Quantifiers in logic and proof-search using permissive-nominal terms and sets

Murdoch J. Gabbay, Claus-Peter Wirth

Abstract

We investigate models of first-order logic designed to give semantics to reductive proof-search systems, with special attention to the so-called γ- and δ-rules controlling quantifiers. The key innovation is the use of syntax and semantics with (finitely-supported) *name-symmetry*, in the style of nominal techniques.

Keywords: Proof-search; nominal techniques; name-symmetry; higher-order logic; quantifiers

Contents

[☆]The authors are grateful for the detailed input of two anonymous referees and acknowledge the support of the Leverhulme Trust and SICSA.

1. Introduction

1.1. Background on reductive proof search

It is not enough to study proof in principle; we may also want to prove things in practice. This means creating notions of derivation that are succinct and susceptible to automation. Propositional logic is decidable; it is when we introduce variables and quantification $\forall$ and so come to predicate logic, that things become interesting.[1]

In brief, this paper presents a new syntax and semantics within which to understand the very special treatment of variables in proof-search. What makes the syntax and semantics special is that we base it on finitely-supported name-symmetry—that is, on nominal terms [UPG04, Gab12b] and nominal sets [GP01, Gab11a].

To see how this works, we need to consider proof-search and some of the difficulties it presents, with special attention to quantifiers. We concentrate on notions of proof-search designed to work backwards from the desired goal towards tautologies; this is often called *reductive*, *analytic*, or *backward* proof-search. Examples include sequent [Gen35], tableau [Fit96], matrix [Wal90], and indexed-formula-tree systems [Die11]. An excellent survey by Fitting is in [DGHP99, Chapter 1]; see also the clearly-written textbooks by Fitting [Fit96] and Smullyan [Smu68].

[1]This paper will consider classical logic, so $\exists$ is considered a dual of $\forall$. Nothing in this paper will depend on that, and a separate treatment of $\exists$ would be quite possible.

The quantifier $\forall$ typically generates a pair of rules. Depending on whether the reader thinks in sequent or tableau style, this pair may be called *left-intro* and *right-intro* rules, or $(\delta\forall)$ and $(\gamma\forall)$ rules (see [Fit96] or the discussion below) respectively. Intuitively:

1. left-intro/$(\gamma\forall)$ means "$\forall x.\phi$ implies $[r/x]\phi$ for any r", where here $[r/x]$ is the usual capture-avoiding substitution of r for x.[2] Call r a **witness**.
2. right-intro/$(\delta\forall)$ means "$[r/x]\phi$ for some *sufficiently generic* r implies $\forall x.\phi$".

Both of these rules are subtle.

The left-intro/$(\gamma\forall)$ intuition has obvious potential for branching and we want to delay the choice of witness as long as possible. This motivates us to introduce a distinct class of variables into our syntax, variously called *existential variables*, *meta-variables*, *proof-search variables*, or (in tableaux) *free variables*. These are variables whose instantiation transforms the proof as a whole, and this instantiation is typically delayed as long as possible. Free/existential variables[3] are written X in this paper and we use some new technology from nominal terms [UPG04] to handle them—but more on that later.

Concerning $(\delta\forall)$: what should 'sufficiently generic' mean? The standard rule, familiar in some form to most readers, is to take a fresh entity variously called a *fresh constant*, a *fresh variable*, an *eigenvariable*, or *parameter*. Here is the $(\delta^-\forall)$ rule of Figure 3, corresponding to Fitting's original free-variable[4] δ-rule [Fit90, Section 7.4] but written in sequent style:

$$\frac{\Gamma \vdash \psi, \Delta \quad (x \text{ fresh for } \Gamma, \Delta)}{\Gamma \vdash \forall x.\psi, \Delta}$$

But this rule is not efficient. A concrete discussion of why, with examples and references, is in Subsection 3.2.6. For this Introduction we merely note the intuition that x is *unnecessarily generic*; by just choosing x 'completely fresh' we do not record—and so cannot take advantage of—information present in the sequent about exactly what variables exist at the moment x was created.

An industry exists in devising rules to prove $\forall a.\phi$ more efficiently:

- Fitting's free-variable δ-rule from the first edition of his book [Fit90, Section 7.4].
- Its 'liberalised' version δ^+, introduced by Hähnle in [HS94] and also treated in the second edition of Fitting's book [Fit96, Section 7.4].
- The rule δ^{++} from [BHS93] (which also includes a particularly nice exposition).
- The rule δ^* [BF95]. This had some errors: one was corrected at the TABLEAUX conference by the authors; another was treated later in [CNA07, Subsection 5.3].

[2]Like this: $(\forall y.\mathsf{P}(y,x))[y/x] = \forall y'.\mathsf{P}(y',y)$. For the specific definitions on the syntax of this paper, see Subsection 2.2 and in particular Figure 2.

[3]There is a potentially confusing terminology clash here. The first author was brought up on sequents, λ-calculi, and semantics, and calls X an existential variable. The second author grew up in the theorem proving community where Smullyan's α, β, γ, and δ provides a *lingua franca*, in which X is called a free variable. This clashes with the notion of 'free variables of' of a predicate or λ-term.

Since we are trying to straddle two communities, we give both possible names where there is any possibility of doubt. In any case, wherever the reader sees X in this paper, this is a variable we try to instantiate to make a proof work out.

[4]In our notation in this paper, based on variable-conditions and permission sets instead of Skolemisation, there is no obvious difference from the rule for a calculus without free variables [Fit90, Section 6.1], because our notion of 'free atoms of' includes the permission sets of the free/existential variables in Γ, Δ.

- The rule δ^{*^*} [CA00]. This had an error which was discussed and corrected in [CNA07, Subsection 5.4].
- The rule δ^{ε} [GA99, Section 4.1].

Efficiency gains arise from fine-grained analysis of name-generation and name-dependency. Generally speaking, the more parsimoniously a rule can generate names and record dependencies, the more efficiently it can be implemented.

These speedups can be significant, and the literature above is computationally well-motivated. The (δ^{+}) of [HS94] allows at least an exponential speedup relative to (δ^{-}) [Fit96], and the (δ^{+^+}) of [BHS93] allows a further exponential speedup relative to (δ^{+}). For more discussion see [BF95, Section 3].

Names are also what nominal techniques, developed by the first author and others, were created to study. So in this paper we turn our nominal toolkit to model the variable behavior typically seen in reductive proof-search.

Just to be clear: this paper is not about creating a more efficient proof-search system and we make no particular claims to efficiency. However, the syntax and semantics are new and throw a new light on the problem of variables in proof-search, and their semantics.[5]

Whether this will lead to better algorithms is an open question, but it may well lead to shorter and better mathematical proofs (and perhaps even contribute to avoiding mistakes in algorithms).

We will use two nominal tools: *permissive-nominal terms* from [DGM10] (a simplification of the *nominal* terms from [UPG04]) and the *nominal sets* semantics from [GP01]. For surveys of the applications of permissive-nominal terms and nominal sets to logic and specification in general, see two survey papers [Gab11a, Gab12b].

This paper will be self-contained; for background theory we include pointers to the literature. We will consider a non-Skolemising version of Fitting's original free-variable δ-rule (see (δ^{-}) in Figure 3) and a novel (δ^{x}) rule (see Figure 5). Both rules are expressed using a nominal syntax.

1.2. *Summary of the paper*

… in a paragraph: rule $(\delta^{-}\forall)$ in Figure 3 is analogous to the usual $\forall$-right rule. It generates a fresh symbol (*atom*, in this paper). As mentioned above, it turns out that this rule (in combination with the other rules) is inefficient for practical proof-search. Rule $(\delta^{\mathsf{x}}\forall)$ in Figure 5 is an example of a 'liberalised δ-rule', of which many have appeared in the literature as cited above. Rule $(\delta^{\mathsf{x}}\forall)$ introduces another kind of variable X. In fact X is *existential*, in the sense that X can be instantiated globally throughout a derivation (see e.g. Subsection 3.2.2). However, (δ^{x}) also adds X to a *maximisation condition* $X{\uparrow}[a]\phi$ (Definition 4.5). Maximisation conditions allow us to restrict the domain over which X can range. Using negation we can (see Subsection 6.2) restrict X to range over elements that try to falsify $\phi[a{\mapsto}X]$, giving X a universal flavour with respect to $\phi[a{\mapsto}X]$.

[5]This paper is not about *Nominal Isabelle* [Urb08], which is an implementation of nominal abstract syntax from [GP99, GP01] in Higher-Order Logic. This paper is about proof-search using nominal terms (which is not the same thing as nominal abstract syntax) with semantics in nominal sets. What these applications have in common is a requirement to handle rich name-based behaviour, which in both cases is addressed with nominal techniques.

We will see in this paper that nominal syntax and semantics can simplify material; they can also lead to new ideas. For instance, the contents of this paper are a simplification of [Wir04], which is 102 pages long; see also [Wir11].[6] This paper is not just a repeat of the previous work and the nominal terms syntax, the maximising hypersequent syntax and rewrites, and nominal semantics, are different, and the semantics in particular is very different indeed. A qualitative change in complexity is apparent: the nominal syntax, semantics, and presentation are simpler.

Conversely, through a study of reductive proof-search we obtain some new and interesting 'nominal' constructions. These include:

- Application of nominal unknowns X from [UPG04, DGM10, Gab12b] directly to proof-search. In nominal unification from [UPG04], unknowns had an existential flavour and ranged over nominal terms. This paper takes the idea much further, making unknowns range over witnesses to higher-order logic predicates. We suspect that this far from exhausts the expressivity and flexibility of permissive-nominal terms, and the application in this paper may be prototypical for many other applications not yet considered.
  For compatibility with the tableau literature, we call unknowns *(free) variables* in this paper.
- Interpretation of existential variables, *and* of universal variables by using maximisation conditions in Section 4. To model what happens in the tableau systems, we use a mechanism (*maximisation condition*) which can make an existential unknown (*free variable*) play the role of a universal parameter for a specific predicate. This is entirely new, from a nominal perspective.
- The construction of the functional model in Subsection 7.2. This construction has been known to the first author since 2006, and was first used in [DG10, DG12a]. See also [DG12b].

1.3. *Map of the paper*

The section and subsection titles should be fairly self-explanatory but it might help to gather up some exposition in an itemised list:

- Subsection 2.1 establishes some of the basic pieces needed for the rest of the paper. This only makes sense in the context of what we will then do with it.
- Nominal terms are discussed in Subsection 2.2. By then we can write down formulae (terms of type o). The term language is just that of higher-order logic, but enriched with proof-search variables which we write X and which are, technically, unknowns from nominal terms.
- In Subsection 2.3 we consider 'free atoms of' and α-equivalence for atoms a. This is off-the-shelf from the theory of permissive-nominal terms, but if the reader has not seen this before it will be new because the nominal terms theory of α-equivalence is actually rather powerful, being based on permutations (symmetry).

[6] [Wir11] simplifies [Wir04] using 'nominal' ideas, and documents this process in successive revisions (the authors are grateful to a SICSA grant which supported this work). The current paper follows a similar project but from the other direction by adding ideas from [Wir04] to nominal syntax and semantics.

- In Subsection 2.4 we introduce instantiation of proof-search variables X. Again, this is off-the-shelf from permissive-nominal terms, but again the uninitiated reader will have to pay attention. We illustrate the definitions with plenty of examples.
- We now have enough to define sequents in Subsection 3.1, followed by detailed examples in Subsection 3.2.
- Section 4 mirrors Section 3 but now we treat *maximising hypersequents*. It is probably fair to say that this is where the material starts to become intrinsically difficult rather than just potentially unfamiliar. Again, we give plenty of examples.
- Section 5 builds the denotation needed to give nominal semantics to the material up to this point: Subsection 5.1 introduces necessary background on nominal sets. The difference from ordinary sets is that atoms and their permutations are taken as primitive so that atoms and binding in the syntax get translated directly to corresponding structure on the nominal sets. Subsection 5.2 takes this further to model the full structure of terms.
- By this point, in Section 6 we have everything we need to interpret sequents and prove soundness of the proof-search 'algorithms' from Sections 3 and 4. As is often the case, with the right definitions in place this is not actually particularly hard. But it is the crux of the paper.
- Section 7 builds two models by hand—one model out of syntax, another model out of 'ordinary' sets. The interest here is to prove that non-trivial models of the preceding material exist. We can also adapt other work to give yet more models, but the models we consider here are sufficient to make the point and to illustrate the nominal semantics.
- Section 8 considers choice, partly because choice is interesting, partly because much other literature appeals to it, and partly because nominal techniques have a somewhat undeserved reputation for causing difficulty with choice. So, we verify explicitly and by hand that it works, and how it works.

2. Nominal terms

2.1. *Atoms, variables, types, and permutations*

We describe the elements from which we build our syntax. This is background material; the reader who wants to understand through examples can look at Example 2.18 (example terms) and Subsection 3.2 (example derivations).

Broadly speaking: *types* are the simple types of higher-order logic with base types for data; *atoms* a are universal variables; *variables* X are existential variables; *constants* are as usual; and *permutations* are as standard for nominal techniques and represent the symmetry implicit in α-conversion and the 'freshness side-conditions' typical of reasoning with α-conversion.

Extensive discussion of these definitions (and why they are generally useful) can be found in [GP01, UPG04, Che05, BU07, Gab11a, Gab12b].

Definition 2.1. Fix a set of **base types** τ. Define **types** α inductively by:

$$\alpha ::= o \mid \tau \mid \alpha \to \alpha$$

Remark 2.2. In Definition 2.1 o will be a type for *truth-values* (see the distinguished constants in Definition 2.8). The base types τ are for data, like numbers $\mathbb{N}$. $\alpha \to \beta$ is a *function type*.

For now these types are just formal elements of an inductively defined set. It is up to us to give them meaning consistent with these intuitions; see for instance the typing rules for terms in Figure 1, and the map from α to a nominal set $[\![\alpha]\!]$ in Definition 5.12.

A minimal interesting set of types would assume just types o (truth-values), i (individuals, to be populated by our favourite data), and function-types. The reader who prefers their types minimal can assume that this is all we have, and no harm will come of it.

Definition 2.3. For each type α fix a disjoint countably infinite set $\mathbb{A}_\alpha$ of **atoms** of that type and define $\mathbb{A} = \bigcup_\alpha \mathbb{A}_\alpha$.

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

Write $\mathit{type}(a)$ for the type of a, so that $a \in \mathbb{A}_{\mathit{type}(a)}$.

Definition 2.4. Let S, T, and U range over finite sets of atoms, not necessarily all of the same type. So for instance if $\mathit{type}(a) = \alpha$ and $\mathit{type}(b) = \beta$ then S might be equal to $\{a, b\}$.

For each S and type α fix a disjoint countably infinite set of **variables** $\mathbb{V}_{\alpha,S}$ and define $\mathbb{V} = \bigcup_{\alpha,S} \mathbb{V}_{\alpha,S}$. X, Y, Z will range over distinct variables.[7]

If $X \in \mathbb{V}_{\alpha,S}$ write $\mathit{pms}(X)$ for S and call this the **permission set** of X, and write $\mathit{type}(X)$ for α and call this the **type** of X.

Remark 2.5. We pause to discuss atoms, variables, and permission sets specifically with regards to the literature on proof-search.

- Atoms a are universal variables; in proof-search $\forall a.\phi$ gets transformed to ϕ where a is fresh. See $(\delta^-\forall)$ in Figure 3.
  In the literature this kind of structure is also called a *parameter* or *eigenvariable*. This is the kind of entity that is introduced by the usual $(\forall\mathbf{R})$ rules of sequent systems.
- Variables X are existential variables, 'Skolemised' over the atoms in their permission set; in proof-search $\neg\forall a.\phi$ gets transformed to $\neg\phi[a{\mapsto}X]$. See $(\gamma\neg\forall)$ in Figure 3.
  This corresponds to the *dummy* variable of [Pra60, Pra83] and [Kan63] and to the *free variable* of [Fit96] and footnote 11 of [Pra60], to the *meta-variable* of planning and constraint solving, and to the *free γ-variable* of [Wir04].

Atoms enter nominal techniques as set-theoretic *urelemente* in Fraenkel-Mostowski set theory; for detailed historical notes see [Gab11a] (especially Remark 2.22). They were originally used to represent atomic variable symbols of abstract syntax with binding [GP01]. Atoms are 'names of which nothing else is known'—just like 'parameters' in proof-search. Also like parameters, atoms can be generated fresh (cf. the И-quantifier [GP01, Gab11a]). So structurally, atoms in Fraenkel-Mostowski sets and parameters in proof-search are similar, and using one to model the other makes sense.

One feature that our system has over parameters, eigenvariables, dummy variables, and so on as cited above, is that permissive-nominal terms are embedded in a broader semantic theory of *nominal techniques* [Gab11a, Gab12b]. This broader theory is the tool that we will use to build our semantics for the quantifier rules and proof-search.

[7] In [UPG04, DGM10] variables are called *unknowns*.

Permutations in nominal techniques are how we express that atoms are symmetric up to permutative renaming. Permutations will feature in the syntax (Definition 2.10) and also in the models (Subsection 5.1):

DEFINITION 2.6. A **permutation** is a bijection on $\mathbb{A}$ such that:

- $a \in \mathbb{A}_\alpha$ if and only if $\pi(a) \in \mathbb{A}_\alpha$ (so π **preserves types**).
- $nontriv(\pi) = \{a \mid \pi(a) \neq a\}$ is finite.

NOTATION 2.7. The following notation may be useful:

- Write id for the **identity** permutation such that $id(a) = a$ for all a.
- Write $\pi' \circ \pi$ for composition, so that $(\pi' \circ \pi)(a) = \pi'(\pi(a))$, and write $\pi^{\text{-}1}$ for inverse, so that $\pi^{\text{-}1} \circ \pi = id = \pi \circ \pi^{\text{-}1}$.
- Suppose $a, b \in \mathbb{A}_\tau$ for some τ. Then write $(a\ b)$ for the **swapping**[8] mapping a to b, b to a, and all other c to themselves, and take $(a\ a) = id$. As a matter of convenience, wherever we write a swapping this comes with an assumption that the two atoms have the same type.

2.2. *Terms and types of terms*

The terms here are a species of permissive-nominal term [DGM10] which simplify nominal terms from [UPG04] by eliminating freshness contexts. For intuitions of terms, see Example 2.18.

DEFINITION 2.8. Fix a set of **constants**, to each of which is associated a type. f, g, h will range over distinct constants; we write $type(\mathsf{f})$ for the type of f.

Assume the following distinguished constants:

$$\top : o \qquad \neg : o \to o \qquad \wedge : o \to o \to o \qquad \forall_\alpha : (\alpha \to o) \to o$$

REMARK 2.9. At the moment the distinguished constants above are just formal symbols, which become part of the syntax of terms in Definition 2.10. What makes them 'distinguished' is how they are interpreted:

- The rewrite rules in Figure 3 give the distinguished constants operational significance in a proof-search framework. So for instance, the rules $(\alpha\neg\wedge)$ and $(\beta\wedge)$ make $\wedge$ behave like a conjunction.
- The nominal algebra axioms in Figure 8 given the distinguished constants significance in a preorder. The axioms make, for instance, $\wedge$ behave like a least upper bound in the preorder.

(Figures 3 and 8 use syntactic sugar from Notation 2.15.)

So the significance of Definition 2.8 is that Definition 2.10 is not just a simply-typed λ-calculus with existential variables (the X)—it is a syntax for higher-order logic with nominal variables.

[8]This terminology is consistent with [GP01]. This is also called a *transposition*.

DEFINITION 2.10. Define **(permissive-nominal) terms** inductively by:

$$\boxed{r ::= a \mid \pi{\cdot}X \mid \mathsf{f} \mid r'r \mid [a]r}$$

Above, $\pi{\cdot}X$ is a formal pair of a permutation π and a variable X; we discuss this in Remark 2.11.

We let r, s, t range over terms. As is standard, r_1r_2 associates to the left (so $r_1r_2r_3$ means $(r_1r_2)r_3$).

REMARK 2.11. We briefly run through the parts of Definition 2.10.

- a is an atom and is discussed in Remark 2.5.
- X is a *free* or *existential* variable. As standard for nominal terms [UPG04] we do not inject X directly into the term syntax; X becomes a term only together with a permutation π. We call the pair $\pi{\cdot}X$ a **moderated** variable.
  We may write $id{\cdot}X$ just as X (Notation 2.15), but there remains a formal distinction between 'X the variable' and '$id{\cdot}X$ the permissive-nominal term'.
  This moderation is important for the permissive-nominal terms theory of α-equivalence; see Remark 2.14 and Example 2.24.
- f is a **constant**; examples of constants are in Definition 2.8.
- $r'r$ is an **application**, and
- $[a]r$ is an **abstraction**.

Example terms follow shortly in Example 2.18, after we have built a typing relation. First, we discuss why atoms are not the same thing as constant symbols:

REMARK 2.12 (Atoms are not constants). Atoms are not constant symbols. The best way to understand this is in the models; models are subject to a symmetry group action permuting atoms, and atoms get translated from the syntax to the denotation in specific ways. In contrast, there are fewer restrictions on how constants are interpreted.

So note in Definition 5.27 how atoms, permutations, and atoms-abstractions get translated directly to corresponding semantic notions which are assumed as primitive in the model. A constant symbol is just interpreted by some element of the model. No special symmetry properties are assumed.

The term $[a]r$ only makes sense because the interpretation of a in the model is specific in such a way that atoms-abstraction $[a]$- has a specific meaning. It makes no sense to talk about $[\mathsf{f}]r$ because there is no restriction on how f is interpreted.

REMARK 2.13 (Permissive-nominal syntax). For the reader familiar with nominal terms, note that the syntax of Definition 2.10 is *permissive-nominal* [DGM09b, DGM09a, DGM10]. *Freshness assertions* like $a\#X$ are replaced with permission sets (Definition 2.4) and assertions like $a \notin pms(X)$.

In this paper permission sets happen to be finite.

In Definition 2.10 we also assume a pairing operator rs. We call this *application*, and Figure 2 defines a relation $=_{\alpha\beta\eta}$ which gives it $\beta\eta$ behaviour. Thus, atoms-abstraction and application will behave like λ-abstraction and application (with existential variables), up to $=_{\alpha\beta\eta}$.

This is orthogonal to the usual theory of nominal terms and permissive-nominal unification. We *also* have a notion of α-equivalence $=_\alpha$ and we are free to ignore the $\beta\eta$

axioms and $=_{\alpha\beta\eta}$—and the type system, if we like—in which case the permissive-nominal syntax up to $=_\alpha$ is standard.

So in this paper we do not consider unification up to $=_\alpha$, but if we wished to (perhaps for an implementation) then the permissive-nominal unification algorithms could be used off-the-shelf.

Remark 2.14 (Why moderated variables?). For the reader unfamiliar with nominal terms, we briefly and intuitively discuss why variables need to be moderated as $\pi{\cdot}X$.

It is not unusual to see quotes of the following form in informal practice:

> *Consider a λ-term $\lambda x.t$ and a variable y not free in t. Then $\lambda x.t$ is α-equivalent to $\lambda y.(t[y/x])$.*

In a nutshell, (permissive-)nominal terms give a formal syntax to the informal quote above. The role of t (a meta-variable ranging over terms) is played by the variable X (called an *unknown* in [UPG04, DGM10]). The role of x and y (object-level variables) is played by atoms a, b (actually identified with *urelemente* in Fraenkel-Mostowski set theory—but we do not have to care about that).

But that is not enough. We also need to model $t[y/x]$. That is handled by the moderation $(b\ a){\cdot}X$ where $b \notin pms(X)$. The permission set describes the free atoms of terms for which X may be instantiated, so if b is not permitted in X then $(b\ a)$ has the value of "replace a in whatever term X becomes, with b", which models $t[y/x]$ which has the value "replace x in whatever term t is, with y".

Now the reader might ask why we use permutations $(b\ a)$ instead of atoms-renamings $[b/a]$. The answer is that permutations form a *group*, are invertible, and have better mathematical and computational properties. More on this in the Conclusions in Subsection 9.2.

