A

Permissive-Nominal Logic (journal version)

Gilles Dowek, INRIA, France. http://www-roc.inria.fr/who/Gilles.Dowek
Murdoch J. Gabbay, Heriot-Watt University, UK. http://www.gabbay.org.uk

Permissive-Nominal Logic (PNL) is an extension of first-order predicate logic in which term-formers can bind names in their arguments.

This allows for direct axiomatisations with binders, such as of the λ-binder of the lambda-calculus or the $\forall$-binder of first-order logic. It also allows us to finitely axiomatise arithmetic, and similarly to axiomatise 'nominal' datatypes-with-binding.

Just like first- and higher-order logic, equality reasoning is not necessary to α-rename.

This gives PNL much of the expressive power of higher-order logic, but models and derivations of PNL are first-order in character, and the logic seems to strike a good balance between expressivity and simplicity.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Set theory*; *Lambda-calculus and related systems*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Metatheory*

General Terms: Theory, Languages

Additional Key Words and Phrases: Permissive-nominal logic, nominal sets, names and binding, permutation

ACM Reference Format:
Dowek, G., Gabbay, M. J., 2010. Permissive-nominal logic (journal version) ACM Trans. Comput. Logic V, N, Article A (January YYYY), 36 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Contents

1 Introduction **2**

2 Permissive-Nominal Logic **4**
2.1 Syntax 4
2.2 Permutation actions and free atoms/unknowns 6
2.3 Free level 1 and level 2 variables 7
2.4 α-equivalence 8
2.5 Substitution 9
2.6 Sequents and derivability 10
2.7 Universal quantification, permission sorts, and *shift*-permutations 11

3 Semantics of permissive-nominal logic **12**
3.1 Permissive-nominal sets 12
3.2 Examples of permissive-nominal sets 13
3.2.1 Atoms 13
3.2.2 Atoms-abstraction 14
3.2.3 Product 14

Thanks to the anonymous referees. Supported by grant RYC-2006-002131 at the Polytechnic University of Madrid.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© YYYY ACM 1529-3785/YYYY/01-ARTA $10.00
DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

3.3 Interpretation and soundness . . . 14
3.3.1 Interpretation of signatures . . . 14
3.3.2 Interpretation of terms . . . 15
3.3.3 Interpretation of propositions . . . 15
3.3.4 Validity and soundness . . . 16
3.4 Completeness . . . 17
3.4.1 Maximally consistent set of propositions . . . 17
3.4.2 The term model . . . 18

4 Specifying arithmetic in permissive-nominal logic 19
4.1 The signature $\dot{\mathcal{L}}$ and the axioms . . . 19
4.2 Comments on the axioms . . . 21

5 A theory of arithmetic in first-order logic 22
5.1 First-order logic $\mathcal{L}$ . . . 22
5.2 Interpretation of first-order logic . . . 23
5.3 A theory of arithmetic in $\mathcal{L}$ . . . 24
5.4 Building an interpretation for $\dot{\mathcal{L}}$ out of one for $\mathcal{L}$ . . . 24

6 More PNL theories 26
6.1 Inductive types . . . 26
6.2 The N quantifier . . . 27
6.3 Freshness $a\#x$ and abstraction . . . 27

7 Cut-elimination 28

8 Related work 29
8.1 Other 'nominal' syntaxes and logics . . . 29
8.2 From nominal terms-in-freshness-context to PNL terms-with-permission-sets 30
8.3 Non-nominal logics . . . 31

9 Further remarks, further work 32

1. INTRODUCTION

In the early 20th century, the expressivity of logics was considered *in principle*. For example, first-order predicate logics with or without term-formers are equally expressive, in principle.

In the early 21st century, more attention is paid to what we like to call 'ergonomics'. First-order predicate logic with term-formers is more ergonomic than first-order predicate logic without term-formers; terms, propositions and proofs are shorter and more natural in the former than in the latter.

Another imperative is for a *weak* logic—the fewer 'bells and whistles' we have to worry about, the easier it will be to verify, implement, and prove its meta-theoretic properties. This can be in tension with being ergonomic, as the example of first-order predicate logic with or without term-formers illustrates.

Thus there enters a fruitful design tension: we aim for logics that are so ergonomic that they 'just work', yet so weak and well-behaved that we can still prove good properties for them.

Now we come to the issue of *binding*. Binding is ubiquitous in logical specifications in mathematics—binding features in function definitions via λ-abstraction, and binders are also used to define sets in comprehension, and to define finite and infinite sums, integrals, derivations, quantifiers, and so on. A logic for mathematics that provides support for this central and essential notion, is likely to be more ergonomic than a logic that does not.

First-order logic is weak, computationally tractable, and has good theoretical properties (unification of first-order terms is decidable; proof-search is simple and well-understood; models are simple). However, first-order logic is unergonomic in the sense that it does not admit term-formers that can bind. Thus it is hard to give direct, finite, first-order axiomatisations of set theory, arithmetic, higher-order logic, or the λ-calculus.

This is one reason that e.g. higher-order logic is often used as a specification language in theory (see [Farmer 2008] for an excellent exposition) and implementations (like Isabelle [Paulson 1990])—higher-order logic has a binder (λ) built-in to terms, and so is more ergonomic.

However, higher-order logic is also stronger than first-order logic, less computationally tractable, and models tend to be more complex. The jump from first- to higher-order logic is quite large.

This motivates the study of direct extensions of first-order predicate logic with term-formers that can bind. The topic of this paper is the construction of one such extension, which we call *permissive-nominal logic* (**PNL**). PNL has a clear first-order flavour, and it admits term-formers that bind.

Technical summary and overview. The main technical contributions of this paper are: the definition of permissive-nominal logic, in particular how it handles freshness side-conditions and how this permits the addition of universal quantification to nominal terms; the Tarski-style models we then construct; and the finite yet fully first-order axiomatisations of substitution, first-order logic, and arithmetic which we then exhibit.

Soundness, completeness, and cut-elimination follow by fairly routine arguments, but we see this as a good sign: that the definition of permissive-nominal logic remains close to first-order logic, while allowing terms with binders.

An overview of this paper is as follows:

— We introduce the syntax and derivation rules of permissive-nominal logic (Section 2). The impatient reader can jump directly to Figure 1 on page 11 and see that these derivation rules are virtually indistinguishable from those of first-order logic; only an extra side-condition in ($\forall$**L**) hints at any difference.[1]
— We prove soundness and completeness (Theorems 3.30 and 3.45) with respect to a suitable notion of *permissive-nominal set* (Definition 3.4).
— We axiomatise arithmetic in PNL and prove a correctness and conservative extension result Theorem 5.21. The axiomatisation of arithmetic is *finite*; the induction schema normally given in first-order logic arithmetic is represented by a single axiom with a universal quantification $\forall X$ over a permissive-nominal terms unknown.
— We indicate how to axiomatise nominal inductive datatypes, the N-quantifier, and semantic freshness (Section 6).
— We prove cut-elimination (Theorem 7.7). The proof is virtually identical to that of first-order logic.

Permissive-nominal logic, nominal logic, and nominal terms. Permissive-nominal logic follows the *nominal logic* of [Pitts 2003] in its name, which coined the term 'nominal', but nominal logic from [Pitts 2003] is a first-order theory of (set of axioms for) the equivariant Fraenkel-Mostowski sets and associated constructions used in [Gabbay and Pitts 2001]. The syntax, semantics, and derivability of PNL are new, as indeed is the application to arithmetic.

The term-language of PNL goes back to the *nominal terms* of [Urban et al. 2004]. It is permissive, which means that the freshness contexts from [Urban et al. 2004] become a kind of sorting constraint called *permission sets*. For more discussion see [Dowek et al. 2010] which

[1] The language of terms is significantly different, though; see Definition 2.13.

introduced permissive-nominal terms, and amongst other things gave correspondences with nominal terms and also higher-order patterns.

In this paper we extend nominal syntax further by introducing *shift*-permutations (Definition 2.9). Also, unlike [Urban et al. 2004; Dowek et al. 2010] in PNL there is quantification over unknowns $\forall X$.

This journal paper follows a conference paper [Dowek and Gabbay 2010]. With respect to that paper, we have made the following changes:

— The treatment of α-equivalence has been streamlined, leading to simplified proofs. Two structural rules $(\alpha_{\mathbf{L}})$ and $(\alpha_{\mathbf{R}})$ have been eliminated.
— The rule $(\mathsf{И})$ from [Dowek and Gabbay 2010] is now part of the axiom rule, further simplifying the proof-theory.
— The notion of permutation includes *shift*-permutations; these permutations 'shift all atoms up by one'. Some readers will see in this a de Bruijn-like 'shift' function [Abadi et al. 1991]. This gives desirable extra power to $\forall$-quantification and, perhaps surprisingly, turns out to be compatible with nominal techniques' characteristic *small support* property.
— We include proofs of completeness by a standard term-model construction, and a sketch proof of cut-elimination.

2. PERMISSIVE-NOMINAL LOGIC

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

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

We admit the possibility of empty tuples, so that $()$ is a sort (the *unit sort*).

Example 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'.

Definition 2.3. A **term-signature** over a sort-signature $(\mathcal{A}, \mathcal{B})$ is a tuple $(\mathcal{F}, \mathcal{P}, ar)$ where:

— $\mathcal{F}$ and $\mathcal{P}$ are disjoint sets of **term-** and **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 $(\alpha_1, \ldots, \alpha_n)\tau$.

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

Notation 2.4. We write $\mathsf{f} : (\alpha)\tau$ for $ar(\mathsf{f}) = (\alpha)\tau$ and similarly we write $\mathsf{P} : \alpha$ for $ar(\mathsf{P}) = \alpha$.

Remark 2.5. The reader familiar with higher-order logic can read $ar(f) = (\alpha)\tau$ as $f : \alpha \to \tau$ and no harm will come of it. We do not do this because we are following a first-order logic notation—and because we want to avoid any possible confusion that $(\alpha \to \alpha) \to \alpha$ might be a sort. It is not.

Example 2.6. A signature for the λ-calculus would have one name-sort ν, one base sort ι, and term-formers $\mathsf{lam} : ([\nu]\iota)\iota$, $\mathsf{app} : (\iota, \iota)\iota$, and $\mathsf{var} : (\nu)\iota$.

A proposition-former for nominal freshness $\#$ would have arity (ν, ι), though the arity $([\nu]\iota)$ would also be possible (this would handle more of the properties of names at the level of the logic). More on this in Subsection 6.3.

Plenty more examples of PNL theories will follow.

Definition 2.7. For each name sort ν fix a disjoint countably infinite set of **atoms** $\mathbb{A}_\nu$ (*level 1 names*).

For each ν also fix a bijective function f_ν from $\mathbb{A}_\nu$ to the integers $\mathbb{Z} = \{0, \text{-}1, 1, \text{-}2, 2, \ldots\}$ (that we can do this follows from our assumption that atoms are countable).

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. 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 co-infinite (their complement with respect to $\mathbb{A}$ is also infinite).

Remark 2.8 (Representing permission sets)**.** A permission set S may be finitely represented

— either as the pair of finite sets (A, B) where $A \subseteq \mathbb{A}^<$ and $B \subseteq \mathbb{A}^>$ and $S = (\mathbb{A}^< \setminus A) \cup B$,
— or perhaps more elegantly as a single finite set $C \subseteq \mathbb{A}$ such that $S = \mathbb{A}^< \Delta C$ where $X \Delta Y = \{z \mid (z \in X \wedge z \notin Y) \vee (z \notin X \wedge z \in Y)\}$ (exclusive or).

Permission sets are a sorting/typing annotation which will be associated to variables in Definition 2.11.

Definition 2.9. Given $a, b \in \mathbb{A}_\nu$ let a **(level 1) swapping** $(a\ b)$ be the bijection on atoms that maps a to b, b to a, and all other c to themselves.
Also define a bijection shift_ν on atoms by:

$$\begin{array}{ll} \mathit{shift}_\nu(a) = f_\nu(f_\nu^{\text{-}1}(a) + 1) & (a \in \mathbb{A}_\nu) \\ \mathit{shift}_\nu(a) = a & a \in \mathbb{A} \setminus \mathbb{A}_\nu \end{array}$$

Let the **(level 1) permutations** be the group of bijections on atoms generated by all swappings and shift_ν.
Call a permutation π **finite** when it is generated just by swappings; thus, when $\mathit{nontriv}(\pi) = \{a{\in}\mathbb{A} \mid \pi(a){\neq}a\}$ is finite. Otherwise, call π **non-finite**.
π will range over permutations. Write $\mathbb{P}$ for the set of all permutations and write $\mathbb{P}_{\text{fin}}$ for the set of all finite permutations.

Remark 2.10. Swappings are used to manage α-equivalence in nominal terms. This is standard and goes back (at least) to [Gabbay and Pitts 2001] and the second author's thesis [Gabbay 2001].

The true importance of shift_ν is that it bijects $\mathbb{A}^<$ with $\mathbb{A}^< \cup \{a\}$ for some $a \notin \mathbb{A}^<$—this cannot be achieved using a *finite* permutation. The relevance of this is that later when we build $\forall X.\phi$, this really will mean 'for all X' even though the permission set S of X makes it range over terms with free atoms in S.

Permutations, like permission sets, easily admit finite representations. shift corresponds via the bijection with numbers to the operation 'add 1'.

Definition 2.11. For each signature $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{P}, ar)$ and each sort α over $(\mathcal{A}, \mathcal{B})$ and permission set S fix a countably infinite set of **unknowns** (*level 2 names*) of that sort and permission set. X, Y, Z will range over distinct unknowns. Write $\mathit{sort}(X)$ for the sort and $\mathit{pmss}(X)$ for the permission set of X.

Remark 2.12. So an unknown X has two type attributes: its *sort* α, which intuitively describes what kind of data it denotes, and its *permission set* S which describes the permitted free

atoms of the terms, and also the nominal support of the denotations, with which it may be associated by a substitution or valuation—see the definitions of substitution and valuation in Definitions 2.32 and 3.19 respectively.

If X has sort 'integers' and permission set $\mathbb{A}^{<}$, then intuitively X represents 'a term denoting an integer, with free atoms in $\mathbb{A}^{<}$'.

Another name for permission set S might be *freshness set*, since equally X represents "terms with free atoms *not* in $\mathbb{A} \setminus S$".[2]

Definition 2.13. For each signature $\mathcal{S}$, define **raw terms** and **raw propositions** over $\mathcal{S}$ by:

$$\frac{(a \in \mathbb{A}_\nu)}{a : \nu} \qquad \frac{\mathtt{r}_1 : \alpha_1 \ \ldots \ \mathtt{r}_n : \alpha_n}{(\mathtt{r}_1, \ldots, \mathtt{r}_n) : (\alpha_1, \ldots, \alpha_n)} \qquad \frac{\mathtt{r} : \alpha \quad (ar(\mathsf{f}) = (\alpha)\tau)}{\mathsf{f}(\mathtt{r}) : \tau}$$

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

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

$$\frac{\texttt{phi prop.}}{\forall X.\texttt{phi prop.}}$$

As in Definition 2.1, we admit the possiblity of empty tuples so that $()$ the empty tuple of terms is a term and has sort $()$.

We will quotient raw terms and propositions by α-equivalence (to obtain terms r and propositions ϕ), later.

Example 2.14. Consider $\mathsf{lam}([b]\mathsf{app}(X, \mathsf{var}(b)))$ where $b \notin pmss(X)$; this represents the λ-term schema $\lambda y.(ty)$ where $y \notin fv(t)$.

Recall that app and lam are term-formers of arities $(\iota, \iota)\iota$ and $([\nu]\iota)\iota$. The sorts of b and X are ν (names) and ι (individuals) respectively.

Remark 2.15. Our version of PNL has connectives $\bot$, $\Rightarrow$, and $\forall$. We could easily add other connectives like $\top$, $\neg$, $\wedge$, $\vee$, and $\exists$. Instead we treat them as a definable extension using the standard 'de Morgan' encoding.

We may write $id{\cdot}X$ just as X.

2.2. Permutation actions and free atoms/unknowns

Nominal techniques suggest handling α-renaming using permutations. To a first approximation, if wherever the reader sees 'permutation action' they substitute 'α-renaming', then they will not go too far wrong.

Notation 2.16. 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.

[2] Via this intuition, permission sets correspond to the *freshness constraints* $a\#X$ of [Urban et al. 2004]. For the reader familiar with freshness constraints, another way to view permission sets is as fixing a single global freshness context with 'enough' freshnesses (the germ of this was already in [Gabbay 2005]) where 'enough' means that for any term, we can always pick an atom not free in that term. However the implications of doing this go beyond a syntactic tweak to nominal terms; permission sets are what make it possible for us to reconcile level 2 quantification $\forall X$ with level 1 atoms-abstraction $[a]r$.

— Write π^{-1} for **inverse**, so $\pi \circ \pi^{-1} = id$.
— Define π^n by $\pi^0 = id$ and $\pi^{n+1} = \pi^n \circ \pi$.

Definition 2.17. Define a (level 1) **permutation action** on syntax by:

$$\begin{array}{rlrl} \pi{\cdot}a &= \pi(a) & \pi{\cdot}(\mathtt{r}_1,\ldots,\mathtt{r}_n) &= (\pi{\cdot}\mathtt{r}_1,\ldots,\pi{\cdot}\mathtt{r}_n) \\ \pi{\cdot}[a]\mathtt{r} &= [\pi(a)]\pi{\cdot}\mathtt{r} & \pi{\cdot}(\pi'{\cdot}X) &= (\pi{\circ}\pi'){\cdot}X \\ \pi{\cdot}\mathsf{f}(\mathtt{r}) &= \mathsf{f}(\pi{\cdot}\mathtt{r}) & & \\ \pi{\cdot}\bot &= \bot & \pi{\cdot}(\mathtt{phi} \Rightarrow \mathtt{psi}) &= (\pi{\cdot}\mathtt{phi}) \Rightarrow (\pi{\cdot}\mathtt{psi}) \\ \pi{\cdot}\mathsf{P}(\mathtt{r}) &= \mathsf{P}(\pi{\cdot}\mathtt{r}) & \pi{\cdot}(\forall X.\mathtt{phi}) &= \forall X.\pi{\cdot}\mathtt{phi} \end{array}$$

Definition 2.18. Let Π range over sort- and permission-set-preserving bijections on unknowns (so $sort(\Pi(X))$=$sort(X)$ and $pmss(\Pi(X))$=$pmss(X)$) 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 2.16.

