A Proof Theory for Generic Judgments

Dale Miller
INRIA-Futurs & École polytechnique
and
Alwen Tiu
École polytechnique & Penn State University

The operational semantics of a computation system is often presented as inference rules or, equivalently, as logical theories. Specifications can be made more declarative and high-level if syntactic details concerning bound variables and substitutions are encoded directly into the logic using term-level abstractions (λ-abstraction) and proof-level abstractions (eigenvariables). When one wishes to use such logical theories to support reasoning about properties of computation, the usual quantifiers and proof-level abstractions do not seem adequate: proof-level abstraction of variables with scope over sequents (*global* scope) as well as over only formulas (*local* scope) seem required for many examples. We will present a sequent calculus which provides this local notion of proof-level abstraction via *generic judgment* and a new quantifier, ∇, which explicitly manipulates such local scope. Intuitionistic logic extended with ∇ satisfies cut-elimination even when the logic is additionally strengthened with a proof theoretic notion of definitions. The resulting logic can be used to encode naturally a number of examples involving abstractions, and we illustrate the uses of ∇ with the π-calculus and an encoding of provability of an object-logic.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Proof Theory*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Specification Techniques*

General Terms: Design, Theory, Verification

Additional Key Words and Phrases: proof search, reasoning about operational semantics, generic judgments, λ-tree syntax, higher-order abstract syntax, ∇-quantifier

1. EIGENVARIABLES AND GENERIC REASONING

In specifying and reasoning about computations involving abstractions, one needs to encode both the static structure of such abstractions and their dynamic structure during computation. One successful approach to such an encoding, generally called *λ-tree syntax* [Miller 2000] (a proof search approach to *higher-order abstract syntax* [Pfenning and Elliott 1988]), uses λ-terms to encode the static structure of abstractions and universally quantified judgments to encode their dynamic structure. Consider in more detail the role of the universal quantifier and eigenvariables in proof search and the specification of computations.

Authors' address: Dale Miller, Laboratoire d'Informatique (LIX), École polytechnique, Palaiseau 91128 CEDEX, France, e-mail: dale@lix.polytechnique.fr. Alwen Tiu, INRIA Lorraine, 615 Rue du Jardin Botanique, 54602 Villers-Les-Nancy, France, e-mail: Alwen.Tiu@loria.fr.

There are, of course, at least a few ways to prove the universally quantified formula $\forall_\tau x.B$. The *extensional* approach attempts to prove $B[t/x]$ for all (closed) terms t of type τ. This rule might involve an infinite number of premises if the domain of the type τ is infinite. If the type τ is defined inductively, a proof by *induction* can replace the need for infinite premises with finite premises (the *base* cases and *inductive* cases) but with the need to discover invariants. Another more *intensional* approach, however, involves introducing a new variable, say, $c : \tau$, that has not been introduced before in the proof, and attempting to prove the formula $B[c/x]$ instead. In natural deduction and sequent calculus proofs, such new variables are called *eigenvariables*, and they are used to prove universally quantified formulas *generically*.

In Gentzen's original presentation of the sequent calculus [Gentzen 1969], eigenvariables are immutable during proof search: once an eigenvariable is introduced (reading proofs bottom-up), it is not used as a site for substitution. In other words, eigenvariables did not vary during proof search: rather they acted more as new, scoped constants. As we now illustrate, it is the proof of cut-elimination that generally requires substitutions for eigenvariables: Assume that the sequent $\Gamma \longrightarrow \forall x.B$ is proved using the introduction of $\forall$ on the right from the premise $\Gamma \longrightarrow B[c/x]$, where c is an eigenvariable and $\Pi(c)$ is a proof of this premise. Similarly, assume that the sequent $\Gamma', \forall xB \longrightarrow C$ is proved using the introduction of $\forall$ on the left from the premise $\Gamma', B[t/x] \longrightarrow C$, where t is some term. To reduce the rank of the cut formula $\forall x.B$ between the sequents $\Gamma \longrightarrow \forall x.B$ and $\Gamma', \forall xB \longrightarrow C$, the eigenvariable c in the sequent calculus proof $\Pi(c)$ must be substituted by t to yield a proof $\Pi(t)$ of $\Gamma \longrightarrow B[t/x]$: in this way, the cut-formula is now the smaller formula $B[t/x]$. In Gentzen, eigenvariables are sites for substitution only in the meta-theory of proofs and not in proofs themselves.

Notice that if the proof of cut-elimination is structured as above, then the intensional interpretation of the universal quantifier entails the extensional interpretation: Given a proof of $\Gamma \longrightarrow \forall x.B$ with premise $\Gamma \longrightarrow B[c/x]$ proved by $\Pi(c)$, then instantiating c with t yields a proof $\Pi(t)$ for $\Gamma \longrightarrow B[t/x]$.

Recent years have witnessed two different developments in the role of eigenvariables in the specification of computation systems.

Eigenvariables as new, scoped constants. Focusing on their *intensional* nature and guarantee of newness in proof search, eigenvariables have been used to encode name restrictions in the π-calculus [Miller 1993], nonces in security protocols [Cervesato et al. 1999], reference locations in imperative programming [Pfenning and Rohwedder 1992; Chirimar 1995; Cervesato and Pfenning 1996; Miller 1996], new assumptions in encodings of natural deduction or sequent calculi [Felty and Miller 1988], and constructors hidden within abstract data-types [Miller 1989]. Eigenvariables also provide an essential aspect of recursive programming with data encoded using λ-tree syntax [Miller 2000]: to move recursively through syntax that is an outermost binder, instantiate the bound variable with an eigenvariable: that is, replace the term-level bound variable with a proof-level bound variable.

Eigenvariables as variables to instantiate. Computation in logic programming can be seen as a (restricted) form of cut-free proof search. Cut and cut-elimination

$$\frac{\Sigma; (\sigma, y : \tau) \triangleright B[y/x], \Gamma \longrightarrow \mathcal{C}}{\Sigma; \sigma \triangleright \nabla_\tau x.B, \Gamma \longrightarrow \mathcal{C}}\ \nabla\mathcal{L} \qquad \frac{\Sigma; \Gamma \longrightarrow (\sigma, y : \tau) \triangleright B[y/x]}{\Sigma; \Gamma \longrightarrow \sigma \triangleright \nabla_\tau x.B}\ \nabla\mathcal{R}$$

Fig. 1. Rules for the ∇-quantifier.

can then be used to reason directly about computation: for example, if A has a cut-free proof (that is, it can be computed) and we know that $A \supset B$ can be proved (possibly with cuts), cut-elimination allows us to conclude that B has a cut-free proof (that is, it can be computed). As we mentioned above, such direct reasoning on logic specification involves instantiations of eigenvariables. Similarly, focusing on their *extensional* nature guaranteed by cut-elimination, enrichments to the sequent calculus have been proposed by [Hallnäs and Schroeder-Heister 1991; Schroeder-Heister 1992; Girard 1992; McDowell and Miller 2000] in which eigenvariables are intended as variables to be substituted during proof search. This enrichment to proof theory (discussed here in Section 4) holds promise for providing proof systems for the direct reasoning about logic specifications (see, for example, the above mentioned papers as well as [McDowell and Miller 2002; McDowell et al. 2003]).

These two approaches are, however, at odds with each other. Consider, for example, the problem of representing restriction of names or nonces using $\forall$ quantification. (The following example can be dualized in the event that a logical specification uses $\exists$ quantification instead of $\forall$, as in, for example, [Cervesato et al. 1999]). A cut-free proof of the formula $\forall x \forall y.P(x, y)$ proceeds by introducing two new and distinct "names" or "nonces" whereas a proof of the expression $\forall z.P(z, z)$ involves just one such item. Of course, in logic, the implication $\forall x \forall y.P(x, y) \supset \forall z.P(z, z)$ holds, so if there is a proof with the two different names, there must be one with those names identified (via cut-elimination), and this is unlikely to be the intended meaning of such quantification. This suggests that when using eigenvariables solely to provide scope and newness to names, one cannot reason directly with the specification using cut-elimination, the centerpiece of proof theory.

Another setting where the difference between the extensional and intensional approaches to universal quantification occurs is when we consider having an assumption that is universally quantified. In Gentzen's sequent system, having $\forall_\tau x.Bx$ as an assumption (that is, on the left of the sequent arrow) is essentially equated to having instead all instances Bt for terms t of type τ. There are cases (one is considered in more detail in Section 6) where we would like to make inferences from an assumption of the form $\forall_\tau x.Bx$ that holds independent of the set of its instances: the fact that such a statement could hold generically (intensionally) provides us with information stronger than examining all instances of it. This is particularly true in many intuitionistic settings where the domain of the type τ might be empty or at least not known to be inhabited.

2. THE ∇-QUANTIFIER

One approach to solving this problem of forcing one quantifier, the $\forall$-quantifier, to have two behaviors that are not entirely compatible, is to extend traditional logics (intuitionistic logic in our case) with a new quantifier. In this paper, we do this by adding the ∇-quantifier: its role will be to provide for variables to be abstracted

with local scope. The syntax of the formula $\nabla_\tau x.B$ is like that for the universal and existential quantifiers. Following Church's Simple Theory of Types [Church 1940], formulas are given the type o, and for all types τ not containing o, ∇_τ is a constant of type $(\tau \to o) \to o$. The expression $\nabla_\tau \lambda x.B$ is usually abbreviated as simply $\nabla_\tau x.B$ or as $\nabla x.B$ if the type information is either simple to infer or not important.

Intuitionistic sequents without the need to account for ∇ are structures of the form

$$\Sigma; B_1, \ldots, B_n \longrightarrow B_0.$$

Here, Σ is a *signature* containing the list of all (explicitly typed) eigenvariables of the sequent. Depending on the domains of applications, there may be an additional fixed set of non-logical constants given. But since this set of non-logical constants does not vary in proof constructions, we choose not to put it explicitly in sequents. The judgment $\Sigma \vdash t : \tau$ means that t is a simply typed λ-term of type τ in which there may appear the (fixed) non-logical constants as well as those eigenvariables in Σ. In the displayed sequent above, $n \geq 0$ and $B_0, B_1, \ldots, B_n$ are formulas (*i.e.*, terms of type o), all of whose free variables are in Σ. Informally, this sequent means that for every substitution θ that maps variables $x : \tau \in \Sigma$ to terms of type τ, if $B_i\theta$ holds for all $i = 1, \ldots, n$, then $B_0\theta$ holds.

To account for the ∇ quantifier, we introduce into sequents a new element of context. Sequents will now have one *global* signature (containing the sequent's eigenvariables) and several *local* signatures, used to scope local variables. More generally, sequents have the structure

$$\Sigma; \sigma_1 \triangleright B_1, \ldots, \sigma_n \triangleright B_n \longrightarrow \sigma_0 \triangleright B_0.$$

Here, $\sigma_0, \ldots, \sigma_n$ are signatures and the other items are as above. We shall consider sequents to be binding structures in the sense that the signatures, both the global and local ones, are abstractions over their respective scopes. The variables in Σ and σ_i will admit α-conversion by systematically changing the names of variables in signatures as well as those in their scope, following the usual convention of the λ-calculus. In general, however, we will assume that the local signatures σ_i contain names different than those in the global signature Σ. The expression $\sigma \triangleright B$ is called a *generic judgment* or simply *judgment*. Equality between judgments follows from the notion of equality of λ-terms, that is, two judgments $\bar{x} \triangleright B$ and $\bar{y} \triangleright C$ are equal if and only if $\lambda\bar{x}.B =_{\beta\eta} \lambda\bar{y}.C$. Since equality between terms, or between judgments, in our logic is always modulo $\beta\eta$, we shall often omit the subscripts $\beta\eta$ when writing the equality symbol. We use script letters $\mathcal{A}$, $\mathcal{B}$, etc. to denote judgments. We write simply B instead of $\sigma \triangleright B$ if the signature σ is empty.

The introduction rules for ∇ are given in Figure 1. The variable y must be new to the variables in σ and Σ (implicit in the definition of sequent). The expression $(\sigma, y : \tau)$ denotes the signature containing the type declaration $y : \tau$ appended to the end of the list σ. Notice that since the left and right rules are essentially the same, this quantifier will be self dual: that is, $\neg\nabla xBx$ is equivalent to $\nabla x\neg Bx$.

$$\frac{}{\Sigma;\sigma \triangleright B,\Gamma \longrightarrow \sigma \triangleright B}\ init \qquad \frac{\Sigma;\Delta \longrightarrow \mathcal{B} \qquad \Sigma;\mathcal{B},\Gamma \longrightarrow \mathcal{C}}{\Sigma;\Delta,\Gamma \longrightarrow \mathcal{C}}\ cut$$

$$\frac{\Sigma;\mathcal{B},\mathcal{B},\Gamma \longrightarrow \mathcal{C}}{\Sigma;\mathcal{B},\Gamma \longrightarrow \mathcal{C}}\ c\mathcal{L} \qquad \frac{\Sigma;\Gamma \longrightarrow \mathcal{C}}{\Sigma;\mathcal{B},\Gamma \longrightarrow \mathcal{C}}\ w\mathcal{L}$$

$$\frac{}{\Sigma;\sigma \triangleright \bot,\Gamma \longrightarrow \mathcal{B}}\ \bot\mathcal{L} \qquad \frac{}{\Sigma;\Gamma \longrightarrow \sigma \triangleright \top}\ \top\mathcal{R}$$

$$\frac{\Sigma;\sigma \triangleright B,\Gamma \longrightarrow \mathcal{D}}{\Sigma;\sigma \triangleright B \wedge C,\Gamma \longrightarrow \mathcal{D}}\ \wedge\mathcal{L} \qquad \frac{\Sigma;\sigma \triangleright C,\Gamma \longrightarrow \mathcal{D}}{\Sigma;\sigma \triangleright B \wedge C,\Gamma \longrightarrow \mathcal{D}}\ \wedge\mathcal{L}$$

$$\frac{\Sigma;\Gamma \longrightarrow \sigma \triangleright B \qquad \Sigma;\Gamma \longrightarrow \sigma \triangleright C}{\Sigma;\Gamma \longrightarrow \sigma \triangleright B \wedge C}\ \wedge\mathcal{R} \qquad \frac{\Sigma;\sigma \triangleright B,\Gamma \longrightarrow \mathcal{D} \qquad \Sigma;\sigma \triangleright C,\Gamma \longrightarrow \mathcal{D}}{\Sigma;\sigma \triangleright B \vee C,\Gamma \longrightarrow \mathcal{D}}\ \vee\mathcal{L}$$

$$\frac{\Sigma;\Gamma \longrightarrow \sigma \triangleright B}{\Sigma;\Gamma \longrightarrow \sigma \triangleright B \vee C}\ \vee\mathcal{R} \qquad \frac{\Sigma;\Gamma \longrightarrow \sigma \triangleright C}{\Sigma;\Gamma \longrightarrow \sigma \triangleright B \vee C}\ \vee\mathcal{R}$$

$$\frac{\Sigma;\Gamma \longrightarrow \sigma \triangleright B \qquad \Sigma;\sigma \triangleright C,\Gamma \longrightarrow \mathcal{D}}{\Sigma;\sigma \triangleright B \supset C,\Gamma \longrightarrow \mathcal{D}}\ \supset\mathcal{L} \qquad \frac{\Sigma;\sigma \triangleright B,\Gamma \longrightarrow \sigma \triangleright C}{\Sigma;\Gamma \longrightarrow \sigma \triangleright B \supset C}\ \supset\mathcal{R}$$

$$\frac{\Sigma,\sigma \vdash t : \tau \qquad \Sigma;\sigma \triangleright B[t/x],\Gamma \longrightarrow \mathcal{C}}{\Sigma;\sigma \triangleright \forall_\tau x.B,\Gamma \longrightarrow \mathcal{C}}\ \forall\mathcal{L} \qquad \frac{\Sigma,h;\Gamma \longrightarrow \sigma \triangleright B[(h\ \sigma)/x]}{\Sigma;\Gamma \longrightarrow \sigma \triangleright \forall x.B}\ \forall\mathcal{R}$$

$$\frac{\Sigma,h;\sigma \triangleright B[(h\ \sigma)/x],\Gamma \longrightarrow \mathcal{C}}{\Sigma;\sigma \triangleright \exists x.B,\Gamma \longrightarrow \mathcal{C}}\ \exists\mathcal{L} \qquad \frac{\Sigma,\sigma \vdash t : \tau \qquad \Sigma;\Gamma \longrightarrow \sigