A Sequent Calculus for Nominal Logic

Murdoch Gabbay
LIX École Polytechnique
gabbay@lix.polytechnique.fr

James Cheney
Cornell University
jcheney@cs.cornell.edu

Abstract

Nominal logic is a theory of names and binding based on the primitive concepts of freshness and swapping, with a self-dual И- (or "new")-quantifier, originally presented as a Hilbert-style axiom system extending first-order logic. We present a sequent calculus for nominal logic called Fresh Logic, or FL, admitting cut-elimination. We use FL to provide a proof-theoretic foundation for nominal logic programming and show how to interpret $FO\lambda^{\nabla}$, another logic with a self-dual quantifier, within FL.

1 Introduction

Names are ubiquitous in logic, theoretical computer science, and practical programming. Names appear as variable symbols in programs and logical formulae, channel names in π-calculus processes, nonces in security protocols, and in many other situations. The ability to reason about names is therefore important to many areas of computer science.

Though integers, strings, or other concrete data are often used as representations of such names, usually the representation does not matter; we take the characteristic properties of names to be:

1. **Renaming (or Equivariance).** Properties concerning names are invariant up to permutative renaming (either globally, or for bound names, locally).
2. **Freshness.** Fresh names can always be generated.

For example, to distribute a substitution under a λ-abstraction $(\lambda x.fx)\{f\mapsto x\}$, it is necessary first to generate a fresh variable name y, and rename x to y in the body of the abstraction. Similarly, to extrude the scope of a π-calculus restriction in $\overline{z}z \mid \nu\langle z\rangle\overline{z}z$ it is necessary to generate a fresh channel name z' and rename z in the body of the restriction. When Microsoft Windows loads a DLL, it rebases it to avoid an address space conflict. The situation is worse when we have an unknown X which may be instantiated *after* we have finished our calculation: how is it possible to choose fresh names, or rename existing names, in an unknown X which will be instantiated after the fact of our calculations?

We give a proof-theoretic account of these phenomena based on a First-Order logic Fresh Logic (FL). It includes primitive **swappings** $(a\ b)$ on terms t for renaming, a И **quantifier** ('New' quantifier) that quantifies over sufficiently fresh names, and rules expressing the facts that fresh names can always be chosen and that all formulae are invariant under permutative renamings.

Related work includes Nominal Logic [10] and FM-HOL [7], Hilbert-style axiom systems both concentrating on expressiveness, and a natural-deduction presentation of Fresh Logic FL$_{ND}$, concentrating on semantics. Nominal Unification [12] and Nominal Rewriting [3] use the same techniques of swapping and freshness in their respective domains, to good effect.

The Fresh Logic of this paper is presented as a Gentzen-style intuitionistic sequent calculus with cut-elimination. As such it provides a direct reading of logical connectives (including И) as proof-search techniques, suitable both for human reasoning and automated deduction. We discuss other innovative design features in the course of the paper. The И quantifier possesses interesting proof-theoretic properties, notably self-duality: it has both a $\forall$ and $\exists$ character, and $\neg \text{И}x.\,P \iff \text{И}x.\,\neg P$. This duality is reflected directly in the symmetric structure of the И-rules of FL.

In addition to presenting our calculus and proving cut-elimination, we also present two applications: we give a uniform proof-theoretic semantics to a logic programming language with И-quantified goals and program clauses, and we relate FL to a similar calculus $FO\lambda^{\nabla}$ [9] which also has a self-dual quantifier. A semantics for Fresh Logic is not presented in this paper, but previous work [5] describes a Kripke-style semantics based on FM sets [4, 6] and gives details of proof-normalization for a natural-deduction version of Fresh Logic.

2 Fresh Logic

Fix a set of **sorts** $\tau, \tau', \ldots \in \mathrm{Sorts}$ and distinguish a **sort of atoms** $\mathtt{A} \in \mathrm{Sorts}$. We assume function sorts $\tau \to \tau'$.

Fix countably infinite sets $\mathbb{V} = \{x, y, z, \ldots\}$ of **variables** and $\{\mathtt{c}, \mathtt{d}, \ldots\} = \mathbb{C}$ of **constants**. Constants and variables are inherently sorted and we may indicate this as $\mathtt{c}:\tau$, $x:\tau'$. We assume a constant $\mathtt{swap}_\tau : \mathtt{A}\to\mathtt{A}\to\tau\to\tau$ exists for each sort τ. Variable names like n, m, n', etc. are typically used for variables of sort $\mathtt{A}$.

Terms are simply-typed λ-terms constructed according to the grammar:

$$s, t, \ldots ::= x \mid \mathtt{c} \mid \lambda x.t \mid t\, t'. \qquad (1)$$

Well-formedness is defined in the usual way; terms are always assumed to be well-formed at appropriate sorts and identified up to $\beta\eta$-equivalence ($\equiv_{\beta\eta}$ or just $\equiv$). We introduce the notion $V(t)$ of 'free variables of t', defined in the usual way. We introduce a **substitution action** $t\{x\mapsto s\}$ defined in the standard (capture-avoiding) way. We introduce the shorthand ts for a list of terms and $s(ts)$ for s applied to the list of arguments ts.

Terms of sort $\mathtt{A}$ are often called **atoms** and written (as above) as a, b, a', etc. A term of the form $\mathtt{swap}_\tau\ a\ a'\ t$ (from now on abbreviated $(a\ a')\cdot t$) is called a **swapping** of a and a' applied to t. Square brackets are used when necessary to avoid ambiguity between ordinary parentheses and swappings. We consider the addition of constants $\mathtt{abs}_\tau$ for FM abstraction $\langle a\rangle t$ [12, 4, 3] in §6.1.

We assume **predicate constant symbols** $p, q, r \ldots \in \mathbb{P}$, each with an arity $\tau_1 \cdots \tau_n$ (i.e., a list of sorts of the arguments of the predicate). For each $\tau \in \mathrm{Sorts}$ we assume distinguished constants $= : \tau\tau$ **equality** and $\# : \mathtt{A}\tau$ **freshness**. **Propositions** or **formulae** are generated by the grammar

$$\begin{array}{rcl} P & ::= & p(ts) \mid P \wedge P \mid P \vee P \mid P \supset P \\ & \mid & \top \mid \bot \mid \forall x.\, P \mid \exists x.\, P \mid \text{И} n.\, P. \end{array} \qquad (2)$$

$P \iff Q$ is shorthand for $P \supset Q \wedge Q \supset P$ and $\neg P$ is shorthand for $P \supset \bot$.

Like terms, propositions have notions of free variables $V(P)$ and substitution $P\{x\mapsto s\}$ defined as usual. We also adopt the notation $(a\ b)\cdot P$ abbreviating $P\{x_1\mapsto(a\ b)\cdot x_1\}\cdots\{x_n\mapsto(a\ b)\cdot x_n\}$, where $V(P) = \{x_1, \ldots, x_n\}$. That is, $(a\ b)\cdot P$ is the result of applying $(a\ b)$ to each free variable of P. For example,

$$\begin{array}{rl} & (a\ b)\cdot[\forall x.q[(c\ d)\cdot x, z]] \\ \equiv & \forall x.q[([(a\ b)\cdot c]\ [(a\ b)\cdot d])\cdot x, (a\ b)\cdot z] \end{array}$$

Formulae are considered equal up to consistent renaming of bound identifiers.

A **(logical) context** is a multiset of propositions, usually written Γ. Write $V(\Gamma)$ for the obvious extension of $V(\cdot)$ to many formulae. We may also write $\Gamma\{x\mapsto s\}$ and $(a\ b)\cdot\Gamma$.

2.1 Judgments and Derivation Rules

A **judgment** is a pair $\langle\Gamma, P\rangle$ of a context and predicate written $\Gamma \vdash P$. Elements of Γ are called hypotheses and P the conclusion. The **valid** or **derivable** judgments are inductively defined by the rules of Figure 1, Figure 2, and Figure 3.

In $(\mathbf{new}\mathbb{A})$ $V(\Gamma, C)$ denotes all variables free in Γ or C, that is $\bigcup\{V(P) \mid P \text{ in } \Gamma \text{ or } P \equiv C\}$. $a \# ts$ denotes a list of assumptions $a \# t_1, \ldots, a \# t_n$.

In $(\forall\mathbf{R})$ and $(\exists\mathbf{L})$, we assume (as usual) that the bound variable x is not already present in the conclusion. In $(\forall\mathbf{L})$ and $(\exists\mathbf{R})$ we assume t and x are of the same sort.