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

$$\begin{array}{rlrl} \Pi{\cdot}a &= a & \Pi{\cdot}(\mathtt{r}_1,\ldots,\mathtt{r}_n) &= (\Pi{\cdot}\mathtt{r}_1,\ldots,\Pi{\cdot}\mathtt{r}_n) \\ \Pi{\cdot}[a]\mathtt{r} &= [a]\Pi{\cdot}\mathtt{r} & \Pi{\cdot}(\pi{\cdot}X) &= \pi{\cdot}(\Pi(X)) \\ \Pi{\cdot}\mathsf{f}(\mathtt{r}) &= \mathsf{f}(\Pi{\cdot}\mathtt{r}) & & \\ \Pi{\cdot}\bot &= \bot & \Pi{\cdot}(\mathtt{phi} \Rightarrow \mathtt{psi}) &= (\Pi{\cdot}\mathtt{phi}) \Rightarrow (\Pi{\cdot}\mathtt{psi}) \\ \Pi{\cdot}\mathsf{P}(\mathtt{r}) &= \mathsf{P}(\Pi{\cdot}\mathtt{r}) & \Pi{\cdot}(\forall X.\mathtt{phi}) &= \forall\Pi(X).\Pi{\cdot}\mathtt{phi} \end{array}$$

2.3. Free level 1 and level 2 variables

Definition 2.19. Suppose A is a set of atoms and π is a level 1 permutation. Suppose U is a set of unknowns and Π is a level 2 permutation. Define $\pi{\cdot}A$ and $\Pi{\cdot}U$ by

$$\pi{\cdot}A = \{\pi(a) \mid a \in A\} \qquad \text{and} \qquad \Pi{\cdot}U = \{\Pi(X) \mid X \in U\}.$$

This is the standard **pointwise** permutation action on sets.

Definition 2.20. Define **free atoms** $fa(\mathtt{r})$ and $fa(\mathtt{phi})$ by:

$$\begin{array}{rlrl} fa(\pi{\cdot}X) &= \pi{\cdot}pmss(X) & fa([a]\mathtt{r}) &= fa(\mathtt{r}) \setminus \{a\} \\ fa(\mathsf{f}(\mathtt{r})) &= fa(\mathtt{r}) & fa((\mathtt{r}_1,\ldots,\mathtt{r}_n)) &= \bigcup fa(\mathtt{r}_i) \\ fa(a) &= \{a\} & & \\ fa(\bot) &= \varnothing & fa(\mathtt{phi} \Rightarrow \mathtt{psi}) &= fa(\mathtt{phi}) \cup fa(\mathtt{psi}) \\ fa(\mathsf{P}(\mathtt{r})) &= fa(\mathtt{r}) & fa(\forall X.\mathtt{phi}) &= fa(\mathtt{phi}) \end{array}$$

Define **free unknowns** $fV(r)$ and $fV(\mathtt{phi})$ by:

$$\begin{array}{rlrl} fV(a) &= \varnothing & fV(\pi{\cdot}X) &= \{X\} \\ fV([a]\mathtt{r}) &= fV(\mathtt{r}) & fV((\mathtt{r}_1,\ldots,\mathtt{r}_n)) &= \bigcup fV(\mathtt{r}_i) \\ fV(\mathsf{f}(\mathtt{r})) &= fV(\mathtt{r}) & & \\ fV(\bot) &= \varnothing & fV(\mathtt{phi} \Rightarrow \mathtt{psi}) &= fV(\mathtt{phi}) \cup fV(\mathtt{psi}) \\ fV(\mathsf{P}(\mathtt{r})) &= fV(\mathtt{r}) & fV(\forall X.\mathtt{phi}) &= fV(\mathtt{phi}) \setminus \{X\} \end{array}$$

Lemma 2.21. *$fa(\pi{\cdot}\mathtt{r}) = \pi{\cdot}fa(\mathtt{r})$ and $fa(\pi{\cdot}\mathtt{phi}) = \pi{\cdot}fa(\mathtt{phi})$.*
Also, $fV(\Pi{\cdot}\mathtt{r}) = \Pi{\cdot}fV(\mathtt{r})$ and $fV(\Pi{\cdot}\mathtt{phi}) = \Pi{\cdot}fV(\mathtt{phi})$.

Proof. By routine inductions on $\mathtt{r}$. □

2.4. α-equivalence

Definition 2.22. Call an equivalence relation $\mathcal{R}$ on terms and on propositions a **congruence** when it is closed under the following rules:

$$\frac{\mathtt{r}_i\ \mathcal{R}\ \mathtt{s}_i \quad 1 \leq i \leq n}{(\mathtt{r}_1,\ldots,\mathtt{r}_n)\ \mathcal{R}\ (\mathtt{s}_1,\ldots,\mathtt{s}_n)} \qquad \frac{\mathtt{r}\ \mathcal{R}\ \mathtt{s} \quad (\mathtt{f}:(\alpha)\tau,\ \mathtt{r},\mathtt{s}:\alpha)}{\mathtt{f}(\mathtt{r})\ \mathcal{R}\ \mathtt{f}(\mathtt{s})}$$

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

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

Definition 2.23. Write $(a\ b)$ for the **(level 1) swapping** permutation which maps a to b, b to a, and all other c to themselves. Similarly write $(X\ Y)$ for the **(level 2) swapping**.

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

$$\frac{(b\ a)\cdot\mathtt{r} =_\alpha \mathtt{s} \quad (b \notin fa(\mathtt{r}))}{[a]\mathtt{r} =_\alpha [b]\mathtt{s}} \qquad \frac{(\pi(a) = \pi'(a) \text{ for all } a{\in}pmss(X))}{\pi\cdot X =_\alpha \pi'\cdot X}$$

$$\frac{(Y\ X)\cdot\mathtt{phi} =_\alpha \mathtt{psi} \quad (Y \notin fV(\mathtt{phi}))}{\forall X.\mathtt{phi} =_\alpha \forall Y.\mathtt{psi}}$$

Example 2.24. We α-convert X and a in $\forall X.\mathsf{P}([a]X)$.

Let $sort(Y) = sort(X)$ and $pmss(Y) = pmss(X) = \mathbb{A}^<$. Suppose $a \in \mathbb{A}^<$ and $b \notin \mathbb{A}^<$. 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.

Furthermore, if we take syntax as above except that $a \notin \mathbb{A}^<$ and $b \notin \mathbb{A}^<$, then we deduce $\forall X.\mathsf{P}([a]X) =_\alpha \forall Y.\mathsf{P}([b]Y)$.

Remark 2.25. Note that α-equivalence is highly symmetric between levels 1 and 2, based on permutations instead of substitutions, and avoids equality reasoning in the logic.

In [Gabbay and Pitts 2001; Pitts 2003; Gabbay 2007a; Gabbay and Cheney 2004; Cheney 2005] it is not in general possible to 'just α-convert' a level 1 abstraction. We must appeal instead to equality reasoning describing atoms-abstraction in nominal sets. But this is harder; derivable equality is more complex than syntactic equivalence.

Lemma 2.26. *For every* π, Π, $\mathtt{r}$, $\mathtt{s}$, $\mathtt{phi}$, *and* $\mathtt{psi}$, *the following hold:*

— $\mathtt{r} =_\alpha \mathtt{s}$ *if and only if* $\pi\cdot\mathtt{r} =_\alpha \pi\cdot\mathtt{s}$ *and similarly* $\mathtt{phi} =_\alpha \mathtt{psi}$ *if and only if* $\pi\cdot\mathtt{phi} =_\alpha \pi\cdot\mathtt{psi}$.
— $\mathtt{r} =_\alpha \mathtt{s}$ *if and only if* $\Pi\cdot\mathtt{r} =_\alpha \Pi\cdot\mathtt{s}$, *and similarly* $\mathtt{phi} =_\alpha \mathtt{psi}$ *if and only if* $\Pi\cdot\mathtt{phi} =_\alpha \Pi\cdot\mathtt{psi}$.

Lemma 2.27. *If* $\mathtt{r} =_\alpha \mathtt{s}$ *then* $fa(\mathtt{r}) = fa(\mathtt{s})$ *and* $fV(\mathtt{r}) = fV(\mathtt{s})$.

Proposition 2.28. $=_\alpha$ *is an equivalence relation on terms and propositions.*

Proof. By a standard argument as in [Fernández and Gabbay 2007], using Lemmas 2.21, 2.26, and 2.27. □

Lemma 2.29. *If* $\pi(a) = a$ *for every* $a \in fa(\mathtt{r})$ *then* $\pi\cdot\mathtt{r} =_\alpha \mathtt{r}$.

Definition 2.30. For each signature $\mathcal{S}$, define **terms** and **propositions** over $\mathcal{S}$ to be raw terms and propositions quotiented by α-equivalence.
r and s will range over terms. ϕ and ψ will range over propositions.

Remark 2.31. Terms and propositions inherit the definitions and properties of raw terms and propositions. Thus for example we may write $fa(r)$ to mean '$fa(\mathtt{r})$ for some $\mathtt{r} \in r$' (Lemma 2.27 proves this is well-defined).

It is possible to construct terms and propositions directly using a variant of nominal syntax-with-binding from [Gabbay and Pitts 2001], tweaked to include permutation and abstraction by unknowns.

It is also possible to retain the definitions above—reasoning on α-equivalence classes of terms—and to use theorems of *abstractive functions* developed in [Gabbay 2007b], which are an alternative 'nominal' way to guarantee well-definedness of functions defined on α-equivalence classes.

We will not dwell on these issues in this paper because we are most interested in what PNL syntax can express rather than thinking about the syntax for its own sake. However, the mathematics to do this exists and is well-understood.

2.5. Substitution

Definition 2.32. A (level 2) **substitution** θ is a function from unknowns to terms such that:

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

θ will range over substitutions.

One kind of substitution will be particularly useful later:

Definition 2.33. Suppose X is an unknown and suppose $t : sort(X)$ and $fa(t) \subseteq pmss(X)$. Define $[X::{=}t]$ by:

$$\begin{array}{ll} [X::{=}t](X) = t & \\ [X::{=}t](Y) = id{\cdot}Y & \text{all other } Y \end{array}$$

By convention (Definition 2.11) X and Y in Definition 2.34 range over *distinct* unknowns:

Definition 2.34. Define $nontriv(\theta)$ by:

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

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

Definition 2.35. Define a **substitution action** by:

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

Remark 2.36. Level 2 substitution $r\theta$ is capturing for level 1 abstraction $[a]$-. For example if $\theta(X) = a$ then $([a]X)\theta = [a]a$. This is the behaviour displayed by the informal meta-level when we write "take t to be x in $\lambda x.t$".

Only atoms in $pmss(X)$ may be captured in this way. Thus for instance, if $a \notin pmss(X)$ then $\theta(X) = a$ is impossible because it would violate the condition $fa(\theta(X)) \subseteq pmss(X)$ in Definition 2.32.

Remark 2.37. The condition $fa(\theta(X)) \subseteq pmss(X)$ in Definition 2.32, and the condition $fa(t) \subseteq pmss(X)$ in Definition 2.33, are necessary for the substitution action in Definition 2.35 to be well-defined.

Consider a name sort ν and suppose $X : \nu$ and $a, b : \nu$. Suppose $a, b \notin pmss(X)$, so that by Definition 2.23 $(a\ b)\cdot X = id\cdot X$.[3]

Suppose we drop the conditions on free atoms of terms, so that we admit $[X::=a]$ as a substitution. Then according to the definitions, $((a\ b)\cdot X)[X::=a] = b$ whereas $(id\cdot X)[X::=a] = a$.

Remark 2.38. In PNL, atoms are data; they are 'bindable constant symbols'. Atoms are not variables; they do not come with a substitution as a primitive in PNL. (Unknowns are variables; they have a substitution action.)

The reader should not expect atoms to populate every sort, like variables do. Atoms populate their own special sorts, name-sorts, which are sorts for 'bindable data'.

We can make atoms populate a base sort (e.g. variable symbols with sort 'λ-terms' or 'functions') with a term-former (Definition 2.3), e.g. var in Sections 4 and 6.

We can make atoms behave like variables using axioms, like those of SUB in Figure 4. In PNL, a substitution action for atoms is a matter of writing suitable axioms. Fortunately, nominal techniques make this fairly easy to do.

2.6. Sequents and derivability

Definition 2.39. Φ and Ψ will range over sets of propositions. We may write ϕ, Φ and Φ, ϕ as shorthand for $\{\phi\} \cup \Phi$. We may write Φ, Ψ as shorthand for $\Phi \cup \Psi$. Finally, define $fV(\phi)$ by

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

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

Definition 2.40 (Derivable sequents)**.** The **derivable sequents** are defined in Figure 1.

Remark 2.41. As standard, the intuition of $\Phi \vdash \Psi$ being derivable is "the conjunction (logical and) of the propositions in Φ entails the disjunction (logical or) of the propositions in Ψ". So for instance, intuitively the axiom rule $(\mathbf{Ax})$ expresses that ϕ if and only if $\pi\cdot\phi$.

The π in $(\mathbf{Ax})$ is deliberate and represents equivariance (preservation of truth under permuting atoms) within permissive-nominal logic.[4] Examples of how π is used follow immediately in Subsection 2.7.

Notation 2.42. We may write $\Phi \vdash \Psi$ as shorthand for '$\Phi \vdash \Psi$ is a derivable sequent'.

We may write $\Phi \nvdash \Psi$ as shorthand for '$\Phi \vdash \Psi$ is not a derivable sequent'.

Remark 2.43. Figure 1 includes rules for $\bot$, $\Rightarrow$, and $\forall$. Following on from Remark 2.15, note that including rules for other connectives like $\top$, $\neg$, $\wedge$, $\vee$, and $\exists$ would be easy. Because the

[3] Remember that we quotient raw terms by α-equivalence to obtain terms so this is now a real equality.

[4] In this paper equivariance surfaces in a variety of technical features of PNL: the π in $(\mathbf{Ax})$; the way permutations distribute into terms in Definition 2.17; Definition 3.16 and how it is used in Definition 3.18; Lemma 3.22; and more.

In fact, equivariance is broad and useful phenomenon. The interested reader is referred to [Gabbay and Pitts 2001, Lemma 4.7] and [Gabbay 2011b, Subsection 4.2], where it is treated in full generality. See also [Gabbay 2011c, Lemma 5.5], where the conditions on free atoms and support in Definitions 2.32 and 3.19 are also exhibited as forms of equivariance.

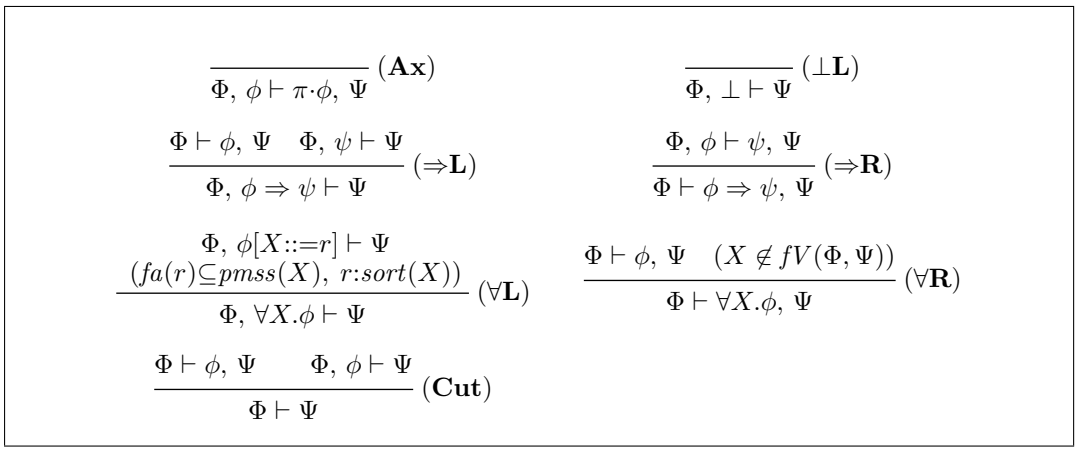

Fig. 1: Sequent derivation rules of Permissive-Nominal Logic

PNL in this paper is classical, we can treat derivation rules for them as a definable extension of what we already have.

We see no inherent difficulty with constructing an intuitionistic version of PNL.

2.7. Universal quantification, permission sorts, and *shift*-permutations

Recall the comment on 'atoms as data' in Remark 2.38. Because of permutations, in certain circumstances free atoms can still behave like variables ranging over distinct atoms (cf. the *permutative convention* of Definition 2.7). Atoms-substitution is not be primitive in PNL, but atoms-permutation is.

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

This goes further. The condition $fa(r) \subseteq pmss(X)$ in $(\forall\mathbf{L})$ might suggest that $\forall X.\phi$ means "$\phi[X::=r]$ for every r such that $fa(r) \subseteq pmss(X)$". Indeed this is so, but what $pmss(X)$ in $\forall X.\phi$ really restricts is *capture*, as we now discuss.

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

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

In other words, we can derive $\mathsf{P}(a)$ from $\forall X.\mathsf{P}(X)$, even if a is not permitted in X.
Thus, in the case of level 2 closed terms (without unknowns), these have finitely many atoms and we can use a finite permutation to place them in $pmss(X)$.

— This may not work for the more general case of a term *with* unknowns; for example there is no finite π such that $fa(\pi{\cdot}(X, a)) \subseteq pmss(X)$ where $a \notin pmss(X)$.
So consider the general case of any sort α and suppose $X : \alpha$ and $pmss(X) = S$. Suppose $\mathsf{Q} : \alpha$. Consider any other $Y : \alpha$ and $pmss(Y) = T$. We will show that $\forall X.\mathsf{Q}(X) \vdash \mathsf{Q}(Y)$ is derivable.

Recall that shift permutation $shift_\nu$ from Definition 2.9 and the definition of $shift^n_\nu$ from Notation 2.16. Using $shift$ permutations we can construct a permutation π such that $S \cup T \subseteq \pi{\cdot}pmss(X)$.[5]
We derive as follows:

$$\dfrac{\dfrac{}{\mathsf{Q}(\pi^{-1}{\cdot}Y) \vdash \mathsf{Q}(Y)}\,(\mathbf{Ax})}{\forall X.\mathsf{Q}(X) \vdash \mathsf{Q}(Y)}\,(\forall\mathbf{L}) \quad [X ::= \pi^{-1}{\cdot}Y]$$

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

$$\forall X.\mathsf{P}([a]X) \nvdash P([a]a), \quad \text{and for all } b,\ \forall X.\mathsf{P}([a]X) \nvdash P([b]b).$$

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

3. SEMANTICS OF PERMISSIVE-NOMINAL LOGIC

Nominal sets were introduced in [Gabbay and Pitts 2001] (they were called 'FM sets'). Technically, a nominal set is a set with a finitely-supported permutation action for atoms. Intuitively, a nominal set is a set with 'free names' in a manner which parallels how names feature in abstract syntax, but without necessarily being syntactic structures.

Names in nominal sets are modelled as atoms. They can be renamed by the permutation action; they can be bound by an atoms-abstraction construction; and they feature a *finite support* property which guarantees that we can always pick a fresh name.

The interested reader can consult a literature which includes [Gabbay and Pitts 2001] or [Gabbay 2011b] for detailed discussions of nominal sets and their applications.

We will interpret PNL using *permissive-nominal* sets. The permissive-nominal sets we use here generalise nominal sets in two ways:

— They allow *infinite support*, since permission sets from Definition 2.7 need not be finite.
— They allow (some) *infinite permutations*, since permutations from Definition 2.9 are generated as a group by swappings and by infinite $shift$ permutations, thus giving $\forall$-quantification extra power as discussed in Subsection 2.7.

The main results are soundness (Theorem 3.30) and completeness (Theorem 3.45). The main definitions are of permissive-nominal sets in Definition 3.4, and of the interpretations of terms and propositions in Definitions 3.20 and 3.25 respectively.

The permissive-nominal development here resembles that in [Gabbay and Pitts 2001]. Definition 3.4 is a little subtle because we ignore infinite permutations when we determine support, whereas equivariance from Definition 3.16 means commuting with *all* permutations.

3.1. Permissive-nominal sets

Recall $\mathbb{P}$ the set of all permutations from Definition 2.9.

Definition 3.1. A **set with a permutation action** X is a pair $(|\mathsf{X}|, \cdot)$ of

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

[5] If permission sets were finite, or if all permutations were finite, then we could not do this in general.

Definition 3.2. Say a set of atoms $A \subseteq \mathbb{A}$ **supports** $x \in |\mathsf{X}|$ when for all finite permutations π, if $\pi(a) = a$ for all $a \in A$ then $\pi{\cdot}x = x$.

Thus, if a permission set S supports x and $\forall a{\in}S.\pi(a) = \pi'(a)$ then $\pi{\cdot}x = \pi'{\cdot}x$.

Remark 3.3. $\mathbb{P}$ contains infinite as well as finite permutations. In the next paragraph we construct an x that is supported by $\varnothing$ (so that $\pi{\cdot}x = x$ for all finite $\pi \in \mathbb{P}$) and yet $\mathit{shift}_\nu(x) \neq x$. This observation is the same as *fuzzy support* from [Gabbay 2007b].

Recall the order f_ν on $\mathbb{A}_\nu$ from Definition 2.7 and consider

$$x = \{\pi{\cdot}(f_\nu(0), f_\nu(\text{-}1), f_\nu(1), f_\nu(\text{-}2), \ldots) \mid \pi \in \mathbb{P}_{\text{fin}}\}$$

the set of finite permutations of $\mathbb{A}_\nu$ written out in order, with the pointwise permutation action. Then $\pi{\cdot}x = x$ for every finite permutation, but $\mathit{shift}_\nu{\cdot}x \neq x$.

Definition 3.4. A **permissive-nominal set** is a set with a permutation action such that every element has a supporting permission set.
X, Y will range over permissive-nominal sets.

Theorem 3.5. *Suppose* X *is a permissive-nominal set. Then every* $x \in |\mathsf{X}|$ *has a unique least supporting set* $\mathit{supp}(x) \subseteq \mathbb{A}$.[6]

As a corollary, if π *is finite and* $\pi(a) = a$ *for all* $a \in \mathit{supp}(x)$ *then* $\pi{\cdot}x = x$.

Theorem 3.5 is familiar from [Gabbay and Pitts 2001], but we do have to be a little bit careful since we are not working with nominal sets. It all works out:

Proof. The corollary is immediate given the definition of support (Definition 3.2).

Define $A = \bigcap\{S \mid S \text{ permission set, supports } x\}$. Also, choose some permission set S that supports x.

Suppose π is finite and $\pi(a) = a$ for all $a \in A$. Write $a_1, \ldots, a_n$ for the atoms in $\mathit{nontriv}(\pi) \cap S$, in some order. Let $b_1, \ldots, b_n$ be some choice of fresh atoms (so $b_i \notin S \cup \mathit{nontriv}(\pi) \cup A$ for $1 \leq i \leq n$). Write $\tau = (b_1\ a_1) \circ \ldots \circ (b_n\ a_n)$. It is routine to check that $(\tau \circ \pi \circ \tau)(a) = a$ for every $a \in S$. Thus $\tau{\cdot}(\pi{\cdot}(\tau{\cdot}x)) = x$. Now $\tau{\cdot}x = x$, and it follows by a routine manipulation that $\pi{\cdot}x = x$ as required. □

Lemma 3.6. *Suppose* X *is a permissive-nominal set and* $x \in |\mathsf{X}|$. *Then* $\mathit{supp}(\pi{\cdot}x) = \pi{\cdot}\mathit{supp}(x)$ *(Definition 2.19).*

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

Corollary 3.7. *Suppose* X *is a permissive-nominal set and* $x \in |\mathsf{X}|$. *Suppose* $b \notin \mathit{supp}(x)$. *Then* $(b\ a){\cdot}x = x$ *implies* $a \notin \mathit{supp}(x)$.

Proof. Suppose $b \notin \mathit{supp}(x)$. We prove the contrapositive. Suppose $a \in \mathit{supp}(x)$. By Lemma 3.6 $\mathit{supp}((b\ a){\cdot}x) = (b\ a){\cdot}\mathit{supp}(x)$. By our suppositions, $(b\ a){\cdot}\mathit{supp}(x) \neq \mathit{supp}(x)$. It follows that $(b\ a){\cdot}x \neq x$. □

3.2. Examples of permissive-nominal sets

3.2.1. Atoms

Definition 3.8. $\mathbb{A}$ the set of atoms can be considered a permissive-nominal set with a natural permutation action $\pi{\cdot}a = \pi(a)$.

The set $\{0, 1\}$ can be considered a permissive-nominal set with the natural **trivial** permutation action $\pi{\cdot}x = x$ for all $\pi \in \mathbb{P}$ and $x \in \{0, 1\}$.

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

[6] $\mathit{supp}(x)$ need not necessarily be a permission set. For instance, $\mathit{supp}(a) = \{a\}$.

3.2.2. Atoms-abstraction

Definition 3.9. Suppose X is a permissive-nominal set and $\mathbb{A}_\nu$ is a set of atoms. Suppose $x \in |\mathsf{X}|$ and $a \in \mathbb{A}_\nu$. Define **atoms-abstraction** $[a]x$ and $[\mathbb{A}_\nu]\mathsf{X}$ by:

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

Remark 3.10. In the definition of $[a]x$ in Definition 3.9 recall that by our permutative convention $b \neq a$. An equivalent and more compact way of writing this is $[a]x = \{(\pi(a), \pi\cdot x) \mid \pi \in fix(supp(x)\backslash\{a\})\}$ where $fix(A) = \{\pi \mid \forall a{\in}A.\pi(a) = a\}$.

Lemma 3.11. *(1)* $[\mathbb{A}_\nu]\mathsf{X}$ *is a permissive-nominal set.*
(2) $[a]x{=}[a]x'$ *if and only if* $x{=}x'$*, for* $a{\in}\mathbb{A}_\nu$ *and* $x{\in}|\mathsf{X}|$*.*
(3) $[a]x{=}[a']x'$ *if and only if* $a'{\notin}supp(x)$ *and* $(a'\ a)\cdot x{=}x'$*, for* $a, a'{\in}\mathbb{A}_\nu$ *and* $x, x'{\in}|\mathsf{X}|$*.*

3.2.3. Product

Definition 3.12. If X_i are permissive-nominal sets for $1 \leq i \leq n$ then define $\mathsf{X}_1 \times \ldots \times \mathsf{X}_n$ by:

$$\begin{array}{c} |\mathsf{X}_1 \times \ldots \times \mathsf{X}_n| = |\mathsf{X}_1| \times \ldots \times |\mathsf{X}_n| \\ \pi\cdot(x_1,\ldots,x_n) = (\pi\cdot x_1,\ldots,\pi\cdot x_n) \end{array}$$

Lemma 3.13. — $supp(a) = \{a\}$.
— $supp([a]x) = supp(x) \setminus \{a\}$.
— $supp((x_1,\ldots,x_n)) = \bigcup\{supp(x_i) \mid 1 \leq i \leq n\}$.

Proof. Proofs are as in [Gabbay and Pitts 2001] or [Gabbay 2011b]. □

3.3. Interpretation and soundness

3.3.1. Interpretation of signatures

Definition 3.14. Suppose $(\mathcal{A}, \mathcal{B})$ is a sort-signature (Definition 2.1).

A **(PNL) interpretation** or **model** $\mathcal{I}$ for $(\mathcal{A}, \mathcal{B})$ consists of an assignment of a nonempty permissive-nominal set $\tau^{\mathcal{I}}$ to each $\tau \in \mathcal{B}$.[7]

We extend an interpretation $\mathcal{I}$ to sorts by:

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

Remark 3.15. Note in Definition 3.14 that a base sort τ is interpreted by a permissive-nominal set $\tau^{\mathcal{I}}$ given in the interpretation, whereas a name sort ν *must* be interpreted by its corresponding set of atoms $\mathbb{A}_\nu$ as fixed in Definition 2.7. This is part of a nominal 'slogan' that *atoms are interpreted by themselves*.

Definition 3.16. Suppose X and Y are sets with a permutation action. Call a function f from $|\mathsf{X}|$ to $|\mathsf{Y}|$ **equivariant** when

$$\forall \pi \in \mathbb{P}.\forall x \in |\mathsf{X}|.\pi\cdot(f(x)) = f(\pi\cdot x). \qquad \text{(equivariance for functions)}$$

Lemma 3.17. *If* f *from* $|\mathsf{X}|$ *to* $|\mathsf{Y}|$ *is equivariant then* $supp(f(x)) \subseteq supp(x)$ *for all* $x \in |\mathsf{X}|$*.*

[7] We favour the word 'interpretation' for assigning a denotational interpretation to a logic, and 'model' for checking whether the interpretation makes a theory (a set of axioms). These senses overlap, in that an interpretation is a model for the empty theory. In practice, we tend to use whichever word seems most appropriate in context.

Proof. If $\pi \in \mathit{fix}(\mathit{supp}(x))$ then $\pi{\cdot}f(x) = f(\pi{\cdot}x)$, and if π is finite then $\pi{\cdot}x = x$. □

Definition 3.18. Suppose $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{P}, ar)$ is a signature (Definition 2.3).
A **(PNL) interpretation** $\mathcal{I}$ for $\mathcal{S}$ consists of the following data:

— An interpretation for the sort-signature $(\mathcal{A}, \mathcal{B})$ (Definition 3.14).
— For every $\mathsf{f} \in \mathcal{F}$ with $ar(\mathsf{f}) = (\alpha)\tau$ an equivariant function $\mathsf{f}^{\mathcal{I}}$ from $[\![\alpha]\!]^{\mathcal{I}}$ to $[\![\tau]\!]^{\mathcal{I}}$ (Definition 3.16).
— For every $\mathsf{P} \in \mathcal{P}$ with $ar(\mathsf{P}) = \alpha$ a finite equivariant function $\mathsf{P}^{\mathcal{I}}$ from $[\![\alpha]\!]^{\mathcal{I}}$ to $\{0,1\}$ (Definition 3.8).

Definition 3.19. 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,

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

ς will range over valuations.

3.3.2. Interpretation of terms

Definition 3.20. 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:

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

Lemma 3.21. *If* $r : \alpha$ *then* $[\![r]\!]^{\mathcal{I}}_{\varsigma} \in [\![\alpha]\!]^{\mathcal{I}}$.

Proof. By a routine induction on r. □

Lemma 3.22. $\pi{\cdot}[\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![\pi{\cdot}r]\!]^{\mathcal{I}}_{\varsigma}$.

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

— The case $\pi'{\cdot}X$. By Definition 3.20 $[\![\pi'{\cdot}X]\!]^{\mathcal{I}}_{\varsigma} = \pi'{\cdot}\varsigma(X)$. Therefore $\pi{\cdot}[\![\pi'{\cdot}X]\!]^{\mathcal{I}}_{\varsigma} = \pi{\cdot}(\pi'{\cdot}\varsigma(X))$. It is a fact of the group action (Definition 3.1) that $\pi{\cdot}(\pi'{\cdot}\varsigma(X)) = (\pi \circ \pi'){\cdot}\varsigma(X)$, and of the permutation action (Definition 2.17) that $\pi{\cdot}(\pi'{\cdot}X) = (\pi \circ \pi'){\cdot}X$. The result follows. □

Lemma 3.23. $\mathit{supp}([\![r]\!]^{\mathcal{I}}_{\varsigma}) \subseteq \mathit{fa}(r)$.

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

— The case $\pi{\cdot}X$. $\mathit{fa}(\pi{\cdot}X) = \pi{\cdot}\mathit{pmss}(X)$ by Definition 2.20. By assumption in Definition 3.19 $\mathit{supp}(\varsigma(X)) \subseteq \mathit{pmss}(X)$.

The cases of a, $[a]r$, and $[a]r$ use parts 1, 2, and 3 of Lemma 3.13. The case of f uses Lemma 3.17. □

3.3.3. Interpretation of propositions

Definition 3.24. Suppose ς is a valuation to an interpretation $\mathcal{I}$. Suppose X is an unknown and $x \in [\![\mathit{sort}(X)]\!]^{\mathcal{I}}$ is such that $\mathit{supp}(x) \subseteq \mathit{pmss}(X)$. Define a function $\varsigma[X::=x]$ 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 3.25. Suppose that $\mathcal{I}$ is an interpretation. Define an **interpretation of propositions** by:

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

Lemma 3.26. $[\![\phi]\!]^{\mathcal{I}}_{\varsigma} = [\![\pi \cdot \phi]\!]^{\mathcal{I}}_{\varsigma}$ *always.*

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

— The case $\forall X.\phi$. Suppose $[\![\forall X.\phi]\!]^{\mathcal{I}}_{\varsigma} = 1$. This means that $[\![\phi]\!]^{\mathcal{I}}_{\varsigma[X::=x]} = 1$ for all $x \in [\![\alpha]\!]^{\mathcal{I}}$ such that $supp(x) \subseteq pmss(X)$. By inductive hypothesis $[\![\pi \cdot \phi]\!]^{\mathcal{I}}_{\varsigma[X::=x]} = 1$ for all $x \in [\![\alpha]\!]^{\mathcal{I}}$ such that $supp(x) \subseteq pmss(X)$. Therefore $[\![\forall X.\pi \cdot \phi]\!]^{\mathcal{I}}_{\varsigma} = 1$. The result follows, since $\pi \cdot (\forall X.\phi) = \forall X.\pi \cdot \phi$.
— The case $\mathsf{P}(r)$. We have $[\![\mathsf{P}(r)]\!]^{\mathcal{I}}_{\varsigma} = \mathsf{P}^{\mathcal{I}}([\![r]\!]^{\mathcal{I}}_{\varsigma})$. As $\mathsf{P}^{\mathcal{I}}$ is equivariant, we get $[\![\mathsf{P}(r)]\!]^{\mathcal{I}}_{\varsigma} = \mathsf{P}^{\mathcal{I}}(\pi \cdot [\![r]\!]^{\mathcal{I}}_{\varsigma})$. By Lemma 3.22 $\pi \cdot [\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![\pi \cdot r]\!]^{\mathcal{I}}_{\varsigma}$. Thus $[\![\mathsf{P}(r)]\!]^{\mathcal{I}}_{\varsigma} = \mathsf{P}^{\mathcal{I}}([\![\pi \cdot r]\!]^{\mathcal{I}}_{\varsigma}) = [\![\pi \cdot \mathsf{P}(r)]\!]^{\mathcal{I}}_{\varsigma}$. □

Lemma 3.27. — $[\![r]\!]^{\mathcal{I}}_{\varsigma[X::=[\![t]\!]^{\mathcal{I}}_{\varsigma}]} = [\![r[X::=t]]\!]^{\mathcal{I}}_{\varsigma}$.
— $[\![\phi]\!]^{\mathcal{I}}_{\varsigma[X::=[\![t]\!]^{\mathcal{I}}_{\varsigma}]} = [\![\phi[X::=t]]\!]^{\mathcal{I}}_{\varsigma}$.