Given that, we can model the quote above as follows:

> *Consider a variable X with $b \notin pms(X)$. Then $[a]id{\cdot}X =_\alpha [b](b\ a){\cdot}X$.*

That is why we need moderated variables. See Examples 2.18 and 2.24, in particular part 2 of Example 2.24.

Notation 2.15. We may use syntactic sugar as follows:

- We may write $\wedge rs$ as $r \wedge s$, and $\forall_\alpha([a]r)$ as $\forall a.r$, and $\mathsf{f}r$ as $\mathsf{f}(r)$.
- We may write $id{\cdot}X$ just as X.
- We may write $([a]r)s$ as $r[a{\mapsto}s]$.
- We may write $r \vee s$ for $\neg(\neg r \wedge \neg s)$ and $r \supset s$ for $\neg r \vee s$ and $\exists a.r$ for $\neg(\forall_\alpha[a]\neg r)$.

In general, we will be very lax about $\vee$, $\supset$, and $\exists$. Technically, they are not primitive, but this is purely a design choice to reduce the number of cases in proofs; we do not really care since it will always be clear and standard how to expand to $\bot$, $\wedge$, and $\forall$.

Definition 2.16. Define a **typing relation** $r : \alpha$ inductively by the rules in Figure 1.

Here and elsewhere, we put side-conditions of derivation rules in brackets.

Notation 2.17. If $r : o$ we may call r a **formula**. We let ϕ, ψ, and χ range over formulae.

It is clear from the typing rules in Figure 1 that $a : \alpha$ (the term a) has type α if and only if $type(a) = \alpha$, and similarly $\mathsf{f} : \alpha$ if and only if $type(\mathsf{f}) = \alpha$ and $id{\cdot}X : \alpha$ (or $X : \alpha$, using our syntactic sugar of writing $id{\cdot}X$ as just X) if and only if $type(X) = \alpha$. We may therefore be lax about the distinction between '$type(x) = \alpha$' and '$x : \alpha$', where x is an atom, constant, or variable. We will usually prefer $x : \alpha$, since it is shorter.

$$\frac{(type(a) = \alpha)}{a : \alpha} \qquad \frac{(type(X) = \alpha)}{\pi{\cdot}X : \alpha} \qquad \frac{(type(\mathsf{f}) = \alpha)}{\mathsf{f} : \alpha}$$

$$\frac{r : \beta \to \alpha \quad s : \beta}{rs : \alpha} \qquad \frac{r : \alpha \quad a : \alpha'}{[a]r : \alpha' \to \alpha}$$

Figure 1: Typing rules for terms

Example 2.18. Suppose a base type τ and atoms a and b of type τ. Suppose constants $\mathsf{P} : \tau{\to}o$ and $\mathsf{Q} : \tau{\to}o$. Finally, suppose a variable $X : o$ and an atom $c : o$. Then here are some example terms:

- $a : \tau$. Intuitively, this term represents 'an assumption/datum called a, of type τ'.
- $\forall a.(\mathsf{P}(a) \land \mathsf{Q}(a)) : o$. Intuitively, this formula represents 'for all a, $P(a)$ and $Q(a)$'.
- $\forall a.c : o$. Intuitively, this formula represents 'for all a, c' (which is logically equivalent to just 'c').
- $\forall a.X : o$. Intuitively, this formula represents 'for all a, the goal X holds'. This X has an existential flavour, and we are supposed to instantiate it. Now $\forall a.X$ is not equivalent to X because, as we shall see, instantiation of X (unlike substitution on a or c) may be *capturing*. Only the atoms in $pms(X)$ may be captured. To see this happen in example derivations, see Subsection 3.2.

2.3. *Free atoms and variables, alpha- and beta-equivalence*

Definition 2.19. If A is a set of atoms define $\pi{\cdot}A = \{\pi(a) \mid a \in A\}$.

Definition 2.20. Define a **permutation action** $\pi{\cdot}r$ and $fa(r)$ the **free atoms** of r and $fv(r)$ the **free variables** of r by:

$$\begin{array}{rclrclrcl}
\pi{\cdot}a &=& \pi(a) & \pi{\cdot}(\pi'{\cdot}X) &=& (\pi \circ \pi'){\cdot}X & \pi{\cdot}\mathsf{f} &=& \mathsf{f} \\
\pi{\cdot}(r'r) &=& (\pi{\cdot}r')(\pi{\cdot}r) & \pi{\cdot}[a]r &=& [\pi(a)]\pi{\cdot}r & && \\
\\
fa(a) &=& \{a\} & fa(\pi{\cdot}X) &=& \pi{\cdot}pms(X) & fa(\mathsf{f}) &=& \varnothing \\
fa(r'r) &=& fa(r') \cup fa(r) & fa([a]r) &=& fa(r) \setminus \{a\} & && \\
\\
fv(a) &=& \varnothing & fv(\pi{\cdot}X) &=& \{X\} & fv(\mathsf{f}) &=& \varnothing \\
fv(r'r) &=& fv(r') \cup fv(r) & fv([a]r) &=& fv(r) & &&
\end{array}$$

Lemma 2.21 notes some easy basic properties of permissive-nominal terms:

Lemma 2.21. 1. *$\pi{\cdot}(\pi'{\cdot}r) = (\pi \circ \pi'){\cdot}r$ and $id{\cdot}r = r$ (the permutation action is a group action).*
2. *$fa(\pi{\cdot}r) = \pi{\cdot}fa(r)$ (*free atoms *commutes with the permutation action, or in the terminology of part 2 of Definition 5.10, it is* equivariant*).*

Proof. By routine inductions on r. For details see [DGM09a, Lemmas 15 and 16].[9] □

[9]Permissive-nominal terms were presented in [DGM09b] (conference paper), [DGM09a] (technical report), and [DGM10] (journal paper). The conference and journal papers versions probably have the better exposition, but the technical report includes exhaustive proofs.

(**stx**α)	$a, b \notin fa(r) \Rightarrow$	$(b\ a)\cdot r$	$=_\alpha$	r
(**stxa**)		$a[a{\mapsto}t]$	$=_{\alpha\beta\eta}$	t
(**stx**$\#$)	$a \notin fa(r) \Rightarrow$	$r[a{\mapsto}t]$	$=_{\alpha\beta\eta}$	r
(**stxapp**)		$(r'r)[a{\mapsto}t]$	$=_{\alpha\beta\eta}$	$(r'[a{\mapsto}t])(r[a{\mapsto}t])$
(**stx**$[]$)	$c \notin fa(t) \Rightarrow$	$([c]r)[a{\mapsto}t]$	$=_{\alpha\beta\eta}$	$[c](r[a{\mapsto}t])$
(**stxid**)		$r[a{\mapsto}a]$	$=_{\alpha\beta\eta}$	r
(**stx**η)	$a \notin fa(r) \Rightarrow$	$[a](ra)$	$=_{\alpha\beta\eta}$	r

Figure 2: Rules for $\alpha\beta\eta$-equivalence

Definition 2.22. A **congruence** is a relation R on terms such that $r\ R\ r'$ and $s\ R\ s'$ imply $[a]r\ R\ [a]r'$ and $rs\ R\ r's'$.
Let α**-equivalence** be the least congruence $=_\alpha$ that includes (**stx**α) from Figure 2.
Let $\alpha\beta\eta$**-equivalence** be the least congruence that includes (**stx**α), (**stxa**) to (**stxid**), and (**stx**η) from Figure 2.

Lemma 2.23. *α-equivalence can be characterised as the least relation such that:*

- *$a =_\alpha a$ and* $\mathsf{f} =_\alpha \mathsf{f}$.
- *If $\pi(a) = \pi'(a)$ for every $a \in pms(X)$ then $\pi\cdot X =_\alpha \pi'\cdot X$.*
- *If $r =_\alpha r'$ and $s =_\alpha s'$ then $rs =_\alpha r's'$ and $[a]r =_\alpha [a]r'$.*
- *If $(b\ a)\cdot r =_\alpha s$ and $b \notin fa(r)$ then $[a]r =_\alpha [b]s$.*

The characterisation of $=_\alpha$ in Definition 2.22 follows [GM07, GM09a]. Lemma 2.23 is modelled on the equivalent definition from [UPG03, UPG04].

Example 2.24. 1. $[a]a =_\alpha [b]b$.
2. If $b \notin pms(X)$ then $[a]X =_\alpha [b](b\ a)\cdot X$.
3. If $a, b \in pms(X)$ then ~~$[a]X =_\alpha [b](b\ a)\cdot X$~~ is *not* the case.
4. $([a]a)t =_{\alpha\beta\eta} t$ and $([a]b)t =_{\alpha\beta\eta} b$. This is just (**stxa**) and (**stx**$\#$).
5. However, if $a \in pms(X)$ then ~~$([a]X)t =_{\alpha\beta\eta} X$~~ is *not* the case. Even though a does not occur syntactically in X, it is considered to be 'free' in X if $a \in pms(X)$. This is because X might be *instantiated* to a; see Subsection 2.4.

Proposition 2.25. *If $r : \alpha$ and $r =_{\alpha\beta\eta} r'$ then $r' : \alpha$.*

2.4. *Variable instantiation*

The variables X and Y are variables in our syntax and they can be instantiated. However, they are associated with a *permission set* which as we shall see in Subsection 3.2.5 is important for soundness. We briefly consider the theory of instantiation for variables:

Definition 2.26. An **instantiation** θ is a function from variables to terms such that:

- If $X : \alpha$ then $\theta(X) : \alpha$.
- $fa(\theta(X)) \subseteq pms(X)$.

If $r : \mathit{type}(X)$ and $\mathit{fa}(r) \subseteq \mathit{pms}(X)$ then let $[X{:=}r]$ be the instantiation mapping X to r and all other Y to $\mathit{id}{\cdot}Y$.

DEFINITION 2.27. Define an **instantiation action** on terms by:

$$\begin{array}{ccc} a\theta = a & (\pi{\cdot}X)\theta = \pi{\cdot}(\theta(X)) & \mathsf{f}\theta = \mathsf{f} \\ (r'r)\theta = r\theta(r'\theta) & ([a]r)\theta = [a](r\theta) & \end{array}$$

REMARK 2.28. It is important to realise that instantiation is capturing for atoms in $\mathit{pms}(X)$. For instance, suppose $a \in \mathit{pms}(X)$ and $b \notin \mathit{pms}(X)$. Then:

- $([a]X)[X{:=}a] = [a]a$. The a in the instantiation $[X{:=}a]$ has been captured by the $[a]X$.
- $([b]X)[X{:=}a] = [b]a$.
- It is impossible to even ask what $([b]X)[X{:=}b]$ is equal to because $[X{:=}b]$ is not an instantiation, since $b \notin \mathit{pms}(X)$. So $b \notin \mathit{pms}(X)$ cannot be captured by an instantiation $[X{:=}b]$, because that instantiation does not exist.
- Also, recall from Example 2.24 that $[b](b\ a){\cdot}X =_\alpha [a]X$. By construction,

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

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

We can only substitute for X a term whose free atoms are in the permission set of X. For instance, if $\mathit{pms}(X) = \{a\}$ and $a : \alpha$ and $b : \alpha$ and $X : \alpha$ then $[X{:=}a]$ is an instantiation and $[X{:=}b]$ is not.

Think of $X : \alpha$ as ranging over terms of type α with free atoms in $\mathit{pms}(X)$. This latter restriction matters for soundness: see Subsection 3.2.5.

Proposition 2.29 notes some familiar properties of permissive-nominal terms:

PROPOSITION 2.29.
1. *$\mathit{fa}(r\theta) \subseteq \mathit{fa}(r)$.*
2. *$\pi{\cdot}(r\theta) = (\pi{\cdot}r)\theta$.*
3. *If $r =_\alpha s$ then $\pi{\cdot}r =_\alpha \pi{\cdot}s$ and $r\theta =_\alpha s\theta$.*
4. *If $r =_{\alpha\beta\eta} s$ then $\pi{\cdot}r =_{\alpha\beta\eta} \pi{\cdot}s$ and $r\theta =_{\alpha\beta\eta} s\theta$.*

Proof.
1. By a routine induction on r. The interesting base case is $\pi{\cdot}X$, where we use our assumpion that $\mathit{fa}(\theta(X)) \subseteq \mathit{pms}(X)$ and part 2 of Lemma 2.21.
2. By a routine induction on r. The interesting base case is $\pi'{\cdot}X$, where we use part 1 of Lemma 2.21.
3. By inductions on the derivation of $r =_\alpha s$. The interesting case is rule $(\mathbf{stx}\alpha)$. We use parts 1 and 2 of this result, and Lemma 2.21.
4. By an easy induction on the derivation of $r =_{\alpha\beta\eta} s$, as for $=_\alpha$.

□

3. Sequents and hypersequents

We now use *hypersequents* (Definition 3.1) to define a rewrite system for proof-search.

The word *hypersequent* follows Avron [Avr96, Avr91], and our hypersequents are essentially identical to the ones he considered—see the first definition of Section 2 of [Avr91]—though the use we put hypersequents to here is different.[10]

A hypersequent is intuitively a conjunction of disjunctions of formulae. In Figures 3 and 4 we define a rewrite system which, intuitively, reduces hypersequents to simpler hypersequents.

If we squint, then we recognise the rules in Figures 3 and 4 as the usual sequent rules [GTL89, Subsection 5.1.5]. They are just written in a 'disjunctive' instead of a 'sequent' form: $(\gamma\neg\forall)$ corresponds to $(\forall\mathbf{L})$; $(\delta^{-}\forall)$ corresponds to $(\forall\mathbf{R})$; $(\alpha\neg\wedge)$ corresponds to $(\wedge\mathbf{L})$; $(\beta\wedge)$ corresponds to $(\wedge\mathbf{R})$; $(\alpha\mathbf{EM})$ corresponds to the axiom rule; $(\alpha\neg\neg)$ to the negation rules; and so on. The difference is that this presentation is somewhat better for the mechanics of proof-search.

If we squint again, then the hypersequent rewrite rules are an abstract rendering of a *tableau* derivation system (specifically, a set-labelled tableau system). For more on tableaux see [Smu68] or [DGHP99].

So Definition 3.1 seems to be quite a nice compromise between the worlds of sequents and tableaux, and the reader can view our hypersequent rewrite system as either a slightly mangled sequent system, or a rather inefficiently-presented set-labelled tableau system, depending on which set of ideas is most familiar, and no harm will come of it. Example derivations are given in Subsection 3.2.

3.1. *The basic definition*

Definition 3.1.
- A **sequent** Φ is a set of formulae $\phi_1 \vee \ldots \vee \phi_n$. Φ will range over sequents.
- A **hypersequent** $\mathcal{H}$ is a set of sequents $\Phi_1 \wedge \ldots \wedge \Phi_m$. $\mathcal{H}$ will range over hypersequents.

The symbol $\wedge$ looks like $\land$ and $\vee$ looks like $\lor$, and indeed they are interpreted equally (Definition 6.1). However, $\wedge$ and $\vee$ are part of the structure of hypersequents, not part of the structure of formulae.[11]

In Definition 3.3 we give rewrite/derivation rules for converting a formula into an equivalent hypersequent.

Notation 3.2. Write $fa(\phi_1 \vee \ldots \vee \phi_n) = \bigcup_i fa(\phi_i)$ and $fa(\Phi_1 \wedge \ldots \wedge \Phi_m) = \bigcup_i fa(\Phi_i)$.

Definition 3.3. We define schemas of hypersequent pairs in Figure 3.

The rules determine a rewrite system on hypersequents, where $\mathcal{H} \Rightarrow \mathcal{H}'$ when $\dfrac{\mathcal{H}}{\mathcal{H}'}$ is an instance of a rule(-schema). We call a sequence of such rewrites a **derivation**.

[10]Hypersequents were independently invented by Michael Gabbay, who approached the first author in great excitement at his new discovery/invention, which he called 'hypersequents'. Sadly, he had committed the sin of being born too late; older people get all the fun.

[11]This is just like the comma , and $\vdash$ of ordinary logic sequents $\phi_1, \ldots, \phi_m \vdash \psi_1, \ldots, \psi_n$, where $\vdash$ is interpreted as implication and comma on the left is interpreted as conjunction and on the right as disjunction.

$$\frac{(\neg\neg\phi \lor \Phi) \land \mathcal{H}}{(\phi \lor \Phi) \land \mathcal{H}}\,(\alpha\neg\neg) \qquad \frac{(\top \lor \Phi) \land \mathcal{H}}{\mathcal{H}}\,(\alpha\top)$$

$$\frac{(\phi \lor \neg\phi \lor \Phi) \land \mathcal{H}}{\mathcal{H}}\,(\alpha\mathbf{EM}) \qquad \frac{(\phi \lor \Phi) \land \mathcal{H} \quad (\phi =_{\alpha\beta\eta} \phi')}{(\phi' \lor \Phi) \land \mathcal{H}}\,(=_{\alpha\beta\eta})$$

$$\frac{(\neg(\phi' \land \phi) \lor \Phi) \land \mathcal{H}}{(\neg\phi' \lor \neg\phi \lor \Phi) \land \mathcal{H}}\,(\alpha\neg\land) \qquad \frac{((\phi' \land \phi) \lor \Phi) \land \mathcal{H}}{(\phi' \lor \Phi) \land (\phi \lor \Phi) \land \mathcal{H}}\,(\beta\land)$$

$$\frac{(\neg\forall a.\phi \lor \Phi) \land \mathcal{H} \quad (r : \mathit{type}(a))}{(\neg\phi[a{\mapsto}r] \lor \neg\forall a.\phi \lor \Phi) \land \mathcal{H}}\,(\gamma\neg\forall) \qquad \frac{(\forall a.\phi \lor \Phi) \land \mathcal{H} \quad (a \notin \mathit{fa}(\Phi))}{(\phi \lor \Phi) \land \mathcal{H}}\,(\delta^{-}\forall)$$

Figure 3: Rules for rewriting hypersequents

$$\frac{((\phi' \lor \phi) \lor \Phi) \land \mathcal{H}}{(\phi' \lor \phi \lor \Phi) \land \mathcal{H}}\,(\alpha\lor) \qquad \frac{((\phi' \supset \phi) \lor \Phi) \land \mathcal{H}}{(\neg\phi' \lor \phi \lor \Phi) \land \mathcal{H}}\,(\alpha\supset)$$

$$\frac{(\neg(\phi' \lor \phi) \lor \Phi) \land \mathcal{H}}{(\neg\phi' \lor \Phi) \land (\neg\phi \lor \Phi) \land \mathcal{H}}\,(\beta\neg\lor) \qquad \frac{(\neg(\phi' \supset \phi) \lor \Phi) \land \mathcal{H}}{(\phi' \lor \Phi) \land (\neg\phi \lor \Phi) \land \mathcal{H}}\,(\beta\neg\supset)$$

$$\frac{(\exists a.\phi \lor \Phi) \land \mathcal{H} \quad (r : \mathit{type}(a))}{(\phi[a{\mapsto}r] \lor \exists a.\phi \lor \Phi) \land \mathcal{H}}\,(\gamma\exists) \qquad \frac{(\neg\exists a.\phi \lor \Phi) \land \mathcal{H} \quad (a \notin \mathit{fa}(\Phi))}{(\neg\phi \lor \Phi) \land \mathcal{H}}\,(\delta^{-}\neg\exists)$$

Figure 4: Derived rules for $\lor$, $\supset$, $\exists$ in hypersequents

Remark 3.4. Recall from Notation 2.15 that we treat $\lor$, $\supset$, and $\exists_\alpha$ as syntactic sugar. For the reader's convenience in Figure 4 we give derived rules for them—we will use them without comment, just as is convenient.

Remark 3.5. We fix no choice of reduction strategy. We tend to write the 'active formula' at the head of the hypersequent in Figures 3 and 4 and in the examples of Subsection 3.2, but this is just a convention. Sequents and hypersequents are sets, they are not ordered, and hypersequent rewriting can rewrite any sequent within a hypersequent,

3.2. *Example derivations*

We now give a few simple but illustrative example derivations. We may elide rule $(=_{\alpha\beta\eta})$. The object of the exercise is to rewrite the hypersequent to $\varnothing$, the empty hypersequent. A precise semantic justification for why this is 'good' is given in Theorems 6.6 and 6.18.

3.2.1. *Derivation of* $(\forall a.\phi)\supset\phi[a{\mapsto}r]$.

This corresponds to $(\forall\mathbf{L})$ (forall left-introduction) and to part 1 of Lemma 5.23:

$$\begin{array}{ll}
(\forall a.\phi)\supset\phi[a{\mapsto}r] & \Longrightarrow\ (\alpha\supset) \\
\neg\forall a.\phi \lor \phi[a{\mapsto}r] & \Longrightarrow\ (\gamma\neg\forall) \\
\neg\phi[a{\mapsto}r] \lor \neg\forall a.\phi \lor \phi[a{\mapsto}r] & \Longrightarrow\ (\alpha\mathbf{EM}) \\
\varnothing &
\end{array}$$

Here and below $\varnothing$ is the empty hypersequent.

It is not hard to transform this derivation into a sequent derivation of $(\forall a.\phi)\supset\phi[a{\mapsto}r]$. The sequent rules are $(\supset\mathbf{R})$, $(\forall\mathbf{L})$, and $(\mathbf{Axiom})$ corresponding precisely to $(\alpha\supset)$, $(\gamma\neg\forall)$, and $(\alpha\mathbf{EM})$. It is not hard to do the same for the examples that now follow.

3.2.2. *Forall implies Exists*

We derive the formula $(\forall a.\mathsf{P}(a))\supset\exists a.\mathsf{P}(a)$ as follows:

$$\begin{array}{ll}
(\forall a.\mathsf{P}(a))\supset\exists a.\mathsf{P}(a) & \Longrightarrow\ (\alpha\supset) \\
\neg\forall a.\mathsf{P}(a) \lor \exists a.\mathsf{P}(a) & \Longrightarrow\ (\gamma\exists),\ \mathit{pms}(X){=}\varnothing \\
\mathsf{P}(X) \lor \neg\forall a.\mathsf{P}(a) \lor \exists a.\mathsf{P}(a) & \Longrightarrow\ (\gamma\neg\forall),\ \mathit{pms}(Y){=}\varnothing \\
\neg\mathsf{P}(Y) \lor \mathsf{P}(X) \lor \neg\forall a.\mathsf{P}(a) \lor \exists a.\mathsf{P}(a) &
\end{array}$$

We implicitly assume above that $\mathit{type}(X) = \mathit{type}(Y) = \mathit{type}(a)$, and we will make similar implicit assumptions in the examples below. There will always be one clear correct choice of type for the variables we introduce in these examples, and other choices will clearly lead to ill-typed syntax.

We also explicitly choose $\mathit{pms}(X) = \mathit{pms}(Y) = \varnothing$. This is just because we have to choose *some* permission sets of X and Y.

If we instantiate X to Y then $\neg\mathsf{P}(X)$ and $\mathsf{P}(X)$ would match the premises of $(\alpha\mathbf{EM})$.

Remark 3.6. The instantiation of X to Y in the derivation above is not a rewrite rule. It is an operation transforming one complete derivation (like the one above this paragraph) into another complete derivation (like the one below this paragraph). The general case is treated in Proposition 6.25. Sometimes it is possible to extend the instantiated derivation, and sometimes not; we will see examples of both cases.

We obtain another valid derivation and in this case we can extend it as follows:

$$\begin{array}{ll}
(\forall a.\mathsf{P}(a))\supset\exists a.\mathsf{P}(a) & \Longrightarrow\ (\alpha\supset),(\gamma\exists),\ \mathit{pms}(Y){=}\varnothing \\
\mathsf{P}(Y) \lor \neg\forall a.\mathsf{P}(a) \lor \exists a.\mathsf{P}(a) & \Longrightarrow\ (\gamma\neg\forall) \\
\neg\mathsf{P}(Y) \lor \mathsf{P}(Y) \lor \neg\forall a.\mathsf{P}(a) \lor \exists a.\mathsf{P}(a) & \Longrightarrow\ (\alpha\mathbf{EM}) \\
\varnothing &
\end{array}$$

3.2.3. *Forall implies Exists (again)*

Another derivation is possible, as follows:

$$\begin{array}{ll}
(\forall a.\mathsf{P}(a))\supset\exists a.\mathsf{P}(a) & \Longrightarrow\ (\alpha\supset),(\gamma\neg\forall),\ \mathit{pms}(Y){=}\varnothing \\
\neg\mathsf{P}(Y) \lor \neg\forall a.\mathsf{P}(a) \lor \exists a.\mathsf{P}(a) & \Longrightarrow\ (\gamma\exists),\ \mathit{pms}(X){=}\varnothing \\
\mathsf{P}(X) \lor \neg\mathsf{P}(Y) \lor \neg\forall a.\mathsf{P}(a) \lor \exists a.\mathsf{P}(a) &
\end{array}$$

Again we instantiate X to Y and use $(\alpha\mathbf{EM})$.