Note that the $(\mathcal{A}\mathbf{L})$ rule may be used to add an instance of any of the swapping axioms listed in $\mathcal{A}$ to Γ (read bottom-up). We write $(\rightsquigarrow)$ to abbreviate an inference following from some collection of $(\mathcal{A}\mathbf{L})$, $(\#\mathbb{A})$, $(\pi\#)$, and equality laws. For example,

$$\frac{\Gamma, a \# (a\ b)\cdot \mathtt{c}(x, y) \vdash C}{\Gamma, b \# x, b \# y \vdash C}\ (\rightsquigarrow)$$

2.2 The nonlogical rules

The rules $(\#\mathbb{A}), (\mathbf{case}\mathbb{A}), (\mathbf{new}\mathbb{A}), (\pi\#), (\pi\mathbf{L})$, and the swapping axioms are referred to as **nonlogical rules**, because they deal with atomic formulae rather than logical connectives. In this section we give some intuition for the meanings and purposes of these rules.

$(\#\mathbb{A})$*:* No atom is fresh for itself.

$(\mathbf{case}\mathbb{A})$*:* For atoms, freshness coincides with inequality and is decidable (even though FL is intuitionistic).

$(\mathbf{new}\mathbb{A})$*:* Read bottom-up this rule says we can always generate a new atom a fresh for any terms ts. This embodies the **Freshness principle** mentioned in the introduction.

$(\pi\#)$*:* If atoms a, b are fresh for t then swapping them in t has no effect.

$(\pi\mathbf{L})$*:* This is **Equivariance** for predicates, also mentioned in the introduction. Validity of P is independent of the particular values of atoms mentioned in P; only equalities or inequalities among atoms matter. For example if $a \# b \supset p(a, b)$ holds for *some* atoms a, b, then $a' \# b' \supset p(a', b')$ holds for *any* a', b'. This can be proved using $(\pi\mathbf{L})$ to swap a for a' and b for b'.

$(\mathcal{A}\mathbf{L})$ *and swapping axioms:* These axioms describe the behavior of swapping. The first three axioms imply swapping a with itself has no effect; swapping is its own inverse; and swapping acts as expected on atoms, respectively. The remaining three axioms imply that swapping is is homomorphic with respect to λ-term structure. Note that in our system, axiom (E1) of [10] becomes derivable:

$$(a\ a')\cdot[(b\ b')\cdot t] = ([(a\ a')\cdot b]\ [(a\ a')\cdot b'])\cdot[(a\ a')\cdot t] \quad (3)$$

$$\frac{\Gamma \vdash P \; \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}\,(\wedge\mathbf{R}) \qquad \frac{\Gamma, P, Q \vdash C}{\Gamma, P \wedge Q \vdash C}\,(\wedge\mathbf{LI})$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}\,(\vee\mathbf{R_1}) \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}\,(\vee\mathbf{R_2})$$

$$\frac{\Gamma, P \vdash C \; \Gamma, Q \vdash C}{\Gamma, P \vee Q \vdash C}\,(\vee\mathbf{L})$$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \supset Q}\,(\supset\mathbf{R}) \qquad \frac{\Gamma \vdash P \; \Gamma, Q \vdash C}{\Gamma, P \supset Q \vdash C}\,(\supset\mathbf{L})$$

$$\frac{}{\Gamma, P \vdash P}\,(\mathbf{Ax}) \; \frac{}{\Gamma, \bot \vdash C}\,(\bot\mathbf{L}) \; \frac{}{\Gamma \vdash \top}\,(\top\mathbf{R})$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash \forall x.\, P}\,(\forall\mathbf{R}) \qquad \frac{\Gamma, P\{x\mapsto t\} \vdash C}{\Gamma, \forall x.\, P \vdash C}\,(\forall\mathbf{L})$$

$$\frac{\Gamma \vdash P\{x\mapsto t\}}{\Gamma \vdash \exists x.\, P}\,(\exists\mathbf{R}) \qquad \frac{\Gamma, P \vdash C}{\Gamma, \exists x.\, P \vdash C}\,(\exists\mathbf{L})$$

$$\frac{\Gamma \vdash P \;\; \Gamma, P \vdash Q}{\Gamma \vdash Q}\,(\mathbf{Cut}) \qquad \frac{\Gamma, P, P \vdash C}{\Gamma, P \vdash C}\,(\mathbf{Ctrct})$$

$$\frac{\Gamma, t = t \vdash C}{\Gamma \vdash C}\,(=\mathbf{r}) \qquad \frac{\Gamma, t' = t, P\{x\mapsto t'\} \vdash C}{\Gamma, t' = t, P\{x\mapsto t\} \vdash C}\,(=\mathbf{s})$$

Figure 1. First-order Logic with Equality

$$\frac{\Gamma, a\#ts \vdash P\{n\mapsto a\}}{\Gamma, a\#ts \vdash \text{И}n.\, P}\,(\text{И}\mathbf{R}) \quad (P \equiv P'[n, ts])$$

$$\frac{\Gamma, a\#ts, P\{n\mapsto a\} \vdash C}{\Gamma, a\#ts, \text{И}n.\, P \vdash C}\,(\text{И}\mathbf{L}) \quad (P \equiv P'[n, ts])$$

$$\frac{\Gamma, n\#ts \vdash C}{\Gamma \vdash C}\,(\mathbf{new}\mathbb{A}) \quad (n \notin V(\Gamma, C))$$

$$\frac{\Gamma, a\#b \vdash C \quad \Gamma, a = b \vdash C}{\Gamma \vdash C}\,(\mathbf{case}\mathbb{A}) \quad \frac{}{\Gamma, a\#a \vdash C}\,(\#\mathbb{A})$$

$$\frac{\Gamma, (a\;a')\cdot t = t \vdash P}{\Gamma,\; a\#t, a'\#t \vdash P}\,(\pi\#) \qquad \frac{\Gamma, P \vdash C}{\Gamma, (a\;b)\cdot P \vdash C}\,(\pi\mathbf{L})$$

Figure 2. Additional Rules for Fresh Logic

$$\frac{\Gamma, A \vdash C}{\Gamma \vdash C}\,(\mathcal{A}\mathbf{L}) \quad (A \in \mathcal{A})$$

$$\mathcal{A} = \left\{ \begin{array}{l} (a\;a)\cdot t = t, \quad (a\;a')\cdot(a\;a')\cdot t = t \\ (a\;a')\cdot a = a', \quad (a\;a')\cdot \mathsf{c} = \mathsf{c}, \\ (a\;a')\cdot[t\;u] = [(a\;a')\cdot t]\;[(a\;a')\cdot u], \\ (a\;a')\cdot \lambda x.t = \lambda x.(a\;a')\cdot[t\{x\mapsto(a\;a')\cdot x\}] \end{array} \right\}$$

Figure 3. Swapping axioms

The axiom involving λ is particularly interesting: to see why the added substitution is needed, consider applying a swapping to the β-equivalent terms $(\lambda x.x)\;t$ and t.

2.3 The И-quantifier rules

The rules $(\text{И}\mathbf{R})$ and $(\text{И}\mathbf{L})$ are symmetric; we consider just $(\text{И}\mathbf{R})$. As discussed starting from Prop. 4.10 in [4], the И-quantifier satisfies the "some/any" equivalences

$$\text{И}a.P \Leftrightarrow \exists a.a \;\#\; xs \wedge P \Leftrightarrow \forall a.a \;\#\; xs \supset P \qquad (4)$$

where $xs = V(\text{И}a.P)$ is a list of all free variables of P except for a. So at first sight a sequent rule such as

$$\frac{\Gamma, a \;\#\; xs \vdash P\{n\mapsto a\}}{\Gamma, a \;\#\; xs \vdash \text{И}n.\, P}. \qquad (5)$$

(where $V(\text{И}a.P) = xs$) might seem appropriate. A deduction rule should be closed under substitution; this is necessary to prove the substitution property which is needed in cut-elimination for $\forall$ and $\exists$. But (5) is not closed under substitution, for substituting a non-variable term such as $\mathsf{c}(y, z)$ for one of the xs results in a non-instance.

To make the deduction rules for И closed under substitution, we use **slices**. Caires and Cardelli have a similar notion of 'free term' [1].

Definition 1. *A* **slice of** P **over** n *is a tuple* (P, n, ys, P', ts) *of* P*, a variable symbol* n*, and:*

1. *Variable symbols* $y_1, \ldots, y_k$ *which we write* ys*, not appearing in* P
2. *A proposition* P' *with* $V(P') = \{n, y_1, \ldots, y_k\}$.
3. *Terms* $t_1, \ldots, t_k$ *which we write* ts*, such that* $n \notin \bigcup_1^k V(t_i)$ *and* $P'\{y_1\mapsto t_1\}\ldots\{y_k\mapsto t_k\} \equiv P$.