Proof. By routine inductions on the definitions of $[\![r]\!]^{\mathcal{I}}_{\varsigma}$ and $[\![\phi]\!]^{\mathcal{I}}_{\varsigma}$ in Definitions 3.20 and 3.25. We consider two cases:

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

$$\begin{array}{rll} [\![\pi \cdot X]\!]^{\mathcal{I}}_{\varsigma[X::=[\![t]\!]^{\mathcal{I}}_{\varsigma}]} & = \pi \cdot [\![t]\!]^{\mathcal{I}}_{\varsigma} & \text{Definition 3.20} \\ & = [\![\pi \cdot t]\!]^{\mathcal{I}}_{\varsigma} & \text{Lemma 3.22} \\ & = [\![(\pi \cdot X)[X::=t]]\!]^{\mathcal{I}}_{\varsigma} & \text{Definition 2.35.} \end{array}$$

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

$$\begin{array}{rll} [\![\mathsf{P}(r)]\!]^{\mathcal{I}}_{\varsigma[X::=[\![t]\!]^{\mathcal{I}}_{\varsigma}]} & = \mathsf{P}^{\mathcal{I}}([\![r]\!]^{\mathcal{I}}_{\varsigma[X::=[\![t]\!]^{\mathcal{I}}_{\varsigma}]}) & \text{Definition 3.25} \\ & = \mathsf{P}^{\mathcal{I}}([\![r[X::=t]]\!]^{\mathcal{I}}_{\varsigma}) & \text{Part 1 of this result} \\ & = [\![\mathsf{P}(r)[X::=t]]\!]^{\mathcal{I}}_{\varsigma} & \text{Definition 3.25.} \end{array}$$

□

Lemma 3.28. *If* $\varsigma(X) = \varsigma'(X)$ *for all* $X \in fV(r)$ *then* $[\![r]\!]^{\mathcal{I}}_{\varsigma} = [\![r]\!]^{\mathcal{I}}_{\varsigma'}$*, and similarly for* ϕ*.*

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

3.3.4. Validity and soundness

Definition 3.29 (Validity)**.** Call the proposition ϕ **valid** in $\mathcal{I}$ when $[\![\phi]\!]^{\mathcal{I}}_{\varsigma} = 1$ for all ς.
Call the sequent $\phi_1, ..., \phi_n \vdash \psi_1, ..., \psi_p$ **valid** in $\mathcal{I}$ when $(\phi_1 \wedge ... \wedge \phi_n) \Rightarrow (\psi_1 \vee ... \vee \psi_p)$ is valid.

Theorem 3.30 (Soundness)**.** *If* $\Phi \vdash \Psi$ *is derivable, then it is valid in all interpretations.*

Proof. By induction on derivations (Figure 1). The case of $(\mathbf{Ax})$ uses Lemma 3.26. The case of $(\forall\mathbf{L})$ uses Lemma 3.27. The case of $(\forall\mathbf{R})$ uses Lemma 3.28. Other rules are routine by unpacking definitions. □

3.4. Completeness

In this subsection we prove Theorem 3.45: a converse to Theorem 3.30, that if ϕ is valid in all models, then ϕ it is derivable.

For this subsection fix the following data:

— A signature $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{P}, ar)$.
— A formula ϕ such that $\nvdash \phi$.

We will construct an interpretation $\mathcal{I}$ and a valuation ς (Definition 3.14) such that $[\![\phi]\!]^{\mathcal{I}}_{\varsigma} = 0$. This suffices to prove the result.

3.4.1. Maximally consistent set of propositions

Definition 3.31. Choose a fixed but arbitrary enumeration of propositions $\xi_1, \xi_2, \xi_3, \ldots$
Define $\Phi_1 = \{\neg\phi\}$. For each $i \geq 1$ we define Φ_{i+1} as follows:

— If $\Phi_i \vdash \xi_i$ then write $\xi = \xi_i$.
— If $\Phi_i \vdash \neg\xi_i$ then write $\xi = \neg\xi_i$.
— If $\Phi_i \nvdash \xi_i$ and $\Phi_i \nvdash \neg\xi_i$ then write $\xi = \xi_i$.

There are now two cases:

— If ξ has the form $\neg\forall X.\xi'$ then we define $\Phi_{i+1} = \Phi_i \cup \{\xi, \neg\xi'[X::=Z]\}$ where Z is some fixed but arbitrary choice of unknown that is not free in any proposition in Φ_i and is such that $pmss(Z) = pmss(X)$ and $sort(Z) = sort(X)$.
— Otherwise, we define $\Phi_{i+1} = \Phi_i \cup \{\xi\}$.

Finally, we define Φ_ω by $\Phi_\omega = \bigcup_i \Phi_i$.

Lemma 3.32. *For every i, $\Phi_i \nvdash \bot$.*

Proof. By induction on i:

— By definition $\Phi_1 = \{\neg\phi\}$. As $\nvdash \phi$, we have $\neg\phi \nvdash \bot$
— Suppose $\Phi_i \nvdash \bot$. Either $\Phi_{i+1} = \Phi_i \cup \{\neg\xi\}$ or $\Phi_{i+1} = \Phi_i \cup \{\neg\xi, \neg\xi'[X::=Z]\}$—we consider the first, simpler case; the second case is similar. Suppose $\Phi_i, \xi \vdash \bot$. It follows that $\Phi_i \vdash \neg\xi$. So we are not in the third case of Definition 3.31 and we are either in the first or the second. So $\Phi_i \vdash \xi$ and thus $\Phi_i \vdash \bot$; a contradiction. □

Lemma 3.33. $\Phi_\omega \nvdash \bot$.

Proof. Assume $\Phi_\omega \vdash \bot$. So there exists a finite subset Γ of Φ_ω such that $\Gamma \vdash \bot$. As Γ is finite it is included in some Φ_i, and $\Phi_i \vdash \bot$, contradicting Proposition 3.32. □

Lemma 3.34. *For every ξ, at least one of $\xi \in \Phi_\omega$ and $\neg\xi \in \Phi_\omega$ holds.*

Proof. We check of Definition 3.31 that for every i, either $\xi_i \in \Phi_{i+1}$ or $\neg\xi_i \in \Phi_{i+1}$. The result follows. □

Lemma 3.35. *For every ξ, if $\neg\forall X.\xi \in \Phi_\omega$ then there exists a Z such that $\neg\xi[X::=Z] \in \Phi_\omega$.*

Proof. There exists an i such that $\xi_i = \neg\forall X.\xi$. Since $\Phi_\omega \vdash \xi_i$ and $\Phi_\omega \nvdash \bot$, we have that $\Phi_\omega \nvdash \neg\xi_i$, and so $\Phi_i \nvdash \neg\xi_i$. Thus $\Phi_{i+1} = \Phi_i \cup \{\neg\forall X.\xi,\ \neg\xi[X::=Z]\}$. The result follows. □

Lemma 3.36. *If $\Phi_\omega \vdash \phi$ then $\phi \in \Phi_\omega$.*

Proof. As, by Lemma 3.33, $\Phi_\omega \nvdash \bot$, if $\Phi_\omega \vdash \phi$ then $\neg\phi \notin \Phi_\omega$. Thus by Lemma 3.34, $\phi \in \Phi_\omega$. □

Corollary 3.37. (1) *$(\phi \Rightarrow \psi) \in \Phi_\omega$ if and only if ($\phi \notin \Phi_\omega$ or $\psi \in \Phi_\omega$).*
(2) *$\forall X.\phi \in \Phi_\omega$ if and only if*
(for every r such that $r : sort(X)$ and $fa(r) \subseteq pmss(X)$, $\phi[X::=r] \in \Phi_\omega$).

Proof. (1) Suppose $(\phi \Rightarrow \psi) \in \Phi_\omega$ and $\phi \in \Phi_\omega$. Then $\Phi_\omega \vdash \psi$ and so by Lemma 3.36 $\psi \in \Phi_\omega$.
Suppose $\phi \notin \Phi_\omega$. By Lemma 3.34 $\neg\phi \in \Phi_\omega$. So $\Phi_\omega \vdash \neg\phi$ and therefore $\Phi_\omega \vdash \phi \Rightarrow \psi$. By Lemma 3.36 $(\phi \Rightarrow \psi) \in \Phi_\omega$.
Suppose $\psi \in \Phi_\omega$. Then $\Phi_\omega \vdash \psi$ and so $\Phi_\omega \vdash \phi \Rightarrow \psi$. By Lemma 3.36 $(\phi \Rightarrow \psi) \in \Phi_\omega$.
(2) Suppose $\forall X.\phi \in \Phi_\omega$. By Lemma 3.36, if $r : sort(X)$ and $\forall(r) \subseteq pmss(X)$ then $\phi[X::=r] \in \Phi_\omega$.
Conversely, suppose $\phi[X::=r] \in \Phi_\omega$ for every r such that $r : sort(X)$ and $\forall(r) \subseteq pmss(X)$. We proceed by contradiction: suppose $\forall X.\phi \notin \Phi_\omega$. By Lemma 3.34 $\neg\forall X.\phi \in \Phi_\omega$ and by Lemma 3.35, there exists a Z such that $\neg\phi[X::=Z] \in \Phi_\omega$. So $\Phi_\omega \vdash \neg\phi[X::=Z]$, and so $\Phi_\omega \vdash \phi[X::=Z]$, and so $\Phi_\omega \vdash \bot$, contradicting Lemma 3.33. □

3.4.2. The term model

Definition 3.38. Define $\mathcal{I}$ by:

— $[\![\alpha]\!]^{\mathcal{I}} = \{r \mid r : \alpha\}$.
— $\mathsf{f}^{\mathcal{I}}$ maps r to $\mathsf{f}(r)$.
— $\mathsf{P}^{\mathcal{I}}$ maps $r_1, \ldots, r_n$ to 1 if $\mathsf{P}(r_1, \ldots, r_n) \in \Phi_\omega$ and to 0 otherwise.

Define ς by

$$\varsigma(X) = X.$$

We endow $[\![\alpha]\!]^{\mathcal{I}}$ with a permutation action given by the action on terms.

Lemma 3.39. *$supp(r) = fa(r)$.*
As a corollary, $[\![\alpha]\!]^{\mathcal{I}}$ from Definition 3.38 is a permissive-nominal set.

Proof. We prove two subset inclusions:

— *Proof that $supp(r) \subseteq fa(r)$.* By Lemma 2.29, if $\pi(a) = a$ for all $a \in fa(r)$ then $r = \pi{\cdot}r$. It follows by the definition of support (Theorem 3.5) that $supp(r) \subseteq fa(r)$.
— *Proof that $fa(r) \subseteq supp(r)$.* Suppose $a \in fa(r)$. Choose some fresh b (so $b \notin fa(r) \cup supp(r)$). By Lemma 2.21 $fa((b\ a){\cdot}r) = (b\ a){\cdot}fa(r)$. Since $(b\ a){\cdot}fa(r) \neq fa(r)$ it follows using Lemma 2.27 that $r \neq (b\ a){\cdot}r$. The result follows by the first part of this result and by Corollary 3.7.

The corollary is immediate, unpacking Definition 3.2. □

Lemma 3.40. (1) *If $ar(\mathsf{f}) = (\alpha)\tau$ then $\mathsf{f}^{\mathcal{I}}$ is well-defined, equivariant, and maps $[\![\alpha]\!]^{\mathcal{I}}$ to $[\![\tau]\!]^{\mathcal{I}}$.*
(2) *If $ar(\mathsf{P}) = \alpha$ then $\mathsf{P}^{\mathcal{I}}$ is well-defined, equivariant, and maps $[\![\alpha]\!]^{\mathcal{I}}$ to $\{0, 1\}$.*

Proof. (1) The only (very) slightly non-trivial part is equivariance. We reason as follows:

$\pi{\cdot}\mathsf{f}^{\mathcal{I}}(r)$		
$=$	$\pi{\cdot}\mathsf{f}(r)$	Definition 3.20
$=$	$\mathsf{f}(\pi{\cdot}r)$	Definition 2.17
$=$	$\mathsf{f}^{\mathcal{I}}(\pi{\cdot}r)$	Definition 3.20

(2) The case of P is similar. □

Proposition 3.41. *$\mathcal{I}$ is an interpretation of the signature $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{P}, ar)$ which we fixed at the beginning of this subsection. In addition, ς is a valuation to $\mathcal{I}$.*

Proof. By Lemma 3.39 each $[\![\alpha]\!]^{\mathcal{I}}$ is a permissive-nominal set. By Lemma 3.40 for each $\mathsf{f} : (\alpha)\tau \in \mathcal{F}$, $\mathsf{f}^{\mathcal{I}}$ is an equivariant map from $[\![\alpha]\!]^{\mathcal{I}}$ to $[\![\tau]\!]^{\mathcal{I}}$ and for each $\mathsf{P} : \alpha \in \mathcal{P}$, $\mathsf{P}^{\mathcal{I}}$ is a finite equivariant function from $[\![\alpha]\!]^{\mathcal{I}}$ to $\{0, 1\}$.
By construction $\varsigma(X) \in [\![sort(X)]\!]^{\mathcal{I}}$ always. By Lemma 3.39 $supp(\varsigma(X)) = supp(X) \subseteq pmss(X)$ always.
The result follows. □

Lemma 3.42. $[\![r]\!]^{\mathcal{I}}_{\varsigma} = r$.

Proof. By a routine induction on the definition of $[\![r]\!]^{\mathcal{I}}_{\varsigma}$ in Definition 3.20. We consider just one case:

— The case of $[\![\pi{\cdot}X]\!]^{\mathcal{I}}_{\varsigma}$. We reason as follows:

$$\begin{array}{rcll} [\![\pi{\cdot}X]\!]^{\mathcal{I}}_{\varsigma} & = & \pi{\cdot}\varsigma(X) & \text{Definition 3.20} \\ & = & \pi{\cdot}X & \text{Definition 3.38.} \end{array}$$

□

Lemma 3.43. *$[\![\xi]\!]^{\mathcal{I}}_{\varsigma} = 1$ if and only if $\xi \in \Phi_\omega$.*

Proof. By induction on the definition of $[\![\xi]\!]^{\mathcal{I}}_{\varsigma}$ (Definition 3.25):

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

$$\begin{array}{rcll} [\![\mathsf{P}(r)]\!]^{\mathcal{I}}_{\varsigma} = 1 & \Leftrightarrow & \mathsf{P}^{\mathcal{I}}([\![r]\!]^{\mathcal{I}}_{\varsigma}) = 1 & \text{Definition 3.25} \\ & \Leftrightarrow & \mathsf{P}^{\mathcal{I}}(r) = 1 & \text{Lemma 3.42} \\ & \Leftrightarrow & \mathsf{P}(r) \in \Phi_\omega & \text{Definition 3.38} \end{array}$$

— The case of $[\![\bot]\!]^{\mathcal{I}}_{\varsigma}$. By definition $[\![\bot]\!]^{\mathcal{I}}_{\varsigma} = 0$. By part 1 of Corollary 3.37, $\bot \notin \Phi_\omega$.

— The case of $[\![\phi \Rightarrow \psi]\!]^{\mathcal{I}}_{\varsigma}$. We reason as follows:

$$\begin{array}{rcll} [\![\phi \Rightarrow \psi]\!]^{\mathcal{I}}_{\varsigma} = 1 & \Leftrightarrow & [\![\phi]\!]^{\mathcal{I}}_{\varsigma} = 0 \text{ or } [\![\psi]\!]^{\mathcal{I}}_{\varsigma} = 1 & \text{Definition 3.25} \\ & \Leftrightarrow & \phi \notin \Phi_\omega \text{ or } \psi \in \Phi_\omega & \text{ind. hyp.} \\ & \Leftrightarrow & \phi \Rightarrow \psi \in \Phi_\omega & \text{Corollary 3.37, part 2} \end{array}$$

— The case of $[\![\forall X.\phi]\!]^{\mathcal{I}}_{\varsigma}$, where $\alpha = \mathit{sort}(X)$ and $S = \mathit{pmss}(X)$.

$$\begin{array}{rcll} [\![\forall X.\phi]\!]^{\mathcal{I}}_{\varsigma} = 1 & \Leftrightarrow & \forall t \in [\![\alpha]\!]^{\mathcal{I}}.\mathit{supp}(t) \subseteq S \Rightarrow [\![\phi]\!]^{\mathcal{I}}_{\varsigma[X::=t]} = 1 & \text{Definition 3.25} \\ & \Leftrightarrow & \forall t \in [\![\alpha]\!]^{\mathcal{I}}.\mathit{supp}(t) \subseteq S \Rightarrow [\![\phi[X::=t]]\!]^{\mathcal{I}}_{\varsigma} = 1 & \text{Lems. 3.28, 3.42} \\ & \Leftrightarrow & [\![\phi[X::=t]]\!]^{\mathcal{I}}_{\varsigma} = 1 \text{ every } t : \alpha \text{ s.t. } \mathit{fa}(t) \subseteq S & \text{Lem. 3.39} \\ & \Leftrightarrow & \phi[X::=t] \in \Phi_\omega \text{ every } t : \alpha \text{ s.t. } \mathit{fa}(t) \subseteq S & \text{ind. hyp.} \\ & \Leftrightarrow & \forall X.\phi \in \Phi_\omega & \text{Cor. 3.37, part 4} \end{array}$$

□

Lemma 3.44. *If $\nvdash \phi$, then there exists a model $\mathcal{I}$ and a valuation ς such that $[\![\phi]\!]^{\mathcal{I}}_{\varsigma} = 0$.*

Proof. As $\neg\phi \in \Phi_0 \subseteq \Phi_\omega$ and $\Phi_\omega \nvdash \bot$, we have $\phi \notin \Phi_\omega$. By Lemma 3.43, we get $[\![\phi]\!]^{\mathcal{I}}_{\varsigma} = 0$. □

As a corollary we get Theorem 3.45:

Theorem 3.45 (Completeness)**.** *If ϕ is valid in all models, then ϕ is derivable.*

4. SPECIFYING ARITHMETIC IN PERMISSIVE-NOMINAL LOGIC

We start by defining the sorts, term-formers, and proposition-formers of a signature $\dot{\mathcal{L}}$ which is suitable for finitely specifying arithmetic in PNL. We then specify its axioms and, in Subsection 4.2, we discuss them in detail.

4.1. The signature $\dot{\mathcal{L}}$ and the axioms

Definition 4.1. A signature $\dot{\mathcal{L}}$ suitable for writing out a PNL theory of first-order logic is given in Figure 2.

We introduce the following syntactic sugar:

— We may write $\mathsf{sub}_o([a]r, t)$ as $r[a{\mapsto}t]$.
— We may write $\mathsf{sub}_\iota([a]r, t)$ as $r[a{\mapsto}t]$.
— We may write both $\approx_\iota$ and $\approx_o$ just as $\approx$.

Examples of this syntactic sugar in use, follow immediately below.

