

# A Logic for Reasoning about Generic Judgments <sup>\*</sup>

Alwen Tiu

Australian National University and National ICT Australia

**Abstract.** This paper presents an extension of a proof system for encoding generic judgments, the logic  $FO\lambda^{\Delta\nabla}$  of Miller and Tiu, with an induction principle. The logic  $FO\lambda^{\Delta\nabla}$  is itself an extension of intuitionistic logic with fixed points and a “generic quantifier”,  $\nabla$ , which is used to reason about the dynamics of bindings in object systems encoded in the logic. A previous attempt to extend  $FO\lambda^{\Delta\nabla}$  with an induction principle has been unsuccessful in modeling some behaviours of bindings in inductive specifications. It turns out that this problem can be solved by relaxing some restrictions on  $\nabla$ , in particular by adding the axiom  $B \equiv \nabla x.B$ , where  $x$  is not free in  $B$ . We show that by adopting the equivariance principle, the presentation of the extended logic can be much simplified. Cut-elimination for the extended logic is stated, and some applications in reasoning about an object logic and a simply typed  $\lambda$ -calculus are illustrated.

## 1 Introduction

This paper aims at providing a framework for reasoning about specifications of deductive systems using *higher-order abstract syntax* [19]. Higher-order abstract syntax is a declarative approach to encoding syntax with bindings using Church’s simply typed  $\lambda$ -calculus. The main idea is to support the notions of  $\alpha$ -equivalence and substitutions in the object syntax by operations in  $\lambda$ -calculus, in particular  $\alpha$ -conversion and  $\beta$ -reduction. There are at least two approaches to higher-order abstract syntax. The *functional programming* approach encodes the object syntax as a data type, where the binding constructs in the object language are mapped to functions in the functional language. In this approach, terms in the object language become values of their corresponding types in the functional language. The *proof search* approach encodes object syntax as expressions in a logic whose terms are simply typed, and functions that act on the object terms are defined via relations, i.e., logic programs. There is a subtle difference between this approach and the former; in the proof search approach, the simple types are inhabited by well-formed expressions, instead of values as in the functional approach (i.e., the abstraction type is inhabited by functions). The proof search approach is often referred to as  *$\lambda$ -tree syntax* [16], to distinguish it from the functional approach. This paper concerns the  $\lambda$ -tree syntax approach.

Specifications which use  $\lambda$ -tree syntax are often formalized using hypothetical and generic judgments in intuitionistic logic. It is enough to restrict to the fragment of first-order intuitionistic logic whose only formulas are those of hereditary Harrop formulas, which we will refer to as the *HH* logic. Consider for instance the problem of defining the data type for untyped  $\lambda$ -terms. One first introduces the following constants:

$$app : tm \rightarrow tm \rightarrow tm \quad abs : (tm \rightarrow tm) \rightarrow tm$$

where the type  $tm$  denotes the syntactic category of  $\lambda$ -terms and  $app$  and  $abs$  encode application and abstraction, respectively. The property of being a  $\lambda$ -term is then defined via the following theory:

$$\bigwedge M \bigwedge N (lam\ M \wedge lam\ N \Rightarrow lam\ (app\ M\ N)) \ \& \\ \bigwedge M ((\bigwedge x. lam\ x \Rightarrow lam\ (M\ x)) \Rightarrow lam\ (abs\ M))$$

where  $\bigwedge$  is the universal quantifier and  $\Rightarrow$  is implication.

---

<sup>\*</sup> Draft, May 8, 2007.

Reasoning about object systems encoded in  $HH$  is reduced to reasoning about the structure of proofs in  $HH$ . McDowell and Miller formalize this kind of reasoning in the logic  $FO\lambda^{\Delta\mathbb{N}}$  [10], which is an extension of first-order intuitionistic logic with fixed points and natural numbers induction. This is done by encoding the sequent calculus of  $HH$  inside  $FO\lambda^{\Delta\mathbb{N}}$  and prove properties about it. We refer to  $HH$  as object logic and  $FO\lambda^{\Delta\mathbb{N}}$  as meta logic. McDowell and Miller considered different styles of encodings and concluded that explicit representations of hypotheses and, more importantly, eigenvariables of the object logic are required in order to capture some statements about object logic provability in the meta logic [11]. One typical example involves the use of hypothetical and generic reasoning as follows: Suppose that the following formula is provable in  $HH$ .

$$\bigwedge x.pxs \Rightarrow \bigwedge y.pyt \Rightarrow pxt.$$

By inspection on the inference rules of  $HH$ , one observes that this is only possible if  $s$  and  $t$  are syntactically equal. This observation comes from the fact that the right introduction rule for universal quantifier, reading the rule bottom-up, introduces new constants, or eigenvariables. The quantified variables  $x$  and  $y$  will be replaced by distinct eigenvariables and hence the only matching hypothesis for  $pxt$  would be  $pxs$ , and therefore  $s$  and  $t$  has to be equal. Let  $\vdash_{HH} F$  denote the provability of the formula  $F$  in  $HH$ . Then in the meta logic, we would want to be able to prove the statement:

$$\forall s\forall t.(\vdash_{HH} \bigwedge x.pxs \Rightarrow \bigwedge y.pyt \Rightarrow pxt) \supset s = t.$$

The question is then how we would interpret the object logic eigenvariables in the meta logic. It is demonstrated in [11] that the existing quantifiers in  $FO\lambda^{\Delta\mathbb{N}}$  cannot be used to capture the behaviours of object logic eigenvariables directly. McDowell and Miller then resort to a non-logical encoding technique (in the sense that no logical connectives are used) which has some similar flavor to the use of deBruijn indices. The use of this encoding technique, however, has a consequence that substitutions in the object logic has to be formalized explicitly.

Motivated by the above mentioned limitation of  $FO\lambda^{\Delta\mathbb{N}}$ , Miller and Tiu later introduce a new quantifier  $\nabla$  to  $FO\lambda^{\Delta\mathbb{N}}$  which allows one to move the binders from the object logic to the meta logic. A generic judgment in the object logic, for instance  $\vdash_{HH} \bigwedge x.Gx$  is reflected in the meta logic as  $\nabla x.\vdash_{HH} Gx$ . This meta logic, called  $FO\lambda^{\Delta\nabla}$  [17], allows one to perform case analyses on the provability of the object logic. Tiu later extended  $FO\lambda^{\Delta\nabla}$  with induction and co-induction rules, resulting in the logic  $\text{Linc}$  [23]. However, some inductive properties about the object logic are not provable in  $\text{Linc}$ . For example, the fact that  $\vdash_{HH} \bigwedge x.Gx$  implies  $\forall t.\vdash_{HH} Gt$  (that is, the extensional property of universal quantification) is not provable in  $\text{Linc}$ . As it is shown in [23], this is partly caused by the fact that  $B \equiv \nabla x.B$ , where  $x$  is not free in  $B$ , is not provable in  $\text{Linc}$  or  $FO\lambda^{\Delta\nabla}$ . In this paper we present the logic  $LG^\omega$ , which is an extension of  $FO\lambda^{\Delta\nabla}$  with natural number induction and with the axiom schemes:

$$\nabla x\nabla y.Bxy \supset \nabla y\nabla x.Bxy \quad \text{and} \quad B \equiv \nabla x.B \tag{1}$$

where  $x$  is not free in  $B$  in the second scheme. We show that inductive properties of  $\lambda$ -tree syntax specifications can be stated directly and in a purely logical fashion, and proved in  $LG^\omega$ .

*Relation to nominal logic* In formulating the proof system for  $LG^\omega$ , it turns out that we can simplify the presentation a lot if we adopt the idea of *equivariant predicates* from nominal logic [20]. That is, provability of a predicate is invariant under permutations of *names*. This is technically done by introducing a countably infinite set of name constants into the logic, and change the identity rule of the logic to allow equivalence under permutations of name constants:

$$\frac{\pi.B = \pi'.B'}{\Gamma, B \vdash B'} \text{ id}$$

where  $\pi$  and  $\pi'$  are permutations on names.  $LG^\omega$  is in fact very close to nominal logic, when we consider only the behaviours of logical connectives. In particular, the quantifier  $\nabla$  in  $LG^\omega$  shares the same properties, in relation to other connectives of the logic, with the  $\forall$  quantifier in nominal logic. However, there are two

important differences in our approach. First, we do not attempt to redefine  $\alpha$ -conversion and substitutions in  $LG^\omega$  in terms of permutations (or *swapping*) and the notion of *freshness* as in nominal logic. Name swapping and freshness constraints are not part of the syntax of  $LG^\omega$ . These notions are present only in the meta theory of the logic. In  $LG^\omega$ , for example, variables are always considered to have empty support, that is,  $\pi.x = x$  for every permutation  $\pi$ . This is because we restrict substitutions to the “closed” ones, in the sense that no name constants can appear in the substitutions. A restricted form of open substitutions can be recovered indirectly at the meta theory of  $LG^\omega$ . The fact that variables have empty support allows one to work with permutation free formulas and terms. So in  $LG^\omega$ , we can prove that  $p x a \supset p x b$ , where  $a$  and  $b$  are names, without using explicit axioms of permutations and freshness. In nominal logic, one would prove this by using the swapping axiom  $p x a \supset p ((a b).x) ((a b).b)$ , where  $(a b)$  denotes a swapping of  $a$  and  $b$ , and then show that  $(a b).x = x$ . The latter might not be valid if  $x$  is substituted by  $a$ , for example. The validity of this formula in nominal logic would therefore depend on the assumption on the support of  $x$ .

The second difference between  $LG^\omega$  and nominal logic is that  $LG^\omega$  allows closed terms (again, in the sense that no name constants appear in them) of type name, while in nominal logic, allowing such terms would lead to an inconsistent theory in nominal logic [20]. As an example, the type  $tm$  in the encoding of  $\lambda$ -terms mentioned previously can be treated as a nominal type in  $LG^\omega$ . This has an important consequence that we do not need to redefine the notion of substitutions for the encoded  $\lambda$ -terms. For example, we can define the (lazy) evaluation relation on untyped  $\lambda$ -terms as the theory:

$$\begin{aligned} eval (abs M) (abs M) &\equiv \top \\ eval (app M N) V &\equiv eval M (abs P) \wedge eval (P N) V \end{aligned}$$

without having to explicitly define substitutions on terms of type  $tm$  inside  $LG^\omega$ . Substitutions in the object language in this case is modelled by  $\beta$ -reduction in the meta-language of  $LG^\omega$ .

The rest of this paper is organized as follows. In Section 2 we introduce a proof system for  $LG^\omega$ . Section 3 states some meta theories of  $LG^\omega$ , in particular cut-elimination and a translation from  $LG^\omega$  without fixed points and induction to  $FO\lambda^\nabla$  with the axioms (1). Section 4 shows an encoding of  $HH$  logic in  $LG^\omega$  and how some properties of the object logic can be formalized in  $LG^\omega$ . Section 5 illustrates the use of  $HH$  to specify the typing judgments of  $\lambda$ -calculus and the evaluation relation on  $\lambda$ -terms. It also shows an example of reasoning about the encoded  $\lambda$ -calculus, by induction on the provability of the typing judgments in the object logic  $HH$ . Section 6 discusses some related and future work. The proofs of the main results in this paper can be found in the appendix.

## 2 A logic for generic judgments

We first define the core fragment of the logic  $LG^\omega$  which does not have fixed point rules or induction. The starting point is the logic  $FO\lambda^\nabla$  introduced in [17].  $FO\lambda^\nabla$  is an extension of a subset of Church’s Simple Theory of Types in which formulas are given the type  $o$ . The core fragment of  $LG^\omega$ , which we refer to as  $LG$ , shares the same set of connectives as  $FO\lambda^\nabla$ , namely,  $\perp$ ,  $\top$ ,  $\wedge$ ,  $\vee$ ,  $\supset$ ,  $\forall_\tau$ ,  $\exists_\tau$  and  $\nabla_\tau$ . The type  $\tau$  in the quantifiers is restricted to that which does not contain the type  $o$ . Hence the logic is essentially first-order. We abbreviate  $(B \supset C) \wedge (C \supset B)$  as  $B \equiv C$ .

The sequents of  $FO\lambda^{\Delta\nabla}$  are expressions of the form

$$\Sigma; \sigma_1 \triangleright B_1, \dots, \sigma_n \triangleright B_n \multimap \sigma_0 \triangleright B_0$$

where  $\Sigma$  is a signature, i.e., a set of eigenvariables scoped over the sequent and  $\sigma_i$  is a local signature, i.e., list of variables locally scoped over  $B_i$ . The introduction rules for  $\nabla$ , reading the rules bottom-up, introduce new local variables to the local signatures, just as the right introduction rule of  $\forall$  introduces new eigenvariables to the signature. The expression  $\sigma_i \triangleright B_i$  is called a local judgment, and is identified up to renaming of variables in  $\sigma_i$ . This enforces a limited notion of equivariance: for example  $a \triangleright pa \multimap b \triangleright pb$  is provable, since both local judgments are equivalent up to renaming of local signatures. However, the judgments  $(a, c) \triangleright pa$  and  $b \triangleright pb$  are considered distinct judgments, and so are  $(a, b) \triangleright qab$  and  $(b, a) \triangleright qab$ . These restrictions are relaxed in  $LG$ .

The sequent presentation of  $LG$  can be simplified, that is, without using the local signatures, if we employ the equivariance principle. For this purpose, we introduce a distinguished set of base types, called *nominal types*, which is denoted with  $\mathcal{N}$ . Nominal types are ranged over by  $\iota$ . We restrict the  $\nabla$  quantifier to nominal types. For each nominal type  $\iota \in \mathcal{N}$ , we assume an infinite number of constants of that type. These constants are called *nominal constants*. We denote the family of nominal constants by  $\mathcal{C}_{\mathcal{N}}$ . The role of the nominal constants is to enforce the notion of equivariance: provability of formulas is invariant under permutations of nominal constants. Depending on the application, we might also assume a set of non-nominal constants, which is denoted by  $\mathcal{K}$ .

We assume the usual notion of capture-avoiding substitutions. Substitutions are ranged over by  $\theta$  and  $\rho$ . Application of substitutions is written in a postfix notation, e.g.,  $t\theta$  is an application of  $\theta$  to the term  $t$ . Given two substitutions  $\theta$  and  $\theta'$ , we denote their composition by  $\theta \circ \theta'$  which is defined as  $t(\theta \circ \theta') = (t\theta)\theta'$ . A *signature* is a set of variables. A substitution  $\theta$  respects a given signature  $\Sigma$  if there exists a set of typed variables  $\Sigma'$  such that for every  $x : \tau$  in the domain of  $\theta$ , it holds that  $\mathcal{K} \cup \Sigma' \vdash \theta(x) : \tau$ . We denote by  $\Sigma\theta$  the minimal set of variables satisfying the above condition. We assume that variables, free or bound, are of a different syntactic category from constants.

**Definition 1.** A permutation on  $\mathcal{C}_{\mathcal{N}}$  is a bijection from  $\mathcal{C}_{\mathcal{N}}$  to  $\mathcal{C}_{\mathcal{N}}$ . The permutations on  $\mathcal{C}_{\mathcal{N}}$  are ranged over by  $\pi$ . Application of a permutation  $\pi$  to a nominal constant  $a$  is denoted with  $\pi(a)$ . We shall be concerned only with permutations which respect types, i.e., for every  $a : \iota$ ,  $\pi(a) : \iota$ . Further, we shall also restrict to permutations which are finite, that is, the set  $\{a \mid \pi(a) \neq a\}$  is finite. Application of a permutation to an arbitrary term (or formula), written  $\pi.t$ , is defined as follows:

$$\begin{aligned} \pi.a &= \pi(a), \text{ if } a \in \mathcal{C}_{\mathcal{N}}. & \pi.c &= c, \text{ if } c \notin \mathcal{C}_{\mathcal{N}}. & \pi.x &= x \\ \pi.(M N) &= (\pi.M) (\pi.N) & \pi.(\lambda x.M) &= \lambda x.(\pi.M) \end{aligned}$$

A permutation involving only two nominal constants is called *swapping*. We use  $(a b)$ , where  $a$  and  $b$  are constants of the same type, to denote the swapping  $\{a \mapsto b, b \mapsto a\}$ .

The *support* of a term (or formula)  $t$ , written  $\text{supp}(t)$ , is the set of nominal constants appearing in it. It is clear from the above definition that if  $\text{supp}(t)$  is empty, then  $\pi.t = t$  for all  $\pi$ . The definition of  $\Sigma$ -substitution implies that for every  $\theta$  and for every  $x \in \text{dom}(\theta)$ ,  $\theta(x)$  has empty support. Therefore  $\Sigma$ -substitutions and permutations commute, that is,  $(\pi.t)\theta = \pi.(t\theta)$ .

A sequent in  $LG^\omega$  is an expression of the form  $\Sigma; \Gamma \vdash C$  where  $\Sigma$  is a signature. The free variables of  $\Gamma$  and  $C$  are among the variables in  $\Sigma$ . The inference rules for the core fragment of  $LG^\omega$ , i.e., the logic  $LG$ , is given in Figure 1. In the rules, the typing judgment  $\Sigma, \mathcal{K}, \mathcal{C}_{\mathcal{N}} \vdash t : \tau$  denotes the typability of  $t : \tau$ , given the typing context  $\Sigma \cup \mathcal{K} \cup \mathcal{C}_{\mathcal{N}}$  in Church's simple type system.

In the  $\nabla\mathcal{L}$  and  $\nabla\mathcal{R}$  rules,  $a$  denotes a nominal constant. In the  $\exists\mathcal{L}$  and  $\forall\mathcal{R}$  rules, we use *raising* [14] to encode the dependency of the quantified variable on the support of  $B$ , since we do not allow  $\Sigma$ -substitutions to mention any nominal constants. In the rules, the variable  $h$  has its type raised in the following way: suppose  $\vec{c}$  is the list  $c_1 : \iota_1, \dots, c_n : \iota_n$  and the quantified variable  $x$  is of type  $\tau$ . Then the variable  $h$  is of type:  $\iota_1 \rightarrow \iota_2 \rightarrow \dots \rightarrow \iota_n \rightarrow \tau$ . This raising technique is similar to that of  $FO\lambda^{\Delta\nabla}$ , and is used to encode explicitly the minimal support of the quantified variable. Its use prevents one from mixing the scopes of  $\forall$  (dually,  $\exists$ ) and  $\nabla$ . That is, it prevents the formula  $\forall x \nabla y. pxy \equiv \nabla y \forall x. pxy$ , and its dual, to be proved.

Looking at the introduction rules for  $\forall$  and  $\exists$ , one might notice the asymmetry between the left and the right introduction rules. The left rule for  $\forall$  allows instantiations with terms containing any nominal constants while the raised variable in the right introduction rule of  $\forall$  takes into account only those which are in the support of the quantified formula. However, we will see that we can extend the dependency of the raised variable to an arbitrary number of fresh nominal constants not in the support without affecting the provability of the sequent (see Lemma 9 and Lemma 10).

We now extend the logic  $LG$  with a proof theoretic notion of equality and fixed points, following on works by Hallnas and Schroeder-Heister [7, 21], Girard [6] and McDowell and Miller [10]. The equality rules are as follows:

$$\frac{\{\Sigma\theta; \Gamma\theta \vdash C\theta \mid (\lambda \vec{c}.t)\theta =_{\beta\eta} (\lambda \vec{c}.s)\theta\}}{\Sigma; \Gamma, s = t \vdash C} \text{eq}\mathcal{L} \qquad \frac{}{\Sigma; \Gamma \vdash t = t} \text{eq}\mathcal{R}$$

$$\begin{array}{c}
\frac{\pi.B = \pi'.B'}{\Sigma; \Gamma, B \vdash B'} \text{id}_\pi \quad \frac{\Sigma; \Gamma \vdash B \quad \Sigma; B, \Delta \vdash C}{\Sigma; \Gamma, \Delta \vdash C} \text{cut} \quad \frac{\Sigma; \Gamma, B, B \vdash C}{\Sigma; \Gamma, B \vdash C} \text{c}\mathcal{L} \\
\frac{}{\Sigma; \Gamma, \perp \vdash C} \perp\mathcal{L} \quad \frac{}{\Sigma; \Gamma \vdash \top} \top\mathcal{R} \\
\frac{\Sigma; \Gamma, B_i \vdash C}{\Sigma; \Gamma, B_1 \wedge B_2 \vdash C} \wedge\mathcal{L}, i \in \{1, 2\} \quad \frac{\Sigma; \Gamma \vdash B \quad \Sigma; \Gamma \vdash C}{\Sigma; \Gamma \vdash B \wedge C} \wedge\mathcal{R} \\
\frac{\Sigma; \Gamma, B \vdash C \quad \Sigma; \Gamma, D \vdash C}{\Sigma; \Gamma, B \vee D \vdash C} \vee\mathcal{L} \quad \frac{\Sigma; \Gamma \vdash B_i}{\Sigma; \Gamma \vdash B_1 \vee B_2} \vee\mathcal{R}, i \in \{1, 2\} \\
\frac{\Sigma; \Gamma \vdash B \quad \Sigma; \Gamma, D \vdash C}{\Sigma; \Gamma, B \supset D \vdash C} \supset\mathcal{L} \quad \frac{\Sigma; \Gamma, B \vdash C}{\Sigma; \Gamma \vdash B \supset C} \supset\mathcal{R} \\
\frac{\Sigma, \mathcal{K}, \mathcal{C}_N \vdash t : \tau \quad \Sigma; \Gamma, B[t/x] \vdash C}{\Sigma; \Gamma, \forall x.B \vdash C} \forall\mathcal{L} \quad \frac{\Sigma, h; \Gamma \vdash B[h\bar{c}/x]}{\Sigma; \Gamma \vdash \forall x.B} \forall\mathcal{R}, h \notin \Sigma, \text{supp}(B) = \{\bar{c}\} \\
\frac{\Sigma; \Gamma, B[a/x] \vdash C}{\Sigma; \Gamma, \nabla x.B \vdash C} \nabla\mathcal{L}, a \notin \text{supp}(B) \quad \frac{\Sigma; \Gamma \vdash B[a/x]}{\Sigma; \Gamma \vdash \nabla x.B} \nabla\mathcal{R}, a \notin \text{supp}(B) \\
\frac{\Sigma, h; \Gamma, B[h\bar{c}/x] \vdash C}{\Sigma; \Gamma, \exists x.B \vdash C} \exists\mathcal{L}, h \notin \Sigma, \text{supp}(B) = \{\bar{c}\} \quad \frac{\Sigma, \mathcal{K}, \mathcal{C}_N \vdash t : \tau \quad \Sigma; \Gamma \vdash B[t/x]}{\Sigma; \Gamma \vdash \exists x.B} \exists\mathcal{R}
\end{array}$$

**Fig. 1.** The inference rules of  $LG$

where  $\text{supp}(s = t) = \{\bar{c}\}$  in the eq $\mathcal{L}$  rule. In the eq $\mathcal{L}$  rule, the substitution  $\theta$  is a *unifier* of  $\lambda\bar{c}.s$  and  $\lambda\bar{c}.t$ . We specify the premise of the rule as a set to mean that every element of the set is a premise. Since the terms  $s$  and  $t$  can be arbitrary higher-order terms, in general the set of their unifiers can be infinite. However, in some restricted cases, e.g., when  $\lambda\bar{c}.s$  and  $\lambda\bar{c}.t$  are *higher-order pattern* terms [13, 18], if both terms are unifiable, then there exists a most general unifier. The applications we are considering are those which satisfy the higher-order pattern restrictions.

**Definition 2.** To each atomic formula, we associate a fixed point equation, or a definition clause, following the terminology of  $FO\lambda^{\Delta\nabla}$ . A definition clause is written  $\forall\bar{x}.p\bar{x} \triangleq B$  where the free variables of  $B$  are among  $\bar{x}$ . The predicate  $p\bar{x}$  is called the head of the definition clause, and  $B$  is called the body. A definition is a set of definition clauses. We often omit the outer quantifiers when referring to a definition clause.

The introduction rules for defined atoms are as follows:

$$\frac{\Sigma; \Gamma, B[\bar{t}/\bar{x}] \vdash C}{\Sigma; \Gamma, p\bar{t} \vdash C} \text{def}\mathcal{L}, p\bar{x} \triangleq B \quad \frac{\Sigma; \Gamma \vdash B[\bar{t}/\bar{x}]}{\Sigma; \Gamma \vdash p\bar{t}} \text{def}\mathcal{R}, p\bar{x} \triangleq B$$

In order to prove the cut-elimination theorem and the consistency of  $LG^\omega$ , we allow only definition clauses which satisfy an *equivariance preserving* condition and a certain positivity condition, so as to guarantee the existence of fixed points.

**Definition 3.** We associate with each predicate symbol  $p$  a natural number, the level of  $p$ . Given a formula  $B$ , its level  $lvl(B)$  is defined as follows:

1.  $lvl(p\bar{t}) = lvl(p)$
2.  $lvl(\perp) = lvl(\top) = 0$
3.  $lvl(B \wedge C) = lvl(B \vee C) = \max(lvl(B), lvl(C))$
4.  $lvl(B \supset C) = \max(lvl(B) + 1, lvl(C))$
5.  $lvl(\forall x.B) = lvl(\nabla x.B) = lvl(\exists x.B) = lvl(B)$ .

A definition clause  $p\bar{x} \triangleq B$  is stratified if  $lvl(B) \leq lvl(p)$  and  $B$  has no free occurrences of nominal constants. We consider only definition clauses which are stratified.

An example that violates the first restriction in Definition 3 is the definition  $p \triangleq p \supset \perp$ . In [21], Schroeder-Heister shows that admitting this definition in a logic with contraction leads to inconsistency. To see why we need the second restriction on name constants, consider the definition  $qx \triangleq (x = a)$ , where  $a$  is a nominal constant. Let  $b$  be a nominal constant different from  $a$ . Using this definition, we would be able to derive  $\perp$ :

$$\frac{\frac{\frac{}{\vdash a = a} \text{eq}\mathcal{R}}{\vdash qa} \text{def}\mathcal{R} \quad \frac{\frac{}{qa \vdash qb} \text{id}_\pi \quad \frac{\frac{}{b = a \vdash \perp} \text{eq}\mathcal{L}}{qb \vdash \perp} \text{def}\mathcal{L}}{qa \vdash \perp} \text{cut}}{\vdash \perp} \text{cut}}{\vdash \perp} \text{cut}$$

In examples and applications, we often express definition clauses with patterns in the heads. Let us consider, for example, a definition clause for lists. We first introduce a type  $lst$  to denote lists of elements of type  $\alpha$ , and the constants

$$nil : lst \quad :: \quad : \alpha \rightarrow lst \rightarrow lst$$

which denote the empty list and a constructor to build a list from an element of type  $\alpha$  and another list. The latter will be written in the infix notation. The definition clause for *lists* is as follows.

$$list\ L \triangleq L = nil \vee \exists_\alpha A \exists_{lst} L'. L = (A :: L') \wedge list\ L'.$$

Using patterns, the above definition of lists can be rewritten as

$$list\ nil \triangleq \top. \quad list\ (A :: L) \triangleq list\ L.$$

We shall often work directly with this patterned notation for definition clauses. For this purpose, we introduce the notion of *patterned definitions*. A *patterned definition clause* is written  $\forall \vec{x}. H \triangleq B$  where the free variables of  $H$  and  $B$  are among  $\vec{x}$ . The stratification of definitions in Definition 3 applies to patterned definitions as well. Since the patterned definition clauses are not allowed to have free occurrences of nominal constants, in matching the heads of the clauses with an atomic formula in a sequent, we need to raise the variables of the clauses to account for nominal constants that are in the support of the introduced formula. Given a patterned definition clause  $\forall x_1 \dots \forall x_n. H \triangleq B$  its raised clause with respect to the list of constants  $c_1 : \iota_1 \dots c_n : \iota_n$  is

$$\forall h_1 \dots \forall h_n. H[h_1 \vec{c}/x_1, \dots, h_n \vec{c}/x_n] \triangleq B[h_1 \vec{c}/x_1, \dots, h_n \vec{c}/x_n].$$

The introduction rules for patterned definitions are

$$\frac{\{\Sigma\theta; B\theta, \Gamma\theta \vdash C\theta\}_\theta}{\Sigma; A, \Gamma \vdash C} \text{def}\mathcal{L} \quad \frac{\Sigma; \Gamma \vdash B\theta}{\Sigma; \Gamma \vdash A} \text{def}\mathcal{R}$$

In the  $\text{def}\mathcal{L}$  rule,  $B$  is the body of the raised patterned clause  $\forall x_1 \dots \forall x_n. H \triangleq B$  and  $(\lambda \vec{c}. H)\theta = (\lambda \vec{c}. A)\theta$  where  $\{\vec{c}\}$  is the support of  $A$ . In the  $\text{def}\mathcal{R}$  rule, we match  $A$  with the head of the clause, i.e.,  $\lambda \vec{c}. A = (\lambda \vec{c}. H)\theta$ . These patterned rules can be derived using the non-patterned definition rules and the equality rules, as shown in [23],

*Natural number induction.* We introduce a type  $nt$  to denote natural numbers, with the usual constants  $z : nt$  (zero) and  $s : nt \rightarrow nt$  (the successor function), and a special predicate  $nat : nt \rightarrow nt \rightarrow o$ . The rules for natural number induction are the same as those in  $FO\lambda^{\Delta\mathbb{N}}$  [10], which are the introduction rules for the predicate  $nat$ .

$$\frac{\vdash Dz \quad j; Dj \vdash D(sj) \quad \Sigma; \Gamma, DI \vdash C}{\Sigma; \Gamma, nat\ I \vdash C} \text{nat}\mathcal{L}$$

$$\frac{}{\Sigma; \Gamma \vdash nat\ z} \text{nat}\mathcal{R} \quad \frac{\Sigma; \Gamma \vdash nat\ I}{\Sigma; \Gamma \vdash nat\ (sI)} \text{nat}\mathcal{R}$$

The logic  $LG$  extended with the equality, definitions and induction rules is referred to as  $LG^\omega$ .

### 3 The meta theory of $LG^\omega$

In this section we investigate some properties of the logic  $LG^\omega$ . We first look at the properties of the  $\nabla$  quantifier in relation to other connectives. The proof of the following proposition is straightforward by inspection on the rules of  $LG$ .

**Proposition 4.** *The following formulas are provable in  $LG$ :*

1.  $\nabla x.(Bx \wedge Cx) \equiv \nabla x.Bx \wedge \nabla x.Cx.$
2.  $\nabla x.(Bx \supset Cx) \equiv \nabla x.Bx \supset \nabla x.Cx.$
3.  $\nabla x.(Bx \vee Cx) \equiv \nabla x.Bx \vee \nabla x.Cx.$
4.  $\nabla x.B \equiv B$ , provided that  $x$  is not free in  $B$ .
5.  $\nabla x \nabla y.Bxy \equiv \nabla y \nabla x.Bxy.$
6.  $\forall x.Bx \supset \nabla x.Bx.$
7.  $\nabla x.Bx \supset \exists x.Bx.$

The formulas (1) – (3) are provable in  $FO\lambda^\nabla$ . The proposition is true also in nominal logic with  $\nabla$  replaced by  $\mathbb{I}$ .

The following properties concern the transformation of derivations. Provability is preserved under  $\Sigma$ -substitutions, permutations and a restricted form of name substitutions.

**Definition 5.** *Given a derivation  $\Pi$  with premise derivations  $\{\Pi_i\}_{i \in \mathcal{I}}$  where  $\mathcal{I}$  is some index set, the measure  $ht(\Pi)$ , the height of  $\Pi$ , is defined as the least upper bound of  $\{ht(\Pi_i) + 1\}_{i \in \mathcal{I}}$ .*

**Lemma 6.** *Substitutions. Let  $\Pi$  be a proof of  $\Sigma; \Gamma \vdash C$  and let  $\theta$  be a  $\Sigma$ -substitution. Then there exists a proof  $\Pi'$  of  $\Sigma\theta; \Gamma\theta \vdash C\theta$  such that  $ht(\Pi') \leq ht(\Pi)$ .*

**Lemma 7.** *Permutations. Let  $\Pi$  be a proof of  $\Sigma; B_1, \dots, B_n \vdash B_0$ . Then there exists a proof  $\Pi'$  of  $\Sigma; \pi_1.B_1, \dots, \pi_n.B_n \vdash \pi_0.B_0$  such that  $ht(\Pi') \leq ht(\Pi)$ .*

**Lemma 8.** *Restricted name substitutions. Let  $\Pi$  be a proof of*

$$\Sigma, x : \iota; B_1, \dots, B_n \vdash B_0.$$

*Then there exists a proof  $\Pi'$  of  $\Sigma; B_1[a_1/x], \dots, B_n[a_n/x] \vdash B_0[a_0/x]$ , where  $a_i \notin \text{supp}(B_i)$  for each  $i \in \{0, \dots, n\}$ , such that  $ht(\Pi') \leq ht(\Pi)$ .*

The next two lemmas are crucial to the cut-elimination proof: they allow one to reintroduce the symmetry between  $\forall\mathcal{L}$  and  $\forall\mathcal{R}$ , and dually, between  $\exists\mathcal{L}$  and  $\exists\mathcal{R}$  rules.

**Lemma 9.** *Support extension. Let  $\Pi$  be a proof of  $\Sigma, h; \Gamma \vdash B[h \bar{a}/x]$  where  $\{\bar{a}\} = \text{supp}(B)$ ,  $h \notin \Sigma$  and  $h$  is not free in  $\Gamma$  and  $B$ . Let  $\bar{c}$  be a list of nominal constants not in the support of  $B$ . Then there exists a proof  $\Pi'$  of  $\Sigma, h'; \Gamma \vdash B[h' \bar{a}\bar{c}/x]$  where  $h' \notin \Sigma$ .*

**Lemma 10.** *Support extension. Let  $\Pi$  be a proof of  $\Sigma, h; B[h \bar{a}/x], \Gamma \vdash C$  where  $\{\bar{a}\} = \text{supp}(B)$ ,  $h \notin \Sigma$  and  $h$  is not free in  $\Gamma$ ,  $B$  and  $C$ . Let  $\bar{c}$  be a list of nominal constants not in the support of  $B$ . Then there exists a proof  $\Pi'$  of  $\Sigma, h'; B[h' \bar{a}\bar{c}/x], \Gamma \vdash C$  where  $h' \notin \Sigma$ .*

The main result on the meta theory of  $LG^\omega$  is the cut-elimination theorem, from which the consistency of the logic follows.

**Theorem 11.** *The cut rule is admissible in  $LG^\omega$ .*

**Corollary 12.** *The logic  $LG^\omega$  is consistent, i.e., it is not the case that both  $A$  and  $A \supset \perp$  are provable.*

Finally, we show that the formulation of  $LG$  is equivalent to  $FO\lambda^\nabla$  extended with the axiom schemes of name permutations and weakening.

**Theorem 13.** *Let  $F$  be a formula which contains no occurrences of nominal constants. Then  $F$  is provable in  $FO\lambda^\nabla$  extended with the axiom schemes  $B \equiv \nabla x.B$  and  $\nabla x \nabla y.Bxy \supset \nabla y \nabla x.Bxy$  if and only if  $F$  is provable in  $LG$ .*

## 4 Encoding an object logic

We now consider an encoding of the logic  $HH$  mentioned in the introduction in  $LG^\omega$ . The encoding of this object logic has been done in  $FO\lambda^{\Delta N}$  by McDowell and Miller [11]. The formalization of the object logic properties in this section follows closely the  $FO\lambda^{\Delta N}$  encoding. The only major difference is that we do not need an explicit encoding of eigenvariables; eigenvariables are mapped to nominal constants in the meta logic  $LG^\omega$ .

The object logic formulas are generated by the following grammar.

$$\begin{aligned} D &::= A \mid G \Rightarrow A \mid \bigwedge_\tau x.D \\ G &::= A \mid tt \mid G \& G \mid A \Rightarrow G \mid \bigwedge_l x.G \mid \bigvee_\tau .G \end{aligned}$$

where  $A$  ranges over atomic (object-level) formula,  $\Rightarrow$ ,  $\&$ ,  $\bigwedge$  and  $\bigvee$  denote implication, conjunction, universal quantifier and existential quantifier, respectively.  $D$  and  $G$  represent definite clauses and goal formulas, respectively. Notice that in goal formulas, universal quantification is restricted to nominal types. The sequent rules for  $HH$  are the standard right introduction rules for the logical connectives plus the *backchaining* rule:

$$\frac{\Gamma, \bigwedge \vec{x}.G \supset A \longrightarrow G\theta}{\Gamma, \bigwedge \vec{x}.G \supset A \longrightarrow A'} \text{ bc, } A\theta = A'$$

This sequent system is complete for the  $HH$  fragment of intuitionistic, as a consequence of *uniform provability* of intuitionistic logic [15].

In order to encode the object-logic formulas into  $LG^\omega$ , we first introduce some types and constants. The object logic formulas are given the type  $prp$ , while atomic formulas are given the type  $atm$ . The formulas of  $HH$  are encoded using the following constants:

$$\begin{aligned} \langle \rangle &: atm \rightarrow prp & tt &: prp & \& &: prp \rightarrow prp \rightarrow prp & \Rightarrow &: atm \rightarrow prp \rightarrow prp \\ \bigwedge_\tau &: (\tau \rightarrow prp) \rightarrow prp & \bigvee_\tau &: (\tau \rightarrow prp) \rightarrow prp \end{aligned}$$

We denote the encoding of an object level formula  $A$  in  $LG^\omega$  with  $\llbracket A \rrbracket$ .

Since the set of definite clauses in the sequents does not change in the proofs in  $HH$ , we will not put them explicitly in the  $HH$  sequents in their encoding in  $LG^\omega$ . Hence hypotheses of  $HH$  sequents are lists of atomic formulas. The object logic sequent is represented using the predicate

$$seq : nt \rightarrow atm\ list \rightarrow prp \rightarrow o$$

where  $atm\ list$  is the type for lists of atomic formulas, with the usual constructors  $nil$  and  $::$ . The natural number in the encoding of sequents will be used as a measure of the length of object logic proofs. Inductive properties about the provability in  $HH$  will be proved using this measure. An object sequent  $\Gamma \longrightarrow A$  is represented as the atomic formula  $(seq_I \llbracket \Gamma \rrbracket \llbracket A \rrbracket)$  in  $LG^\omega$ . We encode definite clauses using a predicate called  $prog : atm \rightarrow prp \rightarrow o$ . A definite clause  $\bigwedge \vec{x}.G \rightarrow A$  is encoded as the definition clause  $\forall \vec{x}.prog A G \triangleq \top$ . The patterned definition of the sequent rules of  $HH$  is given in Figure 2. It uses the following definition clauses.

$$\begin{aligned} list_i nil &\triangleq \top. & list_{(s\ i)} (A :: L) &\triangleq list_i L. \\ list L &\triangleq \exists i.nat\ i \wedge list_i L. \\ elem A (A :: L) &\triangleq \top. & elem A (B :: L) &\triangleq elem A L. \end{aligned}$$

We refer to this definition together with the definition in Figure 2 as  $\mathcal{D}(HH)$  and any additional definite clauses with  $\mathcal{D}(prog)$ .

$$\begin{aligned}
seq_I L tt & \triangleq \top. \\
seq_I L \langle A \rangle & \triangleq elem A L. \\
seq_{(sI)} L (A \& B) & \triangleq seq_I L A \wedge seq_I L B. \\
seq_{(sI)} L (A \Rightarrow B) & \triangleq seq_I (A :: L) B. \\
seq_{(sI)} L (\bigwedge x.Gx) & \triangleq \nabla x.seq_I L Gx. \\
seq_{(sI)} L (\bigvee x.Gx) & \triangleq \exists x.seq_I L Gx. \\
seq_{(sI)} L \langle A \rangle & \triangleq \exists B.prog A B \wedge seq_I L B.
\end{aligned}$$

**Fig. 2.** Definition of an object logic.

*Example:* The formula  $p X \Rightarrow \bigwedge y.p y$  is not provable in the empty theory, whatever the value of  $X$  is. This fact is formalized in  $LG^\omega$  as the formula  $\forall X \forall I. (seq_I nil (p X \Rightarrow \bigwedge y.\langle p y \rangle) \supset \perp)$ . A partial derivation of this formula in  $LG^\omega$  is shown in Figure 3. In the figure the notation  $[p X]$  stands for the list  $(p X :: nil)$ . The derivation is completed by applying  $def\mathcal{L}$  to the topmost sequent, resulting in two matching cases: the identity rule and the backchaining rule. Since we assume no definite clauses, this leaves us with proving the sequent:

$$X, I_2; elem (p X) (p a :: nil) \vdash \perp$$

Applying  $def\mathcal{L}$  to this sequent results in the sequent  $X, I_2; elem (p X) nil \vdash \perp$ , since  $\lambda a.p X$  and  $\lambda a.p a$  are not unifiable. Another application of  $def\mathcal{L}$  gives us empty premise and hence the sequent is provable.  $\square$

$$\frac{\frac{\frac{X, I_2; seq_{I_2} [p X] \langle p a \rangle \vdash \perp}{X, I_2; \nabla y.seq_{I_2} [p X] \langle p y \rangle \vdash \perp} \nabla\mathcal{L}}{X, I_1; seq_{I_1} [p X] (\bigwedge y.\langle p y \rangle) \vdash \perp} def\mathcal{L}}{X, I; seq_I nil (p X \Rightarrow \bigwedge y.\langle p y \rangle) \vdash \perp} def\mathcal{L}}{\vdash \forall X \forall I. (seq_I nil (p X \Rightarrow \bigwedge y.\langle p y \rangle) \supset \perp)} \forall\mathcal{R}; \supset\mathcal{R}$$

**Fig. 3.** A derivation in  $LG^\omega$ .

It is straightforward to see that the structure of the  $HH$  proofs corresponds to the structure of proofs of its encoding in  $LG^\omega$ ; in particular, the backchaining rule in  $HH$  corresponds to the  $def\mathcal{R}$  rule (for the patterned definition) in  $LG^\omega$ . We now state some properties of the encoding of  $HH$  in  $LG^\omega$ .

**Theorem 14.** *Let  $\mathcal{D}(prog)$  be a definition corresponding to a set of definite clauses  $\mathcal{P}$ . Then the sequent  $\mathcal{P}, \Gamma \longrightarrow G$  is derivable in  $HH$  if and only if  $seq_i \llbracket \Gamma \rrbracket \llbracket G \rrbracket$  is derivable in  $LG^\omega$  with the definition  $\mathcal{D}(prog) \cup \mathcal{D}(HH)$  for some natural number  $i$ .*

*Proof.* The proof is by induction on the height of derivations of  $HH$ . The forward direction follows from the definition  $\mathcal{D}(HH)$ . For the other direction, we need only show that only the right-introduction rules are used in deriving the sequent in  $LG^\omega$ . But this follows from the cut-elimination theorem of  $LG^\omega$ .  $\square$

**Theorem 15.** *The following formulas are provable in  $LG^\omega$  with the definition of the object logic  $HH$ :*

1. *Structural rules:*  $\forall L \forall L' \forall G \forall L \forall i. nat i \supset list L \supset list L'$   
 $(\forall A. elem A L \supset elem A L') \supset seq_i L G \supset seq_i L' G.$
2. *Atomic cut:*  $\forall L \forall G \forall A. list L \supset \exists i. (nat i \wedge seq_i L (A \Rightarrow G)) \supset$   
 $\exists i. (nat i \wedge seq_i L \langle A \rangle) \supset \exists i. nat i \wedge seq_i L G.$
3. *Specialization:*  $\forall L \forall G \forall i. nat i \supset list L \supset seq_{(s_i)} L (\bigwedge G) \supset \forall x. seq_i L (G x).$

We conclude this section by a remark that  $\nabla$  is strictly speaking not necessary for capturing object logic provability, as Theorem 15 (3) shows, rather it is the use of nominal constants to model eigenvariables that allows that. The use of  $\nabla$ , however, results in a more natural correspondence between the encoding of  $HH$  and its actual sequent proofs.

## 5 Reasoning about operational semantics

Following McDowell and Miller [11], we use the encoding of  $HH$  in  $LG^\omega$  to specify and reason about the operational semantics of simply typed  $\lambda$ -calculus. Reasoning about more complicated languages like PCF can be done as well using a similar approach (see [11]).

We introduce a type  $ty$  to denote object-level types. The type  $tm$  denotes the object-level  $\lambda$ -terms and is considered a nominal type. The language of the (object-level)  $\lambda$ -terms is encoded using the following constants:

$$app : tm \rightarrow tm \rightarrow tm \quad abs : ty \rightarrow (tm \rightarrow tm) \rightarrow tm$$

which denote application and abstraction, respectively. The object-level type constructor, i.e., the ‘arrow’, is encoded via the constant  $arr : ty \rightarrow ty \rightarrow ty$ . Object-level base types are ranged over by  $\alpha$ .

The evaluation relation and the typing judgments of the simply typed calculus are given as definite clauses below.

$$\begin{aligned} eval (abs T M) (abs T M) &\Leftarrow tt. \\ eval (app M N) V &\Leftarrow \bigvee P. eval M P \ \& \ eval (P N) V. \\ typeof (abs T M) (ar T T') &\Leftarrow \bigwedge x. typeof x T \Rightarrow typeof (Mx) T'. \\ typeof (app M N) T &\Leftarrow \bigvee T'. typeof M (ar T' T) \ \& \ typeof N T'. \end{aligned}$$

It is straightforward to translate these clauses to *prog* clauses.

We state a couple of properties here as formulas in  $LG^\omega$ . In the following theorems, we use the notation  $L \triangleright G$  to denote the formula  $\exists i. nat \ i \wedge seq_i \ L \ G$ . If  $L$  is *nil* we simply write  $\triangleright G$ .

**Theorem 16.** Subject reduction. *The following formula is provable*

$$\forall M \forall V \forall T. \triangleright \langle eval \ M \ V \rangle \wedge \triangleright \langle typeof \ M \ T \rangle \supset \triangleright \langle typeof \ V \ T \rangle.$$

A proof of a similar theorem is given in [11] for the untyped  $\lambda$ -term in the logic  $FO\lambda^{\Delta\mathbb{N}}$ . This proof can be adapted straightforwardly to give a proof for the above theorem. A more interesting property is the determinacy of type assignments, provided that the typing context is well-formed, that is, each variable in the context is assigned a unique type. The well-formedness of a typing context  $L$  is specified as the formula:

$$\begin{aligned} &(\forall X \forall T_1 \forall T_2. elem (typeof \ X \ T_1) \ L \supset elem (typeof \ X \ T_2) \ L \supset T_1 = T_2) \wedge \\ &(\forall X \forall T. elem (typeof \ X \ T) \ L \supset \\ &((\exists M \exists N. T = (app \ M \ N)) \supset \perp) \wedge ((\exists M. T = (abs \ M)) \supset \perp)). \end{aligned}$$

The above formula will be denoted by  $ctx \ L$ .

**Theorem 17.** *The following formula is provable:*

$$\forall L \forall X \forall T_1 \forall T_2. list \ L \supset ctx \ L \supset L \triangleright \langle typeof \ X \ T_1 \rangle \supset L \triangleright \langle typeof \ X \ T_2 \rangle \supset T_1 = T_2.$$

## 6 Related and future work

There have been many previous related works in providing frameworks for higher-order abstract syntax, or more generally abstract syntax with bindings. A non-exhaustive list includes encodings in proof assistants like Coq [4], HOL [24], Isabelle [25], and Twelf [22], categorical frameworks [8], *the theory of context* [9],

nominal logic [20], and proof search frameworks [11, 23]. The approach taken here is similar to the latter; the novelty of our work lies in the use of equivariance principle within the usual style of higher-order abstract syntax specifications. Our aim is not to provide a mathematical foundation for higher-order abstract syntax, as it is in nominal logic, rather we aim at providing a convenient abstraction from such a foundation for those familiar with higher-order syntax specifications.

An immediate future work will be to implement the logic  $LG^\omega$ , possibly on top of an existing proof assistant, and to perform large case studies, in particular, the problem sets put out in the POPLMark Challenge [1].

*Semantics of  $LG$ .* There have been a couple of attempts at giving a semantics for the logic  $FO\lambda^\nabla$ : Cheney and Gabbay proposed an encoding into nominal logic [5, 2], and Miculan and Yemane give a categorical semantics [12]. In both works, it is suggested that extending  $FO\lambda^\nabla$  with the axiom schemes (1) would result in a natural semantics for  $\nabla$ . The work by Miculan and Yemane seems closer to the logic  $LG$  and could very well serve as a basis for finding a categorical model for  $LG$ . There are some similarities between  $LG$  and Nominal Logic, but the treatment of substitutions and the addition of closed terms of type name in  $LG$  make it unobvious whether the support models of Nominal Logic can be used for  $LG$ . We leave the investigation of support models for  $LG$  (or a classical version of  $LG$ ), such as the ones in [20, 3], as a future work.

*Acknowledgement.* The author would like to thank James Cheney for his many helpful remarks and suggestions, in particular those related to Nominal Logic. The author would also like thank David Baelde, Andrew Gacek, Alberto Momigliano, Michael Norrish and the anonymous referees of an earlier draft of the paper for their comments.

## References

1. B. E. Aydemir, A. Bohannon, M. Fairbairn, J. N. Foster, B. C. Pierce, P. Sewell, D. Vytiniotis, G. Washburn, S. Weirich, and S. Zdancewic. Mechanized metatheory for the masses: the POPLMARK challenge. In J. Hurd and T. Melham, editors, *Theorem Proving in Higher Order Logics, 18th International Conference*, Lecture Notes in Computer Science, pages 50–65. Springer, 2005.
2. J. Cheney. A simpler proof theory for nominal logic. In *Proc. FOSSACS’05*, volume 3441 of *Lecture Notes in Computer Science*. Springer, 2005.
3. J. Cheney. Completeness and Herbrand theorems for nominal logic. *Journal of Symbolic Logic*, 7(1):299–320, 2006.
4. J. Despeyroux, A. Felty, and A. Hirschowitz. Higher-order abstract syntax in Coq. In *Second International Conference on Typed Lambda Calculi and Applications*, pages 124–138, April 1995.
5. M. J. Gabbay and J. Cheney. A sequent calculus for nominal logic. In *Proc. 19th IEEE Symposium on Logic in Computer Science (LICS 2004)*, pages 139–148, 2004.
6. J.-Y. Girard. A fixpoint theorem in linear logic. Email to the linear@cs.stanford.edu mailing list, February 1992.
7. L. Hallnäs and P. Schroeder-Heister. A proof-theoretic approach to logic programming. II. Programs as definitions. *Journal of Logic and Computation*, 1(5):635–660, October 1991.
8. M. Hofmann. Semantical analysis of higher-order abstract syntax. In *14th Annual Symposium on Logic in Computer Science*, pages 204–213. IEEE Computer Society Press, 1999.
9. F. Honsell, M. Miculan, and I. Scagnetto. An axiomatic approach to metareasoning on systems in higher-order abstract syntax. In *Proc. ICALP’01*, number 2076 in LNCS, pages 963–978. Springer-Verlag, 2001.
10. R. McDowell and D. Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232:91–119, 2000.
11. R. McDowell and D. Miller. Reasoning with higher-order abstract syntax in a logical framework. *ACM Transactions on Computational Logic*, 3(1):80–136, January 2002.
12. M. Miculan and K. Yemane. A unifying model of variables and names. In *Proc. FOSSACS’05*, volume 3441 of *Lecture Notes in Computer Science*, pages 170 – 186. Springer, 2005.
13. D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.
14. D. Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14(4):321–358, 1992.

15. D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.
16. D. Miller and C. Palamidessi. Foundational aspects of syntax. In P. Degano, R. Gorrieri, A. Marchetti-Spaccamela, and P. Wegner, editors, *ACM Computing Surveys Symposium on Theoretical Computer Science: A Perspective*, volume 31. ACM, September 1999.
17. D. Miller and A. Tiu. A proof theory for generic judgments. *ACM Trans. Comput. Logic*, 6(4):749–783, 2005.
18. T. Nipkow. Functional unification of higher-order patterns. In M. Vardi, editor, *Proc. 8th IEEE Symposium on Logic in Computer Science (LICS 1993)*, pages 64–74. IEEE, June 1993.
19. F. Pfenning and C. Elliott. Higher-order abstract syntax. In *Proceedings of the ACM-SIGPLAN Conference on Programming Language Design and Implementation*, pages 199–208. ACM Press, June 1988.
20. A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003.
21. P. Schroeder-Heister. Cut-elimination in logics with definitional reflection. In D. Pearce and H. Wansing, editors, *Nonclassical Logics and Information Processing*, volume 619 of *LNCS*, pages 146–171. Springer, 1992.
22. C. Schürmann. *Automating the Meta Theory of Deductive Systems*. PhD thesis, Carnegie Mellon University, October 2000.
23. A. Tiu. *A Logical Framework for Reasoning about Logical Specifications*. PhD thesis, Pennsylvania State University, May 2004.
24. C. Urban and M. Norrish. A formal treatment of the Barendregt Variable Convention in rule inductions. In *MERLIN '05: Proceedings of the 3rd ACM SIGPLAN workshop on Mechanized reasoning about languages with variable binding*, pages 25–32, New York, NY, USA, 2005. ACM Press.
25. C. Urban and C. Tasson. Nominal techniques in Isabelle/HOL. In R. Nieuwenhuis, editor, *Proceedings of the 20th International Conference on Automated Deduction (CADE)*, volume 3632 of *LNCS*, pages 38–53. Springer, 2005.

## Appendix A: The meta theory of $LG^\omega$

We use the following more general form of the cut rule in proving the cut elimination theorem:

$$\frac{\Sigma; \Delta_1 \vdash B_1 \quad \cdots \quad \Sigma; \Delta_n \vdash B_n \quad \Sigma; B_1, \dots, B_n, \Gamma \vdash C}{\Sigma; \Delta_1, \dots, \Delta_n, \Gamma \vdash C} mc$$

### A.1 Derivation transformations

We define some transformations of derivations: weakening of hypotheses, substitutions on derivations, permutations and restricted name substitutions. In the following definitions we omit the signatures in the sequents if it is clear from context which signatures we refer to. We denote with  $id$  the identity function on  $\mathcal{C}_N$ .

**Definition 18.** Weakening of hypotheses. *Let  $\Pi$  be a derivation of  $\Sigma; \Gamma \vdash C$ . Let  $\Delta$  be a multiset of formulas whose free variables are among  $\Sigma$ . We define the derivation  $w(\Delta, \Pi)$  of  $\Sigma; \Gamma, \Delta \vdash C$  as follows:*

1. If  $\Pi$  ends with  $eq\mathcal{L}$

$$\frac{\left\{ \frac{\Pi_\theta}{\Sigma\theta; \Gamma'\theta \vdash C\theta} \right\}_\theta}{\Sigma; s = t, \Gamma' \vdash C} eq\mathcal{L}$$

then  $w(\Delta, \Pi)$  is

$$\frac{\left\{ \frac{w(\Delta\theta, \Pi_\theta)}{\Sigma\theta; \Gamma'\theta, \Delta\theta \vdash C\theta} \right\}_\theta}{\Sigma; s = t, \Gamma', \Delta \vdash C} eq\mathcal{L}$$

2. If  $\Pi$  ends with  $nat\mathcal{L}$

$$\frac{\frac{\Pi_1}{\vdash D z} \quad D i \vdash D (s i) \quad \frac{\Pi_2}{D i \vdash D (s i)} \quad \frac{\Pi_3}{D I, \Gamma' \vdash C}}{nat I, \Gamma' \vdash C} nat\mathcal{L}$$

then  $w(\Delta, \Pi)$  is

$$\frac{\frac{\Pi_1}{\vdash D z} \quad D i \vdash D (s i) \quad \frac{\Pi_2}{D i \vdash D (s i)} \quad \frac{w(\Delta, \Pi_3)}{D I, \Gamma', \Delta \vdash C}}{nat I, \Gamma', \Delta \vdash C} nat\mathcal{L}$$

3. If  $\Pi$  ends with the  $mc$  rule

$$\frac{\frac{\Pi_1}{\Delta_1 \vdash B_1} \quad \cdots \quad \frac{\Pi_n}{\Delta_n \vdash B_n} \quad \frac{\Pi'}{B_1, \dots, B_n, \Gamma' \vdash C}}{\Delta_1, \dots, \Delta_n, \Gamma' \vdash C} mc$$

then  $w(\Delta, \Pi)$  is

$$\frac{\frac{\Pi_1}{\Delta_1 \vdash B_1} \quad \cdots \quad \frac{\Pi_n}{\Delta_n \vdash B_n} \quad \frac{w(\Delta, \Pi')}{B_1, \dots, B_n, \Gamma', \Delta \vdash C}}{\Delta_1, \dots, \Delta_n, \Gamma', \Delta \vdash C} mc$$

4. If  $\Pi$  ends with any other rule and has premise derivations  $\Pi_1, \dots, \Pi_n$  then  $w(\Delta, \Pi)$  ends with the same rule with premise derivations  $w(\Delta, \Pi_1), \dots, w(\Delta, \Pi_n)$ .

**Definition 19.** Substitutions on derivations. *If  $\Pi$  is a derivation of  $\Sigma; \Gamma \vdash C$  and  $\theta$  is a  $\Sigma$ -substitution, then we define the derivation  $\Pi\theta$  of  $\Sigma\theta; \Gamma\theta \vdash C\theta$  as follows:*

1. Suppose  $\Pi$  ends with  $\text{eq}\mathcal{L}$ :

$$\frac{\left\{ \frac{\Pi_\rho}{\Sigma\rho; \Gamma'\rho \vdash C\rho} \right\}_\rho}{\Sigma; s = t, \Gamma' \vdash C} \text{eq}\mathcal{L}$$

where each  $\rho$  is a unifier of  $\lambda\vec{c}.s$  and  $\lambda\vec{c}.t$ . Observe that if  $\rho'$  is a unifier of  $(\lambda\vec{c}.s)\theta$  and  $(\lambda\vec{c}.t)\theta$ , then  $\theta \circ \rho'$  is a unifier of  $\lambda\vec{c}.s$  and  $\lambda\vec{c}.t$ . Thus  $\Pi\theta$  is the derivation:

$$\frac{\left\{ \frac{\Pi_{\theta \circ \rho'}}{\Sigma\theta\rho'; \Delta\theta\rho \vdash C\theta\rho} \right\}_{\rho'}}{\Sigma; s\theta = t\theta, \Delta\theta \vdash C\theta} \text{eq}\mathcal{L}$$

2. Suppose  $\Pi$  ends with  $\forall\mathcal{R}$ :

$$\frac{\frac{\Pi_1}{\Sigma; \Gamma \vdash B[h\vec{c}/x]}{\Sigma; \Gamma \vdash \forall x.B} \forall\mathcal{R}}{\Sigma; \Gamma \vdash \forall x.B} \forall\mathcal{R},$$

where  $\{\vec{c}\} = \text{supp}(\forall x.B)$ . Let  $\{\vec{d}\}$  be the support of  $(\forall x.B)\theta$ , which might be smaller than  $\{\vec{c}\}$ . Let  $\rho$  be the substitution  $[\lambda\vec{c}.h'\vec{d}/h]$  where  $h'$  is a new variable not already in  $\Sigma$  and not among the free variables in  $\theta$ . We can assume without loss of generality that  $x$  is not free in  $\theta$ , hence  $((B[h\vec{c}/x])\rho)\theta = (B[h'\vec{d}/x])\theta = (B\theta)[h'\vec{d}/x]$ . Then  $\Pi\theta$  is

$$\frac{\frac{\Pi_1(\rho \circ \theta)}{\Sigma\theta, h'; \Gamma\theta \vdash (B\theta)[h'\vec{d}/x]}{\Sigma\theta; \Gamma\theta \vdash (\forall x.B)\theta} \forall\mathcal{R}}{\Sigma\theta; \Gamma\theta \vdash (\forall x.B)\theta} \forall\mathcal{R}$$

3. Suppose  $\Pi$  ends with  $\exists\mathcal{L}$ : this case is dual to the previous one.

4. If  $\Pi$  ends with any other rule and has premise derivations  $\Pi_1, \dots, \Pi_n$ , then  $\Pi\theta$  ends with the same rule and has premise derivations  $\Pi_1\theta, \dots, \Pi_n\theta$ .

**Definition 20.** Let  $\Pi$  be a proof of  $\Sigma; B_1, \dots, B_n \vdash B_0$  and let  $\vec{\pi} = \pi_0, \dots, \pi_n$  be a list of permutations. We define a derivation  $\langle \vec{\pi} \rangle. \Pi$  of  $\Sigma; \pi_1.B_1, \dots, \pi_n.B_n \vdash \pi_0.B_0$  as follows:

1. Suppose that  $\Pi$  ends with  $\text{id}_\pi$

$$\frac{\pi.B_j = \pi'.B_0}{\Sigma; B_1, \dots, B_n \vdash B_0} \text{id}_\pi.$$

Obverse that  $\pi.\pi_j^{-1}.\pi_j.B = \pi'.\pi_0^{-1}.\pi_0.B'$ . Hence  $\langle \vec{\pi} \rangle. \Pi$  ends with the same rule.

2. Suppose  $\Pi$  ends with  $\text{mc}$ :

$$\frac{\frac{\Pi_1}{\Delta_1 \vdash D_1} \dots \frac{\Pi_m}{\Delta_m \vdash D_m} \quad D_1, \dots, D_m, \Delta_{m+1} \vdash B_0}{B_1, \dots, B_n \vdash B_0} \text{mc}}{\Delta_1 \vdash D_1 \dots \Delta_m \vdash D_m \quad D_1, \dots, D_m, \Delta_{m+1} \vdash B_0} \text{mc}$$

where  $\Delta_1, \dots, \Delta_{m+1}$  are partitions of  $B_1, \dots, B_n$ . Suppose that for each  $i \in \{1, \dots, m+1\}$ ,  $\Delta_i = B_{i1}, \dots, B_{ik_i}$  for some index  $k_i$ . Let  $\vec{\pi}(i)$ , for  $i \in \{1, \dots, m\}$ , be the permutations  $\text{id}, \pi_{i1}, \dots, \pi_{ik_i}$ . Let  $\vec{\pi}(m+1)$  be the permutations

$$\pi_0, \underbrace{\text{id}, \dots, \text{id}}_m, \pi_{(m+1)1}, \dots, \pi_{(m+1)k_{m+1}}$$

We denote with  $\Delta'_i$  the list

$$\pi_{i1}.B_{ij}, \dots, \pi_{ik_i}.B_{ik_i}.$$

Then  $\langle \vec{\pi} \rangle. \Pi$  is the derivation

$$\frac{\frac{\langle \vec{\pi}(1) \rangle. \Pi_1}{\Delta'_1 \vdash D_1} \dots \frac{\langle \vec{\pi}(m) \rangle. \Pi_m}{\Delta'_m \vdash D_m} \quad D_1, \dots, D_m, \Delta'_{m+1} \vdash \pi_0.B_0}{\pi_1.B_1, \dots, \pi_n.B_n \vdash \pi_0.B_0} \text{mc}}{\pi_1.B_1, \dots, \pi_n.B_n \vdash \pi_0.B_0} \text{mc}$$

3. Suppose  $\Pi$  ends with  $\nabla\mathcal{R}$ :

$$\frac{\Pi_1}{\Sigma; B_1, \dots, B_n \vdash B[a/x]} \nabla\mathcal{R}$$

where  $a : \iota \notin \text{supp}(B)$ . Let  $d : \iota$  be a nominal constant such that  $d \notin \text{supp}(B)$  and  $\pi_0(d) = d$ . Such a constant exists since  $\text{supp}(B)$  is finite and  $\pi_0$  is a finite permutation. Thus  $\pi_0.(a d).B_0[a/x] = \pi_0.B_0[d/x]$ . Then  $\langle \vec{\pi} \rangle.\Pi$  is the derivation:

$$\frac{\langle \pi_0.(a d), \dots, \pi_n \rangle.\Pi_1}{\Sigma; \pi_1.B_1, \dots, \pi_n.B_n \vdash \pi_0.B[d/x]} \nabla\mathcal{R}$$

4. Suppose  $\Pi$  ends with  $\nabla\mathcal{L}$ : this case is analogous to previous one.

5. Suppose  $\Pi$  ends with  $c\mathcal{L}$ :

$$\frac{\Pi'}{B_1, \dots, B_j, B_j \dots, B_n \vdash B_0} c\mathcal{L}$$

then  $\langle \vec{\pi} \rangle.\Pi$  is

$$\frac{\langle \pi_1, \dots, \pi_j, \pi_j, \dots, \pi_n \rangle.\Pi'}{\pi_1.B_1, \dots, \pi_j.B_j, \pi_j.B_j \dots, \pi_n.B_n \vdash \pi_0.B_0} c\mathcal{L}$$

6. If  $\Pi$  ends with any other rule and has premise derivations  $\Pi_1, \dots, \Pi_m$ , then  $\langle \vec{\pi} \rangle.\Pi$  ends with the same rule and has premise derivations  $\langle \vec{\pi} \rangle.\Pi_1, \dots, \langle \vec{\pi} \rangle.\Pi_m$ .

**Definition 21.** Let  $\Pi$  be a proof of  $\Sigma, x : \iota; B_1, \dots, B_n \vdash B_0$  and let  $\vec{a} = a_0, \dots, a_n$  be a list of nominal constants such that  $a_i \notin \text{supp}(B_i)$ . We define a derivation  $r(x, \vec{a}, \Pi)$  of  $\Sigma; B_1[a_1/x], \dots, B_n[a_n/x] \vdash B_0[a_0/x]$ , as follows:

1. Suppose  $\Pi$  is

$$\frac{\pi.B_j = \pi'.B_0}{\Sigma, x; B_1, \dots, B_n \vdash B_0} id_\pi.$$

Let  $d : \iota$  be a nominal constant which is not in the support of  $B_j$  and  $B_0$ , and  $\pi(d) = d$  and  $\pi'(d) = d$ . Then  $r(x, \vec{a}, \Pi)$  is

$$\frac{\pi.(a_j d).B_1[a_1/x] = \pi'.(a_0 d).B_0[a_0/x]}{\Sigma; B_1[a_1/x], \dots, B_n[a_n/x] \vdash B_0[a_0/x]} id_\pi$$

2. Suppose  $\Pi$  ends with  $mc$ :

$$\frac{\frac{\Pi_1}{\Sigma, x; \Delta_1 \vdash D_1} \dots \frac{\Pi_m}{\Sigma, x; \Delta_m \vdash D_m} \frac{\Pi'}{\Sigma, x; D_1, \dots, D_m, \Delta_{m+1} \vdash B_0}}{\Sigma, x; B_1, \dots, B_n \vdash B_0} mc$$

where  $\Delta_1, \dots, \Delta_{m+1}$  is a partition of  $B_1, \dots, B_n$ . Suppose that for each  $i \in \{1, \dots, m+1\}$ ,  $\Delta_i = B_{i1}, \dots, B_{ik_i}$  for some index  $k_i$ . Let  $\vec{d} = d_1, \dots, d_m$  be a list of nominal constants such that  $d_i \notin \text{supp}(D_i)$ . Let  $f(i)$ , for  $i \in \{1, \dots, m\}$  be the list  $d_i, a_{i1}, \dots, a_{ik_i}$  and let  $f(m+1)$  be the list

$$a_0, \vec{d}, a_{(m+1)1}, \dots, a_{(m+1)k_{(m+1)}}.$$

Let  $\Delta'_i$  be the list

$$B_{i1}[a_{i1}/x], \dots, B_{ik_i}[a_{ik_i}/x]$$

and let  $\Gamma$  be the list

$$D_1[d_1/x], \dots, D_m[d_m/x], \Delta'_{m+1}.$$

Then  $r(x, \vec{a}, \Pi)$  is the derivation

$$\frac{r(x, f(1), \Pi_1) \quad \dots \quad r(x, f(m), \Pi_m) \quad r(x, f(m+1), \Pi')}{\Sigma; \Delta'_1 \vdash D_1[d_1/x] \quad \dots \quad \Sigma; \Delta'_m \vdash D_m[a_m/x] \quad \Sigma; \Gamma \vdash B_0[a_0/x]} mc$$

3. Suppose  $\Pi$  is

$$\frac{\Pi_1}{\Sigma, x; B_1, \dots, B_n \vdash B[c/y]} \nabla \mathcal{R} .$$

If  $a_0 \neq c$  then  $r(x, \vec{a}, \Pi)$  is

$$\frac{r(x, \vec{a}, \Pi_1)}{\Sigma, x; B_1, \dots, B_n \vdash B[c/y]} \nabla \mathcal{R} .$$

If  $a_0 = c$ , then we swap  $c$  with a fresh constant. Let  $d : \iota$  be a nominal constant not in the support of  $B[c/y]$ . We apply the swapping ( $c$   $d$ ) to the conclusion of the end sequent of  $\Pi_1$  according to the construction in Definition 20 to get a proof  $\Pi_2$  of  $\Sigma, x; B_1, \dots, B_n \vdash B_0[d/y]$ . The derivation  $r(x, \vec{a}, \Pi)$  is constructed as follows:

$$\frac{r(x, \vec{a}, \Pi_2)}{\Sigma; B_1[a_1/x], \dots, B_n[a_n/x] \vdash B[a_0/x, d/y]} \nabla \mathcal{R}$$

4. If  $\Pi$  ends with  $\nabla \mathcal{L}$  apply the same construction as in the previous case.

5. Suppose  $\Pi$  ends with  $\forall \mathcal{R}$

$$\frac{\Pi_1}{\Sigma, x, h; B_1, \dots, B_n \vdash B[h \vec{c}/y]} \forall \mathcal{R} .$$

Let  $\theta = [\lambda \vec{c}. h' \vec{c}x/h]$  where  $h'$  is a variable not in  $\Sigma$ . Apply the construction in Definition 19 to get the proof  $\Pi\theta$  of

$$\Sigma, x, h'; B_1, \dots, B_n \vdash B[h' \vec{a}x/y]$$

Then  $r(x, \vec{a}, \Pi)$  is

$$\frac{r(x, \vec{a}, \Pi\theta)}{\Sigma, h'; B_1[a_1/x], \dots, B_n[a_n/x] \vdash B[a_0/x, (h' \vec{c}a_0)/y]} \forall \mathcal{R} .$$

6. If  $\Pi$  ends with  $\exists \mathcal{L}$ , apply the same construction as in the previous case.

7. Suppose  $\Pi$  ends with  $\exists \mathcal{R}$ :

$$\frac{\Pi_1}{\Sigma, x; B_1, \dots, B_n \vdash B[t/y]} \exists \mathcal{R} .$$

If  $a_0 \notin \text{supp}(B[t/y])$  then  $r(x, \vec{a}, \Pi)$  is

$$\frac{r(x, \vec{a}, \Pi_1)}{\Sigma; B_1[a_1/x], \dots, B_n[a_n/x] \vdash B[a_0/x, t/y]} \exists \mathcal{R} .$$

If  $a_0 \in \text{supp}(B[t/y])$ , we exchange it with a fresh constant. Let  $d$  be a nominal constant distinct from  $a_0$  and not in the support of  $B[t/y]$ . Then  $((a_0 d).B[t/y])[a_0/x] = B[(a_0 d).t/y, a_0/x]$ . We first apply the

construction in Definition 20 to  $\Pi_1$  to get a derivation  $\Pi_2$  of  $\Sigma, x; B_1, \dots, B_n \vdash B[(a_0 d).t/y, a_0/x]$ . The derivation  $r(x, \vec{a}, \Pi)$  is thus

$$\frac{r(x, \vec{a}, \Pi_2)}{\Sigma; B_1[a_1/x], \dots, B_n[a_n/x] \vdash B[(a_0 d).t/y, a_0/x]} \exists \mathcal{R} . \frac{}{\Sigma; B_1[a_1/x], \dots, B_n[a_n/x] \vdash \exists y. B[a_0/x]}$$

8. Suppose  $\Pi$  ends with  $\text{eq}\mathcal{L}$ :

$$\frac{\left\{ \frac{\Pi_\theta}{(\Sigma, x)\theta; B_2\theta, \dots, B_n\theta \vdash B_0\theta} \right\}_\theta}{\Sigma, x; s = t, B_2, \dots, B_n \vdash B_0} \text{eq}\mathcal{L}$$

where each  $\theta$  is a unifier of  $(\lambda\vec{c}.s, \lambda\vec{c}.t)$  and  $\{\vec{c}\} = \text{supp}(s = t)$ . We need to show that for each unifier of  $(\lambda a_1 \lambda\vec{c}.s[a_1/x], \lambda a_1 \lambda\vec{c}.t[a_1/x])$  there is a corresponding unifier for  $\lambda\vec{c}.s$  and  $\lambda\vec{c}.t$ . We can assume without loss of generality that  $x$  is not in the domain of  $\rho$ .

We first show the case where  $x$  is not free in  $\rho$ . It is clear that in this case  $\rho$  is a unifier of  $\lambda\vec{c}.s$  and  $\lambda\vec{c}.t$ . Therefore we apply the procedure recursively to the premise derivation  $\Pi_\rho$ , to get the derivation  $r(x, \vec{a}, \Pi_\rho)$  of

$$\Sigma\rho; (B_2[a_2/x])\rho, \dots, (B_n[a_n/x])\rho \vdash (B_0[a_0/x])\rho.$$

In the other case, where  $x$  is free in the range of  $\rho$ , we show that it can be reduced to the previous case. First we define a substitution  $\rho'$  to be the substitution  $\rho$  where  $x$  is replaced by a new variable  $u$  which is not free in  $\rho$ . Clearly  $\rho'$  is also a unifier of  $\lambda a_1 \lambda\vec{c}.s[a_1/x]$  and  $\lambda a_1 \lambda\vec{c}.t[a_1/x]$ . Moreover, it is more general than  $\rho$ , since  $\rho = [x/u] \circ \rho'$ . Therefore we can apply the construction in the previous case to get a derivation  $r(x, \vec{a}, \Pi_{\rho'})$  and apply the substitution  $[x/u]$  to this derivation, using the procedure in Definition 19, to get a derivation of

$$\Sigma\rho; (B_2[a_2/x])\rho, \dots, (B_n[a_n/x])\rho \vdash (B_0[a_0/x])\rho.$$

The derivation  $r(x, \vec{a}, \Pi)$  is then constructed as follows

$$\frac{\left\{ \frac{\Pi'_\rho}{\Sigma\rho; (B_2[a_2/x])\rho, \dots, (B_n[a_n/x])\rho \vdash (B_0[a_0/x])\rho} \right\}_\rho}{\Sigma; s[a_1/x] = t[a_1/x], \dots, B_n[a_n/x] \vdash B_0[a_0/x]} \text{eq}\mathcal{L}$$

where each  $\Pi'_\rho$  is constructed as explained above.

9. If  $\Pi$  ends with  $\text{c}\mathcal{L}$ :

$$\frac{\Pi'}{B_1, \dots, B_j, B_j, \dots, B_n \vdash B_0} \text{c}\mathcal{L}$$

then  $r(x, \vec{a}, \Pi)$  is

$$\frac{r(x, (a_0, \dots, a_j, a_j, \dots, a_n), \Pi')}{B_1[a_1/x], \dots, B_j[a_j/x], B_j[a_j/x], \dots, B_n[a_n/x] \vdash B_0[a_0/x]} \text{c}\mathcal{L}$$

10. If  $\Pi$  ends with any other rule and has premise derivations  $\Pi_1, \dots, \Pi_n$ , then  $r(x, \vec{a}, \Pi)$  ends with the same rule and has premise derivations  $r(x, \vec{a}, \Pi_1), \dots, r(x, \vec{a}, \Pi_n)$ .

**Lemma 22.** For any derivation  $\Pi$  of  $\Sigma; \Gamma \vdash C$  and any multiset of  $\Sigma$ -formulas  $\Delta$ ,  $w(\Delta, \Pi)$  is a derivation of  $\Sigma; \Gamma, \Delta \vdash C$  and  $\text{ht}(w(\Delta, \Pi)) \leq \text{ht}(\Pi)$ .

**Lemma 23.** For any derivation  $\Pi$  of  $\Sigma; \Gamma \vdash C$  and any  $\Sigma$ -substitution  $\theta$ ,  $\Pi\theta$  is a derivation of  $\Sigma\theta; \Gamma\theta \vdash C\theta$  and  $\text{ht}(\Pi\theta) \leq \text{ht}(\Pi)$ .

**Lemma 24.** For any derivation  $\Pi$  of  $B_1, \dots, B_n \vdash B_0$  and permutations  $\vec{\pi} = \pi_0, \dots, \pi_n, \langle \vec{\pi} \rangle. \Pi$  is a derivation of  $\pi_1.B_1, \dots, \pi_n.B_n \vdash \pi_0.B_0$  and  $ht(\langle \vec{\pi} \rangle. \Pi) \leq ht(\Pi)$ .

**Lemma 25.** For any derivation  $\Pi$  of  $\Sigma, x; B_1, \dots, B_n \vdash B_0$  and any list of nominal constants  $\vec{a} = a_0, \dots, a_n$  such that  $a_i \notin \text{supp}(B_i)$ ,  $r(x, \vec{a}, \Pi)$  is a derivation of  $\Sigma; B_1[a_1/x], \dots, B_n[a_n/x] \vdash B_0[a_0/x]$  and  $ht(r(x, \vec{a}, \Pi)) \leq ht(\Pi)$ .

**Proof of Lemma 6** Follows immediately from Lemma 23. □

**Proof of Lemma 7** Follows immediately from Lemma 24. □

**Proof of Lemma 8** Follows immediately from Lemma 25. □

**Proof of Lemma 9** Suppose  $\vec{c}$  is the list of constants  $c_1 : \iota_1, \dots, c_n : \iota_n$ . Let  $\vec{y} = y_1 : \iota_1, \dots, y_n : \iota_n$  be a list of distinct variables not appearing in  $\Sigma \cup \{h, h'\}$ . We first apply the substitution  $[\lambda \vec{a}. h' \vec{c} \vec{x} / h]$  to the sequent  $\Sigma, h; \Gamma \vdash B[h\vec{a}/x]$ . By Lemma 6, there is a proof  $\Pi_1$  of

$$\Sigma, h', \vec{y}; \Gamma \vdash B[h' \vec{a} \vec{y} / x]$$

The derivation  $\Pi'$  is then obtained by repeatedly applying Lemma 8 to  $\Pi_1$  to change  $\vec{y}$  into  $\vec{c}$ . □

**Proof of Lemma 10** Use the same construction as in the proof of Lemma 9. □

## A.2 Cut reduction

We define a *reduction* relation between derivations, following closely the reduction relation in [10]. For simplicity of presentation, we shall omit the signatures in the sequents in the following reduction of cuts when the signatures are not changed by the reduction or when it is clear from context which signatures should be assigned to the sequents. The redex is always a derivation  $\Xi$  ending with the multicut rule

$$\frac{\frac{\Pi_1}{\Sigma; \Delta_1 \vdash B_1} \quad \dots \quad \frac{\Pi_n}{\Sigma; \Delta_n \vdash B_n} \quad \frac{\Pi}{\Sigma; B_1, \dots, B_n, \Gamma \vdash C} \text{ mc}}{\Sigma; \Delta_1, \dots, \Delta_n, \Gamma \vdash C} .$$

We refer to the formulas  $B_1, \dots, B_n$  produced by the *mc* as *cut formulas*.

If  $n = 0$ ,  $\Xi$  reduces to the premise derivation  $\Pi$ .

For  $n > 0$  we specify the reduction relation based on the last rule of the premise derivations. If the rightmost premise derivation  $\Pi$  ends with a left rule acting on a cut formula  $B_i$ , then the last rule of  $\Pi_i$  and the last rule of  $\Pi$  together determine the reduction rules that apply. We classify these rules according to the following criteria: we call the rule an *essential* case when  $\Pi_i$  ends with a right rule; if it ends with a left rule, it is a *left-commutative* case; if  $\Pi_i$  ends with the *id* rule, then we have an *axiom* case; a *multicut* case arises when it ends with the *mc* rule. When  $\Pi$  does not end with a left rule acting on a cut formula, then its last rule is alone sufficient to determine the reduction rules that apply. If  $\Pi$  ends in a rule acting on a formula other than a cut formula, then we call this a *right-commutative* case. A *structural* case results when  $\Pi$  ends with a contraction or weakening on a cut formula. If  $\Pi$  ends with the *id* rule, this is also an axiom case; similarly a multicut case arises if  $\Pi$  ends in the *mc* rule.

For simplicity of presentation, we always show  $i = 1$ .

Essential cases:

$\wedge \mathcal{R} / \wedge \mathcal{L}$ : If  $\Pi_1$  and  $\Pi$  are

$$\frac{\frac{\Pi'_1}{\Delta_1 \vdash B'_1} \quad \frac{\Pi''_1}{\Delta_1 \vdash B''_1}}{\Delta_1 \vdash B'_1 \wedge B''_1} \wedge \mathcal{R} \quad \frac{\frac{\Pi'}{B'_1, B_2, \dots, B_n, \Gamma \vdash C}}{B'_1 \wedge B''_1, B_2, \dots, B_n, \Gamma \vdash C} \wedge \mathcal{L} ,$$

then  $\Xi$  reduces to

$$\frac{\frac{\Pi'_1}{\Delta_1 \vdash B'_1} \quad \frac{\Pi_2}{\Delta_2 \vdash B_2} \quad \cdots \quad \frac{\Pi_n}{\Delta_n \vdash B_n} \quad \frac{\Pi'}{B'_1, B_2, \dots, B_n, \Gamma \vdash C}}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} mc .$$

The case for the other  $\wedge\mathcal{L}$  rule is symmetric.

$\vee\mathcal{R}/\vee\mathcal{L}$ : If  $\Pi_1$  and  $\Pi$  are

$$\frac{\frac{\Pi'_1}{\Delta_1 \vdash B'_1}}{\Delta_1 \vdash B'_1 \vee B''_1} \vee\mathcal{R} \quad \frac{\frac{\Pi'}{B'_1, B_2, \dots, B_n, \Gamma \vdash C} \quad \frac{\Pi''}{B''_1, B_2, \dots, B_n, \Gamma \vdash C}}{B'_1 \vee B''_1, B_2, \dots, B_n, \Gamma \vdash C} \vee\mathcal{L} ,$$

then  $\Xi$  reduces to

$$\frac{\frac{\Pi'_1}{\Delta_1 \vdash B'_1} \quad \frac{\Pi_2}{\Delta_2 \vdash B_2} \quad \cdots \quad \frac{\Pi_n}{\Delta_n \vdash B_n} \quad \frac{\Pi'}{B'_1, B_2, \dots, B_n, \Gamma \vdash C}}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} mc .$$

The case for the other  $\vee\mathcal{R}$  rule is symmetric.

$\supset\mathcal{R}/\supset\mathcal{L}$ : Suppose  $\Pi_1$  and  $\Pi$  are

$$\frac{\frac{\Pi'_1}{B'_1, \Delta_1 \vdash B''_1}}{\Delta_1 \vdash B'_1 \supset B''_1} \supset\mathcal{R} \quad \frac{\frac{\Pi'}{B_2, \dots, B_n, \Gamma \vdash B'_1} \quad \frac{\Pi''}{B''_1, B_2, \dots, B_n, \Gamma \vdash C}}{B'_1 \supset B''_1, B_2, \dots, B_n, \Gamma \vdash C} \supset\mathcal{L} .$$

Let  $\Xi_1$  be

$$\frac{\left\{ \frac{\Pi_i}{\Delta_i \vdash B_i} \right\}_{i \in \{2..n\}} \quad \frac{\frac{\Pi'}{B_2, \dots, B_n, \Gamma \vdash B'_1} mc}{\Delta_2, \dots, \Delta_n, \Gamma \vdash B'_1} \quad \frac{\Pi'_1}{B'_1, \Delta_1 \vdash B''_1} mc}{\Delta_1, \dots, \Delta_n, \Gamma \vdash B''_1} mc .$$

Then  $\Xi$  reduces to

$$\frac{\frac{\Xi_1}{\dots \vdash B''_1} \quad \left\{ \frac{\Pi_i}{\Delta_i \vdash B_i} \right\}_{i \in \{2..n\}} \quad \frac{\frac{\Pi''}{B''_1, \{B_i\}_{i \in \{2..n\}}, \Gamma \vdash C} mc}{\Delta_1, \dots, \Delta_n, \Gamma, \Delta_2, \dots, \Delta_n, \Gamma \vdash C} mc}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} c\mathcal{L} .$$

We use the double horizontal lines to indicate that the relevant inference rule (in this case,  $c\mathcal{L}$ ) may need to be applied zero or more times.

$\forall\mathcal{R}/\forall\mathcal{L}$ : Suppose  $\Pi_1$  and  $\Pi$  are

$$\frac{\frac{\Pi'_1}{\Sigma, h; \Delta_1 \vdash B'_1[(h\vec{c})/x]}}{\Sigma; \Delta_1 \vdash \forall x.B'_1} \forall\mathcal{R} \quad \frac{\frac{\Pi'}{\Sigma; B'_1[t/x], B_2, \dots, B_n, \Gamma \vdash C}}{\Sigma; \forall x.B'_1, B_2, \dots, B_n, \Gamma \vdash C} \forall\mathcal{L} ,$$

where  $\{\vec{c}\} = \text{supp}(B'_1)$ . Let  $\{\vec{d}\} = \text{supp}(B'_1[t/x]) \setminus \text{supp}(B'_1)$ . Apply Lemma 9 to get a derivation  $\Pi''_1$  of  $\Sigma, h'; \Delta_1 \vdash B'_1[(h\vec{c}\vec{d})/x]$ . The derivation  $\Xi$  reduces to

$$\frac{\frac{\Pi''[\lambda\vec{c}\vec{d}.t/h']}{\Sigma; \Delta_1 \vdash B'_1[t/x]} \quad \left\{ \Sigma; \Delta_i \vdash B_i \right\}_{i \in \{2..n\}} \quad \dots \vdash C}{\Sigma; \Delta_1, \dots, \Delta_n, \Gamma \vdash C} mc .$$

$\exists\mathcal{R}/\exists\mathcal{L}$ : Suppose  $\Pi_1$  and  $\Pi$  are

$$\frac{\frac{\Pi'_1}{\Sigma; \Delta_1 \vdash B'_1[t/x]} \exists\mathcal{R}}{\Sigma; \Delta_1 \vdash \exists x.B'_1} \quad \frac{\frac{\Pi'}{\Sigma, h; B'_1[(h\vec{c})/x], B_2, \dots, B_n, \Gamma \vdash C} \exists\mathcal{L}}{\Sigma; \exists x.B'_1, B_2, \dots, B_n, \Gamma \vdash C} \exists\mathcal{L} ,$$

where  $\{\vec{c}\} = \text{supp}(B'_1)$ . Let  $\{\vec{d}\} = \text{supp}(B'_1[t/x]) \setminus \text{supp}(B'_1)$ . Apply Lemma 10 to  $\Pi'$  to get a derivation  $\Pi''$  of  $\Sigma, h'; \Delta_1 \vdash B'_1[(h'\vec{c}\vec{d})/x]$ . Then  $\Xi$  reduces to

$$\frac{\frac{\Pi'_1}{\Sigma; \Delta_1 \vdash B'_1[t/x]} \quad \dots \quad \frac{\Pi''[\lambda\vec{c}\vec{d}.t/h']}{\Sigma; B'_1[t/x], B_2, \dots, \Gamma \vdash C} mc}{\Sigma; \Delta_1, \dots, \Delta_n, \Gamma \vdash C} mc .$$

$\nabla\mathcal{R}/\nabla\mathcal{L}$ : Suppose  $\Pi_1$  and  $\Pi$  are

$$\frac{\frac{\Pi'_1}{\Delta_1 \vdash B'_1[a/x]} \nabla\mathcal{R}}{\Delta_1 \vdash \nabla x.B'_1} \quad \frac{\frac{\Pi'}{B'_1[b/x], \dots, B_n, \Gamma \vdash C} \nabla\mathcal{L}}{\nabla x.B'_1, \dots, B_n, \Gamma \vdash C} \nabla\mathcal{L} .$$

Apply the construction in Definition 20 to  $\Pi'_1$  to swap  $a$  with  $b$  to get a derivation  $\Pi''_1$  of  $\Delta_1 \vdash B'_1[b/x]$ .  $\Xi$  reduces to

$$\frac{\frac{\Pi''_1}{\Delta_1 \vdash B'_1[b/x]} \quad \dots \quad \frac{\Pi'}{B'_1[b/x], \dots, B_n, \Gamma \vdash C} mc}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} mc .$$

$\text{nat}\mathcal{R}/\text{nat}\mathcal{L}$  : Suppose  $\Pi_1$  is  $\overline{\Delta_1 \vdash \text{nat } z} \text{ nat}\mathcal{R}$  and  $\Pi$  is

$$\frac{\frac{\Pi'}{\vdash D z} \quad \frac{\Pi''}{D j \vdash D (s j)} \quad \frac{\Pi'''}{D z, B_2, \dots, B_n, \Gamma \vdash C}}{\text{nat } z, B_2, \dots, B_n, \Gamma \vdash C} \text{nat}\mathcal{L} .$$

Then  $\Xi$  reduces to

$$\frac{w(\Delta_1, \Pi') \quad \left\{ \Delta_i \vdash B_i \right\}_{i \in \{2..n\}} \quad \frac{\Pi'''}{D z, B_2, \dots, B_n, \Gamma \vdash C} mc}{\Delta_1, \Delta_2, \dots, \Delta_n, \Gamma \vdash C} mc$$

$\text{nat}\mathcal{R}/\text{nat}\mathcal{L}$  : Suppose  $\Pi_1$  is

$$\frac{\frac{\Pi'_1}{\Delta \vdash \text{nat } I} \text{nat}\mathcal{R}}{\Delta_1 \vdash \text{nat } (s I)} \text{nat}\mathcal{R}$$

and  $\Pi$  is

$$\frac{\frac{\Pi'}{\vdash D z} \quad \frac{\Pi''}{D j \vdash D (s j)} \quad \frac{\Pi'''}{D (s I), B_2, \dots, B_n, \Gamma \vdash C}}{\text{nat } (s I), B_2, \dots, B_n, \Gamma \vdash C} \text{nat}\mathcal{L}$$

Let  $\Xi_1$  be

$$\frac{\frac{\Pi'_1 \quad \frac{\frac{\Pi' \quad \Pi''}{\vdash Dz \quad Dj \vdash D(sj)}{\text{nat } I \vdash DI} \text{mc.}}{\Delta_1 \vdash \text{nat } I}}{\Delta_1 \vdash DI} \text{mc.}}{\Delta_1 \vdash DI} \text{mc.}}{\Delta_1 \vdash DI} \text{mc.}} \frac{id_\pi}{\text{nat } \mathcal{L}}$$

Suppose  $\{\vec{c}\} = \text{supp}(I)$ . We apply the procedures in Definition 19 and Definition 21 to  $\Pi''$  to obtain the derivation  $\Pi^\bullet$  of

$$h; D(h\vec{c}) \vdash D(s(h\vec{c})).$$

Let  $\Xi_2$  be

$$\frac{\frac{\Xi_1 \quad \Pi^\bullet[\lambda\vec{c}.I/h]}{\Delta_1 \vdash DI \quad DI \vdash D(sI)} \text{mc.}}{\Delta_1 \vdash D(sI)} \text{mc.}$$

Then  $\Xi$  reduces to

$$\frac{\frac{\Xi_2 \quad \Pi_2 \quad \dots \quad \Pi_n \quad \Pi'''}{\Delta_1 \vdash D(sI) \quad \Delta_2 \vdash B_2 \quad \dots \quad \Delta_n \vdash B_n \quad D(sI), B_2, \dots, B_n, \Gamma \vdash C} \text{mc.}}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} \text{mc.}$$

eq $\mathcal{L}$ /eq $\mathcal{R}$ : If  $\Pi_1$  and  $\Pi$  are

$$\frac{}{\Sigma; \Delta_1 \vdash t = t} \text{eq}\mathcal{R} \quad \frac{\left\{ \frac{\Pi_\theta}{\Sigma\theta; \Gamma\theta \vdash C\theta} \right\}_\theta}{\Sigma; t = t, \Gamma \vdash C} \text{eq}\mathcal{L}$$

then  $\Xi$  reduces to

$$\frac{\frac{\Pi_2 \quad \dots \quad \Pi_n \quad w(\Delta_1, \Pi_\epsilon)}{\Sigma; \Delta_2 \vdash B_2 \quad \dots \quad \Sigma; \Delta_n \vdash B_n \quad \Sigma; \Delta_1, B_2, \dots, B_n, \Gamma \vdash C} \text{mc}}{\Sigma; \Delta_1, \dots, \Delta_n, \Gamma \vdash C} \text{mc}}$$

where  $\epsilon$  is the empty substitution.

def $\mathcal{R}$ /def $\mathcal{L}$ : Suppose  $\Pi_1$  and  $\Pi$  are

$$\frac{\frac{\Pi'_1}{\Delta_1 \vdash B[\vec{t}/\vec{x}]} \text{def}\mathcal{R}}{\Delta_1 \vdash p\vec{t}} \text{def}\mathcal{R} \quad \frac{\frac{\Pi'}{B[\vec{t}/\vec{x}], B_2, \dots, \Gamma \vdash C} \text{def}\mathcal{L}}{p\vec{t}, B_2, \dots, \Gamma \vdash C} \text{def}\mathcal{L} .$$

Then  $\Xi$  reduces to

$$\frac{\frac{\Pi'_1 \quad \Pi_2 \quad \dots \quad \Pi_n \quad \Pi'}{\Delta_1 \vdash B[\vec{t}/\vec{x}] \quad \Delta_2 \vdash B_2 \quad \dots \quad \Delta_n \vdash B_n \quad B[\vec{t}/\vec{x}], \dots, \Gamma \vdash C} \text{mc}}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} \text{mc} .$$

Left-commutative cases:

• $\mathcal{L}$ /• $\mathcal{L}$ : Suppose  $\Pi$  ends with a left rule other than  $c\mathcal{L}$  acting on  $B_1$  and  $\Pi_1$  is

$$\frac{\left\{ \frac{\Pi_1^i}{\Delta_1^i \vdash B_1} \right\}}{\Delta_1 \vdash B_1} \bullet\mathcal{L} ,$$

where  $\bullet\mathcal{L}$  is any left rule except  $\supset\mathcal{L}$ ,  $\text{eq}\mathcal{L}$ , or  $\text{nat}\mathcal{L}$ . Then  $\Xi$  reduces to

$$\left\{ \frac{\frac{\frac{\Pi_1^i}{\Delta_1^i \vdash B_1} \quad \left\{ \frac{\Pi_j}{\Delta_j \vdash B_j} \right\}_{j \in \{2..n\}} \quad B_1, \dots, B_n, \Gamma \vdash C}{\Delta_1^i, \Delta_2, \dots, \Delta_n, \Gamma \vdash C} \text{mc}}{\Delta_1, \Delta_2, \dots, \Delta_n, \Gamma \vdash C} \right\} \bullet\mathcal{L} .$$

$\supset\mathcal{L}/\circ\mathcal{L}$ : Suppose  $\Pi$  ends with a left rule other than  $c\mathcal{L}$  acting on  $B_1$  and  $\Pi_1$  is

$$\frac{\frac{\Pi_1'}{\Delta_1' \vdash D_1'} \quad \frac{\Pi_1''}{D_1'', \Delta_1' \vdash B_1}}{D_1' \supset D_1'', \Delta_1' \vdash B_1} \supset\mathcal{L} .$$

Let  $\Xi_1$  be

$$\frac{\frac{\frac{\Pi_1''}{D_1'', \Delta_1' \vdash B_1} \quad \frac{\Pi_2}{\Delta_2 \vdash B_2} \quad \dots \quad \frac{\Pi_n}{\Delta_n \vdash B_n} \quad B_1, \dots, B_n, \Gamma \vdash C}{D_1'', \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} \text{mc}}{\Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} .$$

Then  $\Xi$  reduces to

$$\frac{w(\Delta_2 \cup \dots \cup \Delta_n \cup \Gamma, \Pi_1') \quad \Xi_1}{\frac{\frac{\Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash D_1'}{\Delta_1' \supset D_1'', \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} \supset\mathcal{L}}{D_1' \supset D_1'', \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} .$$

$\text{nat}\mathcal{L}/\circ\mathcal{L}$ : Suppose  $\Pi$  ends with a left rule other than  $c\mathcal{L}$  acting on  $B_1$  and  $\Pi_1$  is

$$\frac{\frac{\frac{\Pi_1^1}{\vdash D_1 z} \quad \frac{\Pi_1^2}{D_1 j \vdash D_1(sj)} \quad \frac{\Pi_1^3}{D_1 I, \Delta_1' \vdash B_1}}{\text{nat } I, \Delta_1' \vdash B_1} \text{nat}\mathcal{L}}{\Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} .$$

Let  $\Xi_1$  be

$$\frac{\frac{\frac{\Pi_1^3}{D_1 I, \Delta_1' \vdash B_1} \quad \frac{\Pi_2}{\Delta_2 \vdash B_2} \quad \dots \quad \frac{\Pi_n}{\Delta_n \vdash B_n} \quad B_1, \dots, B_n \vdash C}{D_1 I, \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} \text{mc}}{\Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} .$$

Then  $\Xi$  reduces to

$$\frac{\frac{\frac{\Pi_1^1}{\vdash D_1 z} \quad \frac{\Pi_1^2}{D_1 j \vdash D_1(sj)} \quad \frac{\Xi_1}{D_1 I, \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C}}{\text{nat } I, \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} \text{nat}\mathcal{L}}{\Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} .$$

$\text{eq}\mathcal{L}/\circ\mathcal{L}$ : If  $\Pi$  ends with a left rule other than  $c\mathcal{L}$  acting on  $B_1$  and  $\Pi_1$  is

$$\frac{\left\{ \frac{\frac{\Pi^\theta}{\Delta_1' \theta \vdash B_1 \theta}}{s = t, \Delta_1' \vdash B_1} \right\}_\theta \text{eq}\mathcal{L}}{\Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} .$$

then  $\Xi$  reduces to

$$\left\{ \frac{\frac{\frac{\frac{\Pi^\theta}{\Delta_1' \theta \vdash B_1 \theta} \quad \frac{\Pi_2 \theta}{\Delta_2 \theta \vdash B_2 \theta} \quad \dots \quad \frac{\Pi_n \theta}{\Delta_n \theta \vdash B_n \theta} \quad B_1 \theta, \dots, B_n \theta, \Gamma \theta \vdash C \theta}{\Delta_1' \theta, \Delta_2 \theta, \dots, \Delta_n \theta, \Gamma \theta \vdash C \theta} \text{mc}}{s = t, \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \vdash C} \text{eq}\mathcal{L}^\theta \right\} .$$

Right-commutative cases:

$-/\circ\mathcal{L}$ : Suppose  $\Pi$  is

$$\frac{\left\{ B_1, \dots, B_n, \Gamma^i \vdash C \right\}}{B_1, \dots, B_n, \Gamma \vdash C} \circ\mathcal{L} ,$$

where  $\circ\mathcal{L}$  is any left rule other than  $\supset\mathcal{L}$ ,  $\text{eq}\mathcal{L}$ , or  $\text{nat}\mathcal{L}$  (but including  $\text{c}\mathcal{L}$ ) acting on a formula other than  $B_1, \dots, B_n$ . The derivation  $\Xi$  reduces to

$$\frac{\left\{ \frac{\frac{\Pi_1}{\Delta_1 \vdash B_1} \cdots \frac{\Pi_n}{\Delta_n \vdash B_n} \quad B_1, \dots, B_n, \Gamma^i \vdash C}{\Delta_1, \dots, \Delta_n, \Gamma^i \vdash C} \text{mc} \right\}}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} \circ\mathcal{L} ,$$

$-/\supset\mathcal{L}$ : Suppose  $\Pi$  is

$$\frac{B_1, \dots, B_n, \Gamma' \vdash D' \quad B_1, \dots, B_n, D'', \Gamma' \vdash C}{B_1, \dots, B_n, D' \supset D'', \Gamma' \vdash C} \supset\mathcal{L} .$$

Let  $\Xi_1$  be

$$\frac{\frac{\Pi_1}{\Delta_1 \vdash B_1} \cdots \frac{\Pi_n}{\Delta_n \vdash B_n} \quad B_1, \dots, B_n, \Gamma' \vdash D'}{\Delta_1, \dots, \Delta_n, \Gamma' \vdash D'} \text{mc}$$

and  $\Xi_2$  be

$$\frac{\frac{\Pi_1}{\Delta_1 \vdash B_1} \cdots \frac{\Pi_n}{\Delta_n \vdash B_n} \quad B_1, \dots, B_n, D'', \Gamma' \vdash C}{\Delta_1, \dots, \Delta_n, D'', \Gamma' \vdash C} \text{mc} .$$

Then  $\Xi$  reduces to

$$\frac{\frac{\Xi_1}{\Delta_1, \dots, \Delta_n, \Gamma' \vdash D'} \quad \frac{\Xi_2}{\Delta_1, \dots, \Delta_n, D'', \Gamma' \vdash C}}{\Delta_1, \dots, \Delta_n, D' \supset D'', \Gamma' \vdash C} \supset\mathcal{L} .$$

$-/\text{nat}\mathcal{L}$ : Suppose  $\Pi$  is

$$\frac{\frac{\Pi'}{\vdash Dz} \quad \frac{\Pi''}{Dj \vdash D(sj)} \quad B_1, \dots, B_n, DI, \Gamma' \vdash C}{B_1, \dots, B_n, \text{nat } I, \Gamma' \vdash C} \text{nat}\mathcal{L}$$

Let  $\Xi_1$  be

$$\frac{\frac{\Pi_1}{\Delta_1 \vdash B_1} \cdots \frac{\Pi_n}{\Delta_n \vdash B_n} \quad B_1, \dots, B_n, DI, \Gamma' \vdash C}{\Delta_1, \dots, \Delta_n, DI, \Gamma' \vdash C} \text{mc},$$

then  $\Xi$  reduces to

$$\frac{\frac{\Pi'}{\vdash Dz} \quad \frac{\Pi''}{Dj \vdash D(sj)} \quad \frac{\Xi_1}{\Delta_1, \dots, \Delta_n, DI, \Gamma' \vdash C}}{\Delta_1, \dots, \Delta_n, \text{nat } I, \Gamma' \vdash C} \text{nat}\mathcal{L}$$

$-/\text{eq}\mathcal{L}$ : If  $\Pi$  is

$$\frac{\left\{ B_{1\rho}, \dots, B_{n\rho}, \Gamma^\rho \vdash C\rho \right\}}{B_1, \dots, B_n, s = t, \Gamma' \vdash C} \text{eq}\mathcal{L} ,$$

then  $\Xi$  reduces to

$$\frac{\left\{ \frac{\left\{ \Delta_{i\rho} \vdash B_{i\rho} \right\}_{i \in \{1..n\}} \quad \Pi^\rho \quad B_{i\rho}, \dots, \Gamma' \rho \vdash C \rho}{\Delta_1 \rho, \dots, \Delta_n \rho, \Gamma' \rho \vdash C \rho} mc \right\}}{\Delta_1, \dots, \Delta_n, s = t, \Gamma' \vdash C} \text{ eq}\mathcal{L} .$$

$-/\circ\mathcal{R}$ : If  $\Pi$  is

$$\frac{\left\{ B_1, \dots, B_n, \Gamma^i \vdash C^i \right\}}{B_1, \dots, B_n, \Gamma \vdash C} \circ\mathcal{R} ,$$

where  $\circ\mathcal{R}$  is any right rule, then  $\Xi$  reduces to

$$\frac{\left\{ \frac{\Pi_1 \quad \dots \quad \Pi'_n \quad B_1, \dots, B_n, \Gamma^i \vdash C^i}{\Delta_1, \dots, \Delta_n, \Gamma^i \vdash C^i} mc \right\}}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} \circ\mathcal{R} .$$

Multicut cases:

$mc/\circ\mathcal{L}$ : If  $\Pi$  ends with a left rule other than  $c\mathcal{L}$  acting on  $B_1$  and  $\Pi_1$  ends with a multicut and reduces to  $\Pi'_1$ , then  $\Xi$  reduces to

$$\frac{\Pi'_1 \quad \Pi_2 \quad \dots \quad \Pi_n \quad B_1, \dots, B_n, \Gamma \vdash C}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} mc .$$

$-/mc$ : Suppose  $\Pi$  is

$$\frac{\left\{ \{B_i\}_{i \in I^j}, \Gamma^j \vdash D^j \right\}_{j \in \{1..m\}} \quad \{D^j\}_{j \in \{1..m\}}, \{B_i\}_{i \in I'}, \Gamma' \vdash C}{B_1, \dots, B_n, \Gamma^1, \dots, \Gamma^m, \Gamma' \vdash C} mc ,$$

where  $I^1, \dots, I^m, I'$  partition the formulas  $\{B_i\}_{i \in \{1..n\}}$  among the premise derivations  $\Pi_1, \dots, \Pi_m, \Pi'$ . For  $1 \leq j \leq m$  let  $\Xi^j$  be

$$\frac{\left\{ \Delta_i \vdash B_i \right\}_{i \in I^j} \quad \{B_i\}_{i \in I^j}, \Gamma^j \vdash D^j}{\{\Delta_i\}_{i \in I^j}, \Gamma^j \vdash D^j} mc .$$

Then  $\Xi$  reduces to

$$\frac{\left\{ \dots \vdash D^j \right\}_{j \in \{1..m\}} \quad \left\{ \Delta_i \vdash B_i \right\}_{i \in I'} \quad \dots \vdash C}{\Delta_1, \dots, \Delta_n, \Gamma^1, \dots, \Gamma^m, \Gamma' \vdash C} mc .$$

Structural case:

$-/c\mathcal{L}$ : If  $\Pi$  is

$$\frac{B_1, B_1, B_2, \dots, B_n, \Gamma \vdash C}{B_1, B_2, \dots, B_n, \Gamma \vdash C} c\mathcal{L} ,$$

then  $\Xi$  reduces to

$$\frac{\frac{\Delta_1 \vdash B_1 \quad \left\{ \Delta_i \vdash B_i \right\}_{i \in \{1..n\}} \quad \Pi'}{B_1, B_1, B_2, \dots, B_n, \Gamma \vdash C} \text{mc}}{\frac{\Delta_1, \Delta_1, \Delta_2, \dots, \Delta_n, \Delta_n, \Gamma \vdash C}{\Delta_1, \Delta_2, \dots, \Delta_n, \Gamma \vdash C} \text{c}\mathcal{L}} .$$

Axiom cases:

$id_\pi / \circ \mathcal{L}$ : Suppose  $\Pi$  ends with either  $nat\mathcal{L}$  or  $eq\mathcal{L}$  on  $B_1$  and  $\Pi_1$  ends with the  $id_\pi$  rule:

$$\frac{\pi_1.B = \pi_2.B_1}{\Delta'_1, B \vdash B_1} id_\pi$$

Then it is the case that  $B = \pi_1^{-1}.\pi_2.B_1$ . Apply the construction in Definition 20 to  $\Pi$  to get a derivation  $\Pi'$  of  $B, B_2, \dots, B_n, \Gamma \vdash C$ . The derivation  $\Xi$  reduces to

$$\frac{\frac{\Delta_2 \vdash B_2 \quad \dots \quad \Delta_n \vdash B_n \quad w(\Delta'_1, \Pi')}{B, \Delta'_1, B_2, \dots, B_n, \Gamma \vdash C} \text{mc}}{B, \Delta'_1, \Delta_2, \dots, \Delta_n, \Gamma \vdash C} .$$

$-/id_\pi$ : If  $\Pi$  ends with the  $id_\pi$  rule with a matching formula in  $\Gamma$ , i.e., there exists  $C' \in \Gamma$  such that  $\pi.C' = \pi'.C$  for some permutations  $\pi$  and  $\pi'$ , then then  $\Xi$  reduces to

$$\frac{}{\Delta_1, \dots, \Delta_n, \Gamma \vdash C} id_\pi$$

If  $\Pi$  ends with the  $id_\pi$  rule but  $C$  does not match any formula in  $\Gamma$ , then  $C$  must match one of the cut formulas, say  $B_1$ , i.e., there exists permutations  $\pi_1$  and  $\pi_2$  such that  $\pi_1.B_1 = \pi_2.C$ . That is,  $C = \pi_2^{-1}.\pi_1.B_1$ . In this case, we first apply the permutation  $\pi_2^{-1}.\pi_1$  to  $\Pi_1$  according to the construction in Definition 20 to get a derivation  $\Pi'_1$  of  $\Delta_1 \vdash \pi_2^{-1}.\pi_1.B_1$ .  $\Xi$  then reduces to  $w(\Delta_2 \cup \dots \cup \Delta_n \cup \Gamma, \Pi'_1)$ .  $\square$

An inspection of the rules of the logic and this definition will reveal that every derivation ending with a multicut has a reduct. Because we use a multiset as the left side of the sequent, there may be ambiguity as to whether a formula occurring on the left side of the rightmost premise to a multicut rule is in fact a cut formula, and if so, which of the left premises corresponds to it. As a result, several of the reduction rules may apply, and so a derivation may have multiple reducts.

The following lemmas show that the reduction relation is preserved by some of the transformations of derivations defined in the previous section.

### A.3 Normalizability and reducibility

We now define two properties of derivations: normalizability and reducibility. Each of these properties implies that the derivation can be reduced to a cut-free derivation of the same end-sequent. In the following, substitutions mean  $\Sigma$ -substitutions for some signature  $\Sigma$ . The definitions are similar to those by McDowell and Miller [10]. However, since the cut reduction in our case involves several transformations of derivations, other than substitutions and weakening, we need to build this transformations into the definitions of normalizability and reducibility.

**Definition 26.** A height-preserving (HP) transformation  $\mathcal{T}$  is a finite sequence of transformations  $\mathcal{F}_1, \dots, \mathcal{F}_n$  where each  $\mathcal{F}_i$  is one of the transformations described in Definition 18, Definition 19, Definition 20 and Definition 21. The number  $n$  is the order of  $\mathcal{T}$ . The application of  $\mathcal{T}$  to  $\Pi$  is defined as follows:

$$\begin{aligned} \mathcal{T}_0(\Pi) &= \Pi \\ \mathcal{T}_{i+1}(\Pi) &= \mathcal{F}_{i+1}(\mathcal{T}_i(\Pi)) \\ \mathcal{T}(\Pi) &= \mathcal{T}_n(\Pi) \end{aligned}$$

Note that a height-preserving transformation may not be defined for all derivations, and that it may be the identity transformation (i.e., it does nothing). Height-preserving transformations are ranged over by  $\mathcal{T}$ ,  $\mathcal{F}$ ,  $\mathcal{G}$  and  $\mathcal{H}$ .

**Lemma 27.** *Let  $\mathcal{T}$  be a height-preserving transformation. For any derivation  $\Pi$ , if  $\mathcal{T}(\Pi)$  is defined, then  $ht(\mathcal{T}(\Pi)) \leq ht(\Pi)$ .*

**Definition 28.** *We define the set of normalizable derivations to be the smallest set that satisfies the following conditions:*

1. *If a derivation  $\Pi$  ends with a multicut, then it is normalizable if for every height-preserving transformation  $\mathcal{T}$  such that  $\mathcal{T}(\Pi)$  is defined, there is a normalizable reduct of  $\mathcal{T}(\Pi)$ .*
2. *If a derivation ends with any rule other than a multicut, then it is normalizable if the premise derivations are normalizable.*

*These clauses assert that a given derivation is normalizable provided certain (perhaps infinitely many) other derivations are normalizable. If we call these other derivations the predecessors of the given derivation, then a derivation is normalizable if and only if the tree of the derivation and its successive predecessors is well-founded. In this case, the well-founded tree is called the normalization of the derivation.*

The set of normalizable derivations is not empty; the cut-free proofs, for instance, are normalizable.

Since a normalization is well-founded, it has an associated induction principle: for any property  $P$  of derivations, if for every derivation  $\Pi$  in the normalization,  $P$  holds for every predecessor of  $\Pi$  implies that  $P$  holds for  $\Pi$ , then  $P$  holds for every derivation in the normalization.

**Lemma 29.** *If there is a normalizable derivation of a sequent, then there is a cut-free derivation of the sequent.*

*Proof.* Let  $\Pi$  be a normalizable derivation of the sequent  $\Gamma \multimap B$ . We show by induction on the normalization of  $\Pi$  that there is a cut-free derivation of  $\Gamma \multimap B$ .

1. *If  $\Pi$  ends with a multicut, then any of its reducts is one of its predecessors and so is normalizable. One of its reduct, via the empty transformation, is also a derivation of  $\Gamma \multimap B$ , so by the induction hypothesis this sequent has a cut-free derivation.*
2. *Suppose  $\Pi$  ends with a rule other than multicut. Since we are given that  $\Pi$  is normalizable, by definition the premise derivations are normalizable. These premise derivations are the predecessors of  $\Pi$ , so by the induction hypothesis there are cut-free derivations of the premises. Thus there is a cut-free derivation of  $\Gamma \multimap B$ .*

□

The next four lemmas are also proved by induction on the normalization of derivations.

**Lemma 30.** *If  $\Pi$  is a normalizable derivation, then for any substitution  $\theta$  such that  $\Pi\theta$  is defined,  $\Pi\theta$  is normalizable.*

**Lemma 31.** *If  $\Pi$  is normalizable, then for any multiset of formulas  $\Delta$ , if  $w(\Delta, \Pi)$  is defined, then  $w(\Delta, \Pi)$  is normalizable.*

**Lemma 32.** *If  $\Pi$  is normalizable, then for any permutations  $\vec{\pi}$  such that  $\langle \vec{\pi} \rangle.\Pi$  is defined,  $\langle \vec{\pi} \rangle.\Pi$  is normalizable.*

**Lemma 33.** *If  $\Pi$  is normalizable, then for any nominal constants  $\vec{a}$  such that  $r(x, \vec{a}, \Pi)$  is defined,  $r(x, \vec{a}, \Pi)$  is normalizable.*

**Lemma 34.** *If  $\Pi$  is normalizable, then for any height-preserving transformation  $\mathcal{T}$  such that  $\mathcal{T}(\Pi)$  is defined,  $\mathcal{T}(\Pi)$  is normalizable.*

**Definition 35.** *The level of a sequent  $\Gamma \vdash C$  is the level of  $C$ . The level of a derivation  $\Pi$  is the level of its root sequent.*

The definition of reducibility for derivations is done by induction on the level of derivations: in defining the reducibility of level- $i$  derivations, we assume that the reducibility of derivations of level  $j$ , for all  $j < i$  is already defined. In the following definition, when we apply a transformation  $\mathcal{T}$  to a derivation  $\Pi$  of  $B_1, \dots, B_n \vdash B_0$ , we use the notation  $\mathcal{T}(B_i)$  to denote the formula in the root sequent of  $\mathcal{T}(\Pi)$  that results from applying the transformation to  $B_i$ .

**Definition 36.** *Reducibility. For any  $i$ , we define the set of reducible  $i$ -level derivations to be the smallest set of  $i$ -level derivations that satisfies the following conditions:*

1. *If a derivation  $\Pi$  ends with a multicut then it is reducible if for every height-preserving transformation  $\mathcal{T}$  such that  $\mathcal{T}(\Pi)$  is defined, there is a reducible reduct of  $\mathcal{T}(\Pi)$ .*
2. *Suppose the derivation ends with the implication right rule*

$$\frac{\Pi}{B, \Gamma \vdash C} \supset \mathcal{R}$$

*Then the derivation is reducible if  $\Pi$  is reducible and for every height-preserving transformation  $\mathcal{T}$  such that  $\mathcal{T}(\Pi)$  is defined, multiset of formulas  $\Delta$  and reducible derivation  $\Pi'$  of  $\Delta \vdash B'$ , where  $B' = \mathcal{T}(B)$ , the derivation*

$$\frac{\frac{\Pi'}{\Delta \vdash B'} \quad \frac{\mathcal{T}(\Pi)}{B', \Gamma' \vdash C'}}{\Delta, \Gamma' \vdash C'} mc$$

*is reducible.*

3. *If the derivation ends with the implication left rule or the nat rule, then it is reducible if the right premise derivation is reducible and the other premise derivations are normalizable.*
4. *If the derivation ends with any other rule, then it is reducible if the premise derivations are reducible.*

*These clauses assert that a given derivation is reducible provided certain other derivations are reducible. If we call these other derivations the predecessors of the given derivation, then a derivation is reducible only if the tree of the derivation and its successive predecessors is well founded. In this case, the well founded tree is called the reduction of the derivation.*

**Lemma 37.** *If a derivation is reducible, then it is normalizable.*

*Proof.* By induction on the reduction of the derivation. □

**Lemma 38.** *If a derivation  $\Pi$  is reducible, then for any height-preserving  $\mathcal{T}$  such that  $\mathcal{T}(\Pi)$  is defined,  $\mathcal{T}(\Pi)$  is reducible.*

*Proof.* By induction on the reduction of  $\Pi$  and Lemma 34.

#### A.4 Cut elimination

In the following, when we mention  $\mathcal{T}(\Pi)$  we assume implicitly that it is defined. We shall also use the notation  $\underline{B}_{\mathcal{T}}$  to denote  $\mathcal{T}(B)$ , that is the application of the transformation to the formula  $B$ . Similarly, the multiset  $\mathcal{T}(\Delta)$  will be written  $\underline{\Delta}_{\mathcal{T}}$ . We drop the subscript  $\mathcal{T}$  if it is clear from context which transformation we refer to.

**Lemma 39.** For any derivation  $\Pi$  of  $\Sigma; B_1, \dots, B_n, \Gamma \vdash C$  and reducible derivations  $\Pi_1, \dots, \Pi_n$  of  $\Sigma; \Delta_1 \vdash C_1, \dots, \Sigma; \Delta_n \vdash C_n$ , where  $n \geq 0$ , and for any transformations  $\mathcal{T}_1, \dots, \mathcal{T}_n, \mathcal{T}$  such that  $\mathcal{T}_i(\Pi_i)$  is defined and  $\mathcal{T}_i(C_i) = \mathcal{T}(B_i)$ , the derivation  $\Xi$

$$\frac{\frac{\mathcal{T}_1(\Pi_1)}{\Sigma'; \underline{\Delta}_1 \vdash B_1} \quad \dots \quad \frac{\mathcal{T}_n(\Pi_n)}{\Sigma'; \underline{\Delta}_n \vdash B_n} \quad \frac{\mathcal{T}(\Pi)}{\Sigma'; \underline{B}_1, \dots, \underline{B}_n, \underline{\Gamma} \vdash \underline{C}}}{\Sigma'; \underline{\Delta}_1, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{C}} \text{ mc}$$

is reducible.

*Proof.* The proof is by induction on  $ht(\Pi)$  with subordinate induction on  $n$  and on the reductions of  $\Pi_1, \dots, \Pi_n$ . Since the proof does not depend on the order of the inductions on reductions, when we need to distinguish of one the  $\Pi_i$ 's we shall refer to it as  $\Pi_1$  without loss of generality.

We need to show that for every  $\mathcal{T}'$ , the derivation every reduct of  $\mathcal{T}'(\Xi)$  is reducible. If  $n = 0$  then  $\mathcal{T}'(\Xi)$  reduces to  $\mathcal{T}'(\mathcal{T}(\Pi))$ . Since reducibility is preserved by height-preserving transformation, it suffices to consider the case where  $\mathcal{T}$  and  $\mathcal{T}'$  are the identity transformation, that is, we need only to show that  $\Pi$  is reducible. This is proved by case analysis on the last rule of  $\Pi$ . For each case, the results follow from the outer induction hypothesis and Definition 36. The case with  $\supset \mathcal{R}$  requires that height-preserving transformations do not increase the height of the derivations (see Lemma 27). In the cases for  $\supset \mathcal{L}$  and  $nat\mathcal{L}$  we need the additional information that reducibility implies normalizability (see Lemma 37).

For  $n > 0$ , we analyze all possible reductions that apply to  $\mathcal{T}'(\Xi)$  and show that every reduct of  $\mathcal{T}'(\Xi)$  is reducible. We suppose that  $\mathcal{T}'(\Xi)$  is of the following form:

$$\frac{\frac{\mathcal{F}_1(\Pi_1)}{\underline{\Delta}_1 \vdash \underline{C}_1} \quad \dots \quad \frac{\mathcal{F}_n(\Pi_n)}{\underline{\Delta}_n \vdash \underline{C}_n} \quad \frac{\mathcal{F}(\Pi)}{\underline{B}_1, \underline{B}_n, \underline{\Gamma} \vdash \underline{C}}}{\underline{\Delta}_1, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{C}} \text{ mc}$$

where  $\underline{B}_{i\mathcal{F}} = \underline{C}_{i\mathcal{F}_i}$ . In several cases below, we often omit the subscripts  $\mathcal{F}$  or  $\mathcal{F}_i$  when it is clear from context which transformations we refer to. We also often switch between  $\underline{B}_{i\mathcal{F}}$  and  $\underline{C}_{i\mathcal{F}_i}$  to make the inference figures more readable.

Most cases follow immediately from the inductive hypothesis and Definition 36 and Lemma 37, Lemma 38 and Lemma 27. We show here the interesting cases.

$\supset \mathcal{R} / \supset \mathcal{L}$ : Suppose  $\Pi_1$  and  $\Pi$  are

$$\frac{\frac{\Pi'_1}{\Delta_1, B'_1 \vdash B''_1}}{\Delta_1 \vdash B'_1 \supset B''_1} \supset \mathcal{R} \quad \frac{\frac{\Pi'}{B_2, \dots, \Gamma \vdash B'_1} \quad \frac{\Pi''}{B''_1, B_2, \dots, \Gamma \vdash C}}{B'_1 \supset B''_1, B_2, \dots, B_n, \Gamma \vdash C} \supset \mathcal{L} .$$

Let  $\Xi_1$  be the derivation

$$\frac{\frac{\mathcal{F}_2(\Pi_2)}{\underline{\Delta}_2 \vdash \underline{B}_2} \quad \dots \quad \frac{\mathcal{F}_n(\Pi_n)}{\underline{\Delta}_n \vdash \underline{B}_n} \quad \frac{\mathcal{F}_n(\Pi')}{\underline{B}_2, \dots, \underline{B}_n, \underline{\Gamma} \vdash \underline{B}'_1}}{\underline{\Delta}_2, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{B}'_1} \text{ mc}$$

Then  $\Xi_1$  is reducible by induction hypothesis since  $\mathcal{F}$  and  $\mathcal{F}_i$  preserve reducibility (Lemma 38) and do not increase the height of derivations (Lemma 27). Since we are given that  $\Pi_1$  is reducible, by Definition 36, the derivation  $\Xi_2$

$$\frac{\frac{\Xi_1}{\underline{\Delta}_2, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{B}'_1} \quad \frac{\mathcal{F}_1(\Pi'_1)}{\underline{B}'_1, \underline{\Delta}_1 \vdash \underline{B}''_1}}{\underline{\Delta}_1, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{B}''_1} \text{ mc}$$

is reducible as well. Therefore, the reduct of  $\mathcal{T}'(\Xi)$

$$\frac{\dots \vdash \underline{B}_1'' \quad \left\{ \mathcal{F}_i(\Pi_i) \right\}_{i \in \{2..n\}} \quad \mathcal{F}(\Pi'') \quad \underline{B}_1'', \{B_i\}_{i \in \{2..n\}}, \underline{\Gamma} \vdash \underline{C}}{\frac{\underline{\Delta}_1, \dots, \underline{\Delta}_n, \underline{\Gamma}, \underline{\Delta}_2, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{C}}{\underline{\Delta}_1, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{C}} \text{mc}} \text{c}\mathcal{L} .$$

is reducible by the outer induction hypothesis and Definition 36.

$\forall\mathcal{R}/\forall\mathcal{L}$  : Suppose  $\Pi_1$  and  $\Pi$  are

$$\frac{\Pi_1' \quad \underline{\Sigma}, h; \underline{\Delta}_1 \vdash B[h\bar{c}/x]}{\underline{\Sigma}; \underline{\Delta}_1 \vdash \forall x.B} \forall\mathcal{R} \quad \frac{\Pi' \quad \underline{\Sigma}; B[t/x], B_2, \dots, B_n, \underline{\Gamma} \vdash \underline{C}}{\underline{\Sigma}; \forall x.B, B_2, \dots, B_n, \underline{\Gamma} \vdash \underline{C}} \forall\mathcal{L}$$

Applying the transformation  $\mathcal{F}_1$  to  $\Pi_1$  (and similarly,  $\mathcal{F}$  to  $\Pi$ ) might require several transformation be done on the premise of the derivation, e.g., to avoid clashes of nominal constants, etc., so let us suppose that  $\mathcal{F}_1(\Pi_1)$  and  $\mathcal{F}(\Pi)$  are of the following shapes:

$$\frac{\mathcal{G}_1(\Pi_1') \quad \underline{\Sigma}', h; \underline{\Delta}_1 \vdash D[h'\bar{d}/x]}{\underline{\Sigma}'; \underline{\Delta}_1 \vdash \forall x.D} \forall\mathcal{R} \quad \frac{\mathcal{G}(\Pi') \quad \underline{\Sigma}'; D[s/x], \underline{B}_2, \dots, \underline{B}_n, \underline{\Gamma} \vdash \underline{C}}{\underline{\Sigma}'; \forall x.D, \underline{B}_2, \dots, \underline{B}_n, \underline{\Gamma} \vdash \underline{C}} \forall\mathcal{L}$$

where  $\forall x.D = \forall x.B$  and  $D[s/x] = B[t/x]$ . If the support of  $D[s/x]$  is larger than  $\{\bar{d}\}$ , then the reduction rule for  $\forall\mathcal{R}/\forall\mathcal{L}$  requires further transformations be applied to  $\mathcal{G}_1(\Pi_1')$ , i.e., as is described in Lemma 9. So let us suppose that this transformation is applied, resulting in a derivation

$$\frac{\mathcal{G}'_1(\Pi_1')}{\underline{\Sigma}', f; \underline{\Delta}_1 \vdash D[f\bar{c}/x]} .$$

Then  $\mathcal{T}'(\Xi)$  reduces to

$$\frac{\mathcal{G}'_1(\Pi_1')[\lambda\bar{c}.s/f] \quad \mathcal{F}_2(\Pi_2) \quad \dots \quad \mathcal{F}_n(\Pi_n) \quad \mathcal{G}(\Pi') \quad \underline{\Sigma}'; \underline{\Delta}_1 \vdash D[s/x] \quad \underline{\Delta}_2 \vdash \underline{B}_2 \quad \dots \quad \underline{\Delta}_2 \vdash \underline{B}_2 \quad \underline{\Sigma}'; D[s/x], \dots, \underline{\Gamma} \vdash \underline{C}}{\underline{\Sigma}'; \underline{\Delta}_1, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{C}} \text{mc}$$

which is reducible by the outer induction hypothesis.

$\text{nat}\mathcal{R}/\text{nat}\mathcal{L}$  : Suppose  $\Pi_1$  and  $\Pi$  are

$$\frac{\Pi_1' \quad \underline{\Delta}_1 \vdash \text{nat } M}{\underline{\Delta}_1 \vdash \text{nat } \bar{M}} \text{nat}\mathcal{R} \quad \frac{\Pi' \quad \Pi'' \quad \Pi''' \quad \vdash Dz \quad D j \vdash D(sj) \quad D(sM), B_2, \dots, B_n, \underline{\Gamma} \vdash \underline{C}}{\text{nat}(sI), B_2, \dots, B_n, \underline{\Gamma} \vdash \underline{C}} \text{nat}\mathcal{L}$$

then  $\mathcal{F}_1(\Pi_1)$  and  $\mathcal{F}(\Pi)$  are

$$\frac{\mathcal{F}_1(\Pi_1') \quad \underline{\Delta}_1 \vdash \text{nat } I}{\underline{\Delta}_1 \vdash \text{nat } I} \text{nat}\mathcal{R} \quad \frac{\Pi' \quad \Pi'' \quad \mathcal{F}(\Pi''') \quad \vdash Dz \quad D j \vdash D(sj) \quad D(sI), \underline{B}_2, \dots, \underline{B}_n, \underline{\Gamma} \vdash \underline{C}}{\text{nat}(sI), \underline{B}_2, \dots, \underline{B}_n, \underline{\Gamma} \vdash \underline{C}} \text{nat}\mathcal{L}$$

Note that the derivations  $\Pi'$  and  $\Pi''$  are not affected by the transformation  $\mathcal{F}$  since  $D$  is a closed term with no occurrences of nominal constants and  $j$  in  $\Pi''$  is a new eigenvariable. Let  $\Xi_1$  be the derivation

$$\frac{\mathcal{F}_1(\Pi_1') \quad \vdash Dz \quad D j \vdash D(sj) \quad \overline{DI \vdash DI} \text{ id}_\pi}{\underline{\Delta}_1 \vdash \text{nat } I \quad \text{nat } I \vdash DI} \text{nat}\mathcal{L} \quad \frac{\underline{\Delta}_1 \vdash \text{nat } I}{\underline{\Delta}_1 \vdash DI} \text{mc} .$$

Since the height of the right premise is no larger than  $ht(\Pi)$ , and  $\Pi'_1$  is a predecessor of  $\Pi_1$ ,  $\Xi_1$  is reducible by induction on the reduction of  $\Pi_1$ . Let  $\{\vec{c}\}$  be the support of  $I$ . We construct the derivation  $\Pi^\bullet$  of  $h; D(h\vec{c}) \vdash D(s(h\vec{c}))$  from  $\Pi''$  using the procedures described in Definition 19 and Definition 21. Let  $\Xi_2$  be

$$\frac{\frac{\Xi_1 \quad \Pi^\bullet[\lambda\vec{c}.I/h]}{\underline{\Delta}_1 \vdash DI \quad DI \vdash D(sI)} \quad mc.}{\underline{\Delta}_1 \vdash D(sI)}$$

Since  $ht(\Pi^\bullet[\lambda\vec{c}.I/h]) \leq ht(\Pi'')$ , by the outer induction hypothesis,  $\Xi_2$  is also reducible. Therefore the reduct of  $\mathcal{T}'(\Xi)$

$$\frac{\frac{\Xi_2 \quad \mathcal{F}_2(\Pi_2) \quad \mathcal{F}_n(\Pi_n) \quad \mathcal{F}(\Pi''')}{\underline{\Delta}_1 \vdash D(sI) \quad \underline{\Delta}_2 \vdash \underline{B}_2 \quad \dots \quad \underline{\Delta}_n \vdash \underline{B}_n \quad D(sI), \underline{B}_2, \dots, \underline{\Gamma} \vdash \underline{C}} \quad mc.}{\underline{\Delta}_1, \dots, \underline{\Delta}_2, \underline{\Gamma} \vdash \underline{C}}$$

is reducible by the outer induction hypothesis.

$\text{eq}\mathcal{L}/\circ\mathcal{L}$ : Suppose  $\Pi_1$  is

$$\frac{\left\{ \frac{\Pi^\theta}{\underline{\Delta}_1\theta \vdash B_1\theta} \right\}_\theta}{s = t, \underline{\Delta}_1 \vdash B_1} \text{eq}\mathcal{L}$$

then  $\mathcal{F}_1(\Pi_1)$  is

$$\frac{\left\{ \frac{\Pi^{\bullet\rho}}{\underline{\Delta}_1\theta \vdash B_1\theta} \right\}_\rho}{\underline{s} = \underline{t}, \underline{\Delta}_1 \vdash \underline{B}_1} \text{eq}\mathcal{L}$$

where each  $\Pi^{\bullet\rho}$  is obtained from some  $\Pi^\theta$  by the transformations described in Definition 18, Definition 19, Definition 20 and Definition 21. We denote with  $f(\rho)$  the substitution  $\theta$  such that  $\Pi^{\bullet\rho}$  is constructed out of  $\Pi^\theta$ . Thus we can write each  $\Pi^{\bullet\rho}$  as the derivation  $\mathcal{F}_\rho(\Pi^{f(\rho)})$  for some transformation  $\mathcal{F}_\rho$ . The reduct of  $\mathcal{T}'(\Xi)$

$$\frac{\left\{ \frac{\mathcal{F}_\rho(\Pi^{f(\rho)}) \quad \mathcal{F}_2(\Pi_2)\rho \quad \mathcal{F}_n(\Pi_n)\rho \quad \mathcal{F}(\Pi)\rho}{\underline{\Delta}'_1\rho \vdash \underline{B}_1\rho \quad \underline{\Delta}_2\rho \vdash \underline{B}_2\rho \quad \dots \quad \underline{\Delta}_n\rho \vdash \underline{B}_n\rho \quad \underline{B}_1\rho, \dots, \underline{B}_n\rho, \underline{\Gamma}\rho \vdash \underline{C}\rho} \quad mc.}{\underline{\Delta}'_1\rho, \underline{\Delta}_2\rho, \dots, \underline{\Delta}_n\rho, \underline{\Gamma}\rho \vdash \underline{C}\rho} \right\}_\rho}{\underline{s} = \underline{t}, \underline{\Delta}'_1, \underline{\Delta}_2, \dots, \underline{\Delta}_n, \underline{\Gamma} \vdash \underline{C}} \text{eq}\mathcal{L}$$

Each premise derivation of the above derivation is reducible by the induction hypothesis on the reduction of  $\Pi_1$ , since each  $\Pi^{f(\rho)}$  is a predecessor of  $\Pi_1$ . The reduct of  $\mathcal{T}'(\Xi)$  is therefore reducible by Definition 36.

$-/\supset\mathcal{R}$ : Suppose  $\Pi$  is

$$\frac{\mathcal{F}(\Pi')}{\frac{B_1, \dots, B_n, \underline{\Gamma}, C_1 \vdash C_2}{B_1, \dots, B_n, \underline{\Gamma} \vdash C_1 \supset C_2}} \supset\mathcal{R}$$

then  $\mathcal{F}_1(\Pi)$

$$\frac{\mathcal{F}(\Pi')}{\frac{B_1, \dots, B_n, \underline{\Gamma}, C_1 \vdash C_2}{B_1, \dots, B_n, \underline{\Gamma} \vdash C_1 \supset C_2}} \supset\mathcal{R}$$

Let  $\Xi_1$  be

$$\frac{\mathcal{F}_1(\Pi_1) \quad \mathcal{F}_n(\Pi_n) \quad \mathcal{F}(\Pi')}{\underline{\Delta}_1 \vdash \underline{B}_1 \quad \dots \quad \underline{\Delta}_n \vdash \underline{B}_n \quad B_1, \dots, \underline{B}_1, \underline{\Gamma}, \underline{C}_1 \vdash \underline{C}_2}{\underline{\Delta}_1, \dots, \underline{\Delta}_n, \underline{C}_1 \vdash \underline{C}_2}$$

which is reducible by the outer induction hypothesis. Let  $\Xi_2$  be the derivation

$$\frac{\frac{\Xi_1}{\underline{\Delta_1, \dots, \Delta_n, \underline{\Gamma}, C_1} \vdash C_2}}{\underline{\Delta_1, \dots, \Delta_n, \underline{\Gamma}} \vdash C_1 \supset C_2} \supset \mathcal{R} ,$$

which is the reduct of  $\mathcal{T}'(\Xi)$ . To show that  $\Xi_2$  is reducible, we need to show that for any  $\mathcal{T}''$ , and for any derivation  $\Pi''$  of  $\Delta \vdash D$ , where  $D = \mathcal{T}''(\underline{C_1})$ , the derivation  $\Xi_3$

$$\frac{\frac{\Pi''}{\Delta \vdash D} \quad \frac{\mathcal{T}''(\Xi_2)}{D, \underline{\Delta_{1\mathcal{G}_1}}, \dots, \underline{\Delta_{n\mathcal{G}_n}}, \underline{\Gamma_{\mathcal{G}}} \vdash C_{2\mathcal{G}}} mc}{\Delta, \underline{\Delta_{1\mathcal{G}_1}}, \dots, \underline{\Delta_{n\mathcal{G}_n}}, \underline{\Gamma_{\mathcal{G}}} \vdash C_{2\mathcal{G}}}$$

is reducible. Here the transformations  $\mathcal{G}_i$  and  $\mathcal{G}$  are transformations associated with the premise derivations in  $\mathcal{T}''(\Xi_2)$ .  $\Xi_3$  is reducible if for any transformation  $\mathcal{H}$ , every reduct of the derivation  $\mathcal{H}(\Xi_3)$  is reducible. The reduct of  $\mathcal{H}(\Xi_3)$  in this case is:

$$\frac{\frac{\mathcal{H}'(\Pi'')}{\underline{\Delta} \vdash \underline{D}} \quad \frac{\mathcal{H}_1(\Pi_1)}{\underline{\Delta_1} \vdash \underline{B_1}} \quad \dots \quad \frac{\mathcal{H}_n(\Pi_n)}{\underline{\Delta_n} \vdash \underline{B_n}} \quad \frac{\mathcal{H}''(\Pi')}{\underline{D}, \underline{B_1}, \dots, \underline{B_n}, \underline{\Gamma} \vdash \underline{C_2}} mc}{\underline{\Delta}, \underline{\Delta_1}, \dots, \underline{\Delta_n}, \underline{\Gamma} \vdash \underline{C_2}}$$

where  $\mathcal{H}_1, \dots, \mathcal{H}_n$  and  $\mathcal{H}'$  are transformations applied to the premises of  $\mathcal{H}(\mathcal{T}''(\Xi_2))$  and  $\mathcal{H}'$  is the transformation applied to the left premise of  $\mathcal{H}(\Xi_3)$ . This derivation is reducible by the outer induction hypothesis.  $\square$

**Corollary 40.** *Every derivation is reducible.*

*Proof.* This result follows immediately from Lemma 39 with  $n = 0$ .  $\square$

**Proof of Theorem 11** Follows immediately from Corollary 40, Lemma 37 and Lemma 29.  $\square$

## A.5 Correspondence between $LG$ and $FO\lambda^\nabla$

Sequents in  $FO\lambda^\nabla$  are expressions of the form

$$\Sigma; \sigma_1 \triangleright B_1, \dots, \sigma_n \triangleright B_n \vdash \sigma_0 \triangleright B_0.$$

$\Sigma$  is the *signature* of the sequent,  $\sigma_i$  is a list of variables locally scoped over  $B_i$ , and is referred to as *local signature*. The expression  $\sigma_i \triangleright B_i$  is called a *local judgment*, or *judgment* for short. In [17], local judgments are considered equal modulo renaming of their local signatures, e.g.,  $(a, b) \triangleright P a b$  is equal to  $(c, d) \triangleright P c d$ . Local judgments are ranged over by scripted capital letters, e.g.,  $\mathcal{B}, \mathcal{D}$ , etc. For the purpose of proving the correspondence with  $LG$ , however, we will make this renaming step explicit, by including the rules:

$$\frac{\vec{y} \triangleright B', \Gamma \vdash C}{\vec{x} \triangleright B, \Gamma \vdash C} \alpha_{\mathcal{R}}, \lambda \vec{x}. B \equiv_{\alpha} \lambda \vec{y}. B' \quad \frac{\Gamma \vdash \vec{y} \triangleright B'}{\Gamma \vdash \vec{x} \triangleright B} \alpha_{\mathcal{L}}, \lambda \vec{x}. B \equiv_{\alpha} \lambda \vec{y}. B'$$

The inference rules of  $FO\lambda^\nabla$  are given in Figure 4.

We now consider the correspondence between  $LG$  with  $FO\lambda^\nabla$  extended with the following axiom schemes:

$$\nabla x \nabla y. B x y \equiv \nabla y \nabla x. B x y. \quad (2)$$

$$B \equiv \nabla x. B, \text{ provided that } x \text{ is not free in } B. \quad (3)$$

We can equivalently state these two axioms as the following inference rules:

$$\frac{(\vec{x}, b, a, \vec{y}) \triangleright B, \Gamma \vdash C}{(\vec{x}, a, b, \vec{y}) \triangleright B, \Gamma \vdash C} p\mathcal{L} \quad \frac{\Gamma \vdash (\vec{x}, b, a, \vec{y}) \triangleright B}{\Gamma \vdash (\vec{x}, a, b, \vec{y}) \triangleright B} p\mathcal{L}$$

$$\begin{array}{c}
\frac{}{\Sigma; \sigma \triangleright B, \Gamma \vdash \sigma \triangleright B} id \quad \frac{\Sigma; \Delta \vdash B \quad \Sigma; B, \Gamma \vdash C}{\Sigma; \Delta, \Gamma \vdash C} cut \\
\frac{\Sigma; \sigma \triangleright B, \sigma \triangleright C, \Gamma \vdash \mathcal{D}}{\Sigma; \sigma \triangleright B \wedge C, \Gamma \vdash \mathcal{D}} \wedge \mathcal{L} \quad \frac{\Sigma; \Gamma \vdash \sigma \triangleright B \quad \Sigma; \Gamma \vdash \sigma \triangleright C}{\Sigma; \Gamma \vdash \sigma \triangleright B \wedge C} \wedge \mathcal{R} \\
\frac{\Sigma; \sigma \triangleright B, \Gamma \vdash \mathcal{D} \quad \Sigma; \sigma \triangleright C, \Gamma \vdash \mathcal{D}}{\Sigma; \sigma \triangleright B \vee C, \Gamma \vdash \mathcal{D}} \vee \mathcal{L} \quad \frac{\Sigma; \Gamma \vdash \sigma \triangleright B}{\Sigma; \Gamma \vdash \sigma \triangleright B \vee C} \vee \mathcal{R} \\
\frac{}{\Sigma; \sigma \triangleright \perp, \Gamma \vdash B} \perp \mathcal{L} \quad \frac{\Sigma; \Gamma \vdash \sigma \triangleright C}{\Sigma; \Gamma \vdash \sigma \triangleright B \vee C} \vee \mathcal{R} \\
\frac{\Sigma; \Gamma \vdash \sigma \triangleright B \quad \Sigma; \sigma \triangleright C, \Gamma \vdash \mathcal{D}}{\Sigma; \sigma \triangleright B \supset C, \Gamma \vdash \mathcal{D}} \supset \mathcal{L} \quad \frac{\Sigma; \sigma \triangleright B, \Gamma \vdash \sigma \triangleright C}{\Sigma; \Gamma \vdash \sigma \triangleright B \supset C} \supset \mathcal{R} \\
\frac{\Sigma, \sigma \vdash t : \gamma \quad \Sigma; \sigma \triangleright B[t/x], \Gamma \vdash C}{\Sigma; \sigma \triangleright \forall \gamma x.B, \Gamma \vdash C} \forall \mathcal{L} \quad \frac{\Sigma, h; \Gamma \vdash \sigma \triangleright B[(h \sigma)/x]}{\Sigma; \Gamma \vdash \sigma \triangleright \forall x.B} \forall \mathcal{R} \\
\frac{\Sigma, h; \sigma \triangleright B[(h \sigma)/x], \Gamma \vdash C}{\Sigma; \sigma \triangleright \exists x.B, \Gamma \vdash C} \exists \mathcal{L} \quad \frac{\Sigma, \sigma \vdash t : \gamma \quad \Sigma; \Gamma \vdash \sigma \triangleright B[t/x]}{\Sigma; \Gamma \vdash \sigma \triangleright \exists \gamma x.B} \exists \mathcal{R} \\
\frac{\Sigma; (\sigma, y) \triangleright B[y/x], \Gamma \vdash C}{\Sigma; \sigma \triangleright \nabla x B, \Gamma \vdash C} \nabla \mathcal{L} \quad \frac{\Sigma; \Gamma \vdash (\sigma, y) \triangleright B[y/x]}{\Sigma; \Gamma \vdash \sigma \triangleright \nabla x B} \nabla \mathcal{R} \\
\frac{\Sigma; B, B, \Gamma \vdash C}{\Sigma; B, \Gamma \vdash C} c\mathcal{L} \quad \frac{\Sigma; \Gamma \vdash C}{\Sigma; B, \Gamma \vdash C} w\mathcal{L} \quad \frac{}{\Sigma; \Gamma \vdash \sigma \triangleright \top} \top \mathcal{R}
\end{array}$$

Fig. 4. The core inference rules of  $FO\lambda^\nabla$ .

$$\begin{array}{c}
\frac{(\vec{x}, a, \vec{y}) \triangleright B, \Gamma \vdash C}{(\vec{x}\vec{y}) \triangleright B, \Gamma \vdash C} ss\mathcal{L}, a \notin \{\vec{x}, \vec{y}\} \quad \frac{\Gamma \vdash (\vec{x}, a, \vec{y}) \triangleright B}{\Gamma \vdash (\vec{x}\vec{y}) \triangleright B} ss\mathcal{R}, a \notin \{\vec{x}, \vec{y}\} \\
\frac{(\vec{x}\vec{y}) \triangleright B, \Gamma \vdash C}{(\vec{x}, a, \vec{y}) \triangleright B, \Gamma \vdash C} ws\mathcal{L}, a \notin \text{supp}(B) \quad \frac{\Gamma \vdash (\vec{x}\vec{y}) \triangleright B}{\Gamma \vdash (\vec{x}, a, \vec{y}) \triangleright B} ss\mathcal{R}, a \notin \text{supp}(B)
\end{array}$$

Implicit in the above rules is the assumption that variables in local signatures are considered as special constants, much like the nominal constants in  $LG$ . The support of  $B$ , within a local signature  $\sigma$ , is defined similarly as it is in  $LG$ : it is the set  $\{a \in \sigma \mid a \text{ occurs in } B\}$ .

The logical system with the inference rules in Figure 4 together with  $\alpha_{\mathcal{R}}, \alpha_{\mathcal{L}}, p\mathcal{L}, p\mathcal{R}, ss\mathcal{L}, ss\mathcal{R}, ws\mathcal{L}$  and  $ws\mathcal{R}$  is referred to as  $FO\lambda^{\nabla+}$ . In relating  $LG$  and  $FO\lambda^{\nabla+}$ , we map the local signatures to nominal constants, and vice versa. In the following, given a formula  $B$ , we assume a particular enumeration of the nominal constants appearing in  $B$  based the left-to-right order of their appearance in  $B$ .

**Lemma 41.** *If the sequent  $\Sigma; B_1, \dots, B_n \vdash B_0$  is provable in  $LG$  then the sequent*

$$\Sigma; \vec{c}_1 \triangleright B_1, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright B_0$$

where  $\vec{c}_i$  is an enumeration of  $\text{supp}(B_i)$ , is provable in  $FO\lambda^{\nabla+}$ .

*Proof.* Suppose that  $\Pi$  is a proof of  $\Sigma; B_1, \dots, B_n \vdash B_0$ . We construct a proof  $\Pi'$  of

$$\Sigma; \vec{c}_1 \triangleright B_1, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright B_0$$

by induction on  $ht(\Pi)$ . We consider some interesting cases here:

– Suppose  $\Pi$  ends with  $id_\pi$  :

$$\frac{\pi.B_i = \pi'.B_0}{\Gamma', B_i \vdash B_0} id_\pi$$

The permutations  $\pi$  and  $\pi'$  can be imitated by a series of renaming ( $\alpha_{\mathcal{R}}$  and  $\alpha_{\mathcal{L}}$  rules). The derivation  $\Pi'$  is therefore constructed by applying a series of  $\alpha_{\mathcal{R}}, \alpha_{\mathcal{L}}$ , followed by the  $id$  rule.

- Suppose  $\Pi$  ends with  $\supset \mathcal{R}$  : in this case we suppose that  $B_0 = C \supset D$ .

$$\frac{\Pi_1}{\frac{B_1, \dots, B_n, C \vdash D}{B_1, \dots, B_n \vdash C \supset D} \supset \mathcal{R}}$$

By induction hypothesis we have a derivation  $\Pi_2$  of

$$\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n, \vec{a} \triangleright C \vdash \vec{b} \triangleright D$$

We first have to weaken the signatures  $\vec{a}$  and  $\vec{b}$  to  $\vec{c}_0$  before applying the introduction rule for  $\supset$ . That is,  $\Pi'$  is the derivation

$$\frac{\frac{\Pi_2}{\frac{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n, \vec{a} \triangleright C \vdash \vec{b} \triangleright D}{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n, \vec{c}_0 \triangleright C \vdash \vec{c}_0 \triangleright D}^*}}{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright C \supset D} \supset \mathcal{R}}$$

Here the star  $^*$  denotes a series of applications of  $ws\mathcal{L}$ ,  $ws\mathcal{R}$ ,  $p\mathcal{L}$  and  $p\mathcal{R}$ .

- Suppose  $\Pi$  is

$$\frac{\Pi_1}{\frac{B_1, \dots, B_n \vdash C[t/x]}{B_1, \dots, B_n \vdash \exists x.C} \exists \mathcal{R}}$$

It is possible that  $t$  contains new constants that are not in the support of  $C$ . Suppose  $\vec{d}$  is an enumeration of the support of  $C[t/x]$ . The derivation  $\Pi'$  is constructed as follows

$$\frac{\frac{\Pi_2}{\frac{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{d} \triangleright C[t/x]}{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{d} \triangleright \exists x.C} \exists \mathcal{R}}}{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright \exists x.C}^*}}$$

where  $\Pi_2$  is obtained from induction hypothesis applied to  $\Pi_1$ , and the rule  $^*$  denotes a series of applications of  $ss\mathcal{R}$  (for introducing new constants) and  $p\mathcal{R}$  (for rearranging the order of the local signature).

- For other cases, the construction of  $\Pi'$  follows the same pattern as in the previous cases, i.e., by induction hypothesis, followed by some rearranging, extension, or weakening of local signatures.

□

**Lemma 42.** *If the sequent*

$$\Sigma; \vec{c}_1 \triangleright B_1, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright B_0$$

*is provable in  $FO\lambda^{\nabla+}$  then the sequent  $\Sigma; B_1, \dots, B_n \vdash B_0$  is provable in LG*

*Proof.* Suppose  $\Pi$  is a derivation of

$$\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright B_0$$

We construct a derivation  $\Pi'$  of  $B_1, \dots, B_n \vdash B_0$  by induction on  $ht(\Pi)$ . We show here the interesting cases; the other cases follow immediately from induction hypothesis:

- If  $\Pi$  ends with  $id$ ,  $\top\mathcal{R}$ , or  $\perp\mathcal{L}$  then  $\Pi'$  ends with the same rule.
- Suppose  $\Pi$  is

$$\frac{\Pi_1}{\frac{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{d} \triangleright B}{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright B} \alpha_{\mathcal{L}}}}$$

By induction hypothesis, there is a derivation  $\Pi_2$  of  $B_1, \dots, B_n \vdash B$ . To get  $\Pi'$  apply the procedure in Definition 20 to  $\Pi_2$  to rename  $B$  to  $B_0$ .

– Suppose  $\Pi$  is

$$\frac{\Pi_1}{\frac{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright C[(h \vec{c}_0)/x]}{\vec{c}_1 \triangleright B_1, \dots, \vec{c}_n \triangleright B_n \vdash \vec{c}_0 \triangleright \forall x.C} \forall \mathcal{R}}$$

By induction hypothesis, there is a derivation  $\Pi_2$  of  $B_1, \dots, B_n \vdash C[(h \vec{c}_0)/x]$ . Suppose  $\{\vec{d}\} = \text{supp}(C)$ . Then  $\Pi'$  is

$$\frac{\Pi_2[\lambda \vec{c}_0. h' \vec{d}/h]}{\frac{B_1, \dots, B_n \vdash C[h' \vec{d}/x]}{B_1, \dots, B_n \vdash \forall x.C} \forall \mathcal{R}}$$

– If  $\Pi$  ends with  $\exists \mathcal{L}$ , apply the same construction as in the previous case. □

**Proof of Theorem 13:** This result follows immediately from Lemma 41 and Lemma 42. □

## Appendix B. Properties of an object logic

*Proof outline for Theorem 15*

*Proof.* We show here a proof outline for formula (3), the rest can be done by simulating the proof of the same formulas in  $FO\lambda^{\Delta \mathbb{N}}$ . Proving formula (3) reduces to proving the sequent

$$G, i, L; \text{nat } i, \text{list } L, \nabla x. \text{seq}_i L (Gx) \vdash \forall x. \text{seq}_i L (Gx)$$

This sequent is proved by induction on  $i$ , with the induction hypothesis:

$$D = \lambda i. \forall L \forall G. (\forall x. \text{list } (Lx)) \supset (\nabla x. \text{seq}_i (Lx) (Gx)) \supset \forall x. \text{seq}_i (Lx) (Gx).$$

The base case, i.e., the sequent  $\vdash Dz$  is proved by case analyses and induction on lists, that is, to show that

$$\forall L \forall A. (\nabla x. \text{elem } (Ax) (Lx)) \supset \forall x. \text{elem } (Ax) (Lx).$$

The inductive cases correspond to the sequent

$$i, L, G; D i, \forall x. \text{list } (Lx), \nabla x. \text{seq}_{(s i)} Lx Gx \vdash \forall x. \text{seq}_{(s i)} Lx Gx.$$

We first apply the left introduction rule for  $\nabla$ , replacing  $x$  with a fresh nominal constant  $a$ . This is then followed by  $\text{def}\mathcal{L}$  on the object sequent. There are several cases to consider, the interesting case is where  $G$  is instantiated to  $\lambda x. \bigwedge (G' x)$ :

$$i, L, G'; D i, \forall x. \text{list } (Lx), \nabla y. \text{seq}_i (La) (G' ay) \vdash \forall x. \text{seq}_{(s i)} Lx \bigwedge (G' x)$$

This sequent is then proved by using the inductive hypothesis  $Di$ , that is, by instantiating  $L$  and  $G$  in  $Di$  with  $L$  and  $G'$ , followed by the right introduction rule for  $\forall$  and  $\text{def}\mathcal{R}$  rule. □