There is a natural notion of minimal slice, the (unique up to renaming the ys) slice such that P' is as small and the ts are as large as possible. We omit the formal definition. For example, the minimal slice of $p(f(x, n), m)$ over n is $(p(f(x, n), m), n, (y_1, y_2), p(f(y_1, n), y_2), (x, m))$, and the minimal slice of the same term with respect to m is $(p(f(x, n), m), m, (y_1), p(y_1, m), f(x, n))$.

We write $P \equiv P'[n, ts]$ as shorthand for "the minimal slice of P over n is (P, n, ys, P', ts) for some ys". The following are useful technical results:

Lemma 1. *If* $P \equiv P'[n, ts]$ *then for any term* s*,* $P\{n\mapsto s\} \equiv P'\{n\mapsto s\}\{ys\mapsto ts\}$*. Also, for any* n *and* s *such that* $n \notin V(s)$*,* $P\{x\mapsto s\} \equiv P'[n, ts\{x\mapsto s\}]$.

Thus a substitution for n in P does not affect the ts, and minimality of slices over n is not affected by substitutions that do not introduce free occurrences of n.

3 Example deductions

$(\mathbf{D1})$ connects И and $\#$ within Fresh Logic. In the instance of $(\text{И}\mathbf{L})$, $P \equiv (n \# y_1)[n,x]$. $(\mathbf{D2})$ derives an equivalent of $(\pi\#)$ within Fresh Logic. In $(\mathbf{D3})$ $V(A)$ and $V(B)$ are the variables of A and B and $a \# V(A,B)$ is a list of freshness assumptions $a \# x$ for each $x \in V(A) \cup V(B)$.

The reverse entailment $\text{И}n.\,(P \wedge Q) \vdash \text{И}n.\,P \wedge \text{И}n.\,Q$ is similarly derivable. We can also derive similar equivalences for the other connectives such as $\supset$ and $\vee$.

$(\mathbf{D4})$ is one direction of a well-known commutativity property $\vdash (\text{И}n.\,\forall x.\,n \# x \supset P) \iff \forall x.\,\text{И}n.\,P$.

$(\mathbf{D5})$ and $(\mathbf{D6})$ are another well-known relation between $\#$, И, and equality, specifically $\vdash a \# x \iff \text{И}b.\,(b\ a) \cdot x = x$ (see [4, Prop. 4.10]).

4 Metatheoretic Properties

Some basic metatheoretic properties include:

Substitution: If $\Gamma \vdash C$ is derivable then so is $\Gamma\{x \mapsto t\} \vdash C\{x \mapsto t\}$, assuming x and t are sort-compatible.

Weakening: If $\Gamma \vdash C$ is derivable then so is $\Gamma, P \vdash C$.

Moreover, these transformations are all height-preserving.

Uses of $(\mathbf{Ax})$ *can be restricted to atomic* P*:* If $\Gamma \vdash P$ is derivable then it is derivable using only instances of $(\mathbf{Ax})$ where the principal formula is atomic. This follows by induction on P. When $P \equiv \text{И}a.\,Q$ the rule $(\mathbf{new}\mathbb{A})$ is required to obtain a fresh atom for $(\text{И}\mathbf{L})$ and $(\text{И}\mathbf{R})$.

A right-hand form $(\pi\mathbf{R})$ *of* $(\pi\mathbf{L})$ *is admissible:* If $\Gamma \vdash P$ is derivable, then there is also a derivation of $\Gamma \vdash (a\ b) \cdot P$.

The proof is a straightforward induction on derivations.

Uses of $(\pi\mathbf{L})$ *can be restricted to atomic* P*:* If $\Gamma \vdash C$ is derivable, then it has a derivation using only atomic instances of $(\pi\mathbf{L})$.

This is proved by a stronger induction on the size of P that an occurrence of $(\pi\mathbf{R})$ or $(\pi\mathbf{L})$ with a non-atomic principal formula can be converted to a derivation using only atomic $(\pi\mathbf{L})$ and $(\pi\mathbf{R})$ rules. This stronger invariant is needed for cases such as $(\supset\mathbf{L})$ and $(\supset\mathbf{R})$ where π-rules are needed on both sides of $\vdash$. Finally, the atomic $(\pi\mathbf{R})$ rule can be derived from $(\mathbf{Ax})$, $(\pi\mathbf{L})$, and the other nonlogical rules.

General $(\text{И}\mathbf{L})$/$(\text{И}\mathbf{R})$ *slice rules:* Generalized versions of the rules $(\text{И}\mathbf{L})$ and $(\text{И}\mathbf{R})$ which permit arbitrary slices instead of minimal ones are admissible.

The proof requires showing that the nonlogical rules can be used to infer $n \# ts$ for $P'[n, ts]$ minimal from $n \# ts'$ for any slice (P, n, ys, P'', ts) of P.

Soundness/Completeness: A Nominal Logic formula P is a theorem if and only if $\vdash P$ is derivable in FL.

This just requires checking that all the axioms of Nominal Logic are derivable and all the rules of FL are valid proof principles in Nominal Logic.

$$(\mathbf{D1})\quad \dfrac{\dfrac{\dfrac{}{n \# x \vdash a \# x}\,(\mathbf{Ax})}{n \# x \vdash \text{И}n.\,n \# x}\,(\text{И}\mathbf{R})}{\vdash \text{И}n.\,n \# x}\,(\mathbf{new}\mathbb{A}).$$

$$(\mathbf{D2})\quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{n, m \# x \vdash x = x}\,(=\mathbf{r}), (\mathbf{Ax})}{n, m \# x \vdash (n\ m) \cdot x = x}\,(\pi\mathbf{L})}{n, m \# x \vdash \text{И}m.\,(n\ m) \cdot x = x}\,(\text{И}\mathbf{R})}{n, m \# x \vdash \text{И}n.\,\text{И}m.\,(n\ m) \cdot x = x}\,(\text{И}\mathbf{R})}{\vdash \text{И}n.\,\text{И}m.\,(n\ m) \cdot x = x}\,(\mathbf{new}\mathbb{A}), (\mathbf{new}\mathbb{A}).$$

$$(\mathbf{D3})\quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,(\mathbf{Ax}) \quad \dfrac{}{B \vdash B}\,(\mathbf{Ax})}{n \# V(A,B),\ A, B \vdash A \wedge B}\,(\wedge\mathbf{R})}{n \# V(A,B),\ A, \text{И}n.\,B \vdash A \wedge B}\,(\text{И}\mathbf{L})}{n \# V(A,B),\ \text{И}n.\,A, \text{И}n.\,B \vdash A \wedge B}\,(\text{И}\mathbf{L})}{n \# V(A,B),\ \text{И}n.\,A, \text{И}n.\,B \vdash A \wedge B}\,(\wedge\mathbf{L})}{n \# V(A,B),\ \text{И}n.\,A, \text{И}n.\,B \vdash \text{И}n.\,(A \wedge B)}\,(\text{И}\mathbf{R})}{\text{И}n.\,A, \text{И}n.\,B \vdash \text{И}n.\,(A \wedge B)}\,(\mathbf{new}\mathbb{A})}{\text{И}n.\,A \wedge \text{И}n.\,B \vdash \text{И}n.\,(A \wedge B)}\,(\wedge\mathbf{L}).$$

$$(\mathbf{D4})\quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{n \# x \vdash n \# x}\,(\mathbf{Ax}) \quad \dfrac{}{P \vdash P}\,(\mathbf{Ax})}{n \# x \supset P,\ n \# x, vs \vdash P}\,(\supset\mathbf{L})}{\text{И}n.\,\forall x.\,n \# x \supset P,\ n \# x, vs \vdash P}\,(\text{И}\mathbf{L}), (\forall\mathbf{L})}{\text{И}n.\,\forall x.\,n \# x \supset P \vdash \text{И}n.\,P}\,(\mathbf{new}\mathbb{A}), (\text{И}\mathbf{R})}{\text{И}n.\,\forall x.\,n \# x \supset P \vdash \forall x.\,\text{И}n.\,P}\,(\forall\mathbf{R})$$

$$(\mathbf{D5})\quad \dfrac{\dfrac{\dfrac{}{(b\ a) \cdot x = x \vdash (b\ a) \cdot x = x}\,(\mathbf{Ax})}{a \# x, b \# x \vdash (b\ a) \cdot x = x}\,(\pi\#)}{a \# x \vdash \text{И}b.\,(b\ a) \cdot x = x}\,(\text{И}\mathbf{R}), (\mathbf{new}\mathbb{A})$$

$$(\mathbf{D6})\quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a \# x \vdash a \# x}\,(\mathbf{Ax})}{(n\ a) \cdot a \# (n\ a) \cdot (n\ a) \cdot x \vdash a \# x}\,(\rightsquigarrow)}{n \# (n\ a) \cdot x \vdash a \# x}\,(\pi\mathbf{L})}{n \# x, (n\ a) \cdot x = x \vdash a \# x}\,(=\mathbf{s})}{n \# x, a, \text{И}n.\,(n\ a) \cdot x = x \vdash a \# x}\,(\text{И}\mathbf{L})}{\text{И}n.\,(n\ a) \cdot x = x \vdash a \# x}\,(\mathbf{new}\mathbb{A})$$

Figure 4. Example deductions

The most important property is, of course, cut-elimination.

Theorem 1 (Cut-elimination). *If $\Gamma \vdash C$ has a derivation in FL, then it has a cut-free derivation.*

This is proved as usual by first proving that $(\mathbf{Cut})$ is admissible and then using admissibility finitely many times to remove all $(\mathbf{Cut})$ instances from a derivation.

Lemma 2 (Admissibility of Cut). *If $\Gamma \vdash A$ and $\Gamma, A \vdash C$ have cut-free derivations, then so does $\Gamma \vdash C$.*

Proof. Proof is by induction on A, derivation Π of $\Gamma \vdash P$, and derivation Π' of $\Gamma, P \vdash C$. In each case we either reduce the size of A or A stays the same but Π or Π' is smaller. There are many cases, which can be classified into

1. *base cases*: Π or Π' is $(\mathbf{Ax})$, $(\bot\mathbf{L})$, or $(\top\mathbf{R})$
2. *left-commutative cases*: A is not principal in Π.
3. *right-commutative cases*: A is not principal in Π'.
4. *principal cases*: A is principal in both Π and Π'.

We may assume that all instances of $(\mathbf{Ax})$ and $(\pi\mathbf{L})$ are atomic in Π, Π'. Many of the cases involve only standard first-order logic rules, and their proofs are as usual. The additional left- and right-commutative cases for the new rules of FL are also straightforward. If A is principal in both derivations, then there are potentially many new cases to consider. One obvious such case is when Π starts with $(\text{И}\mathbf{R})$ and Π' with $(\text{И}\mathbf{L})$. Except for $(\text{И}\mathbf{L})$ and $(\text{И}\mathbf{R})$, all the additional rules act only on atomic formulae on the left. There are no rules with atomic formulae principal on the right, so the only new combination of rules that leads to a principal cut is $(\text{И}\mathbf{R})$ and $(\text{И}\mathbf{L})$.

Suppose the derivations Π, Π' are respectively of the forms

$$\dfrac{\Gamma, \Gamma' \vdash C, A\{n{\mapsto}a\}}{\Gamma, \Gamma' \vdash C, \text{И}n.A}\ (\text{И}\mathbf{R}) \qquad \dfrac{\Gamma, \Gamma', A\{n{\mapsto}b\} \vdash C}{\Gamma, \Gamma', \text{И}n.A \vdash C}\ (\text{И}\mathbf{L})$$

where $\Gamma' \equiv a \mathrel{\#} ts, b \mathrel{\#} ts$, and $A \equiv A'[n, ts]$. If a and b are the same term then this case is immediate. Otherwise, by appealing to the induction hypothesis twice we can conclude

$$\dfrac{\begin{array}{c}\Gamma, \Gamma' \vdash A\{n{\mapsto}a\} \\ \Gamma', A\{n{\mapsto}a\} \vdash A\{n{\mapsto}b\} \\ \Gamma, \Gamma', A\{n{\mapsto}b\} \vdash C\end{array}}{\Gamma, \Gamma' \vdash C}\ IH, IH \quad .$$

The middle judgment follows by

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\{n{\mapsto}b\} \vdash A\{n{\mapsto}b\}}\ (\mathbf{Ax})}{(a\ b)\cdot ts = ts, (a\ b)\cdot A\{n{\mapsto}a\} \vdash A\{n{\mapsto}b\}}\ (\leadsto)}{a \mathrel{\#} ts, b \mathrel{\#} ts, (a\ b)\cdot A\{n{\mapsto}a\} \vdash A\{n{\mapsto}b\}}\ (\pi\#)}{a \mathrel{\#} ts, b \mathrel{\#} ts, A\{n{\mapsto}a\} \vdash A\{n{\mapsto}b\}}\ (\pi\mathbf{I})$$

where the $(\leadsto)$ step follows because if $(a\ b)\cdot ts = ts$ then

$$\begin{aligned}(a\ b)\cdot A\{n{\mapsto}a\} &= A'(a, ts) = A'(b, (a\ b)\cdot ts) \\ &= A'(b, ts) = A\{n{\mapsto}b\} \qquad (6)\end{aligned}$$

using swapping axioms and Lemma 1. $\square$

5 Variations

5.1 Global vs. Local

$(\mathbf{new}\mathbb{A})$ is nondeterministic since ts is arbitrary. Nominal Logic has the similar axiom $(F4)$, which is $\forall xs.\exists a.a \mathrel{\#} xs$. Though convenient for humans this may be undesirable for search space in automated deduction. Note that a logically equivalent **global** variant of $(\mathbf{new}\mathbb{A})'$ insists a be "maximally fresh"; ts must include all variable symbols in the sequent.

$$\dfrac{\Gamma, n \mathrel{\#} ts \vdash C}{\Gamma \vdash C}\ (\mathbf{new}\mathbb{A})'$$

where $n \notin V(\Gamma, C)$. Certainly every instance of $(\mathbf{new}\mathbb{A})'$ is an instance of $(\mathbf{new}\mathbb{A})$; the converse follows using weakening.

5.2 Substitutive vs. Parametric

$(\forall\mathbf{L}), (\exists\mathbf{R})$ are **substitutive** in that the principal formula in the lower sequent is instantiated with a specific term in the upper sequent. $(\forall\mathbf{R}), (\exists\mathbf{R})$ are **parametric** in that the principal formula in the upper sequent contains an eigenvariable (a syntactically new parameter). In this terminology $(\text{И}\mathbf{L})$ and $(\text{И}\mathbf{R})$ as given are both substitutive.

Parametric versions are possible as self-duality (4) suggests:

$$\dfrac{\Gamma, n \mathrel{\#} ts, P \vdash C}{\Gamma, \text{И}n.P \vdash C}\ (\text{И}\mathbf{L})' \quad (n \notin V(\Gamma, C))$$

$$\dfrac{\Gamma, n \mathrel{\#} ts \vdash P}{\Gamma \vdash \text{И}n.P}\ (\text{И}\mathbf{R})' \quad (n \notin V(\Gamma))$$

where $P \equiv P'[n, ts]$.

Call FL$'$ the logic FL minus $(\text{И}\mathbf{L}), (\text{И}\mathbf{R})$ and plus $(\text{И}\mathbf{L})', (\text{И}\mathbf{R})'$. Clearly $(\text{И}\mathbf{L})', (\text{И}\mathbf{R})'$ are derivable in FL using $(\mathbf{new}\mathbb{A})$ and $(\text{И}\mathbf{L}), (\text{И}\mathbf{R})$. The converse uses $(\pi\mathbf{R})$:

$$\dfrac{\dfrac{\dfrac{\Gamma, a \mathrel{\#} ts \vdash P\{n{\mapsto}a\}}{\Gamma, a \mathrel{\#} ts, m \mathrel{\#} ts \vdash (a\ m)\cdot P\{n{\mapsto}a\}}\ (\pi\mathbf{R}), W}{\Gamma, a \mathrel{\#} ts, m \mathrel{\#} ts \vdash P\{n{\mapsto}m\}}\ (\pi\#)^n, (\leadsto)}{\Gamma, a \mathrel{\#} ts \vdash \text{И}n.P}\ (\text{И}\mathbf{R})' \quad .$$

Similarly for $(\text{И}\mathbf{L})$. Thus FL and FL$'$ have the same consequence relation.

5.3 Sequents vs. Natural Deduction

Two versions of Fresh Logic exist: the one in this paper which we temporarily call FL$_{Seq}$, and another is a Natural Deduction system [5], call it FL_{ND}, which we mentioned previously. FL_{Seq} is better for proof-search and suggests notions of logic programming which we consider below. Conversely, FL_{ND} is (arguably) easier to give a semantics to and we do that in [5]. We do not discuss semantics in this paper but we see no barrier to applying the same Kripke-style model in FM sets [4, 5] as we used for FL_{ND}.

5.4 Atoms as Constants vs. Variables

Another basic design decision is whether to treat **atoms as constants (AAC)** or **atoms as variables (AAV)**. Nominal Unification [12], FL_{ND}, and recent work on Nominal Rewriting [3] make the former choice. Nominal Logic [10], FM-HOL [7], and this paper make the latter choice. Atoms-as-constants makes it much easier to test for term equality and unifiability, since equality and syntactic equality coincide for atoms. For example, the nominal unification problem $(a\ b)\cdot x \approx ?\ x$ has multiple incomparable nominal unifiers if a, b are variables, but only one if a and b are atom-constants.

Any Fresh Logic AAC proof certainly be converted to an AAV proof, by mapping atom constants a, b injectively to new variables a, b and augmenting the hypotheses of all judgments with freshness assertions of the form $a \# b$. For example, an AAC judgment like $p(a, b) \vdash q(c, x)$, where a, b, and c are distinct atoms and x is a variable, would be translated to an AAV sequent $a \# b, a \# c, b \# c, p(a, b) \vdash q(c, x)$.

AAV is strictly more expressive than the AAC used in the works cited above: $(x\ y)\cdot \mathsf{c}(t_1, t_2)$ is not an AAC term, because in AAC swappings are always attached directly to free variables. However if a derivable sequent in the Fresh Logic here is expressible as a sequent in the Fresh Logic in [5], then it is derivable. The derivation uses a case-analysis rule (Const$\mathbb{A}$) [5, Fig.3] which we apply over all possible equalities and inequalities between variables of sort atoms in the sequent.

Thus many interesting variations on the rules of FL are possible, in most cases without changing the essence of the logic. The design decisions made in this paper seem useful for manual reasoning in FL; other applications may find advantage in the alternatives.

6 Uniform Proofs in FL

The following grammars define a core logic programming language, consisting of **first-order nominal Horn clause** goals and program clauses:

$$G \quad ::= \quad \top \mid p(ts) \mid G \wedge G \mid \exists x.G \mid \text{И}a.G \qquad (7)$$

$$D \quad ::= \quad \top \mid p(ts) \mid D \wedge D \mid G \supset D \qquad (8)$$
$$\mid \quad \forall x.D \mid \text{И}a.D$$

The И-free fragment is the Horn clause logic programming language. An important property of such a language is completeness of **uniform proof search** [8]. This is goal-directed proof search which decomposes the goal with right-rules until it is atomic, then uses just left-rules. A language of goals and program clauses is an **abstract logic programming language** when uniform proof search is complete; if $D \vdash G$ then a **uniform** derivation exists.

This may fail in the presence of disjunctive program clauses: $A \vee B \vdash A \vee B$ has a trivial proof using $(\mathbf{Ax})$ but no uniform proof. If we decompose the goal $A \vee B$ to A (or symmetrically B), then $A \vee B \vdash A$ is unprovable. And indeed, $\vee$ and for similar reasons $\exists$ are not allowed in D above.

Miller et al. showed uniform provability complete for first- and higher-order variants of Horn clause languages as well as for **hereditary Harrop formulae**, which include additional goal forms including implication and universal quantification. We now develop nominal logic programming languages which generalize first-order Horn clauses and hereditary Harrop formulae and for which an appropriate notion of uniform proof is complete. This provides a solid proof-theoretic foundation for a **nominal logic programming languages** such as αProlog [2].

Definition 2. *A* ***uniform proof*** *in FL is a sequent proof such that for all subderivations* Π *concluding a non-atomic goal* G*, the final inference rule is either an* R*-rule or* $(\mathbf{new}\mathbb{A})$*. A language of goals and program clauses is an* ***abstract logic programming language*** *if for any set* Γ *of program clauses and goal* G*, the judgment* $\Gamma \vdash G$ *has a proof if and only if it has a uniform proof.*

Note that $(\mathbf{new}\mathbb{A})$ may be applied at any point. This is technically necessary because it cannot be permuted up past $(\exists\mathbf{R})$: the witness t may mention the fresh variable generated by $(\mathbf{new}\mathbb{A})$. The $(\pi\mathbf{L})$ rule also increases proof-search complexity since (modulo the other swapping laws) any two atoms may be swapped. These are important considerations for implementations. Some lessons may be drawn from previous work in FM machine deduction [7].

Call $D, \nabla \vdash G$ an ***fnhc*** **judgment**, where ∇ is a set of atomic formulae. An ***fnhc*** **derivation** is one all of whose judgments are *fnhc*. The atomic formula context ∇ is necessary because atomic equality and freshness formulae are not considered program clauses, but many rules (such as $(\mathbf{new}\mathbb{A})$, $(\text{И}\mathbf{L})$, $(\text{И}\mathbf{R})$) manipulate atomic formulae in Γ.

Theorem 2. *fnhc is an abstract logic programming language.*

Proof. We must show that if $\Gamma \vdash C$ is derivable in FL then there is a uniform proof. It suffices to check that (1) Whenever a *fnhc* sequent $\Gamma \vdash C$ has a proof, it has a proof involving only *fnhc* sequents, and (2) For proofs of *fnhc* sequents, all of the left-rules can be permuted above the right-rules and $(\mathbf{new}\mathbb{A})$. Given these facts, we can transform any proof of a *fnhc* sequent into a *fnhc* derivation and permute L-rules up past R-rules for a uniform proof.

The first part is a straightforward induction based on an existing model by Miller et al. [8]; note that this is where ∇ is needed. For the second part, many of the cases (and their proofs) carry over from the proof for first-order Horn clauses. It is easy to verify that $(\text{И}\mathbf{L})$ and the nonlogical left-rules commute up past right rules and $(\mathbf{new}\mathbb{A})$. This makes sense because И behaves like $\forall$ on the left and like $\exists$ on the right, and $\forall$ and $\exists$ are well-behaved on the left and right respectively. □

We now consider the language of **first-order nominal hereditary Harrop** goals G and program clauses D:

$$\begin{array}{rcll} G & ::= & \top \mid p(ts) \mid G \wedge G \mid \exists x.G \mid \text{И} a.G & (9) \\ & \mid & \forall x.G \mid G \vee G \mid D \supset G & \\ D & ::= & \top \mid p(ts) \mid D \wedge D \mid G \supset D & (10) \\ & \mid & \forall x.D \mid \text{И} a.D & \end{array}$$

Call this language *fnhh*. Define *fnhh* sequents and derivations similarly to the case for *fnhc*.

As with Horn clauses, Miller et al. proved that uniform proofs are complete with respect to intuitionistic provability for both first- and higher-order hereditary Harrop programs. We will show that uniform proofs are complete for *fnhh* programs as well.

Theorem 3. *The language fnhh is an abstract logic programming language.*

Proof. As for *fnhc*, before, this theorem follows from two observations. First, if a *fnhh* sequent has any proof, then it has one in which every sequent is *fnhh*. This is straightforward to prove by induction. Second, any left-rule that can arise in such a proof commutes with every right-rule.

The only cases that were not already checked for *fnhc* involve commuting $(\text{И}\mathbf{L})$ and nonlogical rules upwards past $(\forall\mathbf{R})$, $(\vee\mathbf{R_i})$, and $(\supset\mathbf{R})$. These cases are all straightforward, since И behaves like $\forall$ on the left and nonlogical rules are generally well-behaved. □

Adding atomic freshness and equality goals to *fnhc* or *fnhh* is no problem for uniform proof search since it does not comment on how atomic goals are to be derived. We have already noted that the search space for $D, \nabla \vdash t = u$ or $t \# u$ may be large, but this is a separate issue. In addition, adding program clauses denoting orthogonal (possibly conditional) rewriting rules as in functional logic programming seems unlikely to pose problems.

6.1 Examples

Abstractions (as developed in [4, 11]) may be incorporated into FL by adding an abstraction sort constructor $\langle\mathtt{A}\rangle\tau$ and constants $\mathtt{abs}_\tau : \mathtt{A} \to \tau \to \langle\mathtt{A}\rangle\tau$ for each τ. We write $\langle a\rangle t$ for $\mathtt{abs}_\tau\ a\ t$. The following additional laws are needed for abstractions (cf. axioms (A1) and (A2) of [10]):

$$\frac{\Gamma, \langle a\rangle t = \langle b\rangle u \vdash C}{\Gamma, a\#u, t = (a\ b)\cdot u \vdash C}\ (=\!\mathbf{abs})$$

$$\frac{\Gamma, \langle a\rangle x = y \vdash C}{\Gamma, a \# y \vdash C}\ (\mathbf{absL}) \quad (y{:}\langle\mathtt{A}\rangle\tau, x \notin V(\Gamma, a, y, C))$$

Let $\mathtt{Lam}$ be a sort with constructors $\mathtt{v} : \mathtt{A} \to \mathtt{Lam}$, $\mathtt{a} : \mathtt{Lam} \to \mathtt{Lam} \to \mathtt{Lam}$, and $\mathtt{l} : \langle\mathtt{A}\rangle\mathtt{Lam} \to \mathtt{Lam}$ (not to be confused with λ!). Then for closed terms, equality coincides with α-equivalence in the usual sense: for example, $\mathtt{l}(\langle m\rangle \mathtt{v}(m)) = \mathtt{l}(\langle n\rangle \mathtt{v}(n))$.

Capture-avoiding substitution $\sigma : \mathtt{Lam} \to \mathtt{A} \to \mathtt{Lam} \to \mathtt{Lam}$ can be defined using *fnhc* clauses in what is now a standard FM way [4]. A higher-order version $\sigma : (\mathtt{A} \to \mathtt{Lam}) \to \mathtt{Lam} \to \mathtt{Lam}$ may be defined as follows:

$$\sigma(f, \mathtt{v}(n)) = f\ n \quad \sigma(f, \mathtt{a}(x, y)) = \mathtt{a}(\sigma(f, x), \sigma(f, y))$$
$$m \# f \supset \sigma(f, \mathtt{l}(\langle m\rangle x)) = \mathtt{l}(\langle m\rangle \sigma(f, x)).$$

σ is total despite the precondition on the third clause: by $(\mathbf{new}\mathbb{A})$, $(\mathbf{absL})$, for any abstraction y it is always possible to find an m, x such that $m \# f$ and $y = \langle m\rangle x$.

Note that this suggests an essentially algebraic approach to substitution. For any τ with an injection $\nu \to \tau$, it is possible to directly axiomatise capture-avoiding substitution as an algebraic system of equalities for a ternary operator $\sigma : \tau \to \mathtt{A} \to \tau \to \tau$, using $\#/\text{И}$ to avoid capture. These give 'boiler-plate' substitution actions similar to those offered by higher-order encodings. β/η-conversion, typechecking, and evaluation relations can also be encoded using *fnhc* clauses. For example, $n \# x \supset x \to_\eta \mathtt{l}(n, \mathtt{a}(x, \mathtt{v}(n)))$ defines η-conversion.

Let $\mathtt{Pi}$, $\mathtt{Act}$ be sorts for π-calculus process/action terms with channel names encoded using $\mathtt{A}$ and abstraction. Transition rules can be encoded using *fnhc* clauses, for example:

$$(P_1 \overset{\overline{x}(y)}{\to} P_1' \wedge P_2 \overset{xy}{\to} P_2' \wedge y \# P_1) \supset P_1 \mid P_2 \overset{\tau}{\to} \nu\langle y\rangle(P_1' \mid P_2')$$
$$(P \overset{\overline{x}y}{\to} P' \wedge z \# P \wedge z \# x \wedge z \# P') \supset \nu\langle y\rangle P \overset{\overline{x}(z)}{\to} (y\ z)\cdot P'$$

Bisimulation $\cong$ can be encoded using an *fnhh* clause:

$$\begin{array}{c} \Big(\forall P', x, y.\, (P \overset{\overline{x}y}{\to} P' \supset \exists Q'.\, Q \overset{\overline{x}y}{\to} Q' \wedge P' \cong Q') \\ \wedge \forall P', x.\, \text{И} y.\, (P \overset{\overline{x}(y)}{\to} P' \supset \exists Q'.\, Q \overset{\overline{x}(y)}{\to} Q' \wedge P' \cong Q') \\ \wedge \forall P', x.\, \text{И} y.\, (P \overset{xy}{\to} P' \supset \exists Q'.\, Q \overset{xy}{\to} Q' \\ \qquad \wedge \forall z.\, P'[z/y] \cong Q'[z/y]) \\ \wedge \forall P'.\, (P \overset{\tau}{\to} P' \supset \exists Q'.\, Q \overset{\tau}{\to} Q' \wedge P' \cong Q')\Big) \supset P \cong Q \end{array}$$

Also, we can reason effectively using this definition using additional proof rules for stratified definitions such as those of $FO\lambda^{\Delta\nabla}$ [9]; as discussed at the end of the next section adding such rules to FL appears to be straightforward, but is beyond the scope of this paper.

7 Encoding Generic Judgments

Miller and Tiu's sequent calculus $FO\lambda^{\nabla}$ (a sublanguage of $FO\lambda^{\Delta\nabla}$ [9]) is first-order logic over simply-typed λ-terms with an additional quantifier ∇. Like И, ∇ has symmetric left- and right-introduction rules and is self-dual.

$FO\lambda^{\nabla}$ sequents have the form $\Sigma : \Gamma \vdash \mathcal{A}$, where $\mathcal{A} \equiv \sigma \rhd P$ is a formula P in a **local context** σ, and Γ is a multiset of hypotheses with local contexts. Σ is a global context containing eigenvariables introduced by $(\forall\mathbf{R})$ and $(\exists\mathbf{L})$. The more interesting rules of $FO\lambda^{\nabla}$ are shown in Figure 5; the remaining rules are essentially ordinary sequent rules except for the presence of local contexts. The local contexts σ are used for the parameters introduced by the $(\nabla\mathbf{L})$ and $(\nabla\mathbf{R})$ rules. Local and global environments bind the names they mention, and formulae- and judgments-in-context are considered equal up to α-renaming. Thus $x, y \rhd p(x, y) \equiv y, x \rhd p(y, x)$.

Dependence of global parameters on local parameters is encoded using **raising**. Thus, if the local context of the principal formula is σ, then the parametric rules $(\forall\mathbf{R})$ and $(\exists\mathbf{R})$ introduce a parameter h of type $\tau_\sigma \to \tau$ to represent a parameter of type τ, where τ_σ is the sequence of types of parameters in σ, and the application $h\sigma$ (shorthand for $h\ \sigma_1 \cdots \sigma_n$) is substituted for the quantified variable.

In the substitutive rules $(\forall\mathbf{L})$, $(\exists\mathbf{R})$, the substituted term t must be well-formed in the context Σ, σ where σ is the local context of the principal formula. Together, these rules ensure that a formula's validity is independent of the values of any ∇-quantified variables.

$$\frac{(\Sigma, h); \Gamma \vdash \sigma \rhd A[h\sigma/x]}{\Sigma : \Gamma \vdash \sigma \rhd \forall x.A}\ (\forall\mathbf{R}) \quad (h \notin \Sigma)$$

$$\frac{\Sigma, \sigma \vdash t : \tau \quad \Sigma : \Gamma, \sigma \rhd A[t/x] \vdash C}{\Sigma : \Gamma, \sigma \rhd \forall x{:}\tau.A \vdash C}\ (\forall\mathbf{L})$$

$$\frac{\Sigma, \sigma \vdash t : \tau \quad \Sigma : \Gamma \vdash \sigma \rhd A[t/x]}{\Sigma : \Gamma \vdash \sigma \rhd \exists x{:}\tau.A}\ (\exists\mathbf{R})$$

$$\frac{(\Sigma, h); \Gamma, \sigma \rhd A[h\sigma/x] \vdash C}{\Sigma : \Gamma, \sigma \rhd \exists x.A \vdash C}\ (\exists\mathbf{L}) \quad (h \notin \Sigma)$$

$$\frac{\Sigma : \Gamma \vdash (\sigma, y) \rhd A[y/x]}{\Sigma : \Gamma \vdash \sigma \rhd \nabla x.A}\ (\nabla\mathbf{R}) \quad (y \notin \sigma)$$

$$\frac{\Sigma : \Gamma, (\sigma, y) \rhd A[y/x] \vdash C}{\Sigma : \Gamma, \sigma \rhd \nabla x.A \vdash C}\ (\nabla\mathbf{L}) \quad (y \notin \sigma)$$

Figure 5. Interesting rules of $FO\lambda^{\nabla}$

Miller and Tiu showed how to use $FO\lambda^{\Delta\nabla}$, an extension of $FO\lambda^{\nabla}$ with rules for definitions, as a metalanguage to encode object languages and properties of interest, including a late transition system and simulation/bisimulation for the π-calculus and a simple Horn clause logic programming language interpreter.

∇ and И satisfy several common properties, for example both are self dual and commute with all propositional connectives, and

$$QxQy.Bxy \quad \equiv \quad QyQx.Bxy \tag{11}$$

$$Qx.Bxx \quad \not\subset\not\supset \quad QxQx.Bxy \tag{12}$$

for $Q \in \{\text{И}, \nabla\}$.

However, И and ∇ are not identical:

$$\forall x.Bx \supset \text{И}x.\,Bx, \quad \text{И}x.\,Bx \supset \exists x.Bx, \quad B \supset \text{И}x.\,B \tag{13}$$

(where $x \notin V(B)$), none of these are valid for ∇.

We now give a translation of $FO\lambda^{\nabla}$ into FL. This translation combines two essential ideas:

1. $\nabla x{:}\tau$-quantified local parameters can be modeled with И-quantified *atoms* suitably *injected* into τ.
2. $\forall, \exists$-quantified global parameters can be modeled with $\forall, \exists$-quantified *equivariant* parameters suitably *raised* out of the local context σ.

