\![\beta_n]\!]^{\mathcal{H}} & (\beta_i \text{ not of the form } \lfloor\alpha\rfloor \text{ for at least one } i) \\ [\![\beta' \to \beta]\!]^{\mathcal{H}} &=& [\![\beta']\!]^{\mathcal{H}} \Rightarrow [\![\beta]\!]^{\mathcal{H}} & (\beta' \text{ or } \beta \text{ not of the form } \lfloor\alpha\rfloor) \end{array}$$

Recall $\mathsf{X}^{\sqsupseteq} \Rightarrow \mathsf{Y}^{\sqsupseteq}$ from Definition 5.23 and $\mathsf{X}^{\sqsupseteq} \times \mathsf{Y}^{\sqsupseteq}$ from Definition 5.36.

Remark 7.2. Not all function types are interpreted equally by Definition 7.1.

If a type is the image of a PNL sort then we handle it using the first clause by wrapping it up in ren(-). Otherwise the interpretation is as standard: pairs to product; function types to the (supported) function set. This case-split makes Lemma 8.9 work, which is central to Corollary 8.10 and to Completeness (Theorem 8.11).

Why Lemma 8.9 *could not* work if we did not do this, is indicated in Lemma 7.3. Briefly, $\mathbb{A}_\nu \Rightarrow$ - contains 'exotic elements' making it bigger than $[\mathbb{A}_\nu]$-, which readers familiar with higher-order abstract syntax would expect [14, *exotic terms*].

Perhaps less familiar from Lemma 8.9 is that the free construction ren(-) does not commute with atoms-abstraction or even with cartesian product (which tells us that the associated functor does not have a left adjoint). So even e.g. $\mathbb{A}_\nu \times \mathbb{A}_\nu$ in PmsRen has an 'exotic element'.

The natural maps below in Lemma 7.3 are functors (some are defined in Definition 5.34). Technically, Lemma 7.3 implies that various natural transformations between these functors do not have inverses. More loosely, this is a way of putting some formal measure to the intuition that the translation from (restricted) PNL to HOL cannot be surjective:

Lemma 7.3. *1. The natural map from $ren(\mathbb{A}_\nu)$ to $\mathbb{A}_\nu$ mapping $\rho{\bullet}a$ to $\rho(a)$, is a bijection (cf. Lemma 5.42).*

2. *The natural map from $ren(\mathsf{X}^{\circ} \times \mathsf{Y}^{\circ})$ to $ren(\mathsf{X}^{\circ}) \times ren(\mathsf{Y}^{\circ})$ mapping $\rho{\bullet}(x, y)$ to $(\rho{\bullet}x, \rho{\bullet}y)$ is neither surjective nor injective.*
3. *The natural map from $ren([\mathbb{A}_\nu]\mathsf{X}^{\circ})$ to $[\mathbb{A}_\nu]ren(\mathsf{X}^{\circ})$ mapping $\rho{\bullet}[a]x$ where $a \notin nontriv(\rho)$ to $[a]\rho{\bullet}x$, is not surjective.*
4. *The natural map from $[\mathbb{A}_\nu]\mathsf{Y}^{\sqsupseteq}$ to $\mathbb{A}_\nu \Rightarrow \mathsf{Y}^{\sqsupseteq}$ mapping $[a]x$ to $\lambda a.x$ (see Definitions 5.26 and 5.34), is not surjective.*

Proof. 1. By rule 2 of Definition 5.39.

2. Take $\mathsf{X}^{\circ} = \mathsf{Y}^{\circ} = \mathbb{A}_\nu$. The natural map from $ren(\mathsf{X}^{\circ} \times \mathsf{Y}^{\circ})$ to $ren(\mathsf{X}^{\circ}) \times ren(\mathsf{Y}^{\circ})$ takes $id{\bullet}(a, b)$ to $(id{\bullet}a, id{\bullet}b)$. By equivariance it must map $[a{:=}b]{\bullet}(a, b)$ to $(id{\bullet}b, id{\bullet}b)$. But then it is not injective, since $[a{:=}b]{\bullet}(a, b) \neq id{\bullet}(b, b)$ in $ren(\mathsf{X}^{\circ} \times \mathsf{Y}^{\circ})$.

   Now take $\mathsf{X}^{\circ} = \mathsf{Y}^{\circ} = \mathbb{A}_\nu \times \mathbb{A}_\nu$. It is not hard to see that $([a{:=}b]{\bullet}(a, b), [b{:=}a]{\bullet}(a, b))$ is not in the image of the natural map, so the map is also not surjective.
3. Take $\mathsf{X}^{\circ} = \mathbb{A}_\nu \times \mathbb{A}_\nu$ and consider $[a][a{:=}b]{\bullet}(a, b) \in [\mathbb{A}_\nu]ren(\mathsf{X}^{\circ})$.
4. Take $\mathsf{Y}^{\sqsupseteq} = \mathbb{A}_\nu$ for $\mathbb{A}_\nu$ considered a renaming set as in Definition 5.28. Consider $(b\ a) \in \mathbb{A}_\nu \Rightarrow \mathbb{A}_\nu$, mapping a to b, b to a, and all other c to c. $\square$

7.2. Interpretation of terms

Definition 7.4. A **(HOL) valuation** ϱ to $\mathcal{H}$ is a map on variables $X : \beta$ such that

- $\varrho(X) \in [\![\beta]\!]^{\mathcal{H}}$ for every variable X.
- $\{a \in \mathbb{A} \mid \varrho(a) \neq id{\cdot}a\}$ is finite.[18]

ϱ will range over valuations.

Definition 7.5. Suppose ϱ is a valuation. Suppose X is a variable and $x \in [\![type(X)]\!]^{\mathcal{H}}$. Define a function $\varrho[X{:=}x]$ by:

$$\boxed{(\varrho[X{:=}x])(Y) = \varrho(Y) \quad \text{and} \quad (\varrho[X{:=}x])(X) = x}$$

It is easy to verify that $\varrho[X{:=}x]$ is also a valuation to $\mathcal{H}$.

Definition 7.6. Extend $\mathcal{H}$ to terms as follows:

- $[\![a]\!]^{\mathcal{H}}(\varrho) = \varrho(a)$.
- $[\![X]\!]^{\mathcal{H}}(\varrho) = \varrho(X)$.
- $[\![\mathsf{g}_{\mathsf{f}}]\!]^{\mathcal{H}} = ren(\mathsf{f}^{\iota})$ and $[\![\mathsf{g}_{\mathsf{P}}]\!]^{\mathcal{H}} = ren(\mathsf{P}^{\iota})$ (Definition 5.39).
- $[\![\bot]\!]^{\mathcal{H}}(\varrho) = 0$.
- $[\![\Rightarrow]\!]^{\mathcal{H}}(\varrho) = \lambda x \in \mathbb{B}, y \in \mathbb{B}.max\{1{-}x, y\}$.
- $[\![\forall_{\beta})]\!]^{\mathcal{H}}(\varrho) = \lambda x \in [\![\beta \Rightarrow \mathbb{B}]\!]^{\mathcal{H}}.min\{xy \mid y \in [\![\beta]\!]^{\mathcal{H}}\}$.
- $[\![\lambda a.t]\!]^{\mathcal{H}}(\varrho) = \rho{\cdot}[a]x$ where $[\![t]\!]^{\mathcal{H}}(\varrho) = \rho{\cdot}x$ provided that $t : \lfloor\alpha\rfloor$ for some PNL sort α and $a \in \mathbb{A}_{\nu}$ for some name sort ν and (α-converting if necessary) $a \notin \bigcup_{X \in fv(t)\setminus\{a\}} supp(\varrho(X))$ and $\varrho(a) = id{\cdot}a$.
- $[\![\lambda X.t]\!]^{\mathcal{H}}(\varrho) = \lambda x.[\![t]\!]^{\mathcal{H}}(\varrho[X{:=}x])$ provided that $\lambda X.t : \beta' \to \beta$ where $\beta' \to \beta$ is not equal to $\lfloor[\mathbb{A}_{\nu}]\alpha\rfloor$ for any ν or α.
- $[\![tu]\!]^{\mathcal{H}}(\varrho) = ([a{:=}b] \circ \rho){\cdot}x$ provided that $t : \lfloor[\nu]\alpha\rfloor$ for some PNL name sort ν and sort α, where $[\![u]\!]^{\mathcal{H}}(\varrho) = id{\cdot}b$ (by construction some such $b \in \mathbb{A}_{\nu}$ always exists) and $[\![t]\!]^{\mathcal{H}}(\varrho) = \rho{\cdot}[a]x$, and (renaming if necessary) $a \notin nontriv(\rho) \cup \{b\}$.
- $[\![tu]\!]^{\mathcal{H}}(\varrho) = [\![t]\!]^{\mathcal{H}}(\varrho)[\![u]\!]^{\mathcal{H}}(\varrho)$ provided that $t : \beta$ for β not equal to $\lfloor\alpha\rfloor$ for any PNL sort α (it is a fact that this case and the previous case exhaust all the possibilities for tu).
- $[\![(t_1, \ldots, t_n)]\!]^{\mathcal{H}}(\varrho) = (\bigcup \rho_i){\cdot}(x_1, \ldots, x_n)$ provided that $t_i : \lfloor\alpha_i\rfloor$ for $1 \leq i \leq n$, where $[\![t_i]\!]^{\mathcal{H}} = \rho_i{\cdot}x_i$, and we choose representatives such that $dom(\rho_i) \cap dom(\rho_j) = \varnothing$ for all $1 \leq i \neq j \leq n$.
- $[\![(t_1, \ldots, t_n)]\!]^{\mathcal{H}}(\varrho) = ([\![t_1]\!]^{\mathcal{H}}(\varrho), \ldots, [\![t_n]\!]^{\mathcal{H}}(\varrho))$ provided that there exists some i and β such that $t_i : \beta$ and β is not equal to $\lfloor\alpha\rfloor$ for any PNL sort α.

Remark 7.7. Definition 7.6 propagates to terms the case-split noted in Remark 7.2. We treat terms differently depending on whether they populate the translation of a PNL sort, or not. We must do this because of how we interpreted types in Definition 7.1.

Just to locate where we are, here is an schematic of the overall structure of the proof of completeness:

$$\begin{tikzcd}
\text{PNL syntax} \arrow[r, "\lfloor\text{-}\rfloor^{p}"] \arrow[d, "{[\![\text{-}]\!]^{\iota}_{\varsigma}}"'] & \text{HOL syntax} \arrow[d, "{[\![\text{-}]\!]^{\mathcal{H}}(D(\varsigma))}"] \arrow[dl, dashed, "\textit{not possible}"] \\
\mathsf{PmsPrm} \arrow[r, "ren(\text{-})"'] & \mathsf{PmsRen}
\end{tikzcd}$$

We translated PNL to HOL using $\lfloor\text{-}\rfloor^{p}$ in Definition 4.3. Ideally, to prove completeness we would now give HOL a denotation directly in PmsRen.

Unfortunately this is not possible (the dashed arrow) because $[a]r$ translates to $\lambda a.\lfloor r\rfloor^{p}$ and has nominal denotation as an atoms-abstraction $[a][\![r]\!]^{\iota}_{\varsigma}$. Atoms-abstraction (Definition 5.29) is the graph of a partial function, whereas $\lambda a.\lfloor r\rfloor^{p}$ 'wants' to take denotation as a total function.

So instead we use a commuting square as illustrated; in PmsRen atoms-abstraction *can* be viewed as a total function, as noted in Remark 5.32. Definition 7.6 uses this, and fills in the right-hand arrow. We also need to convert the PNL valuation ς to a HOL valuation $D(\varsigma)$; this comes later in Definition 8.7.

Note that by forming this diagram we give a new semantics to PNL in PmsRen, and thus in particular give a semantics to nominal atoms-abstraction in which it becomes interpreted as a total function.

The top arrow is Definition 4.3; the left-hand arrow is Definition 6.4; and the bottom arrow is Definition 5.39. Lemma 8.8 proves commutativity of the square.

[18] In other words, ϱ restricted to those HOL variables in μ_{ν} that are PNL atoms (condition 5 of Definition 4.2) is a renaming ρ (Definition 5.1).

Lemma 7.8. *Suppose $a \in \mathbb{A}_\nu$ and $b \in \mathbb{A}_\nu$. Suppose $a \notin supp(\varrho(X))$ for every $X \in fv(t)\backslash\{a\}$ (including b). Suppose $\varrho(a) = id{\bullet}a$. Then $[\![t]\!]^{\mathscr{H}}(\varrho[a{:=}id{\bullet}b]) = [a{:=}b]{\bullet}([\![t]\!]^{\mathscr{H}}(\varrho))$.*

Proof. By a routine induction on t. We mention two cases:

- The case t is a. Using the fact that $id{\bullet}b = [a{:=}b]{\bullet}a$ in $\mathbb{A}_\nu$ with the action described in Definition 5.28.
- The case t is X for some HOL variable that is not an atom. By assumption $a \notin supp(\varrho(X))$ and so by Definition 5.4, $\varrho(X) = [a{:=}b]{\bullet}\varrho(X)$. The result follows. □

Remark 7.9. Lemma 7.8 may fail if $a \in supp(\varrho(X))$. For instance, if $\varrho(X) = a$ where $a \in \mathbb{A}_\nu$ and $type(X) = \mu_\nu$ and X is not itself an atom, then $[\![X]\!]^{\mathscr{H}}(\varrho[a{:=}id{\bullet}b]) = id{\bullet}a$ yet $[a{:=}b]{\bullet}([\![X]\!]^{\mathscr{H}}(\varrho)) = [a{:=}b]{\bullet}(id{\bullet}a) = id{\bullet}b$.

We need to check that the denotation of terms populates the denotation of their types, and that β-equivalent terms receive equal denotations.

Lemma 7.10. *If $t : \beta$ then $[\![t]\!]^{\mathscr{H}}(\varrho) \in [\![\beta]\!]^{\mathscr{H}}$.*

Theorem 7.11. $[\![(\lambda X.t)u]\!]^{\mathscr{H}}(\varrho) = [\![t]\!]^{\mathscr{H}}(\varrho[X{:=}[\![u]\!]^{\mathscr{H}}(\varrho)])$.

Proof. There are two cases, depending on whether $\lambda X.t : \lfloor[\mathbb{A}_\nu]\alpha\rfloor$ for some PNL sort, or not.