We assume one atomic sort ν and two base sorts ι and o.
We assume term-formers and proposition-formers as follows:

$$\begin{array}{llll} \dot{0} : \iota & \dot{\mathsf{succ}} : (\iota)\iota & \dot{+} : (\iota,\iota)\iota & \dot{*}: (\iota,\iota)\iota \\ \dot{\bot} : o & \dot{\Rightarrow}: (o,o)o & \dot{\forall} : ([\nu]o)o & \dot{\approx}: (\iota,\iota)o \\ \mathsf{var} : (\nu)\iota & \mathsf{sub}_\iota : ([\nu]\iota,\iota)\iota & \mathsf{sub}_o : ([\nu]o,\iota)o & \\ \approx_\iota : (\iota,\iota) & \approx_o : (o,o) & \epsilon : (o) & \end{array}$$

Fig. 2: Signature suitable for a PNL specification of arithmetic

$$\begin{array}{lllll} (\approx\mathbf{2}) & \forall X', X, Y', Y.(X'\approx X \wedge Y'\approx Y) \Rightarrow X'\ op\ Y' \approx X\ op\ Y & & & op\in\{\dot{+},\dot{*},\dot{\Rightarrow},\dot{\approx}\} \\ (\approx\mathbf{1}) & \forall X', X. & X'\approx X \Rightarrow op(X') \approx op(X) & & op\in\{\dot{\mathsf{succ}}\} \\ (\approx\mathbf{0}) & \forall X. & X \approx X & & \\ (\approx\dot{\forall}) & \forall Z', Z. & Z'\approx Z \Rightarrow \dot{\forall}([a]Z') \approx \dot{\forall}([a]Z) & & \\ (\approx\mathsf{sub}) & \forall X', X, Y', Y.(X'\approx X \wedge Y'\approx Y) \Rightarrow op([a]X', Y') \approx op([a]X, Y) & & & op\in\{\mathsf{sub}_\iota, \mathsf{sub}_o\} \\ (\approx o) & \forall Z', Z. & Z'\approx Z \Rightarrow (\epsilon(Z') \Leftrightarrow \epsilon(Z)) & & \\ (\approx\iota) & \forall X', X. & X'\approx X \Rightarrow \epsilon(X' \mathrel{\dot{\approx}} X) & & \end{array}$$

We fill in sorts as appropriate. Thus, $\dot{\bot} \approx_o \dot{\bot}$ whereas $0 \approx_\iota 0$, and so on. The permission sets of all variables are equal to $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Fig. 3: EQU: axioms for equality as a PNL theory

$$\begin{array}{lllll} (\mathbf{sub}\mathsf{var}) & \forall X.\ \mathsf{var}(a)[a{\mapsto}X] & \approx X & & \\ (\mathbf{sub}\#) & \forall X, Z.\ Z[a{\mapsto}X] & \approx Z & & (pmss(Z) = (b\ a)\cdot\mathbb{A}^<) \\ (\mathbf{sub}\dot{\mathsf{succ}}) & \forall X', X.\ \dot{\mathsf{succ}}(X')[a{\mapsto}X] & \approx \dot{\mathsf{succ}}(X'[a{\mapsto}X]) & & \\ (\mathbf{sub}\dot{+}) & \forall X'', X', X.\ (X'' \mathrel{\dot{+}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathrel{\dot{+}} X'[a{\mapsto}X]) & & \\ (\mathbf{sub}\dot{*}) & \forall X'', X', X.\ (X'' \mathrel{\dot{*}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathrel{\dot{*}} X'[a{\mapsto}X]) & & \\ (\mathbf{sub}\dot{\Rightarrow}) & \forall X'', X', X.\ (X'' \mathrel{\dot{\Rightarrow}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathrel{\dot{\Rightarrow}} X'[a{\mapsto}X]) & & \\ (\mathbf{sub}\dot{\approx}) & \forall X'', X', X.\ (X'' \mathrel{\dot{\approx}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathrel{\dot{\approx}} X'[a{\mapsto}X]) & & \\ (\mathbf{sub}\dot{\forall}) & \forall X, Z.\ (\dot{\forall}([b]Z))[a{\mapsto}X] & \approx \dot{\forall}([b](Z[a{\mapsto}X])) & & (pmss(Z) = (b\ a)\cdot\mathbb{A}^<) \\ (\mathbf{subid}) & \forall X.\ X[a{\mapsto}\mathsf{var}(a)] & \approx X & & \end{array}$$

$a \in \mathbb{A}^<$ and $b \notin \mathbb{A}^<$. The permission set of X'', X', and X is equal to $\mathbb{A}^<$. The permission set of Z is equal to $(b\ a)\cdot\mathbb{A}^<$ (Definition 2.19).

Fig. 4: SUB: axioms for substitution as a PNL theory

Equality. Axioms for equality $\approx$: (ι,ι) and equality $\approx$: (o,o) are given in Figure 3.

Substitution. Axioms for substitution sub_ι and sub_o are given in Figure 4.
We arguably abuse notation in Figure 4 when we use variables of sort ι and o as appropriate not necessarily giving them distinct names (e.g. in $(\mathbf{sub}*)$ X has sort ι, whereas in $(\mathbf{sub}\dot{\Rightarrow})$ we use another variable also written X with sort o).

First-order logic. Axioms reflecting first-order formulas (in a shallow sense) as terms in PNL (the $\dot{\bot}$, $\dot{\Rightarrow}$, and $\dot{\forall}$) are given in Figure 5.

Arithmetic. Given EQU, SUB, and FOL, it is not hard to write axioms for arithmetic in PNL. This is in Figure 6. Later on in Theorem 5.21 we prove that this *is* an axiomatisation of arithmetic in PNL.

$$
\begin{array}{ll}
(\dot{\Rightarrow}) & \forall Z', Z.\ \epsilon(Z' \mathrel{\dot{\Rightarrow}} Z) \Leftrightarrow (\epsilon(Z') \Rightarrow \epsilon(Z)) \\
(\dot{\forall}) & \forall Z.\ \big(\epsilon(\dot{\forall}([a]Z)) \Leftrightarrow \forall X.\epsilon(Z[a{\mapsto}X])\big) \\
(\dot{\bot}) & \epsilon(\dot{\bot}) \Rightarrow \bot
\end{array}
$$

Here Z' and Z have sort o, permission set $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Fig. 5: FOL: axioms for first-order formulas as a PNL theory

$$
\begin{array}{ll}
(\mathbf{PS0}) & \forall X.\ \mathsf{succ}(X) \approx \dot{0} \Rightarrow \bot \\
(\mathbf{PSS}) & \forall X', X.\ \mathsf{succ}(X') \approx \mathsf{succ}(X) \Rightarrow X' \approx X \\
(\mathbf{P{+}0}) & \forall X.\ X \mathbin{\dot{+}} \dot{0} \approx X \\
(\mathbf{P{+}succ}) & \forall X', X.\ X' \mathbin{\dot{+}} \mathsf{succ}(X) \approx \mathsf{succ}(X') \mathbin{\dot{+}} X \\
(\mathbf{P{*}0}) & \forall X.\ X \mathbin{\dot{*}} \dot{0} \approx \dot{0} \\
(\mathbf{P{*}succ}) & \forall X', X.\ X' \mathbin{\dot{*}} \mathsf{succ}(X) \approx (X' \mathbin{\dot{*}} X) \mathbin{\dot{+}} X \\
(\mathbf{PInd}) & \forall Z.\ (\epsilon(Z[a{\mapsto}\dot{0}]) \Rightarrow \\
 & \quad \big(\forall X.(\epsilon(Z[a{\mapsto}X]) \Rightarrow \epsilon(Z[a{\mapsto}\mathsf{succ}(X)]))\big) \Rightarrow \\
 & \quad \forall X.\epsilon(Z[a{\mapsto}X]))
\end{array}
$$

All variables have permission set $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Fig. 6: ARITH: axioms for arithmetic as a PNL theory

4.2. Comments on the axioms

Remark 4.2. In [Gabbay and Mathijssen 2008a] capture-avoiding substitution is equationally axiomatised using nominal algebra in the style of SUB. Soundness and completeness are proved, so providing some formal sense in which the axioms of SUB are 'right'.

In [Gabbay and Mathijssen 2008b] first-order logic is equationally axiomatised using nominal algebra (so the axioms involve only equality). The axioms of FOL are *not* based on those of [Gabbay and Mathijssen 2008b]. In FOL, we take advantage of the stronger language provided by PNL; Because PNL is already a first-order logic, we can use $\bot$, $\Rightarrow$, and $\forall$ directly to capture the behaviour of $\dot{\bot}$, $\dot{\Rightarrow}$, and $\dot{\forall}$. In [Gabbay and Mathijssen 2008b] we had to work a little harder because the ambient logic, nominal algebra, was purely equational.

Remark 4.3. Instead of the axioms for equality EQU, we could directly extend PNL by adding derivation rules Figure 1 as follows:

$$
\frac{\Phi,\ r \approx s, \phi[X::{=}r],\ \phi[X::{=}s] \vdash \Psi \qquad (fa(r) \cup fa(s) \subseteq pmss(X))}{\Phi,\ r \approx s,\ \phi[X::{=}r] \vdash \Psi}\ (\approx\mathbf{S}) \qquad \frac{\Phi,\ r \approx r \vdash \Psi}{\Phi \vdash \Psi}\ (\approx\mathbf{R})
$$

Remark 4.4. Every unknown has a sort, and a permission set.

Different choices of permission set may yield logically equivalent results. For example, in (**sub**lam) it is not vital that $pmss(Z)$ is *exactly* $(b\ a){\cdot}\mathbb{A}^<$. The important point is that $a \notin pmss(Z)$.

Similarly, in (**sub**app) it is not vital that $pmss(X'') = pmss(X')$; when we use the axiom we can instantiate X'' and X' to r'' and r' such that $fa(r'') \neq fa(r')$, and conversely if we take $pmss(X'') \neq pmss(X')$ then we can still instantiate X'' and X' to r'' and r' such that $fa(r'') = fa(r') \subseteq pmss(X'') \cap pmss(X')$. More on this in Section 9.

5. A THEORY OF ARITHMETIC IN FIRST-ORDER LOGIC

We now recall first-order logic $\mathcal{L}$ and write Peano arithmetic in $\mathcal{L}$. Our two main theorems are:

— Theorem 5.6 which states that the PNL theory of first-order logic written in $\dot{\mathcal{L}}$—in symbols this is $\mathsf{EQU} \cup \mathsf{SUB} \cup \mathsf{FOL}$—soundly interprets first-order logic $\mathcal{L}$; and
— Theorem 5.21 which states that the PNL theory of arithmetic *in* the PNL theory of first-order logic, soundly and completely interprets ordinary Peano arithmetic in written in $\mathcal{L}$.[8]

5.1. First-order logic $\mathcal{L}$

We will use the atoms $\mathbb{A}_\nu$ from $\dot{\mathcal{L}}$ in Section 4 as variables of our first-order logic (this is not necessary, but it is convenient). So for this section, $a, b, c, \ldots$ will range over distinct atoms in $\mathbb{A}_\nu$.

Definition 5.1. Define **terms** and **formulas** of $\mathcal{L}$ by:

$$\begin{array}{l} t ::= a \mid 0 \mid \mathit{succ}(t) \mid t + t \mid t * t \\ \xi ::= t \approx t \mid \bot \mid \xi \Rightarrow \xi \mid \forall a.\xi \end{array}$$

Substitution $t'[a::=t]$ and $\xi[a::=t]$ is as usual for first-order logic. We write sequents $\Xi \vdash \chi$ where Ξ and χ are sets of formulas. Derivability is as usual for first-order logic.

Definition 5.2. Define a mapping $(\text{-})^\cdot$ from terms and formulas of $\mathcal{L}$ to terms of $\dot{\mathcal{L}}$ by:

$$\begin{array}{rlrl} (a)^\cdot &= a & (0)^\cdot &= \dot{0} \\ (\mathit{succ}(t))^\cdot &= \dot{\mathsf{succ}}((t)^\cdot) & (t' + t)^\cdot &= (t')^\cdot \dot{+} (t)^\cdot \\ (t' * t)^\cdot &= (t')^\cdot \dot{*} (t)^\cdot & & \\ (t' \approx t)^\cdot &= (t')^\cdot \dot{\approx} (t)^\cdot & (\bot)^\cdot &= \dot{\bot} \\ (\xi' \Rightarrow \xi)^\cdot &= (\xi')^\cdot \dot{\Rightarrow} (\xi)^\cdot & (\forall a.\xi)^\cdot &= \dot{\forall}[a](\xi)^\cdot \end{array}$$

Definition 5.3. Extend $(\text{-})^\cdot$ to first-order logic sequents $\Xi \vdash \chi$ as follows:

$$(\Xi \vdash \chi)^\cdot = \epsilon(\dot{\forall}[a_1] \ldots \dot{\forall}[a_n]((\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l))^\cdot)$$

Here, $\Xi = \{\xi_1, \ldots, \xi_k\}$, $\chi = \{\chi_1, \ldots, \chi_l\}$, and the free variables of Ξ and χ are $\{a_1, \ldots, a_n\}$ (in some order).

Notation 5.4. Write S for $\mathsf{EQU} \cup \mathsf{SUB} \cup \mathsf{FOL}$.

Lemma 5.5. $\mathsf{S} \vdash (t'[a::=t])^\cdot \approx (t')^\cdot[a{\mapsto}(t)^\cdot]$ *and* $\mathsf{S} \vdash (\xi[a::=t])^\cdot \approx (\xi)^\cdot[a{\mapsto}(t)^\cdot]$.

Proof. By routine inductions on t and ξ. □

[8] Missing from this is a proof that $\mathsf{EQU} \cup \mathsf{SUB} \cup \mathsf{FOL}$ soundly and *completely* interprets first-order logic. We believe this to be true and the proof should be an elementary simplification of the more involved case for arithmetic—but it is not worth writing out here.

Arithmetic is (in first-order logic) axiomatised using a *schema*. We use PNL $\forall X$ to express them finitely; see e.g. the $\forall X$ in (**PInd**) of Figure 6 which models the 'every ξ' in (**pind**) in Figure 7.

Finite first-order logic theories (including the empty theory) are unproblematic. In these proofs, soundness and completeness are only a means to the end of demonstrating how we can axiomatise finitely in a nominal first-order logic PNL, structures that without names and binding would require infinite axiom schemes or higher orders.

Theorem 5.6 (Correctness)**.** *If $\Xi \vdash \chi$ is derivable in first-order logic then $\mathsf{S} \vdash (\Xi \vdash \chi)^{\cdot}$ is derivable in PNL.*

Proof. By a long but routine inspection we can check that EQU, SUB, and FOL allow us to model the behaviour of 'real' first-order logic. We use Lemma 5.5. □

5.2. Interpretation of first-order logic

We recall the usual definition of interpretations in first-order logic:

Definition 5.7. A **(first-order logic) interpretation** $\mathcal{M}$ is a **carrier set** M, and elements

$$0^{\mathcal{M}} \in M, \quad succ^{\mathcal{M}} \in M \to M, \quad +^{\mathcal{M}} \in (M \times M) \to M, \quad \text{and} \quad *^{\mathcal{M}} \in (M \times M) \to M.$$

It is convenient to fix some $\mathcal{M}$ from here until Theorem 5.21.

Definition 5.8. Define $Valu_{\mathbb{A}_\nu}(M)$ by:

$$Valu_{\mathbb{A}_\nu}(M) = \{\varepsilon \in \mathbb{A}_\nu \to M \mid \exists A \subseteq \mathbb{A}_\nu.A \text{ finite } \wedge\ \forall a, b \notin A.\varepsilon(a) = \varepsilon(b)\}$$

Call elements of $Valu_{\mathbb{A}_\nu}(M)$ **$\mathbb{A}_\nu$-valuations** (to M). ε will range over $\mathbb{A}_\nu$-valuations.

If $x \in M$ write $\varepsilon[a::=x]$ for the valuation mapping b to $\varepsilon(b)$ and mapping a to x:

$$\varepsilon[a::=x](a) = x$$
$$\varepsilon[a::=x](b) = \varepsilon(b)$$

Give $\varepsilon \in Valu_{\mathbb{A}_\nu}(M)$ and $X \subseteq Valu_{\mathbb{A}_\nu}(M)$ a **pointwise** permutation action:

$$(\pi{\cdot}\varepsilon)(a) = \varepsilon(\pi^{-1}(a)).$$
$$\pi{\cdot}X = \{\pi{\cdot}\varepsilon \mid \varepsilon \in X\}.$$

U, V will range over *finitely-supported* subsets of $Valu_{\mathbb{A}_\nu}(M)$—so there exists some finite $A \subseteq \mathbb{A}_\nu$ such that for all π, if $\pi(a) = a$ for all $a \in A$ then $\pi{\cdot}U = U$.

Remark 5.9. $Valu_{\mathbb{A}_\nu}(M)$ would normally just be called 'the set of valuations'. We are more specific since we separately also have valuations on unknowns X (Definition 3.19).

PNL atoms are serving as variable symbols of $\mathcal{L}$. To conveniently apply nominal techniques, it is useful to restrict to valuations that are finite in the sense given in Definition 5.8. In any case, any term or formula will only contain finitely many atoms.

Definition 5.10. We extend the interpretation to first-order logic syntax as follows:

$$\begin{aligned}
[\![a]\!]^{\mathcal{M}}_{\varepsilon} &= \varepsilon(a)\\
[\![0]\!]^{\mathcal{M}}_{\varepsilon} &= 0^{\mathcal{M}}\\
[\![succ(t)]\!]^{\mathcal{M}}_{\varepsilon} &= succ^{\mathcal{M}}([\![t]\!]^{\mathcal{M}}_{\varepsilon})\\
[\![t' + t]\!]^{\mathcal{M}}_{\varepsilon} &= +^{\mathcal{M}}([\![t']\!]^{\mathcal{M}}_{\varepsilon}, [\![t]\!]^{\mathcal{M}}_{\varepsilon})\\
[\![t' * t]\!]^{\mathcal{M}}_{\varepsilon} &= *^{\mathcal{M}}([\![t']\!]^{\mathcal{M}}_{\varepsilon}, [\![t]\!]^{\mathcal{M}}_{\varepsilon})\\
[\![\bot]\!]^{\mathcal{M}}_{\varepsilon} &= 0\\
[\![\xi' \Rightarrow \xi]\!]^{\mathcal{M}}_{\varepsilon} &= max\{1-[\![\xi']\!]^{\mathcal{M}}_{\varepsilon}, [\![\xi]\!]^{\mathcal{M}}_{\varepsilon}\}\\
[\![\forall a.\xi]\!]^{\mathcal{M}}_{\varepsilon} &= min\{[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a::=x]} \mid x \in M\}\\
[\![t' \approx t]\!]^{\mathcal{M}}_{\varepsilon} &= 1 \text{ if } [\![t']\!]^{\mathcal{M}}_{\varepsilon} = [\![t]\!]^{\mathcal{M}}_{\varepsilon} \text{ and } 0 \text{ otherwise}
\end{aligned}$$

Definition 5.11. Call the formula ξ **valid** in $\mathcal{M}$ when $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon} = 1$ for all ε.

Call $\xi_1, \ldots, \xi_k \vdash \chi_1, \ldots, \chi_l$ **valid** in $\mathcal{M}$ when $(\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l)$ is valid.

$$
\begin{array}{ll}
(\mathbf{ps0}) & \forall a.\ succ(a) \approx 0 \Rightarrow \bot \\
(\mathbf{pss}) & \forall a', a.\ succ(a) \approx succ(a') \Rightarrow a \approx a' \\
(\mathbf{p{+}0}) & \forall a.\ a + 0 \approx a \\
(\mathbf{p{+}succ}) & \forall a', a.\ a' + succ(a) \approx succ(a') + a \\
(\mathbf{p{*}0}) & \forall a.\ a * 0 \approx 0 \\
(\mathbf{p{*}succ}) & \forall a', a.\ a' * succ(a) \approx (a' * a) + a \\
(\mathbf{pind}) & \xi[a::=0] \Rightarrow \\
 & \quad \forall a.(\xi \Rightarrow \xi[a::=succ(a)]) \Rightarrow \\
 & \quad \forall a.\xi \qquad\qquad \text{(every } \xi\text{, every } a\text{)}
\end{array}
$$

Fig. 7: arithmetic: axioms for arithmetic in first-order logic

5.3. A theory of arithmetic in $\mathcal{L}$

Definition 5.12. Define a first-order theory of **arithmetic** by the axioms in Figure 7.
An interpretation is a **model** of arithmetic when $[\![\xi]\!]^{\mathcal{M}} = 1$ for ξ each of $(\mathbf{ps0})$, $(\mathbf{pss})$, $(\mathbf{p{+}0})$, $(\mathbf{p{+}succ})$, $(\mathbf{p{*}0})$, $(\mathbf{p{*}succ})$, and every instance of $(\mathbf{pind})$.

Remark 5.13. $(\mathbf{pind})$ the induction axiom-scheme is of course of particular interest. We therefore unpack what its validity

$$[\![\xi[a::=0] \Rightarrow \forall a.(\xi \Rightarrow \xi[a::=succ(a)]) \Rightarrow \forall a.\xi]\!]^{\mathcal{M}} = 1 \qquad \text{(every } \xi\text{, every } a\text{)}$$

means, in a little more detail. For every a and ξ:

— If $[\![\xi[a::=0]]\!]^{\mathcal{M}}_{\varepsilon} = 1$, and
— if for every $x \in M$, $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a::=x]} = 1$ implies that $[\![\xi[a::=succ(a)]]\!]^{\mathcal{M}}_{\varepsilon[a::=x]} = 1$,
— then for every $x \in M$, $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a::=x]} = 1$.

In $(\mathbf{pind})$ we take 'every a', and in $(\mathbf{PInd})$ we do not. This is because in $(\mathbf{PInd})$, a is α-convertible,

5.4. Building an interpretation for $\dot{\mathcal{L}}$ out of one for $\mathcal{L}$

Recall the PNL signature $\dot{\mathcal{L}}$ from Section 4. Suppose $\mathcal{M}$ is a model of $\mathcal{L}$. We use it to build an interpretation $\mathcal{N}$ of $\dot{\mathcal{L}}$.

Definition 5.14. Extend $\mathcal{L}$ to $\mathcal{L}{+}M$ where we add all elements of M as constants, and extend the interpretation to interpret these constants as themselves in M. (So if $x \in M$ then x is a constant symbol in $\mathcal{L}{+}M$ and $[\![x]\!]^{\mathcal{M}}_{\varepsilon} = x$.)

Define an $\mathbb{A}_\nu$-valuation $\varepsilon_0 \in Valu_{\mathbb{A}_\nu}(M)$ by

$$\varepsilon_0(a) = 0^{\mathcal{M}} \quad \text{always.}$$

If t is a term, we write $[\![t]\!]^{\mathcal{M}}$ for the function $\lambda\varepsilon.[\![t]\!]^{\mathcal{M}}_{\varepsilon}$. If ξ is a formula, we write $[\![\xi]\!]^{\mathcal{M}}$ for the function $\lambda\varepsilon.[\![\xi]\!]^{\mathcal{M}}_{\varepsilon}$.

We now define an interpretation $\mathcal{N}$ for $\dot{\mathcal{L}}$. We give a denotation to the base sorts ι and o of $\dot{\mathcal{L}}$, as follows:

$$
\begin{array}{ll}
\iota^{\mathcal{N}} = \{[\![t]\!]^{\mathcal{M}} \mid t \text{ a term of } \mathcal{L}{+}M\} & \pi{\cdot}[\![t]\!]^{\mathcal{M}} = [\![\pi{\cdot}t]\!]^{\mathcal{M}} \\
o^{\mathcal{N}} = \{[\![\xi]\!]^{\mathcal{M}} \mid \xi \text{ a formula of } \mathcal{L}{+}M\} & \pi{\cdot}[\![\xi]\!]^{\mathcal{M}} = [\![\pi{\cdot}\xi]\!]^{\mathcal{M}}
\end{array}
$$

We give a denotation to the term-formers and proposition-formers of $\dot{\mathcal{L}}$, as follows:

$$
\begin{array}{rlrl}
\mathsf{var}^{\mathcal{N}}\, a\, \varepsilon = \varepsilon(a) & & \mathsf{sub}^{\mathcal{N}}_{o}\,([a]u, v)\,\varepsilon = u(\varepsilon[a::=v\varepsilon]) \\
\dot{0}^{\mathcal{N}}\,\varepsilon = 0^{\mathcal{M}} & & \dot{\Rightarrow}^{\mathcal{N}}(U, V)\,\varepsilon = max\{1{-}U\varepsilon, V\varepsilon\} \\
(\dot{\mathsf{succ}}^{\mathcal{N}}\, u)\,\varepsilon = succ^{\mathcal{M}}(u\varepsilon) & & \dot{\forall}^{\mathcal{N}}\,([a]U)\,\varepsilon = min\{U(\varepsilon[a::=x]) \mid x \in M\} \\
\dot{+}^{\mathcal{N}}\,(u, v)\,\varepsilon = +^{\mathcal{M}}(u\varepsilon, v\varepsilon) & & \dot{\approx}^{\mathcal{N}}(u, v)\,\varepsilon = \approx^{\mathcal{M}}(u\varepsilon, v\varepsilon) \\
\dot{*}^{\mathcal{N}}\,(u, v)\,\varepsilon = *^{\mathcal{M}}(u\varepsilon, v\varepsilon) & & \approx^{\mathcal{N}}_{\iota}\,(u, v) = 1 \text{ if } u{=}v \text{ and } 0 \text{ otherwise} \\
\mathsf{sub}^{\mathcal{N}}_{\iota}\,([a]u, v)\,\varepsilon = u(\varepsilon[a::=v\varepsilon]) & & \approx^{\mathcal{N}}_{o}\,(U, V) = 1 \text{ if } U{=}V \text{ and } 0 \text{ otherwise} \\
\dot{\perp}^{\mathcal{N}}\,\varepsilon = 0 & & \epsilon^{\mathcal{N}}\, U = U(\varepsilon_0)
\end{array}
$$

Here, u and v range over $\iota^{\mathcal{N}}$ and U and V range over $o^{\mathcal{N}}$. We insert brackets where this might increase clarity.

Remark 5.15. For the reader's convenience we indicate the types of some of the symbols above in an informal 'crib sheet':

$$
\begin{array}{rlrl}
\varepsilon : \mathbb{A}_\nu{\to}M & & \mathsf{var}^{\mathcal{N}} : \mathbb{A}_\nu{\to}(\mathbb{A}_\nu{\to}M){\to}M \\
u : (\mathbb{A}_\nu{\to}M){\to}M & & U : (\mathbb{A}_\nu{\to}M){\to}\{0,1\} \\
0^{\mathcal{M}} : M & & \dot{0}^{\mathcal{N}} : (\mathbb{A}_\nu{\to}M){\to}M \\
\approx^{\mathcal{M}} : M^2{\to}\{0,1\} & & \dot{\approx}^{\mathcal{N}} : ((\mathbb{A}_\nu{\to}M){\to}M)^2{\to}(\mathbb{A}_\nu{\to}M){\to}\{0,1\} \\
 & & \dot{\forall}^{\mathcal{N}} : [\mathbb{A}_\nu]((\mathbb{A}_\nu{\to}M){\to}\{0,1\}) \to (\mathbb{A}_\nu{\to}M){\to}\{0,1\}
\end{array}
$$

This crib sheet is only indicative since, for instance, ε is not *any* function in $\mathbb{A}_\nu{\to}M$ (see Definition 5.8).

Lemma 5.16. (1) $[\![t'[a::=t]]\!]^{\mathcal{M}}_{\varepsilon} = [\![t']\!]^{\mathcal{M}}_{\varepsilon[a::=[\![t]\!]^{\mathcal{M}}_{\varepsilon}]}$.
(2) $[\![\xi[a::=t]]\!]^{\mathcal{M}}_{\varepsilon} = 1$ *if only if* $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a::=[\![t]\!]^{\mathcal{M}}_{\varepsilon}]} = 1$.

Lemma 5.17. *The following equalities all hold:*

$$
\begin{array}{rlrl}
\mathsf{var}^{\mathcal{N}}(a) = [\![a]\!]^{\mathcal{M}} & & \mathsf{sub}^{\mathcal{N}}_{\iota}([a][\![t']\!]^{\mathcal{M}}, [\![t]\!]^{\mathcal{M}}) = [\![t'[a::=t]]\!]^{\mathcal{M}} \\
\dot{0}^{\mathcal{N}} = [\![0]\!]^{\mathcal{M}} & & \mathsf{sub}^{\mathcal{N}}_{o}([a][\![\xi]\!]^{\mathcal{M}}, [\![s]\!]^{\mathcal{M}}) = [\![\xi[a::=s]]\!]^{\mathcal{M}} \\
\dot{\mathsf{succ}}^{\mathcal{N}}([\![t]\!]^{\mathcal{M}}) = [\![succ(t)]\!]^{\mathcal{M}} & & \dot{\perp}^{\mathcal{N}} = [\![\perp]\!]^{\mathcal{M}} \\
\dot{+}^{\mathcal{N}}([\![t']\!]^{\mathcal{M}}, [\![t]\!]^{\mathcal{M}}) = [\![t' + t]\!]^{\mathcal{M}} & & \dot{\Rightarrow}^{\mathcal{N}}([\![\xi']\!]^{\mathcal{M}}, [\![\xi]\!]^{\mathcal{M}}) = [\![\xi' \Rightarrow \xi]\!]^{\mathcal{M}} \\
\dot{*}^{\mathcal{N}}\,([\![t']\!]^{\mathcal{M}}, [\![t]\!]^{\mathcal{M}}) = [\![t' * t]\!]^{\mathcal{M}} & & \dot{\forall}^{\mathcal{N}}\,([a][\![\xi]\!]^{\mathcal{M}}) = [\![\forall a.\xi]\!]^{\mathcal{M}} \\
 & & \dot{\approx}^{\mathcal{N}}([\![r]\!]^{\mathcal{M}}, [\![s]\!]^{\mathcal{M}}) = [\![r \approx s]\!]^{\mathcal{M}}
\end{array}
$$

Proof. We compare Definitions 5.14 and 5.10. Most cases are immediate; we consider only the slightly less trivial ones:

$$
\begin{array}{rcll}
\mathsf{var}^{\mathcal{N}}(a) & = & (\lambda a.\lambda\varepsilon.\varepsilon(a))a & \text{Definition 5.14} \\
 & = & (\lambda a.[\![a]\!]^{\mathcal{M}})a & \text{Definition 5.10} \\
 & = & [\![a]\!]^{\mathcal{M}} & \text{fact} \\
\mathsf{sub}^{\mathcal{N}}_{\iota}([a][\![t']\!]^{\mathcal{M}}, [\![t]\!]^{\mathcal{M}}) & = & \lambda\varepsilon.[\![t']\!]^{\mathcal{M}}(\varepsilon[a::=[\![t]\!]^{\mathcal{M}}\varepsilon]) & \text{Definition 5.14} \\
 & = & \lambda\varepsilon.[\![t'[a::=t]]\!]^{\mathcal{M}} & \text{Lemma 5.16}
\end{array}
$$

Other cases are no harder. □

Lemma 5.18. *$\mathcal{N}$ (Definition 5.14) is a PNL interpretation.*

Proof. We must check that:

— *$\iota^{\mathcal{N}}$ and $o^{\mathcal{N}}$ are permissive-nominal sets.*
By routine calculations. (In fact, $\iota^{\mathcal{N}}$ and $o^{\mathcal{N}}$ are *nominal* sets; that is, their elements all have finite support.)

— *The functions defined in Definition 5.14 map elements of $\iota^{\mathcal{N}}$, $o^{\mathcal{N}}$, $[\mathbb{A}]\iota^{\mathcal{N}}$, and $[\mathbb{A}]o^{\mathcal{N}}$ correctly to the appropriate sets.*
By Lemma 5.17.
— *$\epsilon^{\mathcal{N}}$ is equivariant from $o^{\mathcal{N}}$ to $\{0,1\}$.*
By routine calculations using the fact that $(a\ b)\cdot\varepsilon_0 = \varepsilon_0$. □

Lemma 5.19. *If $(\Xi \vdash \chi)^{\cdot}$ is valid in $\mathcal{N}$, then $\Xi \vdash \chi$ is valid in $\mathcal{M}$.*

Proof. We calculate that if $(\Xi \vdash \chi)^{\cdot}$ is valid in $\mathcal{N}$, then

$$[\![(\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l)]\!]^{\mathcal{M}}_{\varepsilon_0} = 1$$

But the proposition written out above is closed, so for all valuations ε, $[\![(\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l)]\!]^{\mathcal{M}}_{\varepsilon} = 1$. □

Recall from Notation 5.4 that we write S for EQU∪SUB∪FOL. Recall also from Definition 5.2 the mapping $(\text{-})^{\cdot}$ from first-order logic $\mathcal{L}$ to PNL terms.

Proposition 5.20. *$\mathcal{N}$ is a model of* S ∪ ARITH.

Proof. By a routine verification. We consider the axiom $(\dot{\forall})$ from Figure 5. We unpack definitions and see that we must prove that for every ξ in $\mathcal{L}{+}M$,

— $\forall x \in M.\varepsilon_0[a::=x] \in [\![\xi]\!]^{\mathcal{N}}$ if and only if
— $\varepsilon_0[a::=[\![t]\!]^{\mathcal{N}}_{\varepsilon_0}] \in [\![\xi]\!]^{\mathcal{N}}$ for every t a term of $\mathcal{L}{+}M$.

This follows, because $\mathcal{L}{+}M$ has a constant symbol for every $x \in M$. Validity of the other axioms is no harder. □

Theorem 5.21. arithmetic, $\Xi \vdash \chi$ *in first-order logic if and only if* S ∪ ARITH $\vdash (\Xi \vdash \chi)^{\cdot}$ *in PNL.*

Proof. We prove two implications. The top-to-bottom implication follows using Theorem 5.6.
For the bottom-to-top implication, we reason as follows: Suppose S ∪ ARITH $\vdash (\Xi \vdash \chi)^{\cdot}$ in PNL. Choose an arbitrary interpretation $\mathcal{M}$ of first-order logic that is a model of arithmetic, with carrier set M. By Soundness (Theorem 3.30) and Proposition 5.20, $(\Xi \vdash \chi)^{\cdot}$ is valid in $\mathcal{N}$. By Lemma 5.19 $\Xi \vdash \chi$ is valid in $\mathcal{M}$. $\mathcal{M}$ was arbitrary, so by completeness of first-order logic [Shoenfield 1967, §4.2] it follows that $\Xi \vdash \chi$ is derivable. □

6. MORE PNL THEORIES

So far we have built PNL and used it to finitely axiomatise arithmetic. In this section we briefly touch on how to express some known 'nominal' constructs within PNL.

6.1. Inductive types

Permissive-nominal logic can express the principles of nominal abstract syntax developed in [Gabbay and Pitts 2001].
Suppose a base sort ι, a name sort ν, and term-formers

$$\mathsf{var} : \nu \to \iota, \quad \mathsf{app} : (\iota, \iota) \to \iota, \quad \text{and} \quad \mathsf{lam} : [\nu]\iota \to \iota.$$

Fix an unknown $U : \iota$ and for brevity write $\phi[U::=r]$ as $\phi(r)$ for every ϕ. Suppose an axiom-scheme, for every ϕ:

$$\begin{array}{l}\phi(\mathsf{var}(a)) \Rightarrow \\ \forall X.(\phi(X) \Rightarrow \phi(\mathsf{lam}([a]X))) \Rightarrow \\ \forall X, Y.(\phi(X) \Rightarrow \phi(Y) \Rightarrow \mathsf{app}(X, Y)) \Rightarrow \\ \qquad \forall X.(\phi(X))\end{array}$$

Here X and Y have sort ι and we make a fixed but arbitrary choice of atom $a \in pmss(X)$.