We introduce a constructor $\mathtt{n}_\tau{:}\mathtt{A} \to \tau$ for each $FO\lambda^{\nabla}$ sort τ that provides the injection required for part (1). For part (2), we introduce a predicate $ev_\tau(x)$ (read "x is equivariant") for each $FO\lambda^{\nabla}$ sort τ which is defined as $\forall a{:}\mathtt{A}.a \mathrel{\#} x$. Equivariance is equivalent to invariance under arbitrary swapping.

The translation of $FO\lambda^{\nabla}$ into FL, written $[\![\text{-}]\!]$, is given as follows:

$$\begin{array}{rcl}
[\![t]\!]_\sigma & = & t \\
[\![P \otimes Q]\!]_\sigma & = & [\![P]\!]_\sigma \otimes [\![Q]\!]_\sigma \quad (\otimes \in \{\wedge, \vee, \supset\}) \\
[\![\forall x{:}\tau.P]\!]_\sigma & = & \forall h{:}\tau_\sigma{\to}\tau.ev(h) \supset [\![P]\!]_\sigma\{x{\mapsto}h\sigma\} \\
[\![\exists x{:}\tau.P]\!]_\sigma & = & \exists h{:}\tau_\sigma{\to}\tau.ev(h) \wedge [\![P]\!]_\sigma\{x{\mapsto}h\sigma\} \\
[\![\nabla x{:}\tau.P]\!]_\sigma & = & \text{И}x{:}\mathtt{A}.[\![P]\!]_{(\sigma,x)}\{x{\mapsto}\mathtt{n}_\tau x\} \\
[\![\sigma \rhd P]\!] & = & \text{И}\sigma{:}\mathtt{A}.[\![P]\!]_\sigma\{\sigma{\mapsto}\mathtt{n}\sigma\}
\end{array}$$

In the cases for $\forall$ and $\exists$, $\tau_\sigma \to \tau$ is shorthand for $\tau_1 \to \cdots \to \tau_n \to \tau$, where the types of the local parameters of σ are $\tau_1, \ldots, \tau_n$. The translation of a multiset Γ is the multiset containing translations of all the elements $\mathcal{A} \in \Gamma$. In addition, we translate term-variable environments $\Sigma = x_1 : t, \ldots, x_n : \tau_n$ to to additional hypotheses $[\![\Sigma]\!] = ev(x_1), \ldots, ev(x_n)$. This set of assumptions helps maintain the invariant that any $\exists/\forall$-quantified $FO\lambda^{\nabla}$ variable corresponds to an equivariant $\exists/\forall$-quantified FL variable. The translation of a judgment $\Sigma : \Gamma \vdash \mathcal{C}$ is $[\![\Sigma]\!], [\![\Gamma]\!] \vdash [\![C]\!]$.

We first state without proof that well-formed $FO\lambda^\nabla$ entities (contexts, propositions, terms) are transformed to corresponding well-formed FL entities. We write $\vdash^L_{wf}$ to distinguish the well-formedness judgments of system L from the consequence judgment $\vdash^L$.

Lemma 3 (Well-formedness preservation). *If $\vdash^{FO\lambda^\nabla}_{wf} \Sigma$ then $\vdash^{FL}_{wf} \Sigma$ in FL and $\vdash^{FL}_{wf} [\![\Sigma]\!]$.*
If $\Sigma, \sigma \vdash^{FO\lambda^\nabla}_{wf} t : \tau$ then $\Sigma \vdash^{FL}_{wf} [\![t]\!]_\sigma : \tau$.
If $\Sigma \vdash^{FO\lambda^\nabla}_{wf} \sigma \triangleright A$ then $\Sigma \vdash^{FL}_{wf} [\![\sigma \triangleright A]\!]$.
If $\Sigma \vdash^{FO\lambda^\nabla}_{wf} \Gamma$ then $\Sigma \vdash^{FL}_{wf} [\![\Gamma]\!]$

Two additional important facts are the invertibility of the $\reflectbox{N}$-rules and the fact that well-formed $FO\lambda^\nabla$ terms are translated to equivariant terms modulo $[\![\Sigma]\!]$. We use the notations $P[\sigma, ts]$ for the appropriate generalization of minimal slices to multiple atom-variables; the notation $\sigma \# xs$ indicates that all the variables among σ are fresh for each other and each $x \in xs$.

Lemma 4 ($\reflectbox{N}$-inversion). *Assume the variables of σ are distinct from $xs = V(\Gamma, C, \reflectbox{N}n.P)$.*
If $\Gamma \vdash \reflectbox{N}\sigma.P$ then $\Gamma, \sigma \# xs \vdash P$.
If $\Gamma, \reflectbox{N}\sigma.P \vdash C$ then $\Gamma, \sigma \# xs, P \vdash C$.

Proof. Straightforward induction. □

Lemma 5 (Equivariance). *If $\Sigma, \sigma \vdash^{FO\lambda^\nabla}_{wf} t : \tau$ then $[\![\Sigma]\!] \vdash^{FL} ev(\lambda\sigma.t)$.*

Proof. By induction on the definition of term well-formedness. For the base case $t = x$, x must appear in Σ, so $ev(t) = ev(x)$ is a hypothesis in $[\![\Sigma]\!]$. The remaining cases follow from these easily verified properties of ev: $ev(\mathtt{c})$ for any constant $\mathtt{c}$; $ev(t) \wedge ev(u) \supset ev(t\ u)$ for any t, u such that $\Sigma \vdash t : \tau' \to \tau$, $\Sigma \vdash u : \tau'$; and $(\forall x.ev(x) \supset ev(t)) \supset ev(\lambda x.t)$ for any t such that $\Sigma, x : \tau' \vdash t : \tau$. □

Theorem 4 (Soundness). *If $\Sigma : \Gamma \vdash^{FO\lambda^\nabla} \mathcal{A}$ then $[\![\Sigma]\!], [\![\Gamma]\!] \vdash^{FL} [\![\mathcal{A}]\!]$*

Proof. The proof is by induction on the structure of a $FO\lambda^\nabla$-derivation. All of the cases for propositional connectives and initial sequents are straightforward induction steps. The interesting cases are those for the quantifiers. We illustrate the representative cases of $(\nabla\mathbf{R})$, $(\forall\mathbf{R})$, and $(\exists\mathbf{R})$; the other three cases are symmetric.

For $(\nabla\mathbf{R})$, we have

$$\frac{\Sigma : \Gamma \vdash (\sigma, y) \triangleright A\{x \mapsto y\}}{\Sigma : \Gamma \vdash \sigma \triangleright \nabla x.A}\ (\nabla\mathbf{R})$$

By induction we have $[\![\Sigma]\!], [\![\Gamma]\!] \vdash [\![(\sigma, y) \triangleright A\{x \mapsto y\}]\!]$. Since $[\![(\sigma, y) \triangleright A\{x \mapsto y\}]\!] = \reflectbox{N}\sigma\reflectbox{N}y.[\![A]\!]_{(\sigma,y)}\{x \mapsto \mathtt{n}y\} = \reflectbox{N}\sigma\reflectbox{N}x.[\![A]\!]_{(\sigma,x)}\{x \mapsto \mathtt{n}x\} = [\![\sigma \triangleright \nabla x.A]\!]$, this derivation suffices.

For $(\forall\mathbf{R})$, we have

$$\frac{(\Sigma, h); \Gamma \vdash \sigma \triangleright A\{x \mapsto h\sigma\}}{\Sigma : \Gamma \vdash \sigma \triangleright \forall x.A}\ (\forall\mathbf{R})$$

By induction we have a derivation of $[\![(\Sigma, h)]\!], [\![\Gamma]\!] \vdash [\![\sigma \triangleright A\{x \mapsto h\sigma\}]\!]$, or, unwinding definitions, $[\![\Sigma]\!], ev(h), [\![\Gamma]\!] \vdash \reflectbox{N}\sigma.[\![A\{x \mapsto h\sigma\}]\!]$. Then by the right-inversion lemma for $\reflectbox{N}$, we may derive

$$\cfrac{\cfrac{[\![\Sigma]\!], ev(h), [\![\Gamma]\!], \sigma \# \Sigma \vdash [\![A]\!]\{x \mapsto h\sigma\}}{[\![\Sigma]\!], [\![\Gamma]\!], \sigma \# \Sigma \vdash ev(h) \supset [\![A]\!]\{x \mapsto h\sigma\}}\ (\supset\mathbf{R})}{[\![\Sigma]\!], [\![\Gamma]\!], \sigma \# \Sigma \vdash \forall h.ev(h) \supset [\![A]\!]\{x \mapsto h\sigma\}}\ (\forall\mathbf{R})\ .$$

Then applying $(\reflectbox{N}\mathbf{R})$ finitely many times we obtain a derivation of $[\![\Sigma]\!], [\![\Gamma]\!] \vdash \reflectbox{N}\sigma.\forall h.ev(h) \supset [\![A]\!]\{x \mapsto h\sigma\}$, as required for the definition of $[\![\sigma \triangleright \forall x.A]\!]$.