3.2.4. *A derivation that fails because atoms are distinct*

We try to derive $(\exists a.\mathsf{P}(a))\supset\forall a.\mathsf{P}(a)$. As standard and as expected, we will fail:

$$\begin{array}{ll}
(\exists a.\mathsf{P}(a))\supset\forall a.\mathsf{P}(a) & \Longrightarrow\ (\alpha\supset), (\delta^-\neg\exists) \\
\neg\mathsf{P}(a) \lor \forall a.\mathsf{P}(a) & \Longrightarrow\ (=_{\alpha\beta\eta}) \\
\neg\mathsf{P}(a) \lor \forall b.\mathsf{P}(b) & \Longrightarrow\ (\delta^-\forall) \\
\neg\mathsf{P}(a) \lor \mathsf{P}(b) &
\end{array}$$

Recall that we will usually elide $(=_{\alpha\beta\eta})$, but for clarity just this once we wrote it out in full.

Now we are stuck. We cannot use $(\alpha\mathbf{EM})$ because a and b are distinct.

3.2.5. *A derivation that fails because of permission sets*

We have not included equality in our system, but suppose we did, with a rule $(\alpha\approx)$ allowing us to delete sequents with $(r\approx r)$ (similar to $(\alpha\top)$).

We derive $\forall a.\exists b.a\approx b$ (and succeed):

$$\begin{array}{ll}
\forall a.\exists b.a\approx b & \Longrightarrow\ (\delta^-\forall) \\
\exists b.a\approx b & \Longrightarrow\ (\gamma^-\exists) \\
a\approx a \lor \exists b.a\approx b & \Longrightarrow\ (\alpha\approx) \\
\varnothing &
\end{array}$$

Now we try to derive $\exists a.\forall b.a\approx b$:

$$\begin{array}{ll}
\exists a.\forall b.a\approx b & \Longrightarrow\ (\gamma^-\exists), \mathit{pms}(X)=\varnothing \\
\forall b.X\approx b \lor \exists a.\forall b.a\approx b & \Longrightarrow\ (\delta^-\forall) \\
X\approx b \lor \exists a.\forall b.a\approx b &
\end{array}$$

Now we are stuck. We cannot use $(\alpha\approx)$ because X is not the same term as b. We cannot globally transform the derivation to instantiate X to b because $b \notin \mathit{pms}(X)$ so that $[X{:=}b]$ is not an instantiation (Definition 2.26). It would not help to try to re-run the derivation with an X' with $\mathit{pms}(X') = \{b\}$ because $(\delta^-\forall)$ would just force us to choose a fresh $b' \notin \mathit{pms}(X')$.

For an example without equality, try to derive $(\forall a.\exists b.\mathsf{P}(a,b)) \supset \exists b.\forall a.\mathsf{P}(a,b)$. The attempted derivation (which fails) is a little longer, but illustrates the same point.

Here, the nominal support restriction on instantiations in Definition 2.26 reflects an important soundness criterion: variables can only vary over terms with free atoms they 'know about', and must not vary over terms with free atoms that were subsequently generated fresh by a (δ^-*) rule, and thus that they do not 'know about'.

3.2.6. *Plato's principle*

Plato's principle [MV95, Subsection 3.3] is represented in our syntax by the formula $\exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a))$.

This can be derived as follows, where we elide rule $(=_{\alpha\beta\eta})$:

$$\begin{array}{ll}
\exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \Longrightarrow\ (\gamma\exists),\ \mathit{pms}(X)=\varnothing \\
(\exists b.\mathsf{P}(b))\supset\mathsf{P}(X) \lor \exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \Longrightarrow\ (\alpha\supset) \\
(\neg\exists b.\mathsf{P}(b)) \lor \mathsf{P}(X) \lor \exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \Longrightarrow\ (\delta^-\neg\exists) \\
\neg\mathsf{P}(b) \lor \mathsf{P}(X) \lor \exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \Longrightarrow\ (\gamma\exists),\ \mathit{pms}(Y)=\{b\},\ (\alpha\supset) \\
\mathsf{P}(Y) \lor \neg\mathsf{P}(b) \lor \mathsf{P}(X) \lor (\neg\exists b.\mathsf{P}(b)) \lor \exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) &
\end{array}$$

Now if we instantiate Y to b then $\mathsf{P}(b)$ and $\neg\mathsf{P}(b)$ would match the premises of $(\alpha\mathbf{EM})$. So we instantiate Y in the derivation so far and extend as follows:

$$\begin{array}{ll}
\exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \Longrightarrow\ (\gamma\exists),\ pms(X){=}\varnothing \\
(\exists b.\mathsf{P}(b))\supset\mathsf{P}(X) \vee \exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \Longrightarrow\ (\alpha\supset) \\
(\neg\exists b.\mathsf{P}(b)) \vee \mathsf{P}(X) \vee \exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \Longrightarrow\ (\delta^{-}\neg\exists) \\
\neg\mathsf{P}(b) \vee \mathsf{P}(X) \vee \exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \Longrightarrow\ (\gamma\exists),\ (\alpha\supset) \\
\mathsf{P}(b) \vee \neg\mathsf{P}(b) \vee \mathsf{P}(X) \vee (\neg\exists b.\mathsf{P}(b)) \vee \exists a.((\exists b.\mathsf{P}(b))\supset\mathsf{P}(a)) & \\
 & \Longrightarrow\ (\alpha\mathbf{EM}) \\
\varnothing &
\end{array}$$

Note how we apply $(\gamma\exists)$ twice; the first time to get $\neg\mathsf{P}(b)$, and the second time to get $\mathsf{P}(b)$. This is because $(\delta^{-}\neg\exists)$ introduces b *fresh* for $pms(X)$—just as $(\forall\mathbf{R})$ introduces a 'fresh variable'. This is related to the issue that we touched on in Subsection 3.2.5.

The reader can find excellent expositions about just how inefficient (δ^{-}) can be in [HS94, BHS93, CA00]. In the Introduction we noted that exponential and non-elementary speedups are possible with so-called 'liberalised' δ-rules. This motivates the $(\delta^{\times})$ rules, which we consider next in Section 4.

4. Maximising hypersequents

We now extend the hypersequents and hypersequent rewriting from Section 3 with *maximisation conditions*. These are subtle, so we provide plenty of exposition in Subsection 4.1 and example derivations follow in Subsection 4.2.

4.1. *The basic definition*

Definition 4.1. A **primitive maximisation condition** is a pair $X{\uparrow}[a]\phi$ where $X:\alpha$ and $a:\alpha$ and $\phi:o$ such that $fa([a]\phi)\subseteq pms(X)$.

Remark 4.2. The intuition of $X{\uparrow}[a]\phi$ is: take X to be a value making $\phi[a{\mapsto}X]$ true. We can read this as a weak form of choice—weak, since we do not necessarily assign the value a term in the syntax. The value is allowed to depend on the atoms in $pms(X)$, and must not depend on any other atoms. Technically, we use nominal techniques to give 'depends on' a precise formal meaning: it means 'has in its support'.

Example maximisers are in Example 4.9. The formal semantic interpretation of $X{\uparrow}[a]\phi$ is the first equation of Definition 6.12; it interprets $X{\uparrow}[a]\phi$ as $(\exists a.\phi)\supset\phi[a{\mapsto}X]$. This can be read as the following slightly more detailed rendering of the intuition of the previous paragraph: "*If* there exists a value making ϕ true, then X is one of those values".

Definition 4.3. A set of primitive maximisation conditions $\mathcal{C}$ induces a relation $\prec$ on variables by letting $\prec$ be the least transitive relation such that

$$Y \prec X \text{ if } (X{\uparrow}[a]\phi)\in\mathcal{C} \text{ and } Y\in fv(\phi).^{12}$$

[12]Recall that by permutative convention X and Y are distinct.

Remark 4.4. Intuitively $Y \prec X$ expresses that the value chosen for Y can influence the choice of maximiser for X—so we can read $Y \prec X$ as the assertion "You need to fix the value for Y before you calculate a value for X".

Definition 4.5. A **maximisation condition** is a set of primitive maximisation conditions $\mathcal{C} = (X_i {\uparrow} [a_i]\phi_i)_1^n$ such that:

1. $\prec$ from Definition 4.3 is a well-founded strict partial order (a well-founded transitive irreflexive relation) on $\mathcal{C}$.
2. $\mathcal{C}$ is **functional**, in the sense that $X{\uparrow}r \in \mathcal{C}$ and $X{\uparrow}s \in \mathcal{C}$ imply that r and s are syntactically identical.

Define $dom(\mathcal{C}) = \{X_1, \dots, X_n\}$ and call this set the **domain** of $\mathcal{C}$. We let $\mathcal{C}$ range over maximisation conditions.

Remark 4.6. In Definition 4.5 we insist that $\prec$ should be well-founded. This is used in Lemma 4.7, and in Proposition 6.17 where we use well-foundedness to 'work our way up' $\prec$ and so generate a maximiser. To see a concrete example of what goes wrong without well-foundedness see Subsection 4.2.3.

Lemma 4.7 is a technical result which is useful for Proposition 6.17. It makes Remark 4.6 formal, and illustrates how well-foundedness of $\prec$ controls the free variables of the predicates in the maximisation condition:

Lemma 4.7. *Suppose $\mathcal{C}$ is a maximisation condition and $\prec$ is the well-founded strict partial order on variables induced by $\mathcal{C}$. Then if $X{\uparrow}[a]\phi \in \mathcal{C}$ and X is $\prec$-minimal in $dom(\mathcal{C})$ then $fv(\phi) \cap dom(\mathcal{C}) = \varnothing$.*

Proof. If $X \in fv(\phi) \cap dom(\mathcal{C})$ then $X \prec X$, contradicting strictness of $\prec$ (and indeed also well-foundedness of $\prec$ and $\prec$-minimality of X in $dom(\mathcal{C})$).

If $Y \in fv(\phi) \cap dom(\mathcal{C})$ then $Y \prec X$ and $Y \in dom(\mathcal{C})$, contradicting $\prec$-minimality of X in $dom(\mathcal{C})$. $\square$

Definition 4.8. A **maximising hypersequent** $\mathcal{P}$ is a pair $\mathcal{C} \vdash \mathcal{H}$ of a maximisation condition and a hypersequent. We may write $\varnothing \vdash \mathcal{H}$ just as $\mathcal{H}$.

Example 4.9. In ordinary hypersequents, variables X have an existential flavour and atoms a have a universal flavour. This is why the $(\delta^- *)$ rules create a fresh atom—this is a fresh 'universal' variable.

A maximisation condition $X{\uparrow}[a]\phi$ restricts X to range over elements making $\phi[a {\mapsto} X]$ as true as possible. Dually, a maximisation condition $X{\uparrow}[a]\neg\phi$ restricts X to range over elements making $\phi[a {\mapsto} X]$ as *false* as possible—so $X{\uparrow}[a]\neg\phi$ makes X behave like a universal variable (the words *universal* and *existential* are a useful intuition but they have their limits: see Remark 6.20 for discussion). We exploit this duality in rules $(\delta^{\mathsf{x}}\forall)$ and $(\delta^{\mathsf{x}}\neg\exists)$ of Figures 5 and 6.

Let us consider some concrete examples of maximisers. These are based on the model which we build in Subsection 7.2, but we will sketch the technical details, which we hope will anyway be fairly self-explanatory:

1. Take ϕ to be $a : o$ so that $[a]\phi$ is just $[a]a$, and take $pms(X) = \varnothing$.
   We use the model $\mathcal{I}$ from Subsection 7.2. There, truth-values are interpreted as $[\![o]\!]$ which is inhabited by maps from atoms-valuations ς, to the set $\{\bot, \top\}$. In symbols, valuations ς and truth-values have types
   $$\Pi_\alpha(\mathbb{A}_\alpha \to [\![\alpha]\!]) \qquad \text{and} \qquad \Pi_\alpha(\mathbb{A}_\alpha \to [\![\alpha]\!]) \to \{\bot, \top\}$$
   respectively.
   The maximisation condition $X \uparrow [a]a$ restricts X to range over the following rather small set:
   $$\{\lambda\varsigma.\top\}$$
   This is not a very interesting example, since ϕ is so simple.
2. Now take ϕ to be $a \vee b : o$ and take $pms(X) = \{b\}$. Then $X \uparrow [a](a \vee b)$ restricts X to be in the set
   $$\{\lambda\varsigma.\top, \quad \lambda\varsigma.\text{if } \varsigma(b){=}\bot \text{ then } \top \text{ else } \bot\}.$$
   This is because there are two functions f on valuations ς, that examine only $\varsigma(b)$ and that make $f(\varsigma) \vee \varsigma(b)$ true: either take $f(\varsigma)$ to be $\top$ always, or let $f(\varsigma)$ be the negation of $\varsigma(b)$. Both are valid maximisers.
3. Suppose (without writing out full definitions) we assume a type for numbers $\mathbb{N}$, and a predicate for equality. Suppose $a, b, X : \mathbb{N}$ and $pms(X) = \{b\}$. Then $X \uparrow [a](a{=}b)$ restricts X to be equal to $\lambda\varsigma.\varsigma(b)$.
   Also, $X \uparrow [a](a{\neq}b)$ restricts X to be any one of the many functions mapping a valuation ς to an element of $\mathbb{N} \setminus \{\varsigma(b)\}$.
4. Note that in the previous example, $pms(X)$ does not *have* to be equal to $\{b\}$, though by Definition 4.1 it must be that $b \in pms(X)$.
   If we ignore this and see what happens if we try to create a maximiser using X where $b \notin pms(X)$ then we see that no such thing exists: to make $(X{=}b)$ (or $(X{\neq}b)$) true, the meaning we give to X must depend on the value of ς at b.

Remark 4.10. The (δ^χ) rules are descended from the *liberalised δ-rule* of [Wir04, page 14], which itself can be understood as a non-Skolemising version of (δ^+) from [HS94], where instead of Skolemisation we use *variable-conditions* [Wir04]. Still, it is probably fair to say that the δ^χ rules and maximisation conditions have an independent existence.

As we hope is now clear, our 'maximisation condition' is a kind of choice, but it is not a unique one. Thus in Example 4.9 we saw that *many* maximisers can exist for the same maximisation condition.

A perennial problem of Hilbert's choice $\epsilon a.\phi$ is that expressions tend to explode exponentially as choices get nested; our syntax is less prone to that. We do not introduce choice $\epsilon a.\phi$ into the syntax or semantics, our maximisation conditions do not force a *specific* choice of maximiser in the semantics, and the syntax manages to remain relatively compact.

Later on in Section 5 onwards we will see how our system admits a semantics in nominal sets.

$$\frac{\mathcal{C} \vdash (\neg\neg\phi \lor \Phi) \land \mathcal{H}}{\mathcal{C} \vdash (\phi \lor \Phi) \land \mathcal{H}}(\alpha\neg\neg) \qquad \frac{\mathcal{C} \vdash (\top \lor \Phi) \land \mathcal{H}}{\mathcal{C} \vdash \mathcal{H}}(\alpha\top)$$

$$\frac{\mathcal{C} \vdash (\phi \lor \neg\phi \lor \Phi) \land \mathcal{H}}{\mathcal{C} \vdash \mathcal{H}}(\alpha\mathbf{EM}) \qquad \frac{\mathcal{C} \vdash (\phi \lor \Phi) \land \mathcal{H} \quad (\phi =_{\alpha\beta\eta} \phi')}{\mathcal{C} \vdash (\phi' \lor \Phi) \land \mathcal{H}}(=_{\alpha\beta\eta})$$

$$\frac{\mathcal{C} \vdash (\neg(\phi' \land \phi) \lor \Phi) \land \mathcal{H}}{\mathcal{C} \vdash (\neg\phi' \lor \neg\phi \lor \Phi) \land \mathcal{H}}(\alpha\neg\land) \qquad \frac{\mathcal{C} \vdash ((\phi' \land \phi) \lor \Phi) \land \mathcal{H}}{\mathcal{C} \vdash (\phi' \lor \Phi) \land (\phi \lor \Phi) \land \mathcal{H}}(\beta\land)$$

$$\frac{\mathcal{C} \vdash (\neg\forall a.\phi \lor \Phi) \land \mathcal{H} \quad (r : \mathit{type}(a))}{\mathcal{C} \vdash (\neg\phi[a{\mapsto}r] \lor \neg\forall a.\phi \lor \Phi) \land \mathcal{H}}(\gamma\neg\forall) \qquad \frac{\mathcal{C} \vdash (\forall a.\phi \lor \Phi) \land \mathcal{H}}{\mathcal{C}, X{\uparrow}[a]\neg\phi \vdash (\phi[a{\mapsto}X] \lor \Phi) \land \mathcal{H}}(\delta^{\mathsf{x}}\forall)$$

Figure 5: Rules for rewriting maximising hypersequents

As a nice side-effect of this—and as was also the case in [Wir04]—we can accommodate both δ^- and δ^{x} in a single syntax, and later on in a single semantics too. This gives us a convenient way of constrasting and understanding what different systems have in common and what sets them apart.

Definition 4.11. We define rewrites on maximising hypersequents in Figures 5 and 6. $(\delta^{\mathsf{x}}\forall)$ and $(\delta^{\mathsf{x}}\neg\exists)$ are subject to the well-formedness conditions that $\mathcal{C}, X{\uparrow}[a]\neg\phi$ and $\mathcal{C}, X{\uparrow}[a]\phi$ respectively must be valid maximisation conditions (see Remark 4.13).

Remark 4.12. The reader can check that the rewrites of Figures 5 and 6 (for maximising hypersequents) are obtained from those of Figures 3 and 4 (for 'plain' hypersequents) by the following procedure:

- Delete the rules $(\delta^-\forall)$ and $(\delta^-\neg\exists)$.
- Add '$\mathcal{C} \vdash$' to all the remaining rules.
- Replace $(\delta^-\forall)$ and $(\delta^-\neg\exists)$ with $(\delta^{\mathsf{x}}\forall)$ and $(\delta^{\mathsf{x}}\neg\exists)$ respectively.

Remark 4.13. We unpack what the well-formedness condition of Definition 4.11 means:

1. $\mathit{type}(X) = \mathit{type}(a)$
2. $\mathit{fa}(\phi)\backslash\{a\} \subseteq \mathit{pms}(X)$ (that is, $\mathit{fa}([a]\phi) \subseteq \mathit{pms}(X)$)
3. The well-foundedness and functionality conditions (conditions 1 and 2 of Definition 4.5) cannot be violated by adding $X{\uparrow}[a]\neg\phi$ or $X{\uparrow}[a]\phi$ to $\mathcal{C}$.

If any of these conditions fail then we do not have a valid instance of a $(\delta^{\mathsf{x}}*)$ rule. Note, however, that X may occur in Φ or $\mathcal{H}$; an example of this is in Subsection 4.2.1.

4.2. *Example derivations using maximising hypersequents*

4.2.1. *Plato's Principle*

We give a derivation of Plato's Principle using $(\delta^{\mathsf{x}}*)$ rules:

$$\frac{\mathcal{C} \vdash ((\phi' \vee \phi) \vee \Phi) \wedge \mathcal{H}}{\mathcal{C} \vdash (\phi' \vee \phi \vee \Phi) \wedge \mathcal{H}}(\alpha\vee) \qquad \frac{\mathcal{C} \vdash ((\phi' \supset \phi) \vee \Phi) \wedge \mathcal{H}}{\mathcal{C} \vdash (\neg\phi' \vee \phi \vee \Phi) \wedge \mathcal{H}}(\alpha\supset)$$

$$\frac{\mathcal{C} \vdash (\neg(\phi' \vee \phi) \vee \Phi) \wedge \mathcal{H}}{\mathcal{C} \vdash (\neg\phi' \vee \Phi) \wedge (\neg\phi \vee \Phi) \wedge \mathcal{H}}(\beta\neg\vee) \qquad \frac{\mathcal{C} \vdash (\neg(\phi' \supset \phi) \vee \Phi) \wedge \mathcal{H}}{\mathcal{C} \vdash (\phi' \vee \Phi) \wedge (\neg\phi \vee \Phi) \wedge \mathcal{H}}(\beta\neg\supset)$$

$$\frac{\mathcal{C} \vdash (\exists a.\phi \vee \Phi) \wedge \mathcal{H} \quad (r : \mathit{type}(a))}{\mathcal{C} \vdash (\phi[a\mapsto r] \vee \exists a.\phi \vee \Phi) \wedge \mathcal{H}}(\gamma\exists) \qquad \frac{\mathcal{C} \vdash (\neg\exists a.\phi \vee \Phi) \wedge \mathcal{H}}{\mathcal{C}, X{\uparrow}[a]\phi \vdash (\neg\phi[a\mapsto X] \vee \Phi) \wedge \mathcal{H}}(\delta^{\mathsf{x}}\neg\exists)$$

Figure 6: Derived rules for $\vee$, $\supset$, $\exists$ in maximising hypersequents

$$\begin{array}{rll}
& \exists a.((\exists b.\mathsf{P}(b)) \supset \mathsf{P}(a)) & \Longrightarrow\ (\gamma\exists),\ \mathit{pms}(X){=}\varnothing \\
& (\exists b.\mathsf{P}(b)) \supset \mathsf{P}(X) \vee \exists a.((\exists b.\mathsf{P}(b)) \supset \mathsf{P}(a)) & \Longrightarrow\ (\alpha\supset) \\
& (\neg\exists b.\mathsf{P}(b)) \vee \mathsf{P}(X) \vee \exists a.((\exists b.\mathsf{P}(b)) \supset \mathsf{P}(a)) & \Longrightarrow\ (\delta^{\mathsf{x}}\neg\exists),\ \mathit{pms}(Y){=}\varnothing \\
Y{\uparrow}[b]\mathsf{P}(b) \vdash & \neg\mathsf{P}(Y) \vee \mathsf{P}(X) \vee \exists a.((\exists b.\mathsf{P}(b)) \supset \mathsf{P}(a)) &
\end{array}$$

We now instantiate X to Y in the rewrites above and extend with $(\alpha\mathbf{EM})$:

$$\begin{array}{rll}
& \exists a.((\exists b.\mathsf{P}(b)) \supset \mathsf{P}(a)) & \Longrightarrow\ (\gamma\exists),\ \mathit{pms}(Y){=}\varnothing \\
& (\exists b.\mathsf{P}(b)) \supset \mathsf{P}(Y) \vee \exists a.((\exists b.\mathsf{P}(b)) \supset \mathsf{P}(a)) & \Longrightarrow\ (\alpha\supset) \\
& (\neg\exists b.\mathsf{P}(b)) \vee \mathsf{P}(Y) \vee \exists a.((\exists b.\mathsf{P}(b)) \supset \mathsf{P}(a)) & \Longrightarrow\ (\delta^{\mathsf{x}}\neg\exists) \\
Y{\uparrow}[b]\mathsf{P}(b) \vdash & \neg\mathsf{P}(Y) \vee \mathsf{P}(Y) \vee \exists a.((\exists b.\mathsf{P}(b)) \supset \mathsf{P}(a)) & \Longrightarrow\ (\alpha\mathbf{EM}) \\
Y{\uparrow}[b]\mathsf{P}(b) \vdash & \varnothing &
\end{array}$$

Compare this derivation with the one using $(\delta^{-}*)$ rules in Subsection 3.2.6. As promised at the end of Subsection 3.2.6, the derivation here is shorter; more complex because of the maximisation conditions, but shorter.