- *The case $t : \lfloor\alpha\rfloor$.* By Definition 7.6 $[\![u]\!]^{\mathscr{H}}(\varrho) = id{\bullet}b$ and $[\![\lambda X.t]\!]^{\mathscr{H}}(\varrho) = \rho{\bullet}[a]x$, for some b, a, and x. α-converting if necessary assume X is equal to a which we choose fresh (so $a \notin nontriv(\rho) \cup \{b\}$ and $a \notin supp(\varrho(Y))$ for every $Y \in fv(t)\backslash\{a\}$ and $\varrho(a) = id{\bullet}a$). Then also by definition $[\![(\lambda a.t)u]\!]^{\mathscr{H}}(\varrho) = ([a{:=}b] \circ \rho){\bullet}x$.
  Thus it suffices to check that $([a{:=}b] \circ \rho){\bullet}x = [\![t]\!]^{\mathscr{H}}(\varrho[a{:=}b])$. This follows using Lemma 7.8.
- *The case $t : \beta$ where β is not equal to $\lfloor\alpha\rfloor$ for any PNL sort α.* This is as standard. □

7.3. Soundness

Lemma 7.12. *If $\varrho(X) = \varrho'(X)$ for all $X \in fv(t)$ then $[\![t]\!]^{\mathscr{H}}(\varrho) = [\![t]\!]^{\mathscr{H}}(\varrho')$.*

Proof. By a routine induction on terms. □

Lemma 7.13. $[\![t]\!]^{\mathscr{H}}(\varrho[X{:=}[\![u]\!]^{\mathscr{H}}(\varrho)]) = [\![t[X{:=}u]]\!]^{\mathscr{H}}(\varrho)$.

Proof. By a routine induction on t. We mention two cases (bearing in mind that in HOL, a variable $X : \beta$ may be an atom in $\mathbb{A}_\nu$ if $\beta = \mu_\nu$):

- *The case t equals X equals $a \in \mathbb{A}_\nu$ for some atom a.*
  By Definition 7.6, $[\![a]\!]^{\mathscr{H}}(\varrho[a{:=}[\![u]\!]^{\mathscr{H}}(\varrho)]) = [\![u]\!]^{\mathscr{H}}(\varrho)$.
- *The case t equals $\lambda Y.t'$.*
  We assume $Y \notin fv(u)$, so $(\lambda Y.t')[X{:=}u] = \lambda Y.(t'[X{:=}u])$, and use the inductive hypothesis. □

Definition 7.14 (*Validity*). Call the proposition ξ **valid** in $\mathscr{H}$ when $[\![\xi]\!]^{\mathscr{H}}(\varrho) = 1$ for all ϱ.
Call the sequent $\xi_1, \dots, \xi_n \overset{\lambda}{\vdash} \chi_1, \dots, \chi_p$ **valid** in $\mathscr{H}$ when $(\xi_1 \wedge \cdots \wedge \xi_n) \Rightarrow (\chi_1 \vee \cdots \vee \chi_p)$ is valid.
If this is true for all $\mathscr{H}$ then write $\xi_1, \dots, \xi_n \overset{\lambda}{\vDash} \chi_1, \dots, \chi_p$.

Theorem 7.15 (*Soundness*). *If $\Xi \overset{\lambda}{\vdash} \chi$ is derivable then $\Xi \overset{\lambda}{\vDash} \chi$.*

Proof. Fix some interpretation $\mathscr{H}$. We work by induction on derivations (Fig. 2). We sketch the two non-trivial cases:

- *The case of* (**h∀L**). We check that $u : type(X)$ implies $[\![\forall X.\xi]\!]^{\mathscr{H}}(\varrho) \leq [\![\xi[X{:=}u]]\!]^{\mathscr{H}}(\varrho)$. We reason as follows:

$$\begin{aligned}
[\![\forall X.\xi]\!]^{\mathscr{H}}(\varrho) &= min\{[\![\lambda X.\xi]\!]^{\mathscr{H}}(\varrho)y \mid y \in [\![type(X)]\!]^{\mathscr{H}}\} && \text{Definition 7.6}\\
&= min\{[\![\xi]\!]^{\mathscr{H}}(\varrho[X{:=}y]) \mid y \in [\![type(X)]\!]^{\mathscr{H}}\} && \text{Definition 7.6}\\
&\leq [\![\xi]\!]^{\mathscr{H}}(\varrho[X{:=}[\![u]\!]^{\mathscr{H}}(\varrho)]) && \text{Fact}\\
&= [\![\xi[X{:=}u]]\!]^{\mathscr{H}}(\varrho) && \text{Lemma 7.13}
\end{aligned}$$

  In the second use of Definition 7.6 above, note that $[\mathbb{A}_\nu]o$ is never of the form $\lfloor[\mathbb{A}_\nu]\alpha\rfloor$ for any α.
- *The case of* (**h∀R**). We use Lemma 7.12 and routine calculations on truth-values. □

8. Completeness of the translation of PNL to HOL

We are now ready to prove completeness (Theorem 8.11) of the translation from Definition 4.3. The proof is subtle; notably Lemma 8.4 and the case of $\forall X.\phi$ in Lemma 8.9 are non-trivial. Some mathematical action also takes place in Lemma 8.8 and the case of $\pi{\cdot}X$ in Lemma 8.9.

8.1. Renamings and HOL propositions

We need a few technical observations about how renamings interact with the denotations of HOL propositions:

Lemma 8.1. *Suppose $G : \mathsf{X}^{\rightleftharpoons} \longrightarrow \mathbb{B}$. Then for every ρ, $G(x) = 1$ implies $G(\rho{\bullet}x) = 1$.*

Proof. From equivariance and the fact that $\rho{\bullet}1 = 1$ in $\mathbb{B}$. □

Corollary 8.2. *Suppose $F : \mathsf{X}^{\circ} \longrightarrow \mathbb{B}$. Then $ren(F)(\rho{\bullet}x) = F(x)$.*

Notation 8.3. Write $\rho{\bullet}\varrho$ for the valuation mapping X to $\rho{\bullet}\varrho(X)$.

Lemma 8.4. *Suppose ξ is a HOL proposition. Then*

- *$[\![\xi]\!]^{\mathcal{H}}(\rho{\bullet}\varrho) = [\![\xi]\!]^{\mathcal{H}}(\varrho)$ for every ρ and ϱ, and*
- *as a corollary, if $X : \beta$ and $x \in [\![\beta]\!]^{\mathcal{H}}$ then $[\![\xi]\!]^{\mathcal{H}}(\varrho[X{:=}x]) = [\![\xi]\!]^{\mathcal{H}}(\varrho[X{:=}\rho{\bullet}x])$.*

Proof. We work by induction on ξ. For each ξ the corollary follows from the first part using a freshening pair of renamings (see Definition 5.19). For the first part, the case of g_{P} is by Corollary 8.2. The case of $\forall$ follows using the second part and some routine calculations. The cases of $\bot$ and $\Rightarrow$ are immediate. □

Remark 8.5. Lemma 8.4 expresses that $[\![\xi]\!]^{\mathcal{H}}$ does not examine atoms for inequality across its arguments (if it did then Lemma 8.4 could not hold, because ρ can identify atoms – make them become equal – in the denotations of variables in ξ). The corollary is even more powerful: we can even apply renamings to the denotations of individual free variables, and still not affect validity.

We use this in the case of $\forall X.\phi$ in Lemma 8.9 to 'jettison' unwanted ρ in the denotation of the quantified variable.

8.2. The completeness proof

Notation 8.6. Suppose $D = [d_1, \dots, d_n]$ is a finite list of distinct atoms in $\mathbb{A}_{\nu_1}, \dots, \mathbb{A}_{\nu_n}$ respectively. Suppose $r : \alpha$ is a PNL term. Then:

- Write $[D]r$ for the PNL term $[d_1] \dots [d_n]r$.
- Write $[\mathbb{A}_D]\alpha$ for the PNL sort $[\mathbb{A}_{\nu_1}] \dots [\mathbb{A}_{\nu_n}]\alpha$.

Definition 8.7. Given a finite list of distinct atoms D, map a PNL valuation ς to a HOL valuation $D(\varsigma)$ defined by

$$\boxed{\begin{array}{llll} D(\varsigma) \quad \text{maps} & X : \alpha & \text{to} & id{\bullet}[D \cap pms(X)]\varsigma(X) \in [\![\lfloor [\mathbb{A}_{D\cap pms(X)}]\alpha \rfloor]\!]^{\mathcal{H}} \quad \text{and} \\ & a : \nu & \text{to} & a \in \mathbb{A}_{\nu} \end{array}}$$

Compare Lemma 8.8 with Lemma 4.12:

Lemma 8.8. *Suppose $x \in [\![\alpha]\!]^{\iota}$ and $nontriv(\pi) \cap supp(x) \subseteq D'$. Then*

$$(id{\bullet}[D']x)\pi{\cdot}D' = id{\bullet}(\pi{\cdot}x)$$

Proof. From Definition 7.6 and rule 2 of Definition 5.39. □

Lemma 8.9 proves that the schematic diagram of Remark 7.7 does indeed commute:

Lemma 8.9. *Suppose $r : \alpha$ and ϕ is a proposition. Then:*

- *If $D \vdash r$ then $[\![\lfloor r \rfloor^{\mathsf{P}}]\!]^{\mathcal{H}}(D(\varsigma)) = id{\bullet}[\![r]\!]^{\iota}(\varsigma)$.*
- *If $D \vdash \phi$ then $[\![\lfloor \phi \rfloor^{\mathsf{P}}]\!]^{\mathcal{H}}(D(\varsigma)) = [\![\phi]\!]^{\iota}(\varsigma)$.*

Proof. By inductions on r and ϕ.

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

$$\begin{array}{lll} [\![\lfloor \pi{\cdot}X \rfloor^{\mathsf{P}}]\!]^{\mathcal{H}}(D(\varsigma)) & = [\![X_D\pi{\cdot}(D \cap pms(X))]\!]^{\mathcal{H}}(D(\varsigma)) & \text{Definition 4.3} \\ & = D(\varsigma)(X_D)\pi{\cdot}(D \cap pms(X)) & \text{Definition 7.6} \\ & = (id{\bullet}[D \cap pms(X)]\varsigma(X))\pi{\cdot}(D \cap pms(X)) & \\ & & \text{Definition 8.7} \\ & = id{\bullet}\pi{\cdot}\varsigma(X) & \text{Lemma 8.8,} \\ & & \quad supp(\varsigma(X)) \subseteq pms(X) \\ & = id{\bullet}[\![\pi{\cdot}X]\!]^{\iota}(\varsigma) & \text{Definition 6.4} \end{array}$$

Note of the penultimate step that by assumption $D \vdash r$, so by Definition 4.7 $nontriv(\pi) \cap pms(X) \subseteq D \cap pms(X)$.
- *The case $[a]r$.* By Definition 4.3

$$[\![\lfloor [a]r \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)) = [\![\lambda a.\lfloor r \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma))$$

By inductive hypothesis $[\![\lfloor r \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)) = id{\bullet}[\![r]\!]^{\mathcal{I}}(\varsigma)$ and so by Definition 7.6 (eliding routine freshness reasoning for a)

$$[\![\lambda a.\lfloor r \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)) = id{\bullet}[a][\![r]\!]^{\mathcal{I}}(\varsigma)$$

Finally by Definition 6.4

$$id{\bullet}[a][\![r]\!]^{\mathcal{I}}(\varsigma) = id{\bullet}[\![[a]r]\!]^{\mathcal{I}}(\varsigma)$$

and by transitivity of equality we are done.

- *The case* $\mathsf{P}(r)$. We reason as follows:

$$\begin{aligned}
[\![\lfloor \mathsf{P}(r) \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)) &= [\![\mathsf{g}_{\mathsf{P}}(\lfloor r \rfloor^D)]\!]^{\mathcal{H}}(D(\varsigma)) && \text{Definition 4.3}\\
&= \mathsf{g}_{\mathsf{P}}^{\mathcal{H}}([\![\lfloor r \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma))) && \text{Definition 6.9}\\
&= \mathsf{g}_{\mathsf{P}}^{\mathcal{H}}(id{\bullet}[\![r]\!]^{\mathcal{I}}(\varsigma)) && \text{part 1}\\
&= ren(\mathsf{P}^{\mathcal{I}})(id{\bullet}[\![r]\!]^{\mathcal{I}}(\varsigma)) && \text{Definition 7.6}\\
&= \mathsf{P}^{\mathcal{I}}([\![r]\!]^{\mathcal{I}}(\varsigma)) && \text{Corollary 8.2}\\
&= [\![\mathsf{P}(r)]\!]^{\mathcal{I}}(\varsigma) && \text{Definition 6.9}
\end{aligned}$$

- *The case* $\forall X.\phi$. Write $\alpha = sort(X)$. From Definition 7.6

$$[\![\lfloor \forall X.\phi \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)) = min\{[\![\lfloor \phi \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)[X{:=}x]) \mid x \in [\![\lfloor [\mathbb{A}_{D\cap pms(X)}]\alpha \rfloor]\!]^{\mathcal{H}}\}$$

By construction in Definition 7.1 every $x \in [\![\lfloor [\mathbb{A}_{D\cap pms(X)}]\alpha \rfloor]\!]^{\mathcal{H}}$ has the form $\rho{\bullet}x'$ for $x' \in [D \cap pms(X)][\![\alpha]\!]^{\mathcal{I}}$. By Lemma 8.4 we have

$$\begin{aligned}
&min\{[\![\lfloor \phi \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)[X{:=}x]) \mid x \in [\![\lfloor [\mathbb{A}_{D\cap pms(X)}]\alpha \rfloor]\!]^{\mathcal{H}}\}\\
&\qquad = min\{[\![\lfloor \phi \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)[X{:=}id{\bullet}x']) \mid x' \in [\![[\mathbb{A}_{D\cap pms(X)}]\alpha]\!]^{\mathcal{I}}\}
\end{aligned}$$

Using Lemma 8.4 again we assume without loss of generality that $supp([D \cap pms(X)]x') \subseteq pms(X) \setminus D \cap pms(X)$, and so:

$$\begin{aligned}
&min\{[\![\lfloor \phi \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)[X{:=}id{\bullet}[D \cap pms(X)]x']) \mid x' \in [\![[\mathbb{A}_{D\cap pms(X)}]\alpha]\!]^{\mathcal{I}}\}\\
&\qquad = min\{[\![\lfloor \phi \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)[X{:=}id{\bullet}x'']) \mid x'' \in [\![\alpha]\!]^{\mathcal{I}}, supp(x'')\subseteq pms(X)\}
\end{aligned}$$

Now we unfold definitions and use the inductive hypothesis which tells us that $D(\varsigma)[X{:=}id{\bullet}[D \cap pms(X)]x''] = D(\varsigma[X{:=}x''])$, and we obtain:

$$\begin{aligned}
&min\{[\![\lfloor \phi \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma)[X{:=}id{\bullet}[D \cap pms(X)]x'']) \mid x'' \in [\![\alpha]\!]^{\mathcal{I}}, supp(x'')\subseteq pms(X)\}\\
&\qquad = min\{[\![\lfloor \phi \rfloor^D]\!]^{\mathcal{H}}(D(\varsigma[X{:=}x''])) \mid x'' \in [\![\alpha]\!]^{\mathcal{I}}, supp(x'')\subseteq pms(X)\}\\
&\qquad = min\{[\![\phi]\!]^{\mathcal{I}}(\varsigma[X{:=}x'']) \mid x'' \in [\![\alpha]\!]^{\mathcal{I}}, supp(x'')\subseteq pms(X)\}\\
&\qquad = [\![\forall X.\phi]\!]^{\mathcal{I}}(\varsigma) \quad \Box
\end{aligned}$$

Corollary 8.10. *Suppose* $\Phi = \{\phi_1, \dots, \phi_n\}$ *and* $\Psi = \{\psi_1, \dots, \psi_p\}$ *and* $D \vdash \Phi$, *and* $D \vdash \Psi$ *(Definition 4.7). Suppose* $\mathcal{I}$ *is a PNL interpretation and suppose* $\phi_1, \dots, \phi_n \models^{\mathsf{PNL}} \psi_1, \dots, \psi_p$ *is not valid in* $\mathcal{I}$.

Then $\mathcal{H}$ *from Definition 7.1 is a HOL interpretation and* $\lfloor \phi_1 \rfloor^D, \dots, \lfloor \phi_n \rfloor^D \models^{\lambda} \lfloor \psi_1 \rfloor^D, \dots, \lfloor \psi_p \rfloor^D$ *is not valid in* $\mathcal{H}$.

Proof. Suppose ς is such that $[\![\phi_1 \wedge \cdots \wedge \phi_n]\!]^{\mathcal{I}}(\varsigma) = 1$ and $[\![\psi_1 \vee \cdots \vee \psi_p]\!]^{\mathcal{I}}(\varsigma) = 0$. We use Lemma 8.9 for $D(\varsigma)$ (Definition 8.7). $\Box$

Theorem 8.11 (*Completeness*)**.** *Suppose* $D \vdash \Phi$ *and* $D \vdash \Psi$. *If* $\Phi \not\models^{\mathsf{PNL}} \Psi$ *then* $\lfloor \Phi \rfloor^D \not\models^{\lambda} \lfloor \Psi \rfloor^D$.

Proof. We use the contrapositive of completeness of restricted PNL (Theorem A.9), then Corollary 8.10, then the contrapositive of HOL soundness (Theorem 7.15). $\Box$

9. Conclusions

We have translated a logic with its own proof-theory, syntax, and sound and complete semantics. Any formal theory specified in the PNL fragment of this paper can be systematically, soundly, and completely translated to HOL.

For the reader interested in nominal techniques, the main contribution of this paper is that in proving completeness of the translation, we have given another semantics of permissive nominal logic, besides the ‘obvious’ one in nominal sets. In this new semantics, a term of the form $[a]t$ is interpreted as a function, like $\lambda a.t$ would be in higher-order logic. This shows at the semantic level an implicit similarity between PNL and HOL (we discuss presheaves in the next Subsection).

For the reader interested in higher-order logic, this paper is of interest because its image is readily identified with the *higher-order patterns* developed by Miller [45] (so that, intuitively, restricted PNL could be thought of as a compact first-order logic and nominal semantics for higher-order patterns).

In this semantics the sort $[\mathbb{A}]\alpha$ is not interpreted as the set of all functions from atoms to the interpretation of α, but as a small subset of this function space. This is an old idea: since Henkin, models of HOL have been constructed to cut down on the full function-space (e.g. to create a complete semantics [2, Section 55]). Moreover in weak HOAS to avoid so-called *exotic terms*, function existence axioms must be weakened in HOL: for instance, the description axiom that entails the existence of a function for all functional relations has to be dropped (an alternative is to introduce an explicit modality [16]). We now have a new view of these 'smaller' function-spaces as being the image of nominal atoms-abstractions via the semantic operations considered in this paper.

9.1. The big picture

It is quite interesting to understand this paper via a contrast between what nominal foundations and 'ordinary' foundations provide.

The denotation of an open term in PNL is an open element of a nominal set (i.e. an element with atoms in its support): so for instance an atom maps just to itself in the denotation, rather than being assigned some other element by a valuation. PNL has binding term-formers because its open denotation in nominal sets provides constructions like atoms, permutations, and atoms-abstraction (Definition 5.29); see Section 6.

In contrast, in HOL there is no concept of open element—there are open *terms*, but their denotation is closed by a valuation. If we want the effect of a free variable then we Skolemise/raise. Indeed, *any* mathematics using technology based on the HOL/ZF (Zermelo–Fraenkel sets) foundations of mathematics *must* handle names either as something like functional arguments or something like numbers; simply because numbers and functions are what HOL/ZF foundations provide.

The translation from PNL to HOL in Section 4 works by raising.[19] However, we can only raise n variables, in order. So the translation has to be on a per-derivation basis, including the (finitely many) atoms of interest in that (finite) derivation. Furthermore, we lose the equivariance (name symmetry) of full PNL. So we can only naturally translate *individual* derivations in *restricted* PNL.

This matters because in losing symmetry we lose what makes nominal techniques so distinctive. So although we show how to translate a complete 'nominal' proof to a complete 'HOL' proof, we also see how the way in which nominal and HOL proofs are manipulated and combined, are different.

The translation is not entirely trivial to define and prove sound, but the technically hardest part in this paper is clearly the proof of its completeness. For this we build a hybrid denotation for HOL in nominal renaming sets which is sound but in which certain function spaces are restricted to be 'not too large'. That motivates the bulk of the technical mathematics of this paper.

9.2. Permissive nominal logic in perspective

Permissive-nominal logic is the endpoint–so far–of an evolution as follows:

- Fraenkel–Mostowski set theory and a first-order axiomatisation by Pitts introduced and described the underlying nominal sets models in first-order logic [41,51].
- Nominal terms introduced a dedicated syntax with two-levels of variable and freshness side-conditions [55].
- Nominal algebra and αProlog inserted nominal terms syntax into formal reasoning systems [37,7].
- Permissive-nominal terms introduced permission sets [13].
- PNL introduced a proof-theory and universal quantifier for nominal terms unknowns [9,10].

Meanwhile in the semantics

- Nominal renaming sets extended nominal sets from a permutation action to a renaming action [29].
- A permissive version of nominal algebra (an equality fragment of PNL) was given semantics in PmsPrm and theories were translated from HOL [38], but this was done purely syntactically without using nominal renaming sets and without considering universal quantification.

The categories PmsPrm and PmsRen from Definition 5.13 are identical to the categories of nominal sets and nominal renaming sets from [41] and [29], except that here we insist on supporting *permission* sets instead of supporting *finite* sets.

The reader familiar with presheaf techniques will see in PmsRen the category $\mathsf{Sets}^{\mathbb{F}}$ (presheaves over the category of finite sets and functions between them). PmsRen corresponds to presheaves (not quite over $\mathbb{F}$, as discussed in the previous paragraph) that preserve pullbacks of pairs of monos [29] and because of this it admits an arguably preferable sets-based presentation. (In the same sense, PmsPrm corresponds to $\mathsf{Sets}^{\mathbb{I}}$.)

If for the sake of argument we set aside the issues of finiteness and preserving pullbacks of monos, then this paper can be summed up as follows: PNL, and thus nominal terms, can be given a semantics in something that looks like $\mathsf{Sets}^{\mathbb{F}}$. This semantics is functional in that atoms-abstractions in $\mathsf{Sets}^{\mathbb{F}}$ can be naturally identified with total functions, though not all of them, which is good. HOL can also be given a semantics in something that looks like $\mathsf{Sets}^{\mathbb{F}}$, and in such a way that it overlaps with the semantics of PNL, as described in Definitions 7.6 and 8.9. We describe and exploit that overlap, in this paper.

19 If we translated PNL to first-order logic then we would probably map atoms to numbers instead. This is future work.

PmsRen from Definition 5.13 is related to the category of (finitely-supported) nominal renaming sets from [29]. Here, the difference that $x \in |\mathsf{X}^{\ni}|$ need not have finite support is significant because it is impossible with a finite renaming to rename $supp(x)$ to be entirely disjoint for some other permission set S. The definitions and proofs in Section 5.2 are delicately revised with respect to those in [29, Section 3]. Thus this paper contributes to the use of non-finitely-supported objects in nominal techniques, building on [29] and also on Cheney's and the second author's considerations of infinitely supported permutation sets [5,21].