For $(\exists\mathbf{R})$, we have

$$\frac{\Sigma, \sigma \vdash t : \tau \quad \Sigma : \Gamma \vdash \sigma \triangleright A\{x \mapsto t\}}{\Sigma : \Gamma \vdash \sigma \triangleright \exists x.A}\ (\exists\mathbf{R})$$

Since $\Sigma, \sigma \vdash t : \tau$, by Lemma 5, we have $[\![\Sigma]\!] \vdash ev(\lambda\sigma.t)$. Moreover $\lambda\sigma.t$ is well-formed at type $\tau_\sigma \to \tau$ since $\Sigma, \sigma \vdash t : \tau$. By induction, we have $[\![\Sigma]\!], [\![\Gamma]\!] \vdash \reflectbox{N}\sigma.([\![A]\!]\{x \mapsto t\})$. Note that since h is new, $[\![A]\!]\{x \mapsto t\} = [\![A]\!]\{x \mapsto h\sigma\}\{h \mapsto \lambda\sigma.t\}$. Using right $\reflectbox{N}$-inversion, then deriving

$$\cfrac{\cfrac{[\![\Sigma]\!] \vdash ev(\lambda\sigma.t) \quad [\![\Sigma]\!], [\![\Gamma]\!], \sigma \# \Sigma \vdash [\![A]\!]\{x \mapsto t\}}{[\![\Sigma]\!], [\![\Gamma]\!], \sigma \# \Sigma \vdash ev(h) \wedge [\![A]\!]\{x \mapsto t\}}\ (\wedge\mathbf{R})}{[\![\Sigma]\!], [\![\Gamma]\!], \sigma \# \Sigma \vdash \exists h.ev(h) \wedge [\![A]\!]\{x \mapsto h\sigma\}}\ (\exists\mathbf{R})$$

and using $(\reflectbox{N}\mathbf{R})$ finitely many times and applying the definition of $[\![\sigma \triangleright \exists x.A]\!]$, we see that we can derive the desired sequent.

This completes the proof. □

Completeness does not hold. However, this is for no deep reason: for example, because $FO\lambda^\nabla$ does not allow weakening on local signatures, the judgment $\sigma \triangleright A \vdash (\sigma, x) \triangleright A$ is not derivable and neither is $\vdash \sigma \triangleright A \supset \nabla x.\ A$ for $x \notin V(A)$, whereas its translation $\vdash \reflectbox{N}\sigma.[\![A]\!] \supset \reflectbox{N}\sigma.\reflectbox{N}x.[\![A]\!]$ is derivable in FL. On the other hand, the translation is at least consistent (that is, $[\![\Sigma : \Gamma \vdash \sigma \triangleright \bot]\!] = [\![\Sigma]\!], [\![\Gamma]\!] \vdash \reflectbox{N}\sigma.\bot$ is underivable, since it has no cut-free proof).

Miller and Tiu designed $FO\lambda^\nabla$ as one of a number of related logics. They are concerned with effectively expressing their case studies and not with relative proof-theoretic strength or semantics; there may be obvious translations between variants of $FO\lambda^\nabla$, which may fail to be complete. Despite its incompleteness, therefore, the translation of $FO\lambda^{\nabla}$ into FL sheds light on the meaning of ∇ and the relation between ∇ and И.

Note that $FO\lambda^{\nabla}$ is not necessarily useful by itself. Either definition rules ($FO\lambda^{\Delta\nabla}$) or equality rules ($FO\lambda^{=\nabla}$) permitting case analysis are necessary for a useful metalogic. We have translations to similar extensions of FL and they pose little difficulty. The important insight is that $FO\lambda^{\Delta\nabla}$ and $FO\lambda^{=\nabla}$ use λ-abstractions to ensure that the distinct locally-scoped parameters of σ cannot be unified, and FL-based extensions use freshness constraints forbidding distinct any globally-scoped atoms from being unified.

8 Conclusions

This paper has presented Fresh Logic as a modular extension of a conventional first-order sequent calculus with equality. It seems to have a good proof-theory, which we have discussed here, and a good semantics, which we discuss elsewhere [5] for a related logic. Fresh Logic is part of a growing body of work including Nominal Unification and Rewriting [12, 3], all of which use the same techniques to incorporate names and binding into existing first-order frameworks. As such FL is a useful part of an 'FM toolkit' for handling names and binding.

We have used FL to develop a proof-theoretic foundation for nominal logic programming with И-quantified goals and program clauses and freshness and equality goals. We have shown that the concept of uniform proofs can be extended to this setting and that uniform proof search is complete for appropriate generalizations of Horn clause and hereditary Harrop formulae. We have given some examples of encoding languages and properties thereof using nominal logic programming. Much remains to be done in developing the semantics and implementation techniques needed for nominal logic programming.

Fresh Logic (and related FM work) and $FO\lambda^{\nabla}$ agree in having self-dual quantifiers И and ∇ to explicitly choose fresh names, as in for example π-calculus bisimulation in §6.1. This choice cannot be modeled using $\forall$ or $\exists$ quantification, because we often need to switch from considering all fresh names to some fresh name and back. Our encoding of $FO\lambda^{\nabla}$ in FL highlights the following contributions of this work: 1. We have for the first time related ∇ to И; their relationship had previously been a mystery. 2. $FO\lambda^{\nabla}$ required substantial changes to the structure of first-order judgments and rules; FL is a modular extension of traditional first-order sequent calculus. 3. The translation gives an interesting semantics to ∇ and suggests how others might be found. This is possible future work. The translation also suggests developing a logic similar in spirit and expressiveness to $FO\lambda^{\nabla}$, but using freshness $\#$ in place of local signatures $\sigma \triangleright$ -. This may be of interest to users and designers of logical frameworks who would like to implement Miller-Tiu-like logics with a minimum of changes.

Additional directions for future work include exploring algebraic theories possible in FL, such as that of capture-avoiding substitution; applying the Curry-Howard correspondence to obtain a dependent type theory with names; and adding definitions (along the lines of $FO\lambda^{\Delta\nabla}$) and structural induction principles to obtain a first-order FL-based metalogic for reasoning about abstract syntax with binding. Higher-order syntax encodings either lack straightforward structural induction principles or require additional well-formedness constraints to exclude exotic terms, whereas these problems do not arise for encodings via freshness, И, abstraction, and swapping, since they are essentially first-order [4, 6]; thus, we believe a metalogic based on FL would be very powerful and useful.

References

[1] Luís Caires and Luca Cardelli. A spatial logic for concurrency (part II). In *CONCUR'2002 Proceedings*, number 2421 in Lecture Notes in Computer Science, 2002.

[2] J. Cheney and C. Urban. System description: Alpha-Prolog, a fresh approach to logic programming modulo alpha-equivalence. In *Proc. UNIF'03*, pages 15–19. Universidad Politecnica de Valencia, 2003. DSIC-II/12/03.

[3] Maribel Fernández, Murdoch Gabbay, and Ian Mackie. Nominal rewriting. Submitted, January 2004.

[4] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.

[5] Murdoch Gabbay. Fresh logic. Submitted, July 2003.

[6] Murdoch J. Gabbay. *A Theory of Inductive Definitions with alpha-Equivalence*. PhD thesis, Cambridge, UK, 2000.

[7] Murdoch J. Gabbay. Automating fraenkel-mostowski syntax. In *TPHOLs, 15th International Conference on Theorem Proving in Higher Order Logics*, August 2002.

[8] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[9] Dale Miller and Alwen Tiu. A proof theory for generic judgments: An extended abstract. In *Proceedings of LICS 2003*, pages 118–127. IEEE, June 2003.

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

[11] M. R. Shinwell, A. M. Pitts, and M. J. Gabbay. FreshML: Programming with binders made simple. In *Eighth ACM SIGPLAN International Conference on Functional Programming (ICFP 2003), Uppsala, Sweden*, pages 263–274. ACM Press, August 2003.

[12] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal unification. In M. Baaz, editor, *Computer Science Logic and 8th Kurt Gödel Colloquium (CSL'03 & KGC), Vienna, Austria. Proccedings*, volume 2803 of *Lecture Notes in Computer Science*, pages 513–527. Springer-Verlag, Berlin, 2003.