4.2.2. *Exists does not imply Forall*

We try to derive the formula $(\exists a.\mathsf{P}(a)) \supset \forall a.\mathsf{P}(a)$:

$$\begin{array}{rll}
& (\exists a.\mathsf{P}(a)) \supset \forall a.\mathsf{P}(a) & \Longrightarrow\ (\alpha\supset), (\delta^{\mathsf{x}}\neg\exists),\ \mathit{pms}(Y){=}\varnothing \\
Y{\uparrow}[a]\mathsf{P}(a) \vdash & \neg\mathsf{P}(Y) \vee \forall a.\mathsf{P}(a) & \Longrightarrow\ (\delta^{\mathsf{x}}\forall),\ \mathit{pms}(X){=}\varnothing \\
Y{\uparrow}[a]\mathsf{P}(a), X{\uparrow}[a]\neg\mathsf{P}(a) \vdash & \neg\mathsf{P}(Y) \vee \mathsf{P}(X) &
\end{array}$$

We note that $\neg\mathsf{P}(Y)$ and $\mathsf{P}(X)$ match the premises of $(\alpha\mathbf{EM})$ if we instantiate Y to X. Unfortunately if we try to do this then we run foul of functionality of maximisation conditions (condition 2 of Definition 4.5); $\{X{\uparrow}[a]\neg\mathsf{P}(a), X{\uparrow}[a]\mathsf{P}(a)\}$ is not functional, since $[a]\neg\mathsf{P}(a)$ and $[a]\mathsf{P}(a)$ are not syntactically identical.

So we are stuck: we cannot use $(\alpha\mathbf{EM})$ because $\neg\mathsf{P}(Y)$ and $\mathsf{P}(X)$ are distinct. We cannot instantiate Y to X because then the instance of $(\delta^{\mathsf{x}}\forall)$ ceases to be an instance.

4.2.3. *A derivation that fails because of the well-foundedness condition*

We illustrate what can go wrong if we drop the well-foundedness condition on $\prec$ from Definition 4.5. Consider the following partial derivation of $\forall a.(a \vee \neg X)$, where $\mathit{type}(a) = \mathit{type}(X) = o$ and $\mathit{pms}(X) = \varnothing$:

$$\begin{array}{rll} & \forall a.(a \vee \neg X) & \Longrightarrow \ (\delta^{\mathsf{x}}\forall), (=_{\alpha\beta\eta}) \\ Y{\uparrow}[a](a \vee \neg X) \vdash & Y \vee \neg X & \end{array}$$

It would be quite nice if we could instantiate Y to X. So we do so, to obtain the following incorrect derivation of $\forall a.(a \vee \neg X)$:

$$\begin{array}{rll} & \forall a.(a \vee \neg X) & \Longrightarrow \ \cancel{(\delta^{\mathsf{x}}\forall)}, (=_{\alpha\beta\eta}) \\ X{\uparrow}[a](a \vee \neg X) \vdash & X \vee \neg X & \Longrightarrow \ (\alpha\vee) \\ X{\uparrow}[a](a \vee \neg X) \vdash & X \boldsymbol{\vee} \neg X & \Longrightarrow \ (\alpha\mathbf{EM}) \\ X{\uparrow}[a](a \vee \neg X) \vdash & \varnothing & \text{Incorrect!} \end{array}$$

We note that the relation $\prec$ induced on variables by $\{X{\uparrow}[a](a \vee \neg X)\}$ is not well-founded, since $X \prec X$.

5. Nominal models

In Subsection 5.1 we set up the basic definitions of nominal sets and in Subsection 5.2 we set up the basic ideas of models that we will need.

For more on nominal sets see [GP01, Gab11a, Gab12b]; to see the notions of model from which the ideas in this paper are derived—though the presentation here is tailored to our application—see [GM11] (nominal Henkin semantics) and [GM08b] or [DG10, DG12a] (nominal algebraic axiomatisations of first-order logic). A specific discussion of how this paper relates to [GM11] and [DG12a] is in Subsection 9.1.

The reader may grant that we want to axiomatise logic to build our models, but why do we need nominal sets? What do e.g. Definition 5.3 or Lemma 5.11 have to do with anything at all?

Nominal sets give us atoms, their permutative symmetries (i.e. equivariance properties), and the notion of finite support. It turns out—this is not obvious, but it is true—that this is just sufficient to give a nice, concise, well-behaved notion of model for systems with names and binding; substitution and quantification are two examples of such systems.

So note

- the conditions on support in Figures 7 and 8;
- how our treatment of atoms allows us to concisely state that $\forall$ is a congruence in Definition 5.16;
- how we impose a condition $a \notin \mathrm{supp}([a]x)$ in Definition 5.12 and we use that condition Lemma 5.15;
- and how we use α-renaming in Corollary 5.24.

These are just examples of a close dance that takes place between the nominal sets semantics, the notion of model, and (in the background) the syntax developed above. It all fits together beautifully, and Subsection 5.1 is (by some magic, it might seem) just what is needed to make this work.

5.1. Background on nominal sets

Definition 5.1. If $A \subseteq \mathbb{A}$ define $\mathit{fix}(A) = \{\pi \mid \forall a \in A.\pi(a) = a\}$.

DEFINITION 5.2. A **set with a permutation action** X is a pair $(|\mathsf{X}|, \cdot)$ of an **underlying set** $|\mathsf{X}|$ and a **permutation action** written $\pi{\cdot}x$ which is a group action on $|\mathsf{X}|$, so that $id{\cdot}x = x$ and $\pi{\cdot}(\pi'{\cdot}x) = (\pi \circ \pi'){\cdot}x$ for all $x \in |\mathsf{X}|$ and permutations π and π'.

Say that $A \subseteq \mathbb{A}$ **supports** $x \in |\mathsf{X}|$ when $\forall \pi. \pi \in \mathit{fix}(A) \Rightarrow \pi{\cdot}x = x$. If there exists a finite set S supporting x then call x **supported** and say that x has **finite support**.

DEFINITION 5.3. Call a set with a permutation action X a **nominal set** when every $x \in |\mathsf{X}|$ is supported. X, Y, Z will range over nominal sets.

EXAMPLE 5.4.
1. $\mathbb{A}$ forms a nominal set where $\pi{\cdot}a = \pi(a)$.
   The atom a is supported by $\{a\}$.
2. If X and Y are nominal sets then $\mathsf{X} \times \mathsf{Y}$ is a nominal set with underlying set $\{(x, y) \mid x \in |\mathsf{X}|, y \in |\mathsf{Y}|\}$ and action $\pi{\cdot}(x, y) = (\pi{\cdot}x, \pi{\cdot}y)$.
   If S supports x and T supports y then $S \cup T$ supports (x, y).
3. If X is a set with a permutation action then $\mathit{pow}'(\mathsf{X})$ is a set with a permutation action, where $\mathit{pow}'(\mathsf{X})$ has underlying set the full powerset of $|\mathsf{X}|$ and permutation action $\pi{\cdot}U = \{\pi{\cdot}u \mid u \in U\}$.
   For example both $\{a\}$ and $\mathbb{A}\backslash\{a\}$ are in $\mathit{pow}'(\mathbb{A})$ and are supported by $\{a\}$. The set $\mathit{comb} = \{a, c, e, g, \dots\}$ of 'every other atom' is also in $\mathit{pow}'(\mathbb{A})$ and supported only by $\mathbb{A}$. In particular, comb has no finite supporting set.
4. If X is a nominal set then $\mathit{pow}(\mathsf{X})$ is a nominal set with underlying set the subset of $|\mathit{pow}'(\mathsf{X})|$ of elements with support.
   Note that $\{a\}$ and $\mathbb{A}\backslash\{a\}$ are both supported by $\{a\}$. So 'a is in the support of x' does not mean the same thing as 'a is a set element of x'. Support measures *asymmetry*, not sets membership.
   It is a fact that $|\mathit{pow}'(\mathbb{A})|$ is the set of all subsets of atoms whereas $|\mathit{pow}(\mathbb{A})|$ is the set of subsets of atoms U such that U or $\mathbb{A} \setminus U$ is finite.
5. Permissive-nominal terms up to α-equivalence (Definition 2.22) with the permutation action from Definition 2.20, form a nominal set. The term r is supported by its free atoms $\mathit{fa}(r)$. For more comment on this see [DGM09b, DGM10].
6. Permissive-nominal terms up to $\alpha\beta\eta$-equivalence (Definition 2.22) with the same permutation action, also form a nominal set. As it happens, the term r is supported by the intersection of the free atoms of all terms $\alpha\beta\eta$-equivalent to r.

REMARK 5.5. The nominal notion of finite support is subtle, but a good starting intuition is that 'a is in the support of x' means '*x depends on a*' or 'x is asymmetric with respect to permuting a'.

It is important to realise that support does not depend on an *a priori* assumption that there is substitution, functional application, abstraction, or sets membership. Support is more elementary than these notions.

However, substitution, application, abstraction, and sets membership can *interact* with support in rich and interesting ways. See for instance the freshness side-conditions in Figure 7.

This is part of what makes nominal semantics so interesting. More extensive discussions of support are in [Gab11a, Gab12b].

Definition 5.6. Suppose X is a nominal set and $x \in |\mathsf{X}|$. Define the **support** of x by $\mathrm{supp}(x) = \bigcap\{A \mid A \text{ supports } x\}$.

For the rest of this subsection we recall basic properties of nominal sets which will help us later, with pointers to full proofs.

Proposition 5.7. $\mathrm{supp}(\pi{\cdot}x) = \{\pi(a) \mid a \in \mathrm{supp}(x)\}$.

Proof. See [Gab11a, Theorem 2.19]. □

Theorem 5.8. *Suppose* X *is a nominal set and* $x \in |\mathsf{X}|$. *Then* $\mathrm{supp}(x)$ *is the unique least set of atoms that supports* x.

Proof. See [Gab11a, Theorem 2.21]. □

Corollary 5.9.
1. *If* $\pi(a) = a$ *for all* $a \in \mathrm{supp}(x)$ *then* $\pi{\cdot}x = x$.
2. *If* $\pi(a) = \pi'(a)$ *for every* $a \in \mathrm{supp}(x)$ *then* $\pi{\cdot}x = \pi'{\cdot}x$.
3. $a \notin \mathrm{supp}(x)$ *if and only if* $\exists b.b \notin \mathrm{supp}(x) \wedge (b\ a){\cdot}x = x$.

Proof. Parts 1 and 2 are just reformulations of Theorem 5.8. Part 3 follows (for details see [Gab12b, Corollary 2.2.7]). □

Definition 5.10.
- Call $x \in |\mathsf{X}|$ **equivariant** when $\mathrm{supp}(x) = \varnothing$.
- Call a function $f \in |\mathsf{X}| \to |\mathsf{Y}|$ **equivariant** when $\pi{\cdot}f(x) = f(\pi{\cdot}x)$ for all permutations π and $x \in |\mathsf{X}|$.

Lemma 5.11.
- $x \in |\mathsf{X}|$ *is equivariant if and only if* $\pi{\cdot}x = x$ *for all* π.
- *If* $f \in |\mathsf{X}| \to |\mathsf{Y}|$ *is equivariant (Definition 5.10) then* $\mathrm{supp}(f(x)) \subseteq \mathrm{supp}(x)$ *for all* $x \in |\mathsf{X}|$.

Proof. The first part is from parts 1 and 3 of Corollary 5.9. For the second part, see [Gab12b, Lemma 2.3.3]. □

5.2. *Models*

We now construct a notion of model. Henkin models [BBK04, Hen50] assume functions, and application in the syntax is translated directly to function application in the semantics. Our model is similar except that we are in a *nominal* setting, so we have names and binding in the semantics, and we translate atoms a and atoms-abstraction $[a]$ directly to atoms and atoms-abstraction in our model.

Definition 5.12. An **interpretation** $\mathcal{I}$ is an assignment to each type α of a nominal set $[\![\alpha]\!]$ together with the following data:

1. For each atom $a \in \mathbb{A}_\alpha$ and constant $\mathsf{f} : \alpha$ elements $a^{\mathcal{I}} \in [\![\alpha]\!]$ and $\mathsf{f}^{\mathcal{I}} \in [\![\alpha]\!]$.
2. For each $x \in [\![\beta]\!]$ and $a \in \mathbb{A}_\alpha$, an element $[a]x \in [\![\alpha{\to}\beta]\!]$ such that $a \notin \mathrm{supp}([a]x)$.
3. For each $x \in [\![\alpha{\to}\beta]\!]$ and $y \in [\![\alpha]\!]$, an element $x \bullet y \in [\![\beta]\!]$.
4. A preorder $\lesssim$ on $[\![o]\!]$.

$\mathcal{I}$ must be *equivariant* in the sense that:

$$\pi{\cdot}a^{\mathcal{I}} = (\pi(a))^{\mathcal{I}} \qquad \pi{\cdot}\mathsf{f}^{\mathcal{I}} = \mathsf{f}^{\mathcal{I}} \qquad \pi{\cdot}[a]x = [\pi(a)]\pi{\cdot}x$$
$$\pi{\cdot}(x \bullet y) = (\pi{\cdot}x) \bullet (\pi{\cdot}y) \qquad x \lesssim y \Rightarrow \pi{\cdot}x \lesssim \pi{\cdot}y$$

Remark 5.13. A few words on Definition 5.12.

- The atom $a \in \mathbb{A}_\alpha$ in the range of the interpretation is an atom of type α; we used these atoms in Definitions 2.3 and 2.10 to build permissive-nominal terms. The element $a^{\mathcal{I}}$ is not an atom, it is just an element of $[\![\alpha]\!]$, but the notation is designed to suggest that we think of $a^{\mathcal{I}}$ as a *model* or *copy* of a in $\mathcal{I}$, and thus the equivariance condition implies that $\mathrm{supp}(a^{\mathcal{I}}) \subseteq \{a\}$.
  So for instance, $\mathrm{supp}(a^{\mathcal{I}}) = \varnothing$ is possible, in which case all atoms of type α map to the same element in $[\![\alpha]\!]$ and $\mathcal{I}$ has a trivial, but perfectly valid, interpretation of atoms.
- Similarly $[a]x$ is not equal to 'abstract a in x' in the sense of Definition 2.10, simply because $x \in [\![\beta]\!]$ is not syntax but semantics. However, $[a]x$ is intended to *model* 'abstract a in x', and if x happens to be the denotation of a term r then indeed $[a]x$ will model $[a]r$. See Definition 5.27.
- Note that we do not assume that $[a]x$ is literally equal to the *nominal atoms-abstraction* of a in x in the sense of equation (35) of [GP01]. Here, $[a]x$ is just the *model* of abstract in $\mathcal{I}$.[13] The condition $a \notin \mathrm{supp}([a]x)$ is therefore necessary.
  To be precise, the map $a, x \mapsto [a]x$ is an instance of an *abstractive function* in the sense of [Gab07b]. See Lemma 5.15.
- By the equivariance condition of Definition 5.12 and part 1 of Lemma 5.11, $\mathrm{supp}(\mathsf{f}^{\mathcal{I}}) = \varnothing$. So in our interpretations, constant symbols must be interpreted as elements with empty support. Treatments of non-equivariant term-formers in nominal terms syntax are in [Gab12c, Subsection 6.3.2] and [Gab12b].

Notation 5.14. For brevity we introduce the following notation:

- We may sugar $([a]x) \bullet y$ to $x[a{\mapsto}y]$ and $\forall^{\mathcal{I}}_\alpha \bullet [a]x$ to $\forall a.x$.
- We may sugar $(\wedge^{\mathcal{I}} \bullet x) \bullet y$ to $x \wedge^{\mathcal{I}} y$ or $x \wedge y$, and $\neg^{\mathcal{I}} \bullet x$ to $\neg x$.
- We may sugar $\neg \top^{\mathcal{I}}$ to $\bot^{\mathcal{I}}$.
- We may sugar $\neg(\neg x \wedge \neg y)$ to $x \vee^{\mathcal{I}} y$ or $x \vee y$, and $\neg x \vee y$ to $x \supset y$, and $(x \supset y) \wedge (y \supset x)$ to $x \Leftrightarrow^{\mathcal{I}} y$.
- We may write "$e \approx e'$" for "$e \lessapprox e'$ and $e' \lessapprox e$".

Lemma 5.15. *Suppose $z \in [\![o]\!]$ and $a, b : \alpha$ and $x \in [\![\alpha]\!]$. Then:*

1. *If $b \notin \mathrm{supp}(z)$ then $z[a{\mapsto}x] = ((b\ a)\cdot z)[b{\mapsto}x]$.*
2. *If $b \notin \mathrm{supp}(z)$ then $\forall a.z = \forall b.(b\ a)\cdot z$.*

Proof. Unpacking the syntactic sugar of Notation 5.14 it suffices to show that $[b](b\ a)\cdot z = [a]z$. Now by assumption $a, b \notin \mathrm{supp}([a]z)$ and by part 1 of Corollary 5.9 $(b\ a)\cdot[a]z = [a]z$. The result follows by equivariance of the interpretation. □

[13] The notion of *model* in Definition 5.16 will impose futher nominal algebra axioms to make $[a]x$ not just an α-abstractor, but also give it $\beta\eta$-style behaviour. Thus, $[a]$ is destined to model λa. Still, $[a]x$ is no more *equal* to a functional abstraction than it is equal to a nominal atoms-abstraction. It is just a model.

$$\begin{array}{llrcl}
(\textbf{moda}) & & a^{\mathcal{I}}[a{\mapsto}x] & = & x \\
(\textbf{mod}\#) & a \notin \mathrm{supp}(z) \Rightarrow & z[a{\mapsto}x] & = & z \\
(\textbf{modapp}) & & (z' \bullet z)[a{\mapsto}x] & = & (z'[a{\mapsto}x]) \bullet (z[a{\mapsto}x]) \\
(\textbf{mod}[]) & c \notin \mathrm{supp}(x) \Rightarrow & ([c]z)[a{\mapsto}x] & = & [c](z[a{\mapsto}x]) \\
(\textbf{modid}) & & z[a{\mapsto}a^{\mathcal{I}}] & = & z \\
(\textbf{mod}\eta) & a \notin \mathrm{supp}(z) \Rightarrow & [a](z \bullet a^{\mathcal{I}}) & = & z
\end{array}$$

Figure 7: Equality in models

$$\begin{array}{llrcl}
(\textbf{Commute}) & & x \wedge y & \approx & y \wedge x \\
(\textbf{Assoc}) & & (x \wedge y) \wedge z & \approx & x \wedge (y \wedge z) \\
(\textbf{Huntington}) & & x & \approx & \neg(\neg x \wedge \neg y) \wedge \neg(\neg x \wedge y) \\
(\forall\textbf{E}) & & \forall a.x & \lessapprox & x \\
(\forall\wedge) & & \forall a.(x \wedge y) & \approx & (\forall a.x) \wedge (\forall a.y) \\
(\forall\vee) & a \notin \mathrm{supp}(y) \Rightarrow & \forall a.(x \vee y) & \approx & (\forall a.x) \vee y
\end{array}$$

Figure 8: Preorder on truth-values

Definition 5.16. Call an interpretation $\mathcal{I}$ a **model** when:

- The axioms in Figure 7 hold.
- The axioms in Figure 8 hold.
- $\lessapprox$ is a **congruence** in the sense that for $x, y \in [\![o]\!]$ and $u \in [\![\alpha]\!]$ and $a \in \mathbb{A}_\alpha$, if $x \lessapprox x'$ and $y \lessapprox y'$ then

$$\neg x' \lessapprox \neg x, \qquad x \wedge y \lessapprox x' \wedge y', \qquad \forall a.x \lessapprox \forall a.x', \qquad \text{and} \qquad x[a{\mapsto}u] \lessapprox x'[a{\mapsto}u].$$

Remark 5.17. Models have a part to model the simply-typed λ-calculus, and a part to model the logic at type o.

- The axioms of Figure 7 handle the simply-typed λ-calculus. Compare with the *nominal Henkin semantics* from [GM11].
- The axioms in Figure 8 handle the logic taking place at type o. These axioms are evolved from [GM08b], where (first-order) logic was axiomatised in nominal algebra. We are using nominal algebra [GM09a] in this paper, but 'secretly' in the sense that we do not develop its formal syntax and semantics.

Broadly speaking, this subsection is obtained by combining [GM11] with [GM08b]. The combination is not off-the-shelf; the systems have been adapted, simplified, and improved; see Subsection 9.1.

Remark 5.18. Our notion of truth in $[\![o]\!]$ is intensional in the sense that for $z', z \in [\![o]\!]$, $z' \approx z$ does not imply $z' = z$. That is, our model allows truth-values to have properties that can distinguish them but do not relate to entailment.

We give an intuitive example; it will be made completely formal in Definition 7.1. Take $[\![o]\!]$ to be some set of predicates, preordered by logical entailment, and take any two distinct predicates ϕ and ψ. Then we expect $\phi \wedge \psi \approx \psi \wedge \phi$ to hold, but $\phi \wedge \psi$ and $\psi \wedge \phi$ are not equal predicates and so will not be equal elements of $[\![o]\!]$.

This modest increase in complexity (and generality) makes it easier to build some models later. In particular, the syntactic model of Subsection 7.1 is easier to construct, because by distinguishing $\approx$ and $=$ we avoid having to take a quotient by logical equivalence, which would be inconvenient. In fact, intensional semantics also have more general justifications from linguistics and computing amongst other fields; see for instance [BBK04] or [Sha85].

REMARK 5.19. Following on from Remark 5.18, note that models themselves are still extensional in the sense that we assume η-equivalence ($(\mathbf{mod}\eta)$ in Figure 7 and, in the syntax, $(\mathbf{stx}\eta)$ in Figure 2). This is not absolutely necessary but it does make things easier because every $z \in [\![\alpha{\to}\beta]\!]$ has the form $[a]z'$ for $z' = z \bullet a$, where $a \notin \text{supp}(z)$. (Such an a always exists because by assumption $\text{supp}(z)$ is finite.)

These are purely design decisions, made to keep the definitions and proofs as simple as possible.

Lemmas 5.20 to 5.23 are basic corollaries of the axioms of being a model:

LEMMA 5.20. *The following all hold:*

- $\text{supp}(a^{\mathcal{I}}) \subseteq \{a\}$.
- $\text{supp}(\mathsf{f}^{\mathcal{I}}) = \varnothing$.
- $\text{supp}([a]x) \subseteq \text{supp}(x) \setminus \{a\}$.
- $\text{supp}(x' \bullet x) \subseteq \text{supp}(x') \cup \text{supp}(x)$.

Proof. Using the equivariance condition on $\mathcal{I}$ and part 3 of Corollary 5.9, and our assumption that $a \notin \text{supp}([a]x)$. □

COROLLARY 5.21. $\text{supp}(\forall a.x) \subseteq \text{supp}(x) \setminus \{a\}$ *and* $\text{supp}(z[a{\mapsto}x]) \subseteq (\text{supp}(z) \setminus \{a\}) \cup \text{supp}(x)$.

Proof. From Lemma 5.20 recalling the syntactic sugar of Notation 5.14. □

LEMMA 5.22. 1. $\forall a.\top^{\mathcal{I}} \approx \top^{\mathcal{I}}$ *and* $\forall a.\bot^{\mathcal{I}} \approx \bot^{\mathcal{I}}$.
2. *If* $z \in [\![o]\!]$ *and* $a \notin \text{supp}(z)$ *then* $\forall a.z \approx z$.

Proof. To prove $\forall a.\top^{\mathcal{I}} \approx \top^{\mathcal{I}}$ we reason as follows, where we use facts of Boolean algebras and congruence properties without comment:

$$\forall a.\top^{\mathcal{I}} \;\approx\; \forall a.(\top^{\mathcal{I}} \vee \top^{\mathcal{I}}) \;\overset{(\forall\vee)}{\approx}\; (\forall a.\top^{\mathcal{I}}) \vee \top^{\mathcal{I}} \;\approx\; \top^{\mathcal{I}}.$$

Here we can use $(\forall\vee)$ because by assumption in Definition 5.12 $\top^{\mathcal{I}}$ is equivariant.

By properties of Boolean algebra $\bot^{\mathcal{I}} \lessapprox \forall a.\bot^{\mathcal{I}}$. Conversely by $(\forall\mathbf{E})$ $\forall a.\bot^{\mathcal{I}} \lessapprox \bot^{\mathcal{I}}$.

For the second part we use the first part. Suppose $a \notin \text{supp}(z)$. Then we reason as follows:

$$\forall a.z \;\approx\; \forall a.(\bot^{\mathcal{I}} \vee z) \;\overset{(\forall\vee)}{\approx}\; (\forall a.\bot^{\mathcal{I}}) \vee z \;\overset{\text{Part 1}}{\approx}\; \bot^{\mathcal{I}} \vee z \;\approx\; z$$

□

LEMMA 5.23. *Suppose* $a : \alpha$.

1. *If* $z \in [\![o]\!]$ *then* $\forall a.z \lessapprox z[a{\mapsto}x]$ *for any* $x \in [\![\alpha]\!]$.

2. *If $z' \lessapprox z[a\mapsto x]$ for all $x \in [\![\alpha]\!]$ and $a \notin \mathrm{supp}(z')$ then $z' \lessapprox \forall a.z$.*

Proof. For the first part, by ($\forall$**E**) $\forall a.z \lessapprox z$. By congruence from Definition 5.16 $(\forall a.z)[a\mapsto x] \lessapprox z[a\mapsto x]$. By Corollary 5.21 and (**mod#**) $(\forall a.z)[a\mapsto x] = \forall a.z$. It follows that $\forall a.z \leq z[a\mapsto x]$.

For the second part, it follows taking $x = a^{\mathfrak{I}}$ and using (**modid**) that $z' \lessapprox z$. By congruence from Definition 5.16 $\forall a.z' \lessapprox \forall a.z$ and by part 2 of Lemma 5.22 $z' \lessapprox \forall a.z$. □

Corollary 5.24. *Suppose $a : \alpha$ and $z \in [\![o]\!]$. Then $\forall a.z$ is a $\lessapprox$-greatest lower bound for $\{z[a\mapsto x] \mid x \in [\![\alpha]\!]\}$.*

Proof. By part 1 of Lemma 5.23 $\forall a.z$ is a lower bound. Now suppose $z' \lessapprox z[a\mapsto x]$ for every $x \in [\![\alpha]\!]$. Renaming if necessary using Lemma 5.15 suppose $a \notin \mathrm{supp}(z')$. We use part 2 of Lemma 5.23. □

We mention another nice corollary of Lemma 5.23:

Lemma 5.25. *If $z \in [\![o]\!]$ then $\forall a.\forall b.z \approx \forall b.\forall a.z$.*

Proof. By ($\forall$**E**) $\forall a.\forall b.z \lessapprox z$. By congruence of $\forall$ in Definition 5.16 $\forall b.\forall a.\forall a.\forall b.z \lessapprox \forall b.\forall a.z$. By definition $\forall b.\forall a.\forall a.\forall b.z = \forall \bullet [b]\forall \bullet [a]\forall \bullet [a]\forall \bullet [b]z$. Now from Lemma 5.11 and equivariance of $\mathfrak{I}$ in Definition 5.12 it follows that $b \notin \mathrm{supp}(\forall \bullet [a]\forall \bullet [a]\forall \bullet [b]z)$ and $a \notin \mathrm{supp}(\forall \bullet [a]\forall \bullet [b]z)$. Thus using ($\forall$**E**) we have $\forall b.\forall a.\forall a.\forall b.z = \forall a.\forall b.z$.

So $\forall a.\forall b.z \lessapprox \forall b.\forall a.z$ and by symmetry we are done. □

5.3. *Valuations*

Definition 5.26. Given a model $\mathfrak{I}$, a **valuation** ζ to $\mathfrak{I}$ is a map on variables such that for every variable X, $\zeta(X) \in [\![\mathit{type}(X)]\!]$ and $\mathrm{supp}(\zeta(X)) \subseteq \mathit{pms}(X)$.

Definition 5.27. Suppose ζ is a valuation to a model $\mathfrak{I}$ and suppose $r : \alpha$. Define an **interpretation** $[\![r]\!]_\zeta \in [\![\alpha]\!]$ by:

$$[\![a]\!]_\zeta = a^{\mathfrak{I}} \qquad [\![\mathsf{f}]\!]_\zeta = \mathsf{f}^{\mathfrak{I}} \qquad [\![\pi\cdot X]\!]_\zeta = \pi\cdot\zeta(X)$$
$$[\![rs]\!]_\zeta = [\![r]\!]_\zeta \bullet [\![s]\!]_\zeta \qquad [\![[a]r]\!]_\zeta = [a][\![r]\!]_\zeta$$

In the rest of this subsection we prove some standard results which will be useful later. The most complex of these is probably Proposition 5.31.

Notation 5.28. Suppose $X : \alpha$ and $x \in [\![\alpha]\!]$. Define $\zeta[X{:=}x]$ by:

$$(\zeta[X{:=}x])(X) = x \qquad (\zeta[X{:=}x])(Y) = \zeta(Y)$$

Lemma 5.29. 1. $[\![\pi\cdot r]\!]_\zeta = \pi\cdot[\![r]\!]_\zeta$
2. $[\![r[X{:=}s]]\!]_\zeta = [\![r]\!]_{\zeta[X:=[\![s]\!]_\zeta]}$
3. $[\![r[a\mapsto s]]\!]_\zeta = [\![r]\!]_\zeta[a\mapsto[\![s]\!]_\zeta]$

Proof. The first part is a routine induction on r. The second part is by a routine induction on r, using the first part for the case $r = \pi{\cdot}X$.

The third part is by unpacking Definition 5.27, recalling from Notation 2.15 that $r[a{\mapsto}s]$ is syntactic sugar for $([a]r)s$ and from Notation 5.14 that $\llbracket r\rrbracket_\varsigma[a{\mapsto}\llbracket s\rrbracket_\varsigma]$ is syntactic sugar for $([a]\llbracket r\rrbracket_\varsigma) \bullet \llbracket s\rrbracket_\varsigma$. $\square$

Lemma 5.30. $\mathrm{supp}(\llbracket r\rrbracket_\varsigma) \subseteq \mathit{fa}(r)$.

Proof. By induction on r.

- *The case of a.* By Lemma 5.20 $\mathrm{supp}(a^{\jmath}) \subseteq \{a\} = \mathit{fa}(a)$.
- *The case of $\pi{\cdot}X$.* By assumption $\mathrm{supp}(\varsigma(X)) \subseteq \mathit{pms}(X)$. So using Proposition 5.7 we have $\mathrm{supp}(\pi{\cdot}\varsigma(X)) = \pi{\cdot}\mathrm{supp}(\varsigma(X)) \subseteq \pi{\cdot}\mathit{pms}(X) = \mathit{fa}(\pi{\cdot}X)$.
- *The case of* f. By Lemma 5.20 $\mathrm{supp}(\mathsf{f}^{\jmath}) = \varnothing = \mathit{fa}(\mathsf{f})$.
- *The case of $r'r$.* We reason as follows:

$$\begin{aligned}
\mathrm{supp}(\llbracket r'r\rrbracket_\varsigma) &= \mathrm{supp}(\llbracket r'\rrbracket_\varsigma \bullet \llbracket r\rrbracket_\varsigma) && \text{Definition 5.27}\\
&\subseteq \mathrm{supp}(\llbracket r'\rrbracket_\varsigma) \cup \mathrm{supp}(\llbracket r\rrbracket_\varsigma) && \text{Lemma 5.20}\\
&\subseteq \mathit{fa}(r') \cup \mathit{fa}(r) && \text{Ind. Hyp.}\\
&= \mathit{fa}(r'r) && \text{Definition 2.20}
\end{aligned}$$

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

$$\begin{aligned}
\mathrm{supp}(\llbracket [a]r\rrbracket_\varsigma) &= \mathrm{supp}([a]\llbracket r\rrbracket_\varsigma) && \text{Definition 5.27}\\
&\subseteq \llbracket r\rrbracket_\varsigma \setminus \{a\} && \text{Condition 2, Def. 5.27}\\
&\subseteq \mathit{fa}(r) \setminus \{a\} && \text{Ind. Hyp.}\\
&= \mathit{fa}([a]r) && \text{Definition 2.20}
\end{aligned}$$

$\square$

Proposition 5.31. 1. *If $r =_\alpha r'$ then $\llbracket r\rrbracket_\varsigma = \llbracket r'\rrbracket_\varsigma$.*
2. *If $r =_{\alpha\beta\eta} r'$ then $\llbracket r\rrbracket_\varsigma = \llbracket r'\rrbracket_\varsigma$.*

Proof. Both parts are by induction on the derivation of $=_\alpha$ and $=_{\alpha\beta\eta}$ respectively. The interesting cases are:

- *Rule* $(\mathbf{stx}\alpha)$. Suppose $a, b \notin \mathit{fa}(r)$. By Lemma 5.30 $a, b \notin \mathrm{supp}(\llbracket r\rrbracket_\varsigma)$. By part 2 of Corollary 5.9 $(b\ a){\cdot}\llbracket r\rrbracket_\varsigma = \llbracket r\rrbracket_\varsigma$. It follows by part 1 of Lemma 5.29 that $\llbracket (b\ a){\cdot}r\rrbracket_\varsigma = \llbracket r\rrbracket_\varsigma$.
- *Rule* $(\mathbf{stxa})$. By $(\mathbf{moda})$.
- *Rule* $(\mathbf{stx}\#)$. Suppose $a \notin \mathit{fa}(r)$. By Lemma 5.30 $a \notin \mathrm{supp}(\llbracket r\rrbracket_\varsigma)$. By $(\mathbf{mod}\#)$ $\llbracket r\rrbracket_\varsigma[a{\mapsto}\llbracket t\rrbracket_\varsigma] = \llbracket r\rrbracket_\varsigma$. The result follows since by construction in Definition 5.27, $\llbracket r\rrbracket_\varsigma[a{\mapsto}\llbracket t\rrbracket_\varsigma] = \llbracket r[a{\mapsto}t]\rrbracket_\varsigma$.
- Similarly, $(\mathbf{stx}\bullet)$ follows from $(\mathbf{modapp})$, $(\mathbf{stx}[])$ follows from $(\mathbf{mod}[])$, and $(\mathbf{stxid})$ follows from $(\mathbf{modid})$.

$\square$

Lemma 5.32. *If $\varsigma(X) = \varsigma'(X)$ for every $X \in \mathit{fv}(r)$ then $\llbracket r\rrbracket_\varsigma = \llbracket r\rrbracket_{\varsigma'}$.*

Proof. By a routine induction on r. $\square$

6. Interpretation of (maximising) hypersequents

6.1. *Interpretation of hypersequents*

For this subsection fix some model $\mathfrak{I}$ (Definition 5.16). ς will range over valuations (Definition 5.26) to $\mathfrak{I}$. Recall also that ϕ ranges over formulae, that is, over terms of type o.

Hypersequents are conjunctions of disjunctions of formulae; we now show how to interpret hypersequents and the notion of hypersequent rewriting (derivations) from Definition 3.3 (we treat maximising hypersequents in Subsection 6.2) in $\mathfrak{I}$, culminating with a soundness result Theorem 6.6.

Definition 6.1. Extend the interpretation of Definition 5.27 to sequents and hypersequents as follows:

$$\boxed{\begin{array}{rclr} [\![\phi_1 \vee \ldots \vee \phi_n]\!]_\varsigma &=& [\![\phi_1]\!]_\varsigma \vee^{\mathfrak{I}} \cdots \vee^{\mathfrak{I}} [\![\phi_n]\!]_\varsigma \qquad & [\![\varnothing]\!]_\varsigma = \bot^{\mathfrak{I}} \\ [\![\Phi_1 \wedge \ldots \wedge \Phi_m]\!]_\varsigma &=& [\![\Phi_1]\!]_\varsigma \wedge^{\mathfrak{I}} \cdots \wedge^{\mathfrak{I}} [\![\Phi_m]\!]_\varsigma \qquad & [\![\varnothing]\!]_\varsigma = \top^{\mathfrak{I}} \end{array}}$$

So $\vee$ and $\wedge$ in hypersequents are interpreted as conjunction and disjunction in the model. The right-hand column may seem confusing, but $\varnothing$ at the top is an empty sequent, whereas $\varnothing$ at the bottom is an empty hypersequent;[14] by convention, if a sequent is empty its denotation is $\bot^{\mathfrak{I}}$ and if a hypersequent is empty its denotation is $\top^{\mathfrak{I}}$.

Proposition 6.2. *If* $\dfrac{\mathcal{H}}{\mathcal{H}'}$ *is an instance of an* $(\alpha *)$ *rule or of* $(\beta\wedge)$ *then* $[\![\mathcal{H}]\!]_\varsigma \approx [\![\mathcal{H}']\!]_\varsigma$.

Proof. $(\alpha\neg\neg)$, $(\alpha\top)$, $(\alpha\mathbf{EM})$, $(\alpha\neg\wedge)$, and $(\beta\wedge)$ follow from (**Commute**), (**Assoc**), and (**Huntington**) (see [McC97] or [Hun33, Postulate 6, page 179]). $(=_{\alpha\beta\eta})$ follows from Proposition 5.31. ☐

Lemma 6.3. *Suppose* $a : \alpha$ *and* $t : \alpha$. *Then* $[\![\phi[a{\mapsto}t]]\!]_\varsigma \lessapprox [\![\exists a.\phi]\!]_\varsigma$ *and* $[\![\forall a.\phi]\!]_\varsigma \lessapprox [\![\phi[a{\mapsto}t]]\!]_\varsigma$.

Proof. We consider just the $\forall$ part; the $\exists$ part follows taking duals. We reason as follows:

$$\begin{array}{rll} [\![\forall a.\phi]\!]_\varsigma &= \forall^{\mathfrak{I}}_\alpha \bullet [a][\![\phi]\!]_\varsigma & \text{Definition 5.27} \\ &\lessapprox [\![\phi]\!]_\varsigma[a{\mapsto}[\![t]\!]_\varsigma] & \text{Part 1 Lemma 5.23} \\ &= [\![\phi[a{\mapsto}t]]\!]_\varsigma & \text{Part 3 Lemma 5.29} \end{array}$$

☐

Corollary 6.4. *If* $\dfrac{\mathcal{H}}{\mathcal{H}'}$ *is an instance of* $(\gamma\neg\forall)$ *or* $(\gamma\exists)$ *then* $[\![\mathcal{H}']\!]_\varsigma \approx [\![\mathcal{H}]\!]_\varsigma$.

Proof. We consider just the case of $(\gamma\exists)$; the case of $(\gamma\neg\forall)$ is dual. Elements of a sequent in a hypersequent are interpreted disjunctively, so it suffices to prove that $[\![\phi[a{\mapsto}r]]\!]_\varsigma \lessapprox [\![\exists a.\phi]\!]_\varsigma$. This follows by Lemma 6.3. ☐

Lemma 6.5. *If* $\dfrac{\mathcal{H}}{\mathcal{H}'}$ *is an instance of* $(\delta^-\forall)$ *or* $(\delta^-\neg\exists)$ *then* $[\![\mathcal{H}']\!]_\varsigma \approx \top^{\mathfrak{I}}$ *implies* $[\![\mathcal{H}]\!]_\varsigma \approx \top^{\mathfrak{I}}$.

[14] Both are empty sets, but of different types.

Proof. We consider just the case of $(\delta^-\forall)$; the case of $(\delta^-\neg\exists)$ is dual. Suppose $\mathcal{H} = (\phi \vee \Phi) \wedge \mathcal{H}'$ where ϕ is the focal formula of the instance of $(\delta^-\forall)$ so that $a \notin \mathit{fa}(\Phi)$.

Suppose $[\![(\phi \vee \Phi) \wedge \mathcal{H}']\!]_\varsigma \approx \top^{\mathfrak{I}}$. By part 1 of Lemma 5.22 $\top^{\mathfrak{I}} \approx \forall a.\top^{\mathfrak{I}}$. Using $(\forall\wedge)$ and $(\forall\vee)$ and Lemma 5.30 and part 2 of Lemma 5.22, we calculate that $\forall a.[\![\phi \vee \Phi]\!]_\varsigma \approx [\![(\forall a.\phi) \vee \Phi]\!]_\varsigma$. Now we did not assume in $(\delta^-\forall)$ that $a \notin \mathit{fa}(\mathcal{H})$ (where we take $\mathit{fa}(\bigwedge\bigvee\phi) = \bigcup \mathit{fa}(\phi)$), but we do know by $(\forall\mathbf{E})$ that $\forall a.[\![\mathcal{H}]\!]_\varsigma \lessapprox [\![\mathcal{H}]\!]_\varsigma$. The result follows. □

Theorem 6.6. *If $\mathcal{H}$ rewrites to $\mathcal{H}'$ by the rules in Definition 3.3 and Figure 3 then $[\![\mathcal{H}']\!]_\varsigma \approx \top^{\mathfrak{I}}$ implies $[\![\mathcal{H}]\!]_\varsigma \approx \top^{\mathfrak{I}}$.*

As a corollary, if $\mathcal{H}$ rewrites to $\varnothing$ then $[\![\mathcal{H}]\!]_\varsigma \approx \top^{\mathfrak{I}}$.

Proof. From Proposition 6.2, Corollary 6.4, and Lemma 6.5, and the fact that $[\![\varnothing]\!]_\varsigma = \top^{\mathfrak{I}}$. □

The result corresponding to Theorem 6.6 for *maximising* hypersequents is Theorem 6.18. Before considering maximising hypersequents, we need to tie up one small loose end:

Lemma 6.7. *If $z \approx \top^{\mathfrak{I}}$ then $z \approx \forall a.z$. Also, $z \approx \top^{\mathfrak{I}}$ if and only if $\forall a.z \approx \top^{\mathfrak{I}}$.*

Proof. Suppose $z \approx \top^{\mathfrak{I}}$. Then $\forall a.z \approx \forall a.\top^{\mathfrak{I}}$ and by part 1 of Lemma 5.22 $\forall a.z \approx \top^{\mathfrak{I}}$. It follows by transitivity of $\approx$ that $z \approx \forall a.z$.

Now suppose $\forall a.z \approx \top^{\mathfrak{I}}$. By $(\forall\mathbf{E})$ $\forall a.z \lessapprox z$. It follows that $z \approx \top^{\mathfrak{I}}$. □

Remark 6.8. In Theorem 6.6 we discuss $[\![\mathcal{H}]\!]_\varsigma \approx \top^{\mathfrak{I}}$. By Lemma 6.7 this equality has the value of a universally quantified closure.

The use of nominal models buys us some simplicity in that we do not need to explicitly quantify or λ-abstract over free atoms. In Theorem 6.6 we do not have to say something of the form 'for all valuations of atoms to denotational elements $[\![\mathcal{H}]\!] = \top$' or '$[\![\lambda x_1, \ldots, x_n.\mathcal{H}]\!] = \lambda x_1, \ldots, x_n.\top$ where $\{x_1, \ldots, x_n\}$ include the variables free in $\mathcal{H}$'. This is all handled automatically when we write $[\![\mathcal{H}]\!]_\varsigma = \top^{\mathfrak{I}}$.

Also, nothing in the models so far insists that there be anything specifically non-nominal to quantify *over*. (This echoes an observation made in a slightly different context in [GM11, Theorem 3.15] that the nominal models of [GM11] always have 'sufficiently many points'.)

It is traditional to add fresh constant symbols or *parameters* to the signature of a language to help us interpret universally quantified formulae. In nominal techniques the need for this structure is fulfilled by the specific and rich structure of atoms.

6.2. *Interpretation of maximising hypersequents*

We now show how to interpret the maximising hypersequents of Section 4 in models. This extra power corresponds nicely to extra structure in the models, which we now discuss:

Definition 6.9. Suppose $a:\alpha$ and $z\in[\![\alpha\to o]\!]$. Call $x\in[\![\alpha]\!]$ a **maximiser** for z, and say that the element x **maximises** z, when

1. $\mathrm{supp}(x)\subseteq\mathrm{supp}(z)$ and
2. $z\bullet x\approx\exists^{\mathcal{I}}_{\alpha}\bullet z$.

Say that $\mathcal{I}$ **has maximisers** for every type α, when every $z\in[\![\alpha\to o]\!]$ has at least one maximiser.
Conversely we can say that $x\in[\![\alpha]\!]$ **minimises** $z\in[\![\alpha\to o]\!]$ when $\mathrm{supp}(x)\subseteq\mathrm{supp}(z)$ and $z\bullet x\approx\forall^{\mathcal{I}}_{\alpha}\bullet z$.

Remark 6.10. We insist on $\mathrm{supp}(x)\subseteq\mathrm{supp}(z)$ in condition 1 of Definition 6.9 to ensure that in Proposition 6.17 when we build $\varsigma[X_i{:=}x_i]_1^n$, this will be a valuation (Definition 5.26).

By part 2 of Lemma 5.23 (and extensionality $(\mathbf{mod}\eta)$) the condition $z\bullet x\approx\exists^{\mathcal{I}}_{\alpha}\bullet z$ is just a fancy way of saying that $z\bullet x'\lessapprox z\bullet x$ for all x', in other words, that x is an element that *maximises* $z\bullet x$.

Another way to understand the condition $\mathrm{supp}(x)\subseteq\mathrm{supp}(z)$ is to view z as an open predicate: open, and thus parameterised over the atoms in its support. Then, whatever values x assumes should depend only on those parameters.

We note here, as we noted in Remark 6.8, that there is nothing in the model to insist that the atoms ever get instantiated to 'actual values'. The point of the nominal models is that we can and do reason directly on open elements, and indeed the distinction between open and closed is no longer equal to the distinction between syntax and semantics.

In particular there is no need to explicitly abstract over the free atoms of z and write 'for all closing λ-abstractions' or 'for all substitutions'. The nominal models make it possible to talk about maximising elements directly and without lifting them to higher types to account for syntactic parameters.

We now take a moment to unpack and verify the dualisation of $\forall$ and $\exists$ with respect to maximisation/minimisation:

Proposition 6.11. *Suppose $z'\in[\![o]\!]$ and $a:\alpha$ and $x\in[\![\alpha]\!]$. Then x minimises $[a]z'$ if and only if x maximises $[a]\neg z'$.*

Proof. Suppose $\mathrm{supp}(x)\subseteq\mathrm{supp}(z')\setminus\{a\}$. It suffices to show that $z'[a\mapsto x]\approx\forall a.z'$ if and only if $(\neg z')[a\mapsto x]\approx\exists a.\neg z'$. But $\exists a.z'$ is syntactic sugar for $\neg\forall a.\neg z'$. The result follows using $(\mathbf{modapp})$ and $(\mathbf{mod}\#)$, and facts of Boolean algebra. □

For the rest of this subsection suppose $\mathcal{I}$ is a model (Definition 5.16), and $\mathcal{I}$ has maximisers.

Definition 6.12. Extend the interpretation of Definition 6.1 to maximising hypersequents as follows:

$$[\![X{\uparrow}[a]\phi]\!]_\varsigma=[\![\exists a.\phi]\!]_\varsigma\supset^{\mathcal{I}}[\![\phi[a\mapsto X]]\!]_\varsigma$$
$$[\![(X_i{\uparrow}[a_i]\phi_i)_1^n]\!]_\varsigma=[\![X_1{\uparrow}[a_1]\phi_1]\!]_\varsigma\wedge^{\mathcal{I}}\cdots\wedge^{\mathcal{I}}[\![X_n{\uparrow}[a_n]\phi_n]\!]_\varsigma$$
$$[\![(X_i{\uparrow}[a_i]\phi_i)_1^n\vdash\mathcal{H}]\!]_\varsigma=[\![(X_i{\uparrow}[a_i]\phi_i)_1^n]\!]_\varsigma\wedge[\![\mathcal{H}]\!]_\varsigma$$

Remark 6.13. In Definition 4.1 we took $fa([a]\phi) \subseteq pms(X)$. From the point of view of the models, it might also suffice to take $fa([a]\phi) = pms(X)$. We allow a subset inclusion because Lemma 5.30 is a subset inclusion, not an equality; it might be that some atoms that appear in a term do not feature in its denotation (think of $\mathit{first\_projection}(a, b)$). It does no harm to leave ourselves a little leeway.