A similar construction as in Section 5.4 has been considered, also in the context of names, though tersely, in Fiore and Turi's paper on the semantics of name and value passing [19]. The reader can compare for example the final two paragraphs of Section 1.3 in [19] with Definition 5.39 from Section 5.4. Fiore and Turi want substitutions to model bisimulation in the presence of name-generation and message-passing; we want renamings to model function application on names. The underlying technical demands overlap and are similar.

Fiore and Turi's framework includes the possibility of arbitrary substitutions for atoms (not just what we call renamings: substitution of atoms for atoms). This was apparent in [19] and is developed greatly in subsequent work by Fiore and Hur [18]. We hypothesise that from the point of view of PNL, their logic and semantics correspond to PNL enriched with substitution actions like those in [9,32], but this remains to be checked.[20]

Levy and Villaret translated nominal unification problems to higher-order unification problems [43]. A similar but more detailed analysis, translating solutions and introducing the same notion of capturable atoms as used in the capture typings in this paper, appears in the paper which introduced permissive nominal terms [13]. See also a journal version of Levy and Villaret's paper [44], which expanded on their previous work by eliminating freshness contexts (in a similar spirit to PNL, we feel, though the details are different). This paper can be viewed as a very considerable extension, refinement, and generalisation of these works: this paper is their grandchild, so to speak, via two other papers [9,38].

The extension of nominal sets to nominal renaming sets is free. This is touched on in Lemma 7.3 when we note that $[a{:=}b]\bullet(a, b)$ and $id\bullet(b, b)$ are distinct elements in $ren(\mathbb{A}_\nu \times \mathbb{A}_\nu)$ in PmsRen; this happens because the free construction 'suspends the non-injectivity' of $[a{:=}b]$ on (a, b). This is as things should be, in order to obtain completeness. The second author has considered a more radical non-free construction [22], which has the effect of extending atoms-abstraction to a total function and in which $[a{:=}b]\cdot x$ really does identify a with b in x in a suitable sense.

As we have emphasised, we translate a fragment of PNL to HOL. In [9] we considered full PNL with *equivariance*, which corresponds to strengthening the axiom rule $(\mathbf{Ax}^{\pi})$ in Fig. 2 from $\overline{\Phi,\ \phi \overset{\pi}{\vDash} \phi,\ \Psi}$ to $\overline{\Phi,\ \phi \vdash \pi\cdot\phi,\ \Psi}$ as illustrated in Fig. 1. This internalises the equivariance assumed in Definition 6.2 and allows us to derive e.g. $\mathsf{P}(a) \vdash \mathsf{P}(b)$.

In the journal version [10] of [9] we strengthen PNL further by allowing a *shift*-permutation. This is a non-finitely-supported bijection on $\mathbb{A}$ similar to a *de Bruijn shift function* $\uparrow$ [1, Subsection 2.2]. Its effect in this paper is to make all permission sets isomorphic up to bijection (e.g. $\mathbb{A}^< \cup \{a\} = \pi\cdot\mathbb{A}^<$ for some π, where $a \notin \mathbb{A}^<$) and this deals with a subtle restriction in the power of universal quantification discussed for instance in [9, Example 2.29]. Briefly, *shift* lets us derive $\forall X.\mathsf{P}(X) \vdash \mathsf{P}(Z)$ where $pms(X) = \mathbb{A}^<$ and $pms(Z) = \mathbb{A}^< \cup \{a\}$ where $a \notin \mathbb{A}^<$, which was not possible in the PNL from [9].

Neither equivariance nor *shift* are translated to HOL in this paper; more on this in the next subsection.

9.3. Future work

We have translated Permissive-Nominal Logic to Higher-Order Logic. The translation is not surjective: all variables are at most second-order; all constants are at most third-order; higher types are not used; and in fact all terms in the image of the translation are *λ-patterns* [45]. In addition, the translation is not total: we have dropped equivariance.

This is with good reason. We have not been able to simulate equivariance in HOL—not without 'cheating' by simply adding it (and causing a blowup in the size of propositions). We have not proved this impossible, but we hypothesise that it cannot be done. We further hypothesise (based on preliminary calculations not included in this paper) that HOL augmented with the ∇-quantifier from [49] would allow us to express equivariance.

It is not currently clear how to extend HOL with a *shift*-like permutation as discussed in [10,27]. This seems reasonable since *shift* would correspond to an infinite renaming.

Some natural theories in PNL might correspond to other fragments of HOL. Notably, it is not known what relation exists between HOL and PNL with the theory of atoms-substitution from [35,10].

Acknowledgements

Thanks to the careful attention of two anonymous referees. We acknowledge the support of DIGITEO in Paris, of grant RYC-2006-002131 at the Polytechnic University of Madrid, and of the Leverhulme Trust in the UK.

[20] Conversely, Fiore and Hur would view PNL as a restriction of their logic *without* substitution. The two points of view are consistent with each other, of course, and it is interesting that different authors are converging on similar systems. It might be worth mentioning that *deduction modulo* by the first author with Hardin and Kirchner was designed to mediate between these kinds of design decisions while retaining proof-theory [15].

Appendix. Soundness and completeness of restricted PNL with respect to non-equivariant models

A.1. Validity and soundness

Definition A.1 (*Validity*)**.** Suppose $\mathcal{I}$ is a non-equivariant interpretation of a signature $\mathcal{S}$ (Definition 6.2). Call the proposition ϕ **valid** in $\mathcal{I}$ when $[\![\phi]\!]^{\mathcal{I}}_{\varsigma} = 1$ for all ς.

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

If this is true for all non-equivariant $\mathcal{I}$ then write $\phi_1, \dots, \phi_n \models^{\pi} \psi_1, \dots, \psi_p$. If this is true for all equivariant $\mathcal{I}$ then write $\phi_1, \dots, \phi_n \models \psi_1, \dots, \psi_p$.

Theorem A.2 (*Soundness*)**.** *1. If $\Phi \vdash^{\pi} \Psi$ is derivable then $\Phi \models^{\pi} \Psi$.*
2. If $\Phi \vdash \Psi$ is derivable then $\Phi \models \Psi$.

Proof. Fix some interpretation $\mathcal{I}$. We work by induction on derivations. The case of ($\forall$**L**) uses Lemma 6.10. The case of ($\forall$**R**) uses Lemma 6.11. Other rules are routine by unpacking definitions.

If the interpretation $\mathcal{I}$ is fully equivariant then it can further be proved that $[\![\phi]\!]^{\mathcal{I}}_{\varsigma} = [\![\pi \cdot \phi]\!]^{\mathcal{I}}_{\varsigma}$ always, so that (**Ax**) is valid. If $\mathcal{I}$ is not fully equivariant, then just (**Ax**$^{\pi}$) is valid. □

Theorem A.3. (**Cut**) *is admissible in both full and restricted PNL.*

Proof. The proof for full PNL is in [10, Section 7] or [27, Subsection 11.2]; the derivation rules are almost exactly those of first-order logic, and so is the proof of cut-elimination. The argument for restricted PNL is identical; we note that none of the cut-eliminating transformations add π to axiom rules unless they are already there, so the same reductions on derivations work also for the restricted system. □

A.2. Completeness

In [10,27] we prove completeness of full PNL with respect to equivariant models, by means of a Herbrand construction (a model built out of syntax). We can leverage this result to concisely prove completeness of restricted PNL with respect to non-equivariant models, without having to repeat the model constructions.

For this subsection, fix the following data:

- A signature $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{P}, ar, \mathcal{X})$.
- A formula ϕ such that $\not\vdash^{\pi} \phi$.