We can also express this finitely, if we axiomatise a sort for predicates (as we did for arithmetic). Here is the axiom-scheme above made finite by using the theories EQU, SUB, and FOL from Section 4:

$$\begin{array}{l}\forall Z.\epsilon(Z[a{\mapsto}\mathsf{var}(a)]) \Rightarrow \\ \quad \forall X.(\epsilon(Z[a{\mapsto}X]) \Rightarrow \epsilon(Z[a{\mapsto}\mathsf{lam}([a]X))) \Rightarrow \\ \quad \forall X,Y.(\epsilon(Z[a{\mapsto}X]) \Rightarrow \epsilon(Z[a{\mapsto}Y]) \Rightarrow \epsilon(Z[a{\mapsto}\mathsf{app}(X,Y)])) \Rightarrow \\ \qquad \forall X.\epsilon(Z[a{\mapsto}X])\end{array}$$

6.2. The И quantifier

Nominal sets support the И-quantifier [Gabbay and Pitts 2001]. PNL also includes the И-quantifier; the way in which it does this is quite interesting, as we shall see in a moment.

И has some distinctive properties which are reflected in nominal logic (NL) and the logic of FM sets (FM):

$$\frac{\forall x.(\mathsf{P}(x) \Rightarrow И a.\mathsf{Q}(a,x))}{\forall x.И a.(\mathsf{P}(x) \Rightarrow \mathsf{Q}(a,x))} \qquad \frac{\forall x.И a.И b.(b\,a){\cdot}x{\approx}x}{И a.И b.\forall x.(a\#x \Rightarrow b\#x \Rightarrow (b\,a){\cdot}x{\approx}x)}$$

Here and below we write a double horizontal line for 'is provably equivalent to'. И appears absent from Permissive-Nominal Logic (PNL). It is 'hiding' in the permission sets. Corresponding propositions are, where $a, b \notin pmss(X)$:

$$\frac{\forall X.(\mathsf{P}(X) \Rightarrow \mathsf{Q}(a,X))}{\forall X.(\mathsf{P}(X) \Rightarrow \mathsf{Q}(a,X))} \qquad \frac{\forall X.(b\,a){\cdot}X \approx X}{\forall X.(b\,a){\cdot}X \approx X}$$

We see from these examples that two things are happening: first, freshness conditions are hard-coded into the syntax by permission sets—and second, so is the И-quantifier.

It is interesting to consider another example. In NL/FM:

$$\frac{И a.\mathsf{P}(a) \wedge И a.\mathsf{Q}(b)}{И a.И b.(\mathsf{P}(a) \wedge \mathsf{Q}(b))} \qquad \frac{И a.\mathsf{P}(a) \wedge И a.\mathsf{Q}(b)}{И a.(\mathsf{P}(a) \wedge \mathsf{Q}(a))}$$

Correspondingly in PNL we have:

$$\frac{\mathsf{P}(a) \wedge \mathsf{Q}(b)}{\mathsf{P}(a) \wedge \mathsf{Q}(b)} \qquad \frac{\mathsf{P}(a) \wedge \mathsf{Q}(b)}{\mathsf{P}(a) \wedge \mathsf{Q}(a)}$$

It is easy to use the rule $(\mathbf{Ax})$ from Figure 1 to construct a derivation proving that $\mathsf{P}(a) \wedge \mathsf{Q}(b)$ and $\mathsf{P}(a) \wedge \mathsf{Q}(a)$ are indeed logically equivalent in Permissive-Nominal Logic.

The π in $(\mathbf{Ax})$ expresses that truth is preserved by permutative renaming, or in symbols: $\vdash \phi \Leftrightarrow \pi{\cdot}\phi$ always.

A permission set S can be viewed in two ways: as giving permission to instantiate using free atoms in S—but also as a form of И for the atoms not in S.

6.3. Freshness $a\#x$ and abstraction

PNL has a notion of syntactic freshness which we identify as a notion of 'free atoms of', and write $fa(t)$ and $fa(\phi)$. Nominal sets also have a *semantic* notion of freshness $a\#x$ given by $a \notin supp(x)$.

As Lemma 3.23 demonstrates, intuitively if a is not free in t then a is fresh for the denotation of t. In symbols: $a \notin fa(t) \Rightarrow a\#[\![t]\!]$ (see [Gabbay 2012, Subsection 7.6] for a kind of converse).

To capture in syntax the effect of a freshness predicate for a name sort ν on a sort α, it suffices to assume EQU (or extend PNL with an equality primitive) and to assume a predicate

$\#$ of arity (ν, α) with an axiom

$$\forall X.(a\#X \Leftrightarrow (b\ a)\cdot X \approx X).$$

Here X has sort α and a and b have sort ν, and $a \in pmss(X)$ and $b \notin pmss(X)$. This is essentially equation 13 in [Gabbay and Pitts 2001], using permission sets to attain the effect of the И-quantifier. See also [Gabbay and Mathijssen 2009, Subsection 5.2] and [Gabbay and Mathijssen 2007, Theorem 5.5] where similar observations were expressed for nominal algebra.

Similarly atoms-abstraction $[a]r$ can be axiomatised not using atoms-abstraction as a term-former $\mathsf{abs} : (\nu, \tau)$ with axiom $\forall X.\mathsf{abs}(b, (b\ a)\cdot X) \approx \mathsf{abs}(a, X)$ where $b \notin pmss(X)$ (cf. [Gabbay and Mathijssen 2009, Subsection 5.1]).

However, it is worthwhile to provide atoms-abstraction as primitive in PNL because it gives us access to PNL α-equivalence which is a structural part of the PNL derivation system: to rename a in $\mathsf{abs}(a, X)$ into $\mathsf{abs}(b, (b\ a)\cdot X)$ requires equality reasoning and axioms; to rename a in $[a]X$ requires nothing but an α-conversion (in this paper, they are actually the same term).

7. CUT-ELIMINATION

Recall the $(\mathbf{Cut})$ rule from Figure 1. In this section we prove that $(\mathbf{Cut})$ is admissible in the presence of the other rules in Figure 1.

Definition 7.1. Suppose $fa(r) \subseteq pmss(X)$ and $r : sort(X)$. Define $\Phi[X::=r]$ by

$$\Phi[X::=r] = \{\phi[X::=r] \mid \phi \in \Phi\}.$$

Lemmas 7.2 and 7.3 are proved by routine arguments like those in [Dowek et al. 2010; Urban et al. 2004]:

Lemma 7.2. $(\pi\cdot r)\theta = \pi\cdot(r\theta)$.

Lemma 7.3. *Suppose $Y \notin fV(t)$. Then*

$$r[Y::=u][X::=t] = r[X::=t][Y::=u[X::=t]].$$

Lemma 7.4. *Suppose $fa(r) \subseteq pmss(X)$ and $r : sort(X)$. Then*

$$\Phi \vdash \Psi \quad \textit{implies} \quad \Phi[X::=r] \vdash \Psi[X::=r].$$

Proof. By a routine induction on derivations. The case of $(\mathbf{Ax})$ uses Lemmas 7.2 and 7.3. The case of $(\forall\mathbf{L})$ uses Lemma 7.3. □

Lemma 7.5. (1) *If there exists a derivation Δ of $\Phi \vdash \psi, \Psi$ then there exists a derivation of $\Phi \vdash \pi\cdot\psi, \Psi$.*
(2) *If there exists a derivation Δ of $\Phi, \phi \vdash \Psi$ then there exists a derivation of $\Phi, \pi\cdot\phi \vdash \Psi$.*

Proof. By a simultaneous induction on Δ. The case of $(\forall\mathbf{L})$ uses Lemma 7.2. (We need the *simultaneous* induction for $(\Rightarrow\mathbf{L})$ and $(\Rightarrow\mathbf{R})$, since parts of the proposition move between left and right.) □

Notation 7.6. An instance of $(\mathbf{Cut})$ rests on two sub-derivations. It is convenient to call them the **left branch** and **right branch** as illustrated:

$$\frac{\begin{array}{cc}\vdots\ \textit{Left branch} & \vdots\ \textit{Right branch}\\ \Phi, \phi \vdash \Psi & \Phi \vdash \phi, \Psi\end{array}}{\Phi \vdash \Psi}\ (\mathbf{Cut})$$

Theorem 7.7 (Cut-elimination)**.** *If $\Phi \vdash \Psi$ is derivable with a derivation that uses $(\mathbf{Cut})$, then it is derivable with a derivation that does not use $(\mathbf{Cut})$.*

Proof. The proof is as for first-order logic. The only differences are a π in $(\mathbf{Ax})$ and a side-condition $fa(r) \subseteq pmss(X)$ in $(\forall\mathbf{L})$. These affect terms and have no effect on the structure of derivations; for the purposes of this proof they are irrelevant.

We commute instances of $(\mathbf{Cut})$ upwards, as usual, following the method of [Dummett 1977, pages 139-145] or [Gabbay 2011a]. At each step, the following measure based on the depth of subderivations and the size of the cut formula, decreases:

— The size of the cut formula, and
— the longest path up the derivation the cut, that the formula persists,

lexicographically ordered.

— The commutation cases between rules for $\Rightarrow$ and $\forall$ are as standard for first-order logic.
— The essential case for $\Rightarrow$ is as standard.
— For the essential case for $\forall$, suppose the subderivation has the following form:

$$\dfrac{\dfrac{\Phi,\ \phi[X::=r] \vdash \Psi}{\Phi,\ \forall X.\phi \vdash \Psi}\,(\forall\mathbf{L}) \qquad \dfrac{\vdots\ \Delta}{\Phi \vdash \phi,\ \Psi}\,(\forall\mathbf{R})}{\Phi \vdash \Psi}\,(\mathbf{Cut})$$

By Lemma 7.4 there is a derivation $\Delta[X::=r]$ of $\Phi \vdash \phi[X::=r],\ \Psi$. We eliminate the essential case as follows:

$$\dfrac{\Phi,\ \phi[X::=r] \vdash \Psi \qquad \dfrac{\vdots\ \Delta[X::=r]}{\Phi \vdash \phi[X::=r],\ \Psi}}{\Phi \vdash \Psi}\,(\mathbf{Cut})$$

— Suppose the subderivation has the following form:

$$\dfrac{\dfrac{}{\Phi,\ \phi \vdash \pi{\cdot}\phi,\ \Psi}\,(\mathbf{Ax}) \qquad \dfrac{\vdots\ \Delta}{\Phi,\ \pi{\cdot}\phi \vdash \Psi}}{\Phi,\ \phi \vdash \Psi}\,(\mathbf{Cut})$$

We use Lemma 7.5 to obtain a derivation Δ' of $\Phi,\ \phi \vdash \Psi$ (the transformations involved in the proof of Lemma 7.5 do not increase the inductive measure).

□

8. RELATED WORK

8.1. Other 'nominal' syntaxes and logics

Compared to other 'nominal' logics, PNL emphasises ergonomics. When we use PNL to axiomatize and build proofs in arithmetic or in set theory, we want our hypotheses to speak about natural numbers or sets, not to speak about atoms and freshness.

Thus, although notions of atom and freshness are important, they should be implicit; i.e. handled automatically by the logic. This is the purpose of permission sets, which allow us to handle freshness and α-renaming separately from logical deduction and equality reasoning.

Axiomatisations. The two best-known 'nominal' logics are probably the *nominal logic* of [Pitts 2003] and *FM set theory*. Both of these are Hilbert-style theories—sets of axioms—in first-order logic. They are axiomatic theories of sets.

FM set theory contains axioms whose intended model is a sets cumulative hierarchy, whereas nominal logic contains axioms only for sets with a finitely-supported permutation

action, with no assumption that they be composed of other sets. For the purposes of this paper, the difference is not important.

Qua logic, PNL is a logic whereas nominal logic and FM set theory are axiomatisations. In addition and closely related to this, we can 'just α-rename' and 'just choose a fresh atom'—as mentioned above we have α-renaming and freshness without appealing to equality reasoning and axioms.

Proof-theories for the И-quantifier. Natural deduction rules for И are proposed e.g. in [Gabbay and Pitts 2001, Proposition 4.10], but these are not closed under substitution. The second author created a proof-theory for И [Gabbay 2007a] with a good notion of proof-normalisation and a completeness proof, followed by an alternative treatment with Cheney [Gabbay and Cheney 2004].

These gave И an operational behaviour as 'pick a locally fresh name'; Cheney then developed another sequent system which gave И an operational behaviour as 'pick a globally fresh name' [Cheney 2005].

The logic of [Cheney 2005] includes 12 infinite axiom-schemes (Figures 3 and 4 of [Cheney 2005]) describing the behaviour of atoms-abstraction from [Gabbay and Pitts 2001]. Thus, α-equivalence (for atoms) is axiomatically handled and does not participate in the proof-theory.

PNL handles both α-equivalence and the И-quantifier very compactly, without recourse to axioms, and indeed requiring neither equality reasoning nor a И-quantifier.

One-and-a-halfth order logic. This logic is designed to represent schematic first-order reasoning (first-order derivations in the presence of 'unknown predicates'). It corresponds roughly to the axiomatisation of first-order logic in Section 4.

Semantic nominal terms. In [Gabbay and Mulligan 2009a] we show how to interpret level 2 variables (unknowns) as infinite lists of distinct level 1 variables (atoms). This allows us to build permissive-nominal term syntax as nominal abstract syntax-style inductive datatypes as proposed in [Gabbay and Pitts 2001]. The aim of this paper is to discuss the logic; not to analyse how its syntax could best be built.

8.2. From nominal terms-in-freshness-context to PNL terms-with-permission-sets

Nominal terms were introduced in [Urban et al. 2004] where a decidable and efficient unification algorithm was demonstrated (see [Calvès and Fernández 2008; Levy and Villaret 2010; Calvès 2010] for the state of the art). Nominal terms have been used in equational specification languages; in rewriting [Fernández and Gabbay 2007] and in universal algebra (the logic of equality) [Gabbay and Mathijssen 2009].

PNL differs from nominal terms in three ways:

— Nominal terms use a finite *freshness context* $a\#X$ whereas PNL uses permission sets, following [Dowek et al. 2009; Dowek et al. 2010] (a 'permission set' S can equally well be considered as a freshness sets $\mathbb{A} \setminus S$).
— PNL predicates include universal quantification over unknowns $\forall X$.
— PNL term syntax includes *shift*-permutations (the implications of this are discussed in Subsection 2.7).

These features optimise PNL for being a first-order style logical foundation for mathematics with binding.

Another way to view this paper is as follows: PNL is the 'obvious' extension of nominal algebra [Gabbay and Mathijssen 2009] (an equational logic based on nominal terms), to a first-order logic.

But how then do we arrive at *permissive* nominal techniques, starting from nominal algebra? Suppose we have an equality axiom $a\#X \vdash \mathsf{f}([a]X) = \mathsf{g}(X)$. We want to write a corresponding first-order axiom. There are two obvious routes to follow:

(1) Assume first-order logic with a sort of atoms ν and some axioms (like nominal logic or FM set theory) and write
$$\forall z, x.z\#x \Rightarrow \mathsf{f}([z]x) = \mathsf{g}(x).$$
Here z and x are variables and freshness $\#$ can be expressed using the axioms of the logic.
The problem with this is that we lose proof-theory; we are just working in a Hilbert-style axiom system.

(2) Imagine a first-order logic with nominal terms in which freshness conditions are attached to quantifiers, so that we can write
$$\forall X{:}\nu_{\#a}.\mathsf{f}([a]X) = \mathsf{g}(X).$$
Here $\nu_{\#a}$ means 'elements of ν for which a is fresh'.
The problem with this is that we have a poor theory of α-equivalence; the freshness context for X does not allow us to rename $[a]X$ to $[b](b\ a)\cdot X$, because there *is* no b fresh for a (more on this in *Implementing PNL* below).

Concerning the second option, we can add 'freshening' axioms or derivation rules to the effect that $\forall X : \nu_{\#a}.\phi$ be logically equivalent to $\forall X{:}\nu_{\#a,b}.\phi$, and so on—this is in essence what the 'freshening' rule (**fr**) of nominal algebra (rule in Figure 2 of [Gabbay and Mathijssen 2009]) does. It should be possible to construct a version of PNL along these lines; the disadvantage would be that once X is instantiated, we can no longer add further fresh atoms, unless we reintroduce (**fr**) into PNL, but even then we would not recover full permissive-nominal α-equivalence.

In PNL we take the idea to the ωth degree; taking a limit of this 'freshening' operation to obtain infinitely many fresh atoms, we arrive at permission sets.

8.3. Non-nominal logics

First- and higher-order logic. As discussed in the Introduction, we see PNL as sitting somewhere in-between these two logics. It is more powerful—and we would claim more ergonomic—than first-order logic, because term-formers can bind. Its advantage over higher-order logic is the smaller and simpler models and generally more first-order character. Its term-syntax supports a decidable unification algorithm: both without *shift* [Dowek et al. 2010] and with [Gabbay 2012].

Logics based on the ∇-quantifier. A family of logics exists based on higher-order patterns and the ∇-quantifier [Miller and Tiu 2003; Tiu 2007; Gacek et al. 2008]. The intended meaning of e.g. $\nabla x.r{=}s$ is '$\lambda x.r{=}\lambda x.s$'. Thus for instance the intended denotation of $\nabla x.\nabla y.x{=}y$ is $\lambda x.\lambda y.x{=}\lambda x.\lambda y.y$, and this is false.

As this example suggests, logics based on ∇ use *raising* and patterns (in brief: higher-order variables applied to finite lists of distinct variables, as in $xx_1 \ldots x_n$) to obtain the effect of capturing substitution and variable dependencies, whereas we use permission sets and a two-level term syntax. Our reading is that ∇ is a way of peeling a single λ-abstraction uniformly off all terms and pushing it 'into the meta-level'. Or, to put it another way: ∇ generates a fresh λ-abstracted variable.

The main philosophical difference here is that ∇ is designed to assume α-equivalence and treats variables as a 'wire' which must always be bound, either by λ in a term or possibly by a top-level ∇; in contrast nominal techniques treat names as global and permutable and break α-equivalence down into names and permutation. In a separate journal paper submitted for publication, we relate these by translating permissive-nominal logic to higher-order

logic in the style of e.g. a translation of permissive-nominal term unification to higher-order pattern unification [Dowek et al. 2010] or nominal algebra to algebra over higher-order terms [Gabbay and Mulligan 2009b].

However, note that raising can cause a linear expansion in the size of a term (because what is represented by X in this paper would be represented by $xx_1 \dots x_n$ in a logic based on raising), and can also cause 'silly' β-redexes (since the mechanism which encodes dependency is the same mechanism which encodes computation). This is one of the motivations for CMTT discussed below.

Contextual modal type theory (CMTT). CMTT [Nanevski et al. 2008] is a two-level system; typing contexts split into two halves; Δ and Φ. The two levels are different from the two levels used in (permissive-)nominal terms. Variables $u : A[\Phi] \in \Delta$ range over representations of code; variables $x : A \in \Phi$ range over denotation.

This addresses a problem that is mostly orthogonal to what PNL tries to achieve. To the extent that this could be represented in PNL, it would be represented at the level of sorts—one sort for code, another for denotation (values).

Logics based on CMTT are consistent and have a well-studied proof-theory, so individual semantic models can be constructed using normal forms, but the question "what is the general class of structures which these syntactic models represent?"' has no answer we know of. That is, no general sets-based class of models has has been given for CMTT. Developing such models—perhaps using techniques borrowed from nominal sets—would be interesting future work.

Further logics. Coming from other threads of research in computer science are logics designed to enrich first-order logic directly with binders without thinking specifically about inductive reasoning. We note in particular *binding logic* [Dowek et al. 2002] and *λ-logic* [Beeson 2004].

Binding logic enriched first-order terms with binders but forbade capture and turned out to be a little too weak.

λ-logic takes a direct approach of enriching first-order terms with λ-abstraction. The approach to binding taken by PNL is somewhat more general and is certainly different, in that it allows us to treat names as 'bindable constants'. That is, we can compare names for *in*equality as names, while at the same time we can give them the behaviour of variables by axiomatising e.g. substitution for them, if we wish.

9. FURTHER REMARKS, FURTHER WORK

We have seen how permissive-nominal logic (PNL) extends first-order logic with term-formers that can bind. We have given PNL a nominal-sets based semantics and shown it sound and complete. We have considered a finite axiomatisation of arithmetic, based on a finite axiomatisation of first-order logic in PNL, and proven it correct. Finally, we have proved cut-elimination.

In most respects PNL behaves just like first-order logic. However, its 'nominal' constructs let it ergonomically perform many of the tasks that would require much more powerful constructs in e.g. higher-order logic.

We do not claim that PNL makes nominal terms like those used in [Urban et al. 2004; Fernández and Gabbay 2007] obsolete. The argument for permissive-nominal terms is that they are a more abstract and powerful mathematical model with which to do proofs; for we use them to 'just quotient' by α-equivalence, and we use them to reconcile α-equivalence of atoms with $\forall X$. But that story is entirely compatible with prevous work. For instance, though permission sets have the form $(\mathbb{A}^{<} \cup A) \setminus B$ (Definition 2.7), in practice we only seem to specify restrictions like $a \in \mathit{pmss}(X)$ and $b \notin \mathit{pmss}(X)$—and these look like freshness environments. In the discussion of Subsection 2.7 we saw why: permission sets control capture, and we only care about controlling capture for the finitely many atoms mentioned explicitly in an axiom.

Furthermore the use of shift-permutations (Definition 2.9) means that the exact choice of permission set 'does not really matter'—more on this below. In a sense, freshness contexts live on in this paper and remain useful, as an emergent property of how we interact with a more abstract underlying mathematical model given by permissive-nominal terms.

We do not claim that PNL is the ultimate logic, whatever that means. However, we do see PNL as a significant step forward in the continuing search for logics suitable for axiomatising the systems with binding which are so common in computer science, as briefly discussed in Section 8. We hope that PNL will turn out to be a 'sweet spot' amongst such systems—fairly simple, yet usefully expressive and with good theoretic properties.

We will now briefly discuss some of the design decisions and design alternatives available to us in creating logics in the spirit of PNL.

Unknowns of name sort, and atoms. A swapping with unknowns, as in $(X\ Y)\cdot r$ where X and Y have a name sort ν, is not primitive syntax in PNL. This is *atoms as variables* as in [Gabbay and Cheney 2004] and [Pitts 2003], as opposed to the *atoms as constants* approach of nominal terms [Urban et al. 2004] which is inherited by this paper.

The axioms of nominal logic [Pitts 2003, Section 5] can be copied over to endow term-formers abs and swap (see below) with the right properties; since the logic of [Pitts 2003] is already a set of axioms, there is no harm in doing this also in PNL. Alternatively, we can 'promote' behaviour from atoms to unknowns: Suppose a sort α and a name sort Atm. Suppose $\mathit{sort}(X) = \alpha$ and $a \in \mathit{pmss}(X)$. Suppose a term-former $\mathsf{abs} : (\mathsf{Atm}, \alpha)[\mathsf{Atm}]\alpha$. The axiom $\forall X, Y.(X{\approx}a \Rightarrow \mathsf{abs}(X,Y){\approx}[a]Y)$ 'promotes' atoms-abstraction, to an abstraction by unknowns of sort atom (over terms of sort α). Similarly for a term-former $\mathsf{swap} : (\mathsf{Atm}, \mathsf{Atm}, \alpha)\alpha$.

The question phrased in semantic terms is as follows:

— Should every inhabitant of the denotation of a name sort ν to be referenced in PNL syntax by an atom?
— Should every inhabitant of the semantics of name sorts, to be referenced by a closed term?

The answers to these questions matter, but not for this paper. It is not unusual (indeed, it is very common) for there to be more elements in a type than there are closed terms. Furthermore, we have proved completeness in Subsection 3.4, so any extra elements in name sorts cannot 'make anything false'.

Yet it is reasonable to ask in future work whether we could exclude 'non-standard atoms' in the same way that for example we might try to exclude non-standard numbers from models of first-order theories of arithmetic. (Note that this is a general issue with a two-level syntax, and is not specific to PNL.)

We believe that this is possible. The idea is in the proof-theory for N from 'Fresh Logic' [Gabbay 2007a]; see $(\mathbf{Exhaust}\mathbb{A})$ in Figure 3, and Subsection 5.5.

Extending sorts. It would be a good idea to introduce sort-formers and polymorphism into the sorting system, so that e.g. we can conveniently axiomatise a substitution action on an infinite class of sorts. We see no difficulty in doing this—it is a definitional extension of what we already have.

Another interesting extension is to assume, for every sort α, an associated name sort ν_α. This would allow us to talk about 'the level 1 variables associated with α' in the same way that we can already talk about 'the level 2 variables of sort α'.

Design of permission sets. There is design freedom in the choice of permission sets. We briefly sketch some of the options.

We can have *more* permission sets. For instance, we can take $\mathbb{A}$ uncountable and permission sets all countably infinite sets. We can also add finite permission sets, enabling us e.g. to reason about properties that only hold of (level 1) closed terms, or terms with a finitely bounded number of free atoms.

We can have *fewer* permission sets. For instance we could take permission sets to be $\mathbb{A}^< \setminus A$, or $\pi{\cdot}\mathbb{A}^<$ for finite π.

We can also have *much* fewer permission sets—one, to be precise. PNL would work just as well if we took $\mathbb{A}^<$ as the single unique permission set. The effect of larger or smaller permission sets can then be obtained using permutations. For example if $pmss(X) = \mathbb{A}^<$ and $pmss(Y) = \mathit{shift}_\tau{\cdot}pmss(X)$ and $sort(X) = \tau = sort(Y)$ then the effect of $\forall Y.\mathsf{P}(Y)$ can be obtained by the logically equivalent $\forall X.\mathsf{P}(\mathit{shift}_\tau{\cdot}X)$. Using further shift-permutations and conjugation by finite permutations, any of the permission sets of Definition 2.7 can be obtained.

Note that we never want $\mathbb{A}$ to be a permission set. If we had that, then we would not be able to 'choose a fresh atom' and e.g. would be unable to α-convert a in $[a]X$ if $pmss(X) = \mathbb{A}$.

Taking a more abstract view, a natural generalisation of Definition 2.7 is an *equivariant nominal join semi-lattice that does not contain* $\top$. So specifically for sets of atoms, this means that if S and T are permission-sets then so are $\pi{\cdot}S$ and $S \cup T$, and $\mathbb{A}$ is not a permission set. To illustrate how this works, note that if S is a permission set and $\mathbb{A} \setminus S$ is finite then it easily follows using equivariance and sets unions that $\mathbb{A}$ is a permission set. So to insist that $\mathbb{A}$ is not a permission set, is really to insist that every permission set is coinfinite.

The design decision made in Definition 2.7 is simple, effective, and direct, and it allows us to express capture-avoidance conditions easily without complex 'emulations' involving shift.

PNL without shift. We can restrict PNL by dropping shift-permutations (but retaining permission sets as defined), yielding a logic that could be called *PNL without shift*. This is what was considered in the conference version of this paper [Dowek and Gabbay 2010].

This is less ergonomic, but in a certain sense it is just as powerful. It all depends on whether we want to be able to change our mind about a permission set in mid-derivation.

This is a similar issue as appears e.g. in the design of a sequent system, whether we allow weakening as an explicit sequent rule (so that we can weaken mid-derivation), or integrate weakening into the axiom rule (so we have to anticipate the other propositions needed in the sequent).

The isomorphism between $pmss(X)$ and $pmss(X) \cup \{a\}$ for $a \notin pmss(X)$ is explicit in full PNL and an implicit fact in PNL without shift.

Unification of permissive-nominal terms without shift was considered in [Dowek et al. 2010]. Subsequently to writing this paper, that theory was re-cast using shift [Gabbay 2012].

Note that nominal algebra satisfies an HSPA result whereas permissive-nominal algebra with shift satisfies an HSP result; details are elsewhere [Gabbay 2009; Gabbay 2012] but what is relevant to this discussion is that the extra expressivity which shift gives, can make a real, mathematically measurable, difference.

Implementing PNL. An implementation of PNL could follow the lines of a first-order theorem-prover, since the proof-rules in Figure 1 are so like those of first-order logic. The term-language would be richer and would include names and binding.

There would be many design choices, some of which we have touched on above: polymorphism in sorts; choice of permission sets (perhaps even adding variables to permission sets); whether or not to include shift; whether or not to exclude 'non-standard' atoms using an $(\mathbf{Exhaust}\mathbb{A})$ rule like that in Figure 3 of [Gabbay 2007a]; and so on.

Another point is how much of the infinity of permission sets we should expose to the user. To discuss this further, we must draw together several strands that have run through this paper from the beginning.

At the start of this paper we introduced permission sets, which guarantee infinite supplies of fresh atoms for every unknown. This culminated in Definition 2.30 with the permissive-nominal 'just quotient' α-equivalence, which is different from the notion of α-equivalence used in nominal terms in e.g. [Urban et al. 2004; Fernández and Gabbay 2007; Gabbay and

Mathijssen 2009]. In Subsection 8.2 we gave a sense in which PNL is obtained from nominal terms and nominal algebra by adding universal quantifiers while taking a limit of extending freshness contexts in the sense of nominal terms.

But then at the start of this section we noted that for any *concrete* derivation we only care about the finitely many atoms explicitly mentioned, thus for any concrete derivation we only care about finite freshness information after all.

So we have a choice, when implementing PNL, whether to *present* the user with $\mathbb{A}^<$ and $\mathbb{A}^>$ directly as we did in Definition 2.7, or to present a nominal terms syntax in which a (possibly but not necessarily finite) context of freshness assertions Δ is carried and may be extended as needed (nominal algebra does this using a freshness rule $(\mathbf{fr})$ [Gabbay and Mathijssen 2009]; an idea taken from [Gabbay 2007a]).

A specific disadvantage is that we would lose the permissive-nominal 'just quotient syntax' theory of α-equivalence used in Definition 2.30.

At the moment it seems unclear how much this matters from the point of view of an implementation. After all, in an implementation we will have a specific goal with specific and (finitely many) atoms for which the user has chosen specific names. So will the user even appreciate explicit access to an infinite stock of fresh atoms? Or, will the user prefer a freshness context to be extended as needed? Note that the implementation might need fresh names when α-renaming during resolution, so a resolution step might extend the freshness context with finitely but unboundedly many new names. An advantage of presenting $\mathbb{A}^>$ explicitly is that these names are honestly presented to the user from the start.

As with any logic, there are many ways to present it. Yet, the underlying mathematics remains essentially the same.

Summary. PNL addresses problems of mathematical specification with names and binding. It provides a first-order logic environment which allows us to formally express the 'informal meta-level', complete with names and binding. As such, the most exciting potential application of PNL is as a logical foundation—as a meta-theory for mathematics—intermediate in power between first- and higher-order logic. We believe that, perhaps with some fairly modest extensions, it would make an expressive, ergonomic, and practical alternative meta-language for mechanised mathematics.

REFERENCES

Abadi, M., Cardelli, L., Curien, P.-L., and Lévy, J.-J. 1991. Explicit substitutions. *Journal of Functional Programming 1*, 4, 375–416.

Beeson, M. 2004. Lambda logic. In *Second International Joint Conference on Automated Reasoning (IJCAR 2004)*. Lecture Notes in Computer Science Series, vol. 3097. Springer, 460–474.

Calvès, C. 2010. Complexity and implementation of nominal algorithms. Ph.D. thesis, King's College London.

Calvès, C. and Fernández, M. 2008. A polynomial nominal unification algorithm. *Theoretical Computer Science 403*, 285–306.

Cheney, J. 2005. A simpler proof theory for nominal logic. In *FoSSaCS*. Lecture Notes in Computer Science Series, vol. 3441. Springer, 379–394.

Dowek, G. and Gabbay, M. J. 2010. Permissive Nominal Logic. In *Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP 2010)*. ACM Press, 165–176.

Dowek, G., Gabbay, M. J., and Mulligan, D. P. 2009. Permissive Nominal Terms and their Unification. In *Proceedings of the 24th Italian Conference on Computational Logic (CILC'09)*.

Dowek, G., Gabbay, M. J., and Mulligan, D. P. 2010. Permissive Nominal Terms and their Unification: an infinite, co-infinite approach to nominal techniques (journal version). *Logic Journal of the IGPL 18*, 6, 769–822.

Dowek, G., Hardin, T., and Kirchner, C. 2002. Binding logic: Proofs and models. In *Proceedings of the 9th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2002)*. Springer, 130–144.

Dummett, M. 1977. *Elements of intuitionism* 1 Ed. Clarendon Press.

Farmer, W. M. 2008. The seven virtues of simple type theory. *Journal of Applied Logic 3*, 6, 267–286.

Fernández, M. and Gabbay, M. J. 2007. Nominal rewriting (journal version). *Information and Computation 205*, 6, 917–965.

Gabbay, M. 2011a. A proof-theoretic treatment of lambda-reduction with cut-elimination: lambda calculus as a logic programming language. *Journal of Symbolic Logic 76,* 2, 673–699.

Gabbay, M. J. 2001. A Theory of Inductive Definitions with alpha-Equivalence. Ph.D. thesis, University of Cambridge, UK.

Gabbay, M. J. 2005. A NEW calculus of contexts. In *Proceedings of the 7th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming (PPDP 2005)*. ACM Press, 94–105.

Gabbay, M. J. 2007a. Fresh Logic. *Journal of Applied Logic 5,* 2, 356–387.

Gabbay, M. J. 2007b. A General Mathematics of Names. *Information and Computation 205,* 7, 982–1011.

Gabbay, M. J. 2009. Nominal Algebra and the HSP Theorem. *Journal of Logic and Computation 19,* 2, 341–367.

Gabbay, M. J. 2011b. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic 17,* 2, 161–229.

Gabbay, M. J. 2011c. Two-level nominal sets and semantic nominal terms: an extension of nominal set theory for handling meta-variables. *Mathematical Structures in Computer Science 21*, 997–1033.

Gabbay, M. J. 2012. Nominal terms and nominal logics: from foundations to meta-mathematics. In *Handbook of Philosophical Logic*. Vol. 17. Kluwer.

Gabbay, M. J. and Cheney, J. 2004. A Sequent Calculus for Nominal Logic. In *Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS 2004)*. IEEE Computer Society, 139–148.

Gabbay, M. J. and Mathijssen, A. 2007. A Formal Calculus for Informal Equality with Binding. In *WoLLIC'07: 14th Workshop on Logic, Language, Information and Computation*. Lecture Notes in Computer Science Series, vol. 4576. Springer, 162–176.

Gabbay, M. J. and Mathijssen, A. 2008a. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing 20,* 4-5, 451–479.

Gabbay, M. J. and Mathijssen, A. 2008b. One-and-a-halfth-order Logic. *Journal of Logic and Computation 18,* 4, 521–562.

Gabbay, M. J. and Mathijssen, A. 2009. Nominal universal algebra: equational logic with names and binding. *Journal of Logic and Computation 19,* 6, 1455–1508.

Gabbay, M. J. and Mulligan, D. P. 2009a. Semantic nominal terms. In *Proceedings of the 2nd International Workshop on the Theory and Applications of Abstraction, Substitution and Naming (TAASN 2009)*.

Gabbay, M. J. and Mulligan, D. P. 2009b. 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, 64–73.

Gabbay, M. J. and Pitts, A. M. 2001. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing 13,* 3–5, 341–363.

Gacek, A., Miller, D., and Nadathur, G. 2008. Combining generic judgments with recursive definitions. In *Proceedings of the 23rd IEEE Symposium on Logic in Computer Science (LICS 2008)*. IEEE Computer Society Press, 33–44.

Levy, J. and Villaret, M. 2010. An efficient nominal unification algorithm. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA 2010)*. Leibniz International Proceedings in Informatics (LIPIcs) Series, vol. 6. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 209–226.

Miller, D. and Tiu, A. 2003. 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, 118–127.

Nanevski, A., Pfenning, F., and Pientka, B. 2008. Contextual modal type theory. *ACM Transactions on Computational Logic (TOCL) 9,* 3, 1–49.

Paulson, L. C. 1990. Isabelle: the next 700 theorem provers. In *Logic and Computer Science*, P. Odifreddi, Ed. Academic Press, 361–386.

Pitts, A. M. 2003. Nominal logic, a first order theory of names and binding. *Information and Computation 186,* 2, 165–193.

Shoenfield, J. 1967. *Mathematical Logic*. Addison-Wesley.

Tiu, A. 2007. A logic for reasoning about generic judgments. *Electronic Notes in Theoretical Computer Science 174,* 5, 3–18.

Urban, C., Pitts, A. M., and Gabbay, M. J. 2004. Nominal Unification. *Theoretical Computer Science 323,* 1–3, 473–497.