Remark 6.14. In Remarks 4.2 and 4.10 we described maximisers as a form of choice. Now that we have defined the formal semantics we can be more precise about this. Definition 6.12 *does not make a choice*, in the sense of choosing an element. It simply takes the conjunction of $[\![X{\uparrow}[a]\phi]\!]_\varsigma$ (a truth-value) with $[\![\mathcal{H}]\!]_\varsigma$ (another truth-value). This is why we described maximisers as *non-Skolemising*.

What gives this conjunction the effect of a choice is, specifically, the $\exists\varsigma$ in Theorem 6.18 and Corollary 6.19.

It might help to think of maximisers as a form of *delayed* choice.

We mention this particularly in case the reader is tempted to imagine that maximisers are just a syntactic repackaging of Hilbert's choice ϵ. They are not, and this is made semantically precise in Section 8.

Lemma 6.15. *Suppose $a : \alpha$ and $X : \alpha$. Then*

$$[\![X{\uparrow}[a]\phi]\!]_\varsigma \wedge [\![\exists a.\phi]\!]_\varsigma \lessapprox [\![\phi[a{\mapsto}X]]\!]_\varsigma$$
$$[\![X{\uparrow}[a]\neg\phi]\!]_\varsigma \wedge [\![\phi[a{\mapsto}X]]\!]_\varsigma \lessapprox [\![\forall a.\phi]\!]_\varsigma$$

Proof. We just unpack Definition 6.12 and use Definition 5.27 and properties of Boolean algebra. □

Corollary 6.16. *If $\dfrac{\mathcal{P}}{\mathcal{P}'}$ is an instance of $(\delta^{\mathsf{X}}\forall)$ or $(\delta^{\mathsf{X}}\neg\exists)$ then $[\![\mathcal{P}']\!]_\varsigma \lessapprox [\![\mathcal{P}]\!]_\varsigma$.*

Proof. We consider just the case of $(\delta^{\mathsf{X}}\forall)$; the case of $(\delta^{\mathsf{X}}\neg\exists)$ is dual. Suppose

$$\mathcal{P} = (X_i{\uparrow}[a_i]\phi_i)_1^n \vdash (\forall a.\phi \vee \Phi) \wedge \mathcal{H} \quad \text{and}$$
$$\mathcal{P}' = (X_i{\uparrow}[a_i]\phi_i)_1^n, X{\uparrow}[a]\neg\phi \vdash (\phi[a{\mapsto}X] \vee \Phi) \wedge \mathcal{H}.$$

We unpack Definition 6.12 and use Lemma 6.15. □

Proposition 6.17. *Suppose $\mathcal{C}$ is a maximisation condition. Then there exists a valuation ς such that $[\![\mathcal{C}]\!]_\varsigma \approx \top^{\mathcal{I}}$.*

Proof. Choose *any* valuation ς_1. We work by induction on the partial order $\prec$ on variables induced by $\mathcal{C}$ (Definition 4.5) to construct a sequence $\varsigma_1, \varsigma_2, \ldots, \varsigma_n$ such that ς_n is the required valuation.

Start with a $\prec$-minimal element $X_i \in dom(\mathcal{C})$, so that by functionality of $\mathcal{C}$ (Definition 4.5), $\mathcal{C} = \{X_i{\uparrow}[a_i]\phi_i\} \cup \mathcal{C}'$ where $\mathcal{C}' = \mathcal{C}\backslash\{X_i{\uparrow}[a_i]\phi_i\}$. We assumed X_i is $\prec$-minimal so by Lemma 4.7 we have $fv(\phi_i) \cap dom(\mathcal{C}) = \varnothing$. By assumption $\mathcal{I}$ has maximisers so there exists an x_i maximising $[\![\phi_i]\!]_{\varsigma_1}$. Set $\varsigma_2 = \varsigma_1[X_i{:=}x_i]$.

Repeat this process for $\mathcal{C}'$ and work our way up $\prec$ to arrive at $\varsigma_n = \varsigma_1[X_i{:=}x_i]_1^n$ where n is the cardinality of $dom(\mathcal{C})$. Using Lemma 5.32 we can verify that x_i maximises $[\![[a_i]\phi_i]\!]_{\varsigma_n}$ for each $1{\leq}i{\leq}n$, as required.

We must also check that ζ_n is a valuation; it suffices to show that $\mathrm{supp}(x_i) \subseteq pms(X_i)$ for each i. This follows from condition 1 of Definition 6.9, Lemma 5.30, and Definition 4.1. □

Theorem 6.18 is to maximising hypersequents as Theorem 6.6 was to hypersequents:

THEOREM 6.18. *If $\mathcal{P}$ rewrites to $\mathcal{P}'$ by the rules of Definition 4.11 then*

$$\exists\zeta.[\![\mathcal{P}']\!]_\zeta \approx \top^{\mathcal{J}} \quad \textit{implies} \quad \exists\zeta.[\![\mathcal{P}]\!]_\zeta \approx \top^{\mathcal{J}}.$$

As a corollary, if $\mathcal{P}$ rewrites to $\mathcal{C} \vdash \varnothing$ then $\exists\zeta.[\![\mathcal{P}]\!]_\zeta \approx \top^{\mathcal{J}}$.

Proof. From Proposition 6.2, Corollary 6.4, Corollary 6.16, and Proposition 6.17 (existence of maximisers). □

COROLLARY 6.19. *If $\varnothing \vdash \mathcal{H}$ rewrites to $\mathcal{C} \vdash \varnothing$ by the rules of Definition 4.11 then $\exists\zeta.[\![\mathcal{H}]\!] \approx \top^{\mathcal{J}}$.*

Proof. Direct from Theorem 6.18 using the fact that any ζ satisfies the empty maximisation condition. □

REMARK 6.20. When nominal unknowns X, Y, Z were introduced in [UPG03, UPG04] as unification variables, they had a flavour of existential variables; that is, variables ranging over elements that we try to 'find' to 'make something true'. This is made formal in Theorem 6.18 by the use of $\exists\zeta$; we try to find values that make the denotation of the problem be true.

Atoms are 'universal' in a sense made formal by the following lemma: if $[\![\mathcal{P}]\!]_\zeta = \top^{\mathcal{J}}$ and $a \in \mathbb{A}_\alpha$ then by $(\mathbf{mod}\#)$ from Figure 7 also $[\![\mathcal{P}]\!]_{\zeta}[a\mapsto x] = \top^{\mathcal{J}}$ for every $x \in [\![\alpha]\!]$. Part 3 of Lemma 5.29 relates this semantic substitution to a natural notion of substituting atoms for terms in problems.

Condition 2 of Definition 6.9 and Definition 6.12 make formal a sense in which maximisation conditions restrict existential variables X to make a predicate $\phi[a\mapsto X]$ as true as possible. Using negation, we can then give X a *universal* flavour, by restricting it to range over elements that try to make $\neg\phi[a\mapsto X]$ as true as possible, i.e. $\phi[a\mapsto X]$ as *false* as possible. This is happens in $(\delta^X\forall)$ in Figure 5.

Thus, two distinct notions of 'universal' variable are explained in our nominal semantics—we easily map this to the syntax in Subsection 6.3. Note that variables do not obsolete atoms. We still need atoms to name lemmas, or, in λ-calculus terminology, we still need to be able to write $[a]r$ and apply this to s to make $([a]r) \bullet s$, or write $[a]\phi$ and apply $\forall$ to it to make $\forall \bullet [a]\phi$.

In view of the above, the reader should take the tag 'universal' for atoms a, and 'existential' for variables X, with a pinch of salt.[15] We can always form $\exists a.\phi$ or insert a maximisation condition $X\!\uparrow\![a]\neg\phi$. Ultimately, the precise meanings of atoms and variables are given by the semantics. For the reader's convenience we recall the critical definitions: 5.27, 6.1, 6.9, and 6.12.

[15] A thread of research is devoted to giving X, Y, Z a *universal* flavour: we can generalise nominal terms with λX or $\forall X$ [GL09, DG12a], or make X behave like a schema variable [GM10] (in the sense of 'an axiom-schema'). More on this in Subsection 9.1.
The interaction between a and X is really about names and capture, and such notions arise in a great number of situations—one of which is proof-search.

6.3. *Instantiating X in (maximising) hypersequents*

We can now come full circle and show how the existential flavour of X can be folded back down into the syntax. In other words, we show how to instantiate X in a hypersequent or maximising hypersequent. With the maths we have so far, this is not hard.

We need a little notation:

Definition 6.21. Suppose $s : \mathit{type}(X)$ and $\mathit{fa}(s) \subseteq \mathit{pms}(X)$ so that $[X{:=}s]$ is an instantiation (Definition 2.26). Suppose $\mathcal{H} = \bigwedge \bigvee \phi_i$ is a hypersequent. Then define

$$\mathcal{H}[X{:=}s] = \bigwedge \bigvee (\phi_i[X{:=}s]).$$

If $\mathcal{C} = (X_i{\uparrow}[a_i]\phi_i)_1^n$ and $X \notin \mathit{dom}(\mathcal{C})$ (so $X \notin \{X_i \mid 1{\leq}i{\leq}n\}$) then define

$$\mathcal{C}[X{:=}s] = (X_i{\uparrow}[a_i](\phi_i[X{:=}s]))_1^n.$$

Suppose $\mathcal{P} = (\mathcal{C} \vdash \mathcal{H})$ and $X \notin \mathit{dom}(\mathcal{C})$. Define

$$\mathcal{P}[X{:=}s] = (\mathcal{C}[X{:=}s] \vdash \mathcal{H}[X{:=}s]).$$

Finally, if $X \notin \mathit{fv}(\phi)$ define[16]

$$(\mathcal{C}, X{\uparrow}[a]\phi \vdash \mathcal{H})[X{:=}s] = \big(\mathcal{C}[X{:=}s] \vdash ((\exists a.\phi) \supset (\phi[a{\mapsto}s])) \wedge \mathcal{H}[X{:=}s]\big).$$

We take a moment to state the obvious:

Lemma 6.22. *Suppose $s : \mathit{type}(X)$ and $\mathit{fa}(s) \subseteq \mathit{pms}(X)$, so that $[X{:=}s]$ is an instantiation. Suppose $\mathcal{C} \vdash \mathcal{H}$ is a maximising hypersequent.*

Then if $\mathcal{C}[X{:=}s]$ is a valid maximisation condition then $(\mathcal{C} \vdash \mathcal{H})[X{:=}s]$ is a maximising hypersequent.

Lemma 6.22 is interesting not for the logical content (which is almost directly tautological) but for highlighting the condition that $\mathcal{C}[X{:=}s]$ be a valid maximisation condition. Recall from Subsection 4.2.3 that this depends on $\prec$ remaining well-founded. Whether that holds depends on the precise structures of $\mathcal{C}$ and s.

Lemma 6.23. *Suppose $s : \mathit{type}(X)$ and $\mathit{fa}(s) \subseteq \mathit{pms}(X)$ so that $[X{:=}s]$ is an instantiation. Suppose $\mathcal{P} = (\mathcal{C} \vdash \mathcal{H})$ is a maximising hypersequent and $\mathcal{C}[X{:=}s]$ is a valid maximisation condition.*

Then $[\![\mathcal{P}[X{:=}s]]\!]_\varsigma = [\![\mathcal{P}]\!]_{\varsigma[X:=[\![s]\!]_\varsigma]}$.

Proof. By unpacking Definitions 6.12 and 6.21 and using part 2 of Lemma 5.29. □

Theorem 6.24. *Suppose $s : \mathit{type}(X)$ and $\mathit{fa}(s) \subseteq \mathit{pms}(X)$ so that $[X{:=}s]$ is an instantiation. Suppose $\mathcal{P} = (\mathcal{C} \vdash \mathcal{H})$ is a maximising hypersequent and $\mathcal{C}[X{:=}s]$ is a valid maximisation condition. Then:*

1. *If $\exists\varsigma.[\![\mathcal{P}[X{:=}s]]\!]_\varsigma = \top^{\mathfrak{I}}$ then $\exists\varsigma.[\![\mathcal{P}]\!]_\varsigma = \top^{\mathfrak{I}}$.*
2. *If $\forall\varsigma.[\![\mathcal{P}]\!]_\varsigma \lessapprox [\![\mathcal{P}']\!]_\varsigma$ then $\forall\varsigma.[\![\mathcal{P}[X{:=}s]]\!]_\varsigma \lessapprox [\![\mathcal{P}'[X{:=}s]]\!]_\varsigma$.*

Proof. Both parts are from Lemma 6.23. □

[16]We will know $X \notin \mathit{fv}(\phi)$ by condition 1 in Definition 4.5 (well-foundedness) so $(\exists a.\phi)[X{:=}s] = \exists a.\phi$ and $(\phi[a{\mapsto}X])[X{:=}s] = \phi[a{\mapsto}s]$.

With what we now have, we can put the instantiations of Subsections 3.2 and 4.2 (see Remark 3.6) in a more general context:

Proposition 6.25. *Suppose $s : type(X)$ and $fa(s) \subseteq pms(X)$ so that $[X{:=}s]$ is an instantiation.*

1. *Suppose $\mathcal{H}$ and $\mathcal{H}'$ are hypersequents. Then $\mathcal{H} \Rightarrow \mathcal{H}'$ implies $\mathcal{H}[X{:=}s] \Rightarrow \mathcal{H}'[X{:=}s]$.*
2. *Suppose $\mathcal{C}$ and $\mathcal{C}'$ are maximisation conditions and $X \notin dom(\mathcal{C}) \cup dom(\mathcal{C}')$. Suppose $\mathcal{C}[X{:=}s]$ and $\mathcal{C}'[X{:=}s]$ are valid maximisation conditions.*
   Then $(\mathcal{C} \vdash \mathcal{H}) \Rightarrow (\mathcal{C}' \vdash \mathcal{H}')$ implies $(\mathcal{C} \vdash \mathcal{H})[X{:=}s] \Rightarrow (\mathcal{C}' \vdash \mathcal{H}')[X{:=}s]$.

Proof. For part 1 of this result we check each of the rules in Figure 3 and see that they are preserved by instantiating variables: By part 4 of Proposition 2.29 the side-condition $\phi =_{\alpha\beta\eta} \phi'$ in $(=_{\alpha\beta\eta})$ is preserved . By Proposition 2.25 the side-condition $r : type(a)$ in $(\gamma\neg\forall)$ is preserved. Using part 1 of Proposition 2.29 the side-condition $a \notin fa(\Phi)$ is preserved.

Part 2 of this result follows using Lemma 6.22 and some routine calculations on syntax. ◻

7. Building models by hand

We set about building some concrete instances of the notion of model given in Subsection 5.2. Two possibilities naturally present themselves: build a model out of quotiented syntax; or build a nominal model out of some standard model of higher-order logic over ordinary (non-nominal) sets. These are Subsections 7.1 and 7.2 respectively. We also return to these in Subsection 8.3, when we consider choice.

7.1. *The syntactic model*

Recall from Definition 2.22 the definition of $=_{\alpha\beta\eta}$ as the least congruence containing $(\mathbf{stx}\alpha)$, rules $(\mathbf{stxa})$ to $(\mathbf{stxid})$, and $(\mathbf{stx}\eta)$.

Definition 7.1. Let $\lessapprox$ be the least preorder congruence containing $=_{\alpha\beta\eta}$ and also such that:

$$\begin{array}{lll}
\phi \wedge \psi \approx \psi \wedge \phi & & (\phi \wedge \psi) \wedge \chi \approx \phi \wedge (\psi \wedge \chi) \\
\phi \approx \neg(\neg\phi \wedge \neg\psi) \wedge \neg(\neg\phi \wedge \psi) & & \\
\forall a.\phi \lessapprox \phi & & \\
\forall a.(\phi \wedge \psi) \approx (\forall a.\phi) \wedge (\forall a.\psi) & & a \notin fa(\psi) \Rightarrow \forall a.(\phi \vee \psi) \approx (\forall a.\phi) \vee \psi
\end{array}$$

Here, as before in Notation 5.14, we write "$\phi \approx \psi$" as shorthand for "$\phi \lessapprox \psi$ and $\psi \lessapprox \phi$".

Definition 7.2. Define $[\![\alpha]\!]^{\mathcal{S}} = \{[r]_{\alpha\beta\eta} \mid r : \alpha\}$. Give this the *pointwise* permutation action $\pi{\cdot}[r]_{\alpha\beta\eta} = [\pi{\cdot}r]_{\alpha\beta\eta}$.

It is easy to check that Definition 7.2 determines a nominal set, in which $[r]_{\alpha\beta\eta}$ is supported by $fa(r)$.

Definition 7.3. Define an interpretation $\mathcal{S}$ (Definition 5.12) by assigning to each type α the nominal set $[\![\alpha]\!]^{\mathcal{S}}$, along with the following data:

- $a^{\mathcal{S}} = [a]_{\alpha\beta\eta}$ and $f^{\mathcal{S}} = [f]_{\alpha\beta\eta}$.
- $[a][r]_{\alpha\beta\eta} = [[a]r]_{\alpha\beta\eta}$.
- $[r]_{\alpha\beta\eta} \bullet [s]_{\alpha\beta\eta} = [rs]_{\alpha\beta\eta}$.
- $[\phi]_{\alpha\beta\eta} \lessapprox [\psi]_{\alpha\beta\eta}$ if $\phi \lessapprox \psi$.

Lemma 7.4. *$r =_{\alpha\beta\eta} s$ if and only if $\pi{\cdot}r =_{\alpha\beta\eta} \pi{\cdot}s$. As a corollary, Definition 7.3 is equivariant in the sense of Definition 5.12.*

Proof. We note that Definition 2.22 is symmetric in permuting atoms, and therefore so is the relation $=_{\alpha\beta\eta}$ that it defines. $\square$

Remark 7.5. One of the fundamental observations of nominal techniques is that *atoms are symmetric*, and this can be exploited to give fast, convenient, and fully-rigorous proofs of real theorems in real maths papers, such as in [Gab07a],[17] or here in Lemma 7.4. This is made formal by the general nominal meta-principle of *equivariance* for symmetric predicates [Gab11a, Theorem 4.4]. A concrete proof of Lemma 7.4 by induction on the derivation of $r =_{\alpha\beta\eta} s$ is also possible; that would amount to verifying the symmetries inductively.

Lemma 7.6. *$a \notin \mathrm{supp}([[a]r]_{\alpha\beta\eta})$.*

Proof. By $(\mathbf{stx}\alpha)$ if b is fresh (so $b \notin fa(r) \cup \mathrm{supp}([[a]r]_{\alpha\beta\eta})$) then $[a]r =_{\alpha} [b](b\ a){\cdot}r$. It follows that $(b\ a){\cdot}[[a]r]_{\alpha\beta\eta} = [[a]r]_{\alpha\beta\eta}$ and so by part 3 of Corollary 5.9 that $a \notin \mathrm{supp}([[a]r]_{\alpha\beta\eta})$.[18] $\square$

Corollary 7.7. *Definition 7.3 is an interpretation.*

Proof. From Lemmas 7.6 and 7.4. $\square$

We will need Lemma 7.8 in a moment. This is a special case of a more general result [Gab12b, Lemma 7.6.2]:

Lemma 7.8. *If $a \notin \mathrm{supp}([r]_{\alpha\beta\eta})$ then there exists some $r' \approx r$ such that $a \notin fa(r')$.*

Proof. Suppose $a \notin \mathrm{supp}([r]_{\alpha\beta\eta})$. Choose fresh b, so $b \notin fa(r) \cup \mathrm{supp}([r]_{\alpha\beta\eta})$. By part 1 of Corollary 5.9 $(b\ a){\cdot}[r]_{\alpha\beta\eta} = [r]_{\alpha\beta\eta}$ and it follows that $(b\ a){\cdot}r \in [r]_{\alpha\beta\eta}$, that is, $(b\ a){\cdot}r =_{\alpha\beta\eta} r$. We take $r' = (b\ a){\cdot}r$ and note by Lemma 2.21 that $a \notin fa(r')$. $\square$

Proposition 7.9 *looks* like a trivial replay of $(\mathbf{stxa})$ to $(\mathbf{stx}id)$ and $(\mathbf{stx}\eta)$ from Figure 2, but this is deceptive. The freshness side-conditions of the model equalities refer to support of equivalence classes of terms; the freshness side-conditions of the syntactic equalities refer to free atoms of a term.

Proposition 7.9. *The axioms $(\mathbf{moda})$ to $(\mathbf{mod}id)$ and $(\mathbf{mod}\eta)$ from Figure 7 are valid in $\mathcal{S}$.*

Proof. We consider just the case of $(\mathbf{mod}\#)$; the other axioms are no harder. Suppose $a \notin \mathrm{supp}([r]_{\alpha\beta\eta})$. By Lemma 7.8 there exists some r' such that $r' =_{\alpha\beta\eta} r$ and a does not occur at all in r'. In particular, it is not hard to prove that $a \notin fa(r')$. By $(\mathbf{stx}\#)$ $r'[a{\mapsto}t] =_{\alpha\beta\eta} r'$. By congruence of $=_{\alpha\beta\eta}$, also $r[a{\mapsto}t] =_{\alpha\beta\eta} r$ and so $[r[a{\mapsto}t]]_{\alpha\beta\eta} = [r]_{\alpha\beta\eta}$ as required. $\square$

Proposition 7.10. *The axioms $(\mathbf{Commute})$ to $(\forall\vee)$ from Figure 8 are valid in $\mathcal{S}$.*

[17]This paper was submitted in 2003.

[18]Technically, we also need to check that $[[a]r]_{\alpha\beta\eta}$ has finite support, so that we can choose a b fresh for it. This also follows from the symmetries of atoms; for the general statement of the relevant nominal meta-principle, see [Gab11a, Theorem 4.7].

Proof. Most follow directly from the preorder on terms in Definition 7.1. As for Proposition 7.9 we just have to be a little careful about the freshness side-conditions.

There is only one of these: we must show that if $a \notin \mathrm{supp}([\psi]_{\alpha\beta\eta})$ then $\forall a.([\phi]_{\alpha\beta\eta} \vee [\psi]_{\alpha\beta\eta}) \approx (\forall a.[\phi]_{\alpha\beta\eta}) \vee [\psi]_{\alpha\beta\eta}$.

So suppose $a \notin \mathrm{supp}([\psi]_{\alpha\beta\eta})$. Using Lemma 7.8 let ψ' be such that $\psi' =_{\alpha\beta\eta} \psi$ and $a \notin fa(\psi')$. By assumption $\forall a.(\phi \vee \psi') =_{\alpha\beta\eta} (\forall a.\phi) \vee \psi'$. By the congruence properties of $=_{\alpha\beta\eta}$ we are done. $\square$

The arguments above suffice to prove that we have a model:

Theorem 7.11. *$\mathcal{S}$ is a model in the sense of Definition 5.16.*

7.2. *The functional model $\mathcal{F}$*

We now show how to obtain a model in the sense of Definition 5.16 from an 'ordinary' functional model of higher-order logic.

Definition 7.12. A **functional model** $\mathcal{F}$ is an assignment as follows:

- To each base type τ assign a non-empty 'ordinary' set $[\![\tau]\!]^{\mathcal{F}}$.
- To each constant f assign an element $\mathsf{f}^{\mathcal{F}} \in [\![type(\mathsf{f})]\!]^{\mathcal{F}}$.

We extend $[\![\text{-}]\!]^{\mathcal{F}}$ to all types by

$$[\![o]\!]^{\mathcal{F}} = \{\bot, \top\} \text{ (Booleans)} \quad \text{and} \quad [\![\alpha{\to}\beta]\!]^{\mathcal{F}} = [\![\alpha]\!]^{\mathcal{F}} \Rightarrow [\![\beta]\!]^{\mathcal{F}} \text{ (function space).}$$

Fix some functional model $\mathcal{F}$. We now work towards Definition 7.19, in which we construct a nominal model $\mathcal{I}$ in the sense of Definition 5.16, out of $\mathcal{F}$:

Definition 7.13. For each type α write $\mathbb{A}_\alpha \Rightarrow [\![\alpha]\!]^{\mathcal{F}}$ for the set of functions from atoms of type α to $[\![\alpha]\!]^{\mathcal{F}}$. We let ς range over elements of $\prod_\alpha \mathbb{A}_\alpha \Rightarrow [\![\alpha]\!]^{\mathcal{F}}$ (product over all types) and give ς a permutation action by

$$(\pi\cdot\varsigma)(a) = \varsigma(\pi^{\text{-}1}(a)).$$

Remark 7.14. It is a fact that the set $\prod_\alpha \mathbb{A}_\alpha \Rightarrow [\![\alpha]\!]^{\mathcal{F}}$ forms a set with a permutation action (Definition 5.2), but it is not necessarily a nominal set (Definition 5.3). Because of how we use the ς in what follows, this will not be a problem.

Where does the action in Definition 7.13 come from, anyway? It is a special case of the standard *conjugation action* $(\pi\cdot\varsigma)(a) = \pi\cdot\varsigma(\pi^{\text{-}1}(a))$ where we use the trivial action $\pi\cdot x = x$ for every $x \in [\![\alpha]\!]^{\mathcal{F}}$. This is entirely standard; one place where this is discussed specifically in a 'nominal' context is [Gab12b, Definition 2.4.2].

Definition 7.15. Write $[\![\beta]\!]$ for

- the set of functions f from $\prod_\alpha \mathbb{A}_\alpha \Rightarrow [\![\alpha]\!]^{\mathcal{F}}$ to $[\![\beta]\!]^{\mathcal{F}}$ such that for each f there exists a finite set A_f such that if $\varsigma(a) = \varsigma'(a)$ for every $a \in A_f$ then $f(\varsigma) = f(\varsigma')$,
- with a permutation action defined by

$$(\pi\cdot f)(\varsigma) = f(\pi^{\text{-}1}\cdot\varsigma).$$

Lemma 7.16. *$[\![\beta]\!]$ from Definition 7.15 determines a nominal set.*

Proof. It is routine to verify that the permutation action is indeed a group action.

Suppose $\pi(a) = a$ for every $a \in A_f$ (Definition 7.15). By Definition 7.15 $(\pi \cdot f)(\varsigma) = f(\pi^{-1} \cdot \varsigma)$. By Definition 7.13 $(\pi^{-1} \cdot \varsigma)(a) = \varsigma(\pi(a))$. Now by assumption $\varsigma(a) = \varsigma(\pi(a))$ for every $a \in A_f$. Therefore, $(\pi^{-1} \cdot \varsigma)(a) = \varsigma(a)$ for every $a \in A_f$, and so $f(\varsigma) = f(\pi^{-1} \cdot \varsigma)$, and so $(\pi \cdot f)(\varsigma) = f(\varsigma)$. Thus, f has finite support (and is supported by A_f). □

Notation 7.17 is clearly in the same spirit as Notation 5.28:

Notation 7.17. Suppose $a : \alpha$ and $x \in [\![\alpha]\!]^{\mathcal{F}}$. Define $\varsigma[a{:=}x]$ by:

$$(\varsigma[a{:=}x])(a) = x \qquad (\varsigma[a{:=}x])(b) = \varsigma(b)$$

Lemma 7.18. *If $\varsigma(a) = \varsigma'(a)$ for every $a \in \mathrm{supp}(f)$ then $f(\varsigma) = f(\varsigma')$.*

Proof. Suppose $\varsigma(a) = \varsigma'(a)$ for every $a \in \mathrm{supp}(f)$. Recall the definition of A_f from Definition 7.15. It suffices to show that if $a \in A_f \setminus \mathrm{supp}(f)$ and $a : \alpha$ and $x \in [\![\alpha]\!]^{\mathcal{F}}$ then $f(\varsigma[a{:=}x]) = f(\varsigma)$.

Choose fresh $b : \alpha$ (so $b \notin A_f$). By part 1 of Corollary 5.9 $(b\ a) \cdot f = f$, since $a, b \notin \mathrm{supp}(f)$. We reason as follows:

$$f(\varsigma[a{:=}x]) \overset{b \notin A_f}{=} f(\varsigma[a{:=}x][b{:=}\varsigma(a)]) \overset{(b\,a)\cdot f = f}{=} f(\varsigma[b{:=}x]) \overset{b \notin A_f}{=} f(\varsigma)$$

□

Definition 7.19. We define an interpretation $\mathcal{I}$ (Definition 5.12) by assigning to each type α the nominal set $[\![\alpha]\!]$ from Definition 7.15, along with the following data:

- $a^{\mathcal{I}}(\varsigma) = \varsigma(a)$.
- $\mathsf{f}^{\mathcal{I}}(\varsigma) = \mathsf{f}^{\mathcal{F}}$.
- If $f \in [\![\beta]\!]$ and $a \in \mathbb{A}_\alpha$ then $([a]f)(\varsigma) = \lambda x {\in} [\![\alpha]\!]^{\mathcal{F}}. f(\varsigma[a{:=}x])$.
- If $f \in [\![\alpha \to \beta]\!]$ and $g \in [\![\alpha]\!]$ then $(f \bullet g)(\varsigma) = f(\varsigma)(g(\varsigma))$.
- If $f, g \in [\![o]\!]$ then define $f \lessapprox g$ when for every ς, $f(\varsigma) = \top$ implies $g(\varsigma) = \top$.

Also, we interpret $\bot$, $\top$, $\neg$, $\wedge$, and $\forall_\alpha$ from Definition 2.8 as follows:

$$\begin{array}{ll} \bot^{\mathcal{I}}(\varsigma) = \bot & (f \wedge^{\mathcal{I}} g)(\varsigma) = f(\varsigma) \wedge g(\varsigma) \\ \top^{\mathcal{I}}(\varsigma) = \top & \\ (\neg^{\mathcal{I}} f)(\varsigma) = \neg(f(\varsigma)) & (\forall^{\mathcal{I}}_\alpha)(\varsigma) = \lambda y {\in} [\![\alpha \to \beta]\!]^{\mathcal{F}}. \bigwedge_{x \in [\![\alpha]\!]^{\mathcal{F}}} yx \end{array}$$

Above, $\wedge$ and $\bigwedge$ refer to the standard greatest lower bound operation on the set $\{\bot, \top\}$. We now set about checking that Definition 7.19 determines a model in the sense of Definition 5.16. We sketch the non-trivial parts.

Lemma 7.20. *Suppose $a \in \mathbb{A}_\alpha$ and $f \in [\![\beta]\!]$. Then $a \notin \mathrm{supp}([a]f)$.*

Proof. Using Lemma 7.18 it suffices to show that $([a]f)(\varsigma[a{:=}x]) = ([a]f)(\varsigma)$ for all ς. This follows from the definitions, because the information in $\varsigma(a)$ gets overwritten by $[a]f$. □

Lemma 7.21. *Definition 7.19 is equivariant, that is:*

- $\pi \cdot a^{\mathcal{I}} = (\pi(a))^{\mathcal{I}}$.
- $\pi \cdot \mathsf{f}^{\mathcal{I}} = \mathsf{f}^{\mathcal{I}}$ *for* f *one of* $\bot$, $\wedge$, *or* $\forall_\alpha$.

- $\pi{\cdot}[a]f = [\pi(a)]\pi{\cdot}f$.
- $\pi{\cdot}(f \bullet g) = (\pi{\cdot}f) \bullet (\pi{\cdot}g)$.

Proof. We consider each case in turn:

- $(\pi{\cdot}a^{\mathfrak{I}})(\varsigma) = (\pi^{\text{-}1}{\cdot}\varsigma)(a) = \varsigma(\pi(a))$ and $(\pi(a))^{\mathfrak{I}}(\varsigma) = \varsigma(\pi(a))$.
- We consider just the case of $\forall_\alpha$. We note that $\forall^{\mathfrak{I}}_\alpha(\varsigma) = \forall^{\mathfrak{I}}_\alpha(\varsigma')$ for all ς and ς'. It follows that $\pi{\cdot}\forall^{\mathfrak{I}}_\alpha = \forall^{\mathfrak{I}}_\alpha$.
- We reason as follows:

$$\begin{aligned}(\pi{\cdot}[a]f)(\varsigma) &= ([a]f)(\pi^{\text{-}1}{\cdot}\varsigma) \\ &= \lambda x{\in}[\![\alpha]\!]^{\mathcal{F}}.f((\pi^{\text{-}1}{\cdot}\varsigma)[\pi(a){:=}x]) \\ ([\pi(a)]\pi{\cdot}f)(\varsigma) &= \lambda x{\in}[\![\alpha]\!]^{\mathcal{F}}.f((\pi^{\text{-}1}{\cdot}\varsigma)[\pi(a){:=}x])\end{aligned}$$

- The case of $f \bullet g$ is routine.

□

Recall from Notation 5.14 that $f[a{\mapsto}g]$ is syntactic sugar for $([a]f) \bullet g$. We unpack this concretely:

Lemma 7.22. *Suppose $f \in [\![\alpha{\to}\beta]\!]$ and $a \in \mathbb{A}_\alpha$ and $g \in [\![\alpha]\!]$. Then $(f[a{\mapsto}g])(\varsigma) = f(\varsigma[a{:=}g(\varsigma)])$.*

Proof. We just unpack Definition 7.19:

$$(([a]f) \bullet g)(\varsigma) = ([a]f)(\varsigma)(g(\varsigma)) = f(\varsigma[a{:=}g(\varsigma)])$$

□

Remark 7.23. Note in passing that if $f, g \in [\![o]\!]$ then $f \approx g$ if and only if $f = g$ so the functional model (unlike the syntactic model of Subsection 7.1) is extensional on truth-values (cf. Remark 5.18).

Proposition 7.24. *Definition 7.19 satisfies the axioms of Figures 7 and Figure 8.*

Proof. We use Lemma 7.22 without comment:

- *Rule* (**moda**). $(a^{\mathfrak{I}}[a{\mapsto}h])(\varsigma) = (\varsigma[a{:=}h(\varsigma)])(a) = h(\varsigma)$.
- *Rule* (**mod#**). Suppose $a \notin \mathrm{supp}(f)$. By Lemma 7.18 $f(\varsigma) = f(\varsigma[a{:=}h(\varsigma)])$.
- *Rule* (**modapp**). $(f' \bullet f)(\varsigma[a{:=}h(\varsigma)]) = f'(\varsigma[a{:=}h(\varsigma)])(f(\varsigma[a{:=}h(\varsigma)]))$.
- *Rule* (**mod[]**). Suppose $c \notin \mathrm{supp}(h)$.

$(([c]f)[a{\mapsto}h])(\varsigma) = \lambda g.f(\varsigma[c{:=}g(\varsigma), a{:=}h(\varsigma)])$	Def. 7.19, Lem. 7.22
$([c](f[a{\mapsto}h]))(\varsigma) = \lambda g.f(\varsigma[c{:=}g(\varsigma), a{:=}h(\varsigma[c{:=}g(\varsigma)])])$	Def. 7.19, Lem. 7.22
$= \lambda g.f(\varsigma[c{:=}g(\varsigma), a{:=}h(\varsigma)])$	Lemma 7.18

- *Rule* (**modid**). $(f[a{\mapsto}a^{\mathfrak{I}}])(\varsigma) = f(\varsigma[a{:=}\varsigma(a)]) = f(\varsigma)$.
- (**Commute**), (**Assoc**), and (**Huntington**) are routine.
- Rules ($\forall$**E**) and ($\forall\wedge$) are routine.
- *Rule* ($\forall\vee$). Suppose $a : \alpha$ and $a \notin \mathrm{supp}(g)$ and $f, g \in [\![o]\!]$. Then

$$
\begin{aligned}
(\forall_\alpha \bullet [a](f \vee g))(\varsigma) &= \textstyle\bigwedge_{x\in[\![\alpha]\!]^{\mathcal{F}}} \big(f(\varsigma[a{:=}x]) \vee g(\varsigma[a{:=}x])\big) && \text{Definition 7.19}\\
&= \textstyle\bigwedge_{x\in[\![\alpha]\!]^{\mathcal{F}}} (f(\varsigma[a{:=}x]) \vee g(\varsigma)) && \text{Lemma 7.18}\\
&= (\textstyle\bigwedge_{x\in[\![\alpha]\!]^{\mathcal{F}}} f(\varsigma[a{:=}x])) \vee g(\varsigma) && \text{Fact}\\
&= ((\forall_\alpha \bullet [a]f) \vee g)(\varsigma) && \text{Definition 7.19}
\end{aligned}
$$

□

THEOREM 7.25. *Definition 7.19 determines a model in the sense of Definition 5.16.*

Proof. This is Lemmas 7.20 and 7.21 and Proposition 7.24. □

8. Choice

We consider how Hilbert's ϵ can be accommodated in our framework.

A word on notation: we write ϵ for the choice constant in syntax (Definition 8.1), and the ever-so-slightly-more-italic ε for a choice function in semantics (Definition 8.6).

8.1. *Syntax and axiom*

In Definition 2.8 we assumed a fixed but arbitrary set of constants (so long as it contained $\bot$, $\wedge$, and $\forall_\alpha$). Now we add to this list:

DEFINITION 8.1. For each type α assume a constant $\epsilon_\alpha : (\alpha{\to}o){\to}o$. Write $\epsilon_\alpha[a]r$ as $\epsilon_\alpha a.r$ or $\epsilon a.r$.

Add to the definition of $=_{\alpha\beta\eta}$ in Figure 2 the following axiom (and change $=_{\alpha\beta\eta}$ to $=_{\alpha\beta\eta\varepsilon}$ accordingly):

$$(\mathbf{stx}\epsilon) \qquad \forall a.\phi =_{\alpha\beta\eta\varepsilon} \phi[a{\mapsto}\epsilon a.\neg\phi]$$

We call this syntax and the notion of equality $=_{\alpha\beta\eta\varepsilon}$ a syntax **with choice**.

REMARK 8.2. Recall from Notation 2.15 that $\exists a.\phi$ is syntactic sugar for $\neg(\forall_\alpha[a]\neg\phi)$ (for some α), so an alternative to $(\mathbf{stx}\epsilon)$ is $\exists a.\phi =_{\alpha\beta\eta\varepsilon} \phi[a{\mapsto}\epsilon a.\phi]$. These expressions are recognisable as Hilbert and Bernays' formulae $(\varepsilon_\mathbf{2})$ (for $\forall$) and $(\varepsilon_\mathbf{1})$ (for $\exists$) of [HB70, page 15].

REMARK 8.3. The reader familiar with nominal techniques might wonder: are nominal techniques not supposed to be inconsistent with choice? For instance [Gab11a, Theorem 10.11] proves that the axiom of choice is inconsistent with the axioms underlying nominal techniques.[19]

The choice used here is an *equivariant* choice, which is consistent with nominal techniques; see Example 8.5. This makes Lemma 8.4 particularly relevant:

LEMMA 8.4. *Suppose $a : \alpha$. Then:*

- $\pi\cdot(\epsilon a.\phi) = \epsilon\pi(a).\pi\cdot\phi$.
- $(\epsilon a.\phi)[b{\mapsto}s] =_{\alpha\beta\eta\varepsilon} \epsilon a.(\phi[b{\mapsto}s])$ *provided* $a \notin fa(s)$.

[19]This observation has an excellent pedigree. Nominal techniques are based on Fraenkel-Mostowski set theory, which was developed specifically to prove the independence of the axiom of choice from the other axioms of set theory (with atoms). Historical notes to the earlier literature are in [Gab11a, Remark 2.22].

Proof. For the first part we unpack Definition 2.20, bearing in mind that ϵ_α is just a constant symbol (one of the f from Definition 2.8):

$$\pi\cdot(\epsilon a.\phi) = \pi\cdot(\epsilon_\alpha[a]\phi) = (\pi\cdot\epsilon_\alpha)\pi\cdot[a]\phi = \epsilon_\alpha[\pi(a)]\pi\cdot\phi = \epsilon\pi(a).\pi\cdot\phi.$$

For the second part we unpack Figure 2, reasoning as follows:

$$\begin{aligned}
(\epsilon a.\phi)[b\mapsto s] &= (\epsilon_\alpha[a]\phi)[b\mapsto s] && \text{Definition}\\
&= \epsilon_\alpha[b\mapsto s](([a]\phi)[b\mapsto s]) && (\mathbf{stxapp})\\
&= \epsilon_\alpha([a](\phi[b\mapsto s])) && (\mathbf{stx\#}),\ (\mathbf{stx[]}),\ a\notin fa(s)\\
&= \epsilon a.(\phi[b\mapsto s]) && \text{Definition}
\end{aligned}$$

□

Example 8.5. We give concrete examples of equivariant and non-equivariant choice:

1. *Non-equivariant choice.* Consider the set of all unordered pairs of distinct atoms $\mathcal{P} \subseteq pow(\mathbb{A})$ (pow is from Example 5.4). So for example $\{a,b\} \in \mathcal{P}$ and $\{a\} \notin \mathcal{P}$ and $\{a,b,c\} \notin \mathcal{P}$.
   Then it is a fact that no choice function f from $\mathcal{P}$ to $\mathbb{A}$ is equivariant. That is, for every choice function f there exist π and a and b such that $\pi(f(\{a,b\})) \neq f(\{\pi(a),\pi(b)\})$.
2. *Equivariant choice.* Consider the set of all unordered pairs of distinct natural numbers $\mathcal{N} \subseteq \mathbb{N}$. So for example $\{1,2\} \in \mathcal{N}$ and $\{1\} \notin \mathcal{N}$ and $\{1,2,3\} \notin \mathcal{N}$. Give $\mathbb{N}$ the *trivial* permutation action $\pi\cdot x = x$.
   Then every choice function g from $\mathcal{N}$ to $\mathbb{N}$ (e.g. g mapping $\{x,y\}$ to the lesser of x and y) is equivariant. That is, $\pi\cdot(g(\{x,y\})) = g(\{\pi\cdot x,\pi\cdot y\})$.
   Here, the choice function is equivariant because the permutation actions are trivial, or more loosely: there are no atoms to permute.
3. *Equivariant choice, with atoms.* Consider the nominal set $\mathbb{A}\otimes\mathbb{A}$ of ordered pairs of distinct atoms, thought of as a two-element lists. Then there is a natural equivariant choice function which maps (a,b) to a.

Examples 2 and 3 might look silly, but in fact they are not. There is nothing inherently non-equivariant about choice, even choosing elements involving atoms—if we do so symmetrically, that is, if we do not have to create order on atoms to do so. In example 2 above, numbers are inherently ordered; in example 3 we assumed the order was packaged with the input as a pair, so that we can canonically and equivariantly choose the 'first element' of the pair.

Two more elaborate examples of equivariant choice functions are in Subsections 8.3.1 and 8.3.2. However, the principle on which they work remains the same (see Remark 8.11).

8.2. *Denotation*

Now we show how choice gets interpreted denotationally. Fix a model $\mathcal{I}$.

Definition 8.6. A **choice function** ε_α is an equivariant function in $[\![\alpha\to o]\!] \to [\![\alpha]\!]$ which satisfies:

1. $(\forall^{\mathcal{I}}_\alpha \bullet [a]z) \approx z[a\mapsto\varepsilon_\alpha([a]\neg z)]$ for every $z \in [\![o]\!]$ and $a : \alpha$.
2. $\varepsilon_\alpha([a]z)[b\mapsto y] = \varepsilon_\alpha([a](z[b\mapsto y]))$ for every $z \in [\![o]\!]$ and $a : \alpha$, and every β, $b : \beta$, and $y \in [\![\beta]\!]$.

Say that $\mathcal{I}$ has **choice functions** when it has a choice function for every α.

Remark 8.7. It may or may not be the case that ε_α can be represented internally in $\mathcal{I}$ in the sense that there exists some family of elements $x_\alpha \in [\![(\alpha{\to}o){\to}\alpha]\!]$ such that

$$x_\alpha \bullet f = \varepsilon_\alpha(f) \quad \text{for every} \quad f \in [\![\alpha{\to}o]\!].$$

Suppose there is. Then we can interpret ϵ_α from Subsection 8.1 by taking $\epsilon^{\mathcal{I}}_\alpha = x_\alpha$:

Theorem 8.8. *If $\mathcal{I}$ has choice functions then it has maximisers (Definition 6.9).*

Proof. It suffices to show that for every $a : \alpha$ and $z \in [\![o]\!]$ there exists some x with $z[a{\mapsto}x] = \forall^{\mathcal{I}}_\alpha \bullet [a]z$. We take $x = \varepsilon_\alpha([a]\neg z)$. $\square$

8.3. *The syntactic and functional models with choice functions*

Theorems 7.11 and 7.25 suffice for interpreting hypersequents and the rules $(\alpha *)$, $(\beta *)$, $(\gamma *)$, and $(\delta^- *)$—that is, up to but not including Section 4.

To interpret the $(\delta^\chi *)$ rules from Definition 4.11, and thus to interpret maximising hypersequents, we need models with maximisers. By Theorem 8.8 it suffices to show a model has *choice functions* (Definition 8.6).

This can be accommodated in the syntactic model and is already present in the functional model.[20]

8.3.1. *Choice in the syntactic model*

For the syntactic model, we use the syntax with choice ϵ_α from Definition 8.1. By Theorem 8.8 it suffices to show that $\mathcal{S}$ has choice functions. We obtain this by replaying the proofs of Subsection 7.1 for syntax and syntactic equivalence with choice $=_{\alpha\beta\eta\varepsilon}$ from Definition 8.1; the extra axiom $(\mathbf{stx}\epsilon)$ from Definition 8.1 changes nothing in the proofs above, and by construction we get a choice function, which maps $[[a]\phi]_{\alpha\beta\eta\varepsilon} \in [\![o]\!]^{\mathcal{S}}$ equivariantly (by part 1 of Lemma 8.4) to $[\epsilon a.\phi]_{\alpha\beta\eta\varepsilon} \in [\![\mathit{type}(a)]\!]^{\mathcal{S}}$.

8.3.2. *Choice in the functional model*

For the functional model (Definition 7.19) we construct a choice function by hand as follows:

Definition 8.9. For each $y \in [\![\alpha{\to}o]\!]^{\mathcal{F}}$ make some choice $\epsilon(y)$ of element $x \in [\![\alpha]\!]^{\mathcal{F}}$ such that $y(x) = \top$, if such an x exists, and otherwise let $\epsilon(y)$ be any element of $[\![\alpha]\!]^{\mathcal{F}}$.

Define ε_α to map $f \in [\![\alpha{\to}o]\!]$ to $\lambda\varsigma.\epsilon(f(\varsigma)) \in [\![\alpha]\!]$. Thus,

$$\varepsilon_\alpha(f)(\varsigma) = \epsilon(f(\varsigma)).$$

Theorem 8.10. *The family of functions $\varepsilon_\alpha : [\![\alpha{\to}o]\!]{\to}[\![o]\!]$ is a choice function in the sense of Definition 8.6.*

Proof. It suffices to check the following:

[20]...provided that your mathematical foundation includes the axiom of choice. If you are not sure whether your foundation includes choice, then it does.

1. *Proof that* $\pi\cdot(\varepsilon_\alpha \bullet f) = \varepsilon_\alpha \bullet (\pi\cdot f)$. We just manipulate definitions, as follows:

$$\begin{aligned}
(\pi\cdot(\varepsilon_\alpha \bullet f))(\varsigma) &= (\varepsilon_\alpha \bullet f)(\pi^{-1}\cdot\varsigma) && \text{Definition 7.15}\\
&= \epsilon(f(\pi^{-1}\cdot\varsigma)) && \text{Definition 8.9}\\
&= \epsilon((\pi\cdot f)(\varsigma)) && \text{Definition 7.15}\\
&= \varepsilon_\alpha \bullet (\pi\cdot f). && \text{Definition 8.9}
\end{aligned}$$

2. *Proof that* $(\forall a.z) \approx z[a\mapsto\varepsilon_\alpha([a]\neg z)]$. It suffices to show that $(\forall a.z)(\varsigma) = \top$ implies $(z[a\mapsto\varepsilon_\alpha([a]\neg z)])(\varsigma) = \top$. This follows by construction from Definition 8.9.
3. *Proof that* $\varepsilon_\alpha(f)[a\mapsto g] = \varepsilon_\alpha(f[a\mapsto g])$.

$$\begin{aligned}
(\varepsilon_\alpha(f)[a\mapsto g])(\varsigma) &= \varepsilon_\alpha(f)(\varsigma[a{:=}g(\varsigma)]) \\
&= \epsilon(f(\varsigma[a{:=}g(\varsigma)]))
\end{aligned}
\qquad
\begin{aligned}
(\varepsilon_\alpha(f[a\mapsto g]))(\varsigma) &= \varepsilon_\alpha(f[a{:=}g])(\varsigma) \\
&= \epsilon(f(\varsigma[a{:=}g(\varsigma)]))
\end{aligned}$$

□

Remark 8.11. From the point of view of equivariant choice, the construction of the maximising element $\varepsilon_\alpha(f)$ in Definition 8.9 is a glorified version of part 2 of Example 8.5: $[\![\alpha]\!]^{\mathcal{F}}$ from Definition 7.12 is an 'ordinary set' (or if the reader prefers; it is a nominal set with the trivial action $\pi\cdot x = x$), and we can choose elements from it freely without affecting equivariance.

In contrast, the construction of the maximising element $[\epsilon a.\phi]_{\alpha\beta\eta\varepsilon}$ in Subsection 8.3.1 resembles part 3 of Example 8.5: terms may certainly contain atoms and have a non-trivial permutation action, but we can still choose a canonical maximising term $\epsilon a.\phi$, and indeed, that is precisely what ϵ is designed to provide.

9. Conclusions

We have presented a new, more abstract syntax and semantics for δ-rules. We are not trying to be algorithmically efficient, but we are optimising for nice and compact syntax and semantics. This is quite hard to get right using ordinary syntax and semantics (e.g. using simply-typed λ-calculus and sets and functions between them). Instead we used *nominal terms* and *nominal sets*.

The 'nominal' approach is based not on functions but on *symmetric names*. Thus, nominal sets are sets with a group action, and nominal terms include variables X with free atoms $pms(X)$ that are considered to occur in X (e.g. the free atoms of X is $pms(X)$) but they occur in no particular order.[21] As discussed in Remark 4.10, this gives the nominal syntax an expressivity usually attained by Skolemisation or raising, but $pms(X)$ is not ordered (unlike arguments to a function) and the nominal semantics are first-order.

Nominal semantics may be unfamiliar, but they are fairly elementary; the technical requirements are no greater than for other semantics. Open terms map directly to open elements; and name-related aspects of nominal terms, like permutations and atoms-abstraction, map directly to corresponding *semantic* operations on nominal sets.

Because of this, operations that we are accustomed to seeing only on syntax, like substitution and binding, now also occur in the semantics.

[21]In [Gab12a] we argue that X is, in some abstract mathematical sense that we make formal, actually a symmetry-breaking operation.

The similarity of syntax and semantics can make it look like nothing is happening: Definition 5.16 could be mistaken for a Hilbert axiomatisation; Definition 5.27 might seem to just translate syntax trivially to semantics. But this is deceptive: Subsection 7.2 constructed a *functional model*, and if that model is least somewhat familiar, in [GM11] and even more so in [Gab11b] we build (topological) models that are not derived from either syntax or functions, in which substitution and abstraction are interpreted completely independently of the apparatus of syntax and functions—and yet they still satisfy the axioms and have a natural construction.

Thus when we have written $z[a{\mapsto}x]$ in this paper and read this 'z with a replaced by x', this is an *algebraic generalisation* of substitution[22] which operates on semantic objects.

What this contributes to proof-search is simplicity: proof-search is by design full of different kinds of variables that interact in complex ways. Our denotation allows us to interpret this directly—and in an implementation and algorithm-independent manner—in semantics.

9.1. *Nominal provenance of this paper*

Proof-search. This paper pulls together ideas from a number of sources, notably the specific view of proof-search represented for instance in [Wir04] (which uses a broadly higher-order semantics for variable-dependency) and [Wir11] (which takes a more nominal perspective).

Setting the specific application to proof-search aside for now: where does the underlying nominal machinery of this paper come from?

Nominal algebra and one-and-a-halfth order logic. The idea of equational axiomatisations of mathematics in a nominal context goes back to *nominal algebra*, which was developed to axiomatise capture-avoiding substitution and first-order logic [GM06b, GM06a], which in turn goes back to the *nominal rewrite* systems of [FGM04, FG07].

This was developed further in [GM08b, GM08a]. In particular, the axiomatisation in Figures 7 and 8 is a direct descendent of the *one-and-a-halfth order logic* of [GM06b, GM08b]. This technology was used in [GM08c, GM10] to represent schemas of first-order derivations.

In a nutshell: [GM08b] considered derivability in the presence of nominal unknowns X, whereas [GM08c] also treated the proof-theory. This paper considers both of these, and adds semantics for how we look for instantiations of the X.[23]

Permissive-nominal syntax. The work in [GM08c, GM10] motivated the development of *permissive*-nominal terms, an idea which goes back to [DGM09b, DGM09a, DGM10]. This is pulled together in [DG12a] to give a finite axiomatisation of substitution, first-order logic, and arithmetic in permissive-nominal logic.

In all the above, there had been no explicit consideration of proof-search. It is simplistic, but reasonable, to sum things up as follows: this paper uses the same basic syntactic (and semantic; see below) ideas, but instead of studying, say, incomplete derivation schemas (one-and-a-halfth order logic) or $r = s$ (nominal algebra) or $\forall X$ (permissive-nominal

[22]The axioms are in Figure 7. When reading this figure it is useful to recall the notation used in it, from Notation 5.14.

[23]There is a little more to it than that. This paper treats a higher-order syntax; that is, we assume $\beta\eta$.

logic), we study hypersequents and maximisation conditions $X{\uparrow}[a]\phi$, making X behave explicitly like a proof-search variable.

Models. An investigation of models followed, in several styles. This includes translations of theories in permissive-nominal syntax to higher-order terms [GM09b, DG12b]—see also [LV12], which translates a simpler syntax and does not consider theories, but follows the same general idea—and it includes the topological model of first-order logic [Gab11b] mentioned above (essentially, a topological treatment of [GM08b] and of a first-order version of Figures 7 and 8 from this paper).

Nominal Henkin semantics and permissive-nominal logic. Of most relevance to this paper is the *nominal Henkin semantics* of [GM11], which considered nominal models of $\alpha\beta$-equivalence. That and *permissive-nominal logic* [DG10, DG12a] are the most immediate ancestors of this paper. So for the expert in nominal techniques we conclude with a brief technical comparison of some specific design choices of these papers.

In [GM11] atoms and variables do not have a fixed type, so we need typing contexts. This paper fixes a global typing context in Definition 2.3. Concretely that means that Definitions 3.3 and 5.9 of [GM11] and the associated proofs have subscripts Γ (typing contexts), whereas Definition 5.12 of this paper and the associated proofs, do not. This is purely a design issue. The choice made here suffices for our needs and gives a cleaner presentation.

Permissive-nominal logic syntax [DG10, DG12a] has, like this paper, a single fixed typing (we called it a *sorting*). For instance Definition 3.20 of [DG12a] has no Γ subscripts, like Definition 5.12 here. The major *difference* is that permissive-nominal logic sorts are first-order: permissive-nominal logic has no function-sorts, and it has α-equivalence but not the $\beta\eta$-equivalence from Figure 2. This is deliberate: permissive-nominal logic has a quantifier $\forall X$ and β- and η-equivalence are axiomatisable, which was part of the design brief of permissive-nominal logic.

Of course the specifics of hypersequents, maximisation conditions, and application to proof-search considered in this paper, are new to nominal techniques.

Variables as well-orderings of atoms. A word on the broader mathematical context. In [Gab12a] we identify X with a well-ordering of $pms(X)$. From this perspective, the instantiation action for variables (Definition 2.27, in this paper) is a compact way to present an infinite *Skolemisation* (Subsections 4.2 and 4.3 of Chapter 1 of [DGHP99] or Definition 53 of Chapter 3 of [DGHP99]) or infinite *raising* (Section 5 of [Mil92]), or a *de Bruijn index* [dB72].

Indeed if we take $a_1, \ldots, a_n$ to be $pms(X)$ in some order then intuitively $X =_{\alpha\beta\eta} f a_1 \ldots a_n$ where $f = \lambda a_1. \ldots \lambda a_n.X$. Instantiating X is like substituting f for a 'closed term' of higher order (a term r with $fa(r) = \varnothing$).

However, X is not *identical* to Skolemisation:

- There is no order on $pms(X)$ and the type of X does not change depending on its permission set.
- The nominal semantics for atoms and atoms-abstraction are inherently first-order. In [GM11] we illustrate this with a sound and complete Henkin semantics for higher-order logic, resembling the semantics in Section 5. See also the first-order meaning given to nominal atoms-abstraction in [GP01, Lemma 5.1].

For a more general view see [DG10, DG12a, Gab12b]. To see nominal instantiation viewed as an equivariant function see [Gab12b, Lemma 3.4.3] or [Gab12a, Lemma 2.33].

To see explicit translations of permissive-nominal theories to higher-order theories based on these ideas, see [GM09b, DG12b]—in which it is clear how the nominal permutative properties of atoms are lost in the translation [DG12b, Subsection 1.2].

9.2. Why permutations

In Remark 2.14 we justified the use of moderated variables $\pi\cdot X$, and noted that $(b\ a)\cdot X$ where $b \notin pms(X)$ models the informal vernacular '$t[y/x]$ where we take y not free in t'.

We noted that nominal techniques use permutations $(b\ a)$ rather than the (more natural—or rather, more *familiar*) atoms-renaming $[b/a]$, or even substitution $[a{\mapsto}b]$.

Why? What are the advantages of the *permutative* model of names and binding?

The underlying point is that groups are a more restricted structure than monoids, and they have inverses. Atoms-renamings and substitutions form monoids, and do not always have inverses. As is often the case, favouring the more restricted class of structures gives us better behaviour and more theorems.

We take a moment to list just two concrete ways in which that helps us:

Thanks to the use of a group of permutations instead of a monoid of atoms-renamings, atoms can conveniently be identified with urelemente and their natural permutation action on the Fraenkel-Mostowski sets universe. This is a fancy way of saying that nominal sets from Section 5 are nice to work with.

Also, the permutation action of Definition 2.20 is directly definable on permissive-nominal syntax *without* triggering α-conversions. Thus, the use of permutations allows us to modularise the definitions; first defining the permutation action and *then* defining α-equivalence (Definition 2.22), rather than having to define both by a simultaneous induction, as would be the case if we used atoms-renamings.

This does not exhaust the advantages of the nominal treatment of names and bindings. It gives a flavour of its convenience. In two words these are: *modularity* and *foundations*.

9.3. Future work

We hope this paper will provide a *lingua franca* for expressing constructs in proof-search, and more generally for meta-variables in proofs. We can also hope that our nominally symmetric models might help to develop new and improved algorithms, to cross-check soundness of algorithms that are independently developed, or even to help with the general design of theorem-provers.

In [DGM10, Gab12b] we allow infinite permission sets in terms (in this paper $pms(X)$ is taken to be a finite set of atoms). The benefit of allowing 'larger' permission sets is not visible in this paper, but when we come to consider induction an infinite stock of atoms may be useful to generate and name as many intermediate lemmas in inductive derivations as we like. That is: in the presence of more complicated notions of derivation, we may not discover all the atoms we require until after we have started constructing a derivation. Thus, it might be useful to allow larger, infinite, permission sets. We might even permit infinite permission sets directly in the semantics.

This is not a problem. Permissive-nominal techniques from [DGM10, DG12a, Gab12b] already assume infinite permission sets. The finite permission sets of this paper are a useful simplification of that idea and they are all we need for now, but if required there should be no difficulty in generalising to 'larger' infinite permission sets.

We have not considered completeness. The logic of this paper is higher-order, for which incomplete semantics are standard. However our semantics resembles the nominal Henkin semantics of [GM11] and our semantics might be complete for our rules—perhaps modulo some simple adaptations.

In this paper we tried to be accessible to a broad spectrum of readers, who are not *necessarily* experts in nominal techniques or even expert in proof-search. Attempting a proof of completeness would have detracted from that, but it remains an interesting question. Thus, for the mathematically and logically inclined, as well as for the implementor, there remains much work to be done to extend and develop on these ideas.

We conclude with a technical point, but one which illustrates the flexibility of our techniques. Our analysis of substitution is fairly simple-minded; we assume atoms-abstraction $[a]r$ and application rs and take $r[a{\mapsto}s]$ to be just $([a]r)s$. This is fine for some situations—for instance in a mathematical model such as this paper, or in a tactics-driven theorem-prover—but there are other situations—for instance for a confluence proof, or in a dependent type theory—where we might wish to distinguish between those reducts that are there because we are trying to reduce, and those reducts that are there because we have not yet reduced them. Then, we might like to introduce an explicit substitution. This raises the question of what $X[a{\mapsto}r]$ should reduce to.

One answer is to allow simultaneous substitutions of atoms of terms and suspend these on X (just as we currently suspend permutations as $\pi{\cdot}X$). Another, equivalent but perhaps more elegant answer, is to treat X as a tuple of the atoms in its permission set and $X[a{\mapsto}r]$ as that tuple with a replaced by r. More on this in [Gab12a].

For us, these questions are a feature and not a bug: our nominal syntax lets us be as abstract, or as concrete, as we like. We can think about implementation if we want to, or ignore it insouciantly if we do not. This is up to us.

Bibliography

[Avr91] Arnon Avron. Hypersequents, logical consequence and intermediate logics for concurrency. *Annals of mathematics and artificial intelligence*, 4(3–4):225–248, 1991.

[Avr96] Arnon Avron. The method of hypersequents in the proof theory of propositional non-classical logics. In Wilfrid Hodges, Martin Hyland, Charles Steinhorn, and John Truss, editors, *Logic: from foundations to applications*, pages 1–32. Oxford University Press, 1996.

[BBK04] Christoph Benzmüller, Chad E. Brown, and Michael Kohlhase. Higher-order semantics and extensionality. *Journal of Symbolic Logic*, 69:1027–1088, 2004.

[BF95] Matthias Baaz and Christian G. Fermüller. Non-elementary speedups between different versions of tableaux. In *Proceedings of the 4th International Workshop on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX'95)*, pages 217–230. Springer, 1995.

[BHS93] Bernhard Beckert, Reiner Hähnle, and Peter H. Schmitt. The even more liberalized delta-rule in free variable semantic tableaux. In Georg Gottlob, Alexander Leitsch, and Daniele Mundici, editors, *Proceedings of the third Kurt Gödel Colloquium (KGC'93)*, volume 713 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 1993.

[BU07] Stefan Berghofer and Christian Urban. A head-to-head comparison of de Bruijn indices and names. *Electronic Notes in Theoretical Computer Science*, 174(5):53–67, 2007.

[CA00] Domenico Cantone and Marianna Asmundo. A further and effective liberalization of the δ-rule in free variable semantic tableaux. In Ricardo Caferra and Gernot Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Computer Science*, pages 408–414. Springer, 2000.

[Che05] James Cheney. Nominal logic and abstract syntax. *SIGACT News (logic column 14)*, 36(4):47–69, 2005.

[CNA07] Domenico Cantone and Marianna Nicolosi-Asmundo. A sound framework for delta-rule variants in free variable semantic tableaux. *Journal of Automated Reasoning*, 38:31–56, 2007.

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

[DG10] Gilles Dowek and Murdoch J. Gabbay. Permissive Nominal Logic. In *Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP 2010)*, pages 165–176. ACM Press, 2010.

[DG12a] Gilles Dowek and Murdoch J. Gabbay. Permissive Nominal Logic (journal version). *Transactions on Computational Logic*, 13(3), 2012.

[DG12b] Gilles Dowek and Murdoch J. Gabbay. PNL to HOL: from the logic of nominal sets to the logic of higher-order functions. *Theoretical Computer Science*, 451:38–69, 2012.

[DGHP99] Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handbook of Tableau Methods*. Kluwer, 1999.

[DGM09a] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. Technical Report HW-MACS-TR-0062, Heriot-Watt, February 2009.

[DGM09b] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. In *Proceedings of the 24th Italian Conference on Computational Logic (CILC'09)*, 2009.

[DGM10] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification: an infinite, co-infinite approach to nominal techniques (journal version). *Logic Journal of the IGPL*, 18(6):769–822, 2010.

[Die11] Dominik Dietrich. *Proof Planning with Compiled Strategies*. Phd thesis, Dept. Informatics, Saarland University, 2011.

[FG07] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting (journal version). *Information and Computation*, 205(6):917–965, June 2007.

[FGM04] Maribel Fernández, Murdoch J. Gabbay, and Ian Mackie. Nominal Rewriting Systems. In *Proceedings of the 6th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming (PPDP 2004)*, pages 108–119. ACM Press, August 2004.

[Fit90] Melvin Fitting. *First-order Logic and Automated Theorem Proving*. Texts and monographs in computer science. Springer, 1 edition, 1990. Second edition is [Fit96].

[Fit96] Melvin Fitting. *First-order Logic and Automated Theorem Proving*. Texts and monographs in computer science. Springer, 2 edition, 1996. First edition is [Fit90].

[GA99] Martin Giese and Wolfgang Ahrendt. Hilbert's ϵ-terms in automated theorem proving. In Neil Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1617 of *Lecture Notes in Computer Science*, pages 662–662. Springer, 1999.

[Gab07a] Murdoch J. Gabbay. Fresh Logic. *Journal of Applied Logic*, 5(2):356–387, June 2007.

[Gab07b] Murdoch J. Gabbay. A General Mathematics of Names. *Information and Computation*, 205(7):982–1011, July 2007.

[Gab11a] Murdoch J. Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic*, 17(2):161–229, 2011.

[Gab11b] Murdoch J. Gabbay. Stone duality for First-Order Logic: a nominal approach. In *Howard Barringer Festschrift*. December 2011.

[Gab12a] Murdoch J. Gabbay. Meta-variables as infinite lists in nominal terms unification and rewriting. *Logic Journal of the IGPL*, 2012.

[Gab12b] Murdoch J. Gabbay. Nominal terms and nominal logics: from foundations to meta-mathematics. In *Handbook of Philosophical Logic*, volume 17. Kluwer, 2012.

[Gab12c] Murdoch J. Gabbay. Unity in nominal equational reasoning: The algebra of equality on nominal sets. *Journal of Applied Logic*, 10:199–217, June 2012.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen [Investigations into logical deduction]. *Mathematische Zeitschrift 39*, pages 176–210,405–431, 1935. Translated in [Sza69], pages 68–131.

[GL09] Murdoch J. Gabbay and Stéphane Lengrand. The lambda-context calculus (extended version). *Information and computation*, 207:1369–1400, December 2009.

[GM06a] Murdoch J. Gabbay and Aad Mathijssen. Capture-avoiding Substitution as a Nominal Algebra. In *ICTAC 2006: Theoretical Aspects of Computing*, volume 4281 of *Lecture Notes in Computer Science*, pages 198–212, November 2006.

[GM06b] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order logic. In *Proceedings of the 8th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP 2006)*, pages 189–200. ACM, July 2006.

[GM07] Murdoch J. Gabbay and Aad Mathijssen. A Formal Calculus for Informal Equality with Binding. In

WoLLIC'07: 14th Workshop on Logic, Language, Information and Computation, volume 4576 of *Lecture Notes in Computer Science*, pages 162–176. Springer, July 2007.

[GM08a] Murdoch J. Gabbay and Aad Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing*, 20(4-5):451–479, June 2008.

[GM08b] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic. *Journal of Logic and Computation*, 18(4):521–562, August 2008.

[GM08c] Murdoch J. Gabbay and Dominic P. Mulligan. One-and-a-halfth Order Terms: Curry-Howard for Incomplete Derivations. In *Proceedings of 15th Workshop on Logic, Language and Information in Computation (WoLLIC 2008)*, volume 5110 of *Lecture Notes in Artificial Intelligence*, pages 180–194. Springer, June 2008.

[GM09a] Murdoch J. Gabbay and Aad Mathijssen. Nominal universal algebra: equational logic with names and binding. *Journal of Logic and Computation*, 19(6):1455–1508, December 2009.

[GM09b] Murdoch J. Gabbay and Dominic P. Mulligan. Universal algebra over lambda-terms and nominal terms: the connection in logic between nominal techniques and higher-order variables. In *Proceedings of the 4th International Workshop on Logical Frameworks and Meta-Languages (LFMTP 2009)*, pages 64–73. ACM, August 2009.

[GM10] Murdoch J. Gabbay and Dominic P. Mulligan. One-and-a-halfth Order Terms: Curry-Howard for Incomplete Derivations (journal version). *Information and Computation*, 208:230–258, March 2010.

[GM11] Murdoch J. Gabbay and Dominic P. Mulligan. Nominal Henkin Semantics: simply-typed lambda-calculus models in nominal sets. In *Proceedings of the 6th International Workshop on Logical Frameworks and Meta-Languages (LFMTP 2011)*, volume 71 of *EPTCS*, pages 58–75, September 2011.

[GP99] Murdoch J. Gabbay and Andrew M. Pitts. A New Approach to Abstract Syntax Involving Binders. In *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS 1999)*, pages 214–224. IEEE Computer Society Press, July 1999.

[GP01] Murdoch J. Gabbay and Andrew M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing*, 13(3–5):341–363, July 2001.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Cambridge University Press, 1989.

[HB70] David Hilbert and Paul Bernays. *Grundlagen der Mathematik (volume II)*. Number 50 in Grundlehren der mathematischen Wissenschaften. Springer, second edition, 1970.

[Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15:81–91, 1950.

[HS94] Reiner Hähnle and Peter H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13(2):211–222, 1994.

[Hun33] Edward V. Huntington. New sets of independent postulates for the algebra of logic with special reference to Whitehead and Russell's "Principia Mathematica". *Transactions of the American Mathematical Society*, 35(1):274–304, January 1933.

[Kan63] Stig Kanger. A simplified proof method for elementary logic. In Jörg Siekmann and Graham Wrightson, editors, *Automation of Reasoning volume 1 - classical papers on Computational Logic 1957-1966*, pages 364–371. Springer, 1963.

[LV12] Jordi Levy and Mateu Villaret. Nominal unification from a higher-order perspective. *Transactions on Computational logic (TOCL)*, 13(2), 2012.

[McC97] William McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19:263–276, 1997.

[Mil92] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14(4):321–358, 1992.

[MV95] Wilfried Meyer-Viol. *Instantial Logic — An Investigation into Reasoning with Instances*. PhD thesis, University of Utrecht, 1995. ILLC dissertation series 1995–11.

[Pra60] Dag Prawitz. An improved proof procedure. *Theoria: A Swedish Journal of Philosophy*, 26:102–139, 1960.

[Pra83] Dag Prawitz. An improved proof procedure. In Jörg Siekmann and Graham Wrightson, editors, *Automation of Reasoning volume 1 - classical papers on Computational Logic 1957-1966*, pages 159–199. Springer, 1983.

[Sha85] Stewart Shapiro, editor. *Intensional mathematics*. North-Holland, 1985.

[Smu68] Raymond Smullyan. *First-order logic*. Springer, 1968. Reprinted by Dover, 1995.

[Sza69] M. E. Szabo, editor. *Collected Papers of Gerhard Gentzen*. North Holland, 1969.

[UPG03] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. In *Proceedings of the 17th International Workshop on Computer Science Logic (CSL 2003)*, volume 2803 of *Lecture Notes in Computer Science*, pages 513–527. Springer, December 2003.

[UPG04] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, September 2004.

[Urb08] Christian Urban. Nominal reasoning techniques in Isabelle/HOL. *Journal of Automatic Reasoning*, 40(4):327–356, 2008.

[Wal90] Lincoln A. Wallen. *Automated Proof Search in Non-Classical Logics — efficient matrix proof methods for modal and intuitionistic logics*. MIT Press, 1990. Phd thesis.

[Wir04] Claus-Peter Wirth. Descente infinie + Deduction. *Logic Journal of the IGPL*, 12(1):1–96, 2004.

[Wir11] Claus-Peter Wirth. A simplified and improved free-variable framework for Hilbert's epsilon as an operator of indefinite committed choice. *CoRR*, abs/1104.2444, 2011.