Definition A.4. Define a new signature $\mathcal{S}^{\pi}$ as follows:

- $\mathcal{A}^{\pi} = \mathcal{A}$ and $\mathcal{B}^{\pi} = \mathcal{B} \cup \{\tau^{\pi}\}$ (so we have the same atom sorts and the same base sorts, plus one extra base sort τ^{π}).
- $\mathcal{F}^{\pi} = \mathcal{F}$ and $\mathcal{P}^{\pi} = \mathcal{P}$ (so we have the same term- and proposition-formers).
- If $\mathsf{f} \in \mathcal{F}$ then $ar^{\pi}(\mathsf{f}) = ar(\mathsf{f})$ (the term-formers are identical).
- If $\mathsf{P} \in \mathcal{P}$ and $ar(\mathsf{P}) = \alpha$ then $ar^{\pi}(\mathsf{P}) = (\tau^{\pi}, \alpha)$ (so proposition-formers take one extra argument of sort τ^{π}).
- $\mathcal{X}^{\pi} = \mathcal{X} \cup \{Z^{\pi}_{i,S} \mid i \in \mathbb{N}, S \text{ a permission set}\}$ where $sort(Z^{\pi}_{i,S}) = \tau^{\pi}$ (so we add unknowns of sort τ^{π}).

Now fix some particular unknown Z^{π} with $sort(Z^{\pi}) = \tau^{\pi}$ and such that $fa(\phi) \subseteq pms(Z^{\pi})$.

Definition A.5. Define a translation -$^{\pi}$ from PNL propositions in the signature $\mathcal{S}$ to PNL propositions in the signature $\mathcal{S}^{\pi}$ by mapping $\mathsf{P}(r)$ to $\mathsf{P}(Z^{\pi}, r)$ and extending this in the natural way to all predicates.

Our proof depends on the following technical lemma about restricted PNL:

Lemma A.6. *If $\Phi \vdash \Psi$ is derivable in full PNL then there exists a derivation $\mathfrak{B}$ such that every sequent $\Phi' \vdash \Psi'$ in $\mathfrak{B}$ satisfies $fa(\Phi') \cup fa(\Psi') \subseteq fa(\Phi) \cup fa(\Psi)$.*

Proof. By cut-elimination of restricted PNL (Theorem A.3) if a derivation of $\Phi \vdash \Psi$ exists then a cut-free derivation exists. We now examine the derivation rules in Fig. 2 and the definition of free atoms in Definition 2.15 and note that the rules ($\Rightarrow$**L**), ($\Rightarrow$**R**), ($\forall$**L**), and ($\forall$**R**) do not increase the free atoms moving from below the line to above the line.[21] □

Lemma A.7. *$\pi \cdot r = \pi' \cdot r$ if and only if $\pi(a) = \pi'(a)$ for every $a \in fa(r)$, and similarly for ϕ.*

See [27, Lemma 3.2.9] or [13, Lemma 4.15].

Proposition A.8. *If $\Phi^{\pi} \vdash \Psi^{\pi}$ in PNL and $fa(\Phi) \cup fa(\Psi) \subseteq pms(Z^{\pi})$ then $\Phi \models^{\pi} \Psi$.*

[21] ($\forall$**R**) and ($\forall$**L**) can increase the free *unknowns*—but not the free atoms.

Proof. Using cut-elimination of full PNL (Theorem A.3) assume a cut-free PNL derivation $\mathfrak{B}$ of $\Phi^{\pi} \vdash \Psi^{\pi}$. Because of Lemma A.6, the condition on free atoms holds of every sequent in $\mathfrak{B}$. Because of the form of the derivation rules in Fig. 1, $\mathfrak{B}$ cannot instantiate Z^{π}.

So we can go through the entire syntax of $\mathfrak{B}$ and delete Z^{π} to obtain a structure that is a candidate for being a derivation in restricted PNL of $\Phi \overset{\overline{\pi}}{\vdash} \Psi$.

The only non-trivial thing to check is that valid instances of (**Ax**) are transformed to valid instances of (**Ax**$^{\pi}$). Suppose we deduce $\Phi^{\pi}, \psi^{\pi} \vdash \pi'\cdot\psi^{\pi}, \Psi^{\pi}$ using (**Ax**). By assumption $\pi'\cdot\psi^{\pi} = \psi'^{\pi}$ for some ψ'. It follows that $\pi'\cdot Z^{\pi} = id\cdot Z^{\pi}$ (recall from Section 2.3 that we quotient by α-equivalence) and so by Lemma A.7 that $\pi'(a) = a$ for all $a \in pms(Z^{\pi})$. By assumption $fa(\Phi) \cup fa(\Psi) \cup fa(\psi) \cup fa(\psi') \subseteq pms(Z^{\pi})$ and so by Lemma A.7 $\psi = \psi'$, and we are done. □

Theorem A.9. *If $\Phi \overset{\overline{\pi}}{\models} \Psi$ then $\Phi \overset{\overline{\pi}}{\vdash} \Psi$.*

Proof. We prove the contrapositive, that if $\Phi \overset{\overline{\pi}}{\nvdash} \Psi$ then $\Phi \overset{\overline{\pi}}{\nvDash} \Psi$. Suppose $\Phi \overset{\overline{\pi}}{\nvdash} \Psi$. Using the constructions above we augment to a signature $\mathcal{S}^{\pi}$ (Definition A.4) with some Z^{π} with $fa(\Phi) \cup fa(\Psi) \subseteq pms(Z^{\pi})$. Thus by Proposition A.8 $\Phi^{\pi} \nvdash \Psi^{\pi}$.

By completeness of full PNL with respect to equivariant models ([10, Theorem 3.45], [27, Theorem 9.4.15]) we have that $\Phi^{\pi} \nvDash \Psi^{\pi}$. So there exists an equivariant model $\mathcal{I}$ and valuation ς to $\mathcal{I}$ such that $[\![\Phi]\!]^{\mathcal{I}}_{\varsigma} = 1$ and $[\![\Psi]\!]^{\mathcal{I}}_{\varsigma} = 0$. It is now routine to convert $\mathcal{I}$ into a non-equivariant model of the original signature $\mathcal{S}$ by taking $\mathsf{P}^{\mathcal{H}}(x) = \mathsf{P}^{\mathcal{I}}(\varsigma(Z^{\pi}), x)$. □

References

[1] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, Jean-Jacques Lévy, Explicit substitutions, Journal of Functional Programming 1 (4) (1991) 375–416.
[2] Peter B. Andrews, An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof, Academic Press, 1986.
[3] Christoph Benzmüller, Chad E. Brown, Michael Kohlhase, Higher-order semantics and extensionality, Journal of Symbolic Logic 69 (2004) 1027–1088.
[4] Mikolaj Bojanczyk, Laurent Braud, Bartek Klin, Slawomir Lasota, Towards nominal computation, in: Proceedings of the 39th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, POPL 2012, ACM Press, January 2012, pp. 401–412.
[5] James Cheney, Completeness and Herbrand theorems for nominal logic, Journal of Symbolic Logic 71 (2006) 299–320.
[6] Alonzo Church, A formulation of the simple theory of types, Journal of Symbolic Logic (1940) 56–68.
[7] James Cheney, Christian Urban, Alpha-prolog: a logic programming language with names, binding and alpha-equivalence, in: Bart Demoen, Vladimir Lifschitz (Eds.), Proceedings of the 20th International Conference on Logic Programming, ICLP 2004, in: Lecture Notes in Computer Science, vol. 3132, Springer, 2004, pp. 269–283.
[8] Nicolaas G. de Bruijn, Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem, Indagationes Mathematicae 5 (34) (1972) 381–392.
[9] Gilles Dowek, Murdoch J. Gabbay, Permissive nominal logic, in: Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP 2010, ACM Press, 2010, pp. 165–176.
[10] Gilles Dowek, Murdoch J. Gabbay, Permissive nominal logic (journal version), Transactions on Computational Logic 13 (3) (2012).
[11] Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, Joachim Posegga (Eds.), Handbook of Tableau Methods, Kluwer, 1999.
[12] Gilles Dowek, Murdoch J. Gabbay, Dominic P. Mulligan, Permissive nominal terms and their unification, in: Proceedings of the 24th Italian Conference on Computational Logic, CILC'09, 2009.
[13] Gilles Dowek, Murdoch J. Gabbay, 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) (2010) 769–822.
[14] Joëlle Despeyroux, André Hirschowitz, Higher-order abstract syntax with induction in COQ, in: LPAR'94, in: Lecture Notes in Computer Science, vol. 822, Springer, 1994, pp. 159–173.
[15] Gilles Dowek, Thérèse Hardin, Claude Kirchner, Theorem proving modulo, Rapport de Recherche 3400, Institut National de Recherche en Informatique et en Automatique, April 1998.
[16] Joëlle Despeyroux, Frank Pfenning, Carsten Schürmann, Primitive recursion for higher-order abstract syntax, Theoretical Computer Science 266 (1–2) (2001) 1–57.
[17] William M. Farmer, The seven virtues of simple type theory, Journal of Applied Logic 3 (6) (2008) 267–286.
[18] Fiore Marcelo, Chung-Kil Hur, Second-order equational logic, in: Proceedings of the 19th EACSL Annual Conference on Computer Science Logic, CSL 2010, in: Lecture Notes in Computer Science, Springer, 2010.
[19] Marcelo Fiore, Daniele Turi, Semantics of name and value passing, in: Proceedings of the 16th IEEE Symposium on Logic in Computer Science, LICS 2001, IEEE Computer Society Press, 2001, pp. 93–104.
[20] Murdoch J. Gabbay, Fresh logic, Journal of Applied Logic 5 (2) (2007) 356–387.
[21] Murdoch J. Gabbay, A general mathematics of names, Information and Computation 205 (7) (2007) 982–1011.
[22] Murdoch J. Gabbay, A study of substitution, using nominal techniques and Fraenkel–Mostowski sets, Theoretical Computer Science 410 (12–13) (2009) 1159–1189.
[23] Murdoch J. Gabbay, Foundations of nominal techniques: logic and semantics of variables in abstract syntax, Bulletin of Symbolic Logic 17 (2) (2011) 161–229.
[24] Murdoch J. Gabbay, Stone duality for first-order logic: a nominal approach, in: Howard Barringer Festschrift, December 2011.
[25] 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 (2011) 997–1033.
[26] Murdoch J. Gabbay, Meta-variables as infinite lists in nominal terms unification and rewriting, Logic Journal of the IGPL (2012).
[27] Murdoch J. Gabbay, Nominal terms and nominal logics: from foundations to meta-mathematics, in: Handbook of Philosophical Logic, vol. 17, Kluwer, 2012.
[28] Gerhard Gentzen, Untersuchungen über das logische Schließen [Investigations into logical deduction], Mathematische Zeitschrift 39 (1935), 176–210, 405–431. Translated in [54], pages 68–131.
[29] Murdoch J. Gabbay, Martin Hofmann, Nominal renaming sets, in: Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2008, Springer, November 2008, pp. 158–173.
[30] Murdoch J. Gabbay, Stéphane Lengrand, The lambda-context calculus, Electronic Notes in Theoretical Computer Science 196 (January) (2008) 19–35.
[31] Murdoch J. Gabbay, Stéphane Lengrand, The lambda-context calculus (extended version), Information and computation 207 (December) (2009) 1369–1400.
[32] Murdoch J. Gabbay, Aad Mathijssen, Capture-avoiding Substitution as a Nominal Algebra, in: ICTAC 2006: Theoretical Aspects of Computing, in: Lecture Notes in Computer Science, vol. 4281, November 2006, pp. 198–212.
[33] Murdoch J. Gabbay, 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, ACM, July 2006, pp. 189–200.

[34] Murdoch J. Gabbay, Aad Mathijssen, A formal calculus for informal equality with binding, in: WoLLIC'07: 14th Workshop on Logic, Language, Information and Computation, in: Lecture Notes in Computer Science, vol. 4576, Springer, July 2007, pp. 162–176.

[35] Murdoch J. Gabbay, Aad Mathijssen, Capture-avoiding substitution as a nominal algebra, Formal Aspects of Computing 20 (4–5) (2008) 451–479.

[36] Murdoch J. Gabbay, Aad Mathijssen, One-and-a-halfth-order Logic, Journal of Logic and Computation 18 (4) (2008) 521–562.

[37] Murdoch J. Gabbay, Aad Mathijssen, Nominal universal algebra: equational logic with names and binding, Journal of Logic and Computation 19 (6) (2009) 1455–1508.

[38] Murdoch J. Gabbay, 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, ACM, August 2009, pp. 64–73.

[39] Murdoch J. Gabbay, Aad Mathijssen, A nominal axiomatisation of the lambda-calculus, Journal of Logic and Computation 20 (2) (2010) 501–531.

[40] Fabio Gadducci, Marino Miculan, Ugo Montanari, About permutation algebras, (pre)sheaves and named sets, Higher Order Symbolic Computation 19 (2–3) (2006) 283–304.

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

[42] J. Roger Hindley, Jonathan P. Seldin, Lambda-Calculus and Combinators, An Introduction, 2nd ed., Cambridge University Press, 2008.

[43] Jordi Levy, Mateu Villaret, Nominal unification from a higher-order perspective, in: Rewriting Techniques and Applications, Proceedings of RTA 2008, in: Lecture Notes in Computer Science, vol. 5117, Springer, 2008.

[44] Jordi Levy, Mateu Villaret, Nominal unification from a higher-order perspective, Transactions on Computational Logic (TOCL) 13 (2011).

[45] Dale Miller, A logic programming language with lambda-abstraction, function variables, and simple unification, Journal of Logic and Computation 1 (4) (1991) 497–536.

[46] Dale Miller, Logic, higher order, in: Stuart Shapiro (Ed.), Encyclopedia of Artificial Intelligence, 2nd ed., Wiley, 1992, Available online from the author's webpage.

[47] Dale Miller, Unification under a mixed prefix, Journal of Symbolic Computation 14 (4) (1992) 321–358.

[48] Richard Mayr, Tobias Nipkow, Higher-order rewrite systems and their confluence, Theoretical Computer Science 192 (1998) 3–29.

[49] Dale Miller, Alwen Tiu, A proof theory for generic judgments (extended abstract), in: Proceedings of the 18th IEEE Symposium on Logic in Computer Science, LICS 2003, IEEE Computer Society Press, 2003, pp. 118–127.

[50] Frank Pfenning, Conal Elliott, Higher-order abstract syntax, in: PLDI (Programming Language Design and Implementation), ACM Press, 1988, pp. 199–208.

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

[52] Dag Prawitz, Natural Deduction: A Prooof-Theoretical Study, Almquist and Wiksell, 1965, Reprinted by Dover, 2006.

[53] Raymond Smullyan, First-Order Logic, Springer, 1968, Reprinted by Dover, 1995.

[54] M.E. Szabo (Ed.), Collected Papers of Gerhard Gentzen, North Holland, 1969.

[55] Christian Urban, Andrew M. Pitts, Murdoch J. Gabbay, Nominal unification, Theoretical Computer Science 323 (1–3) (2004) 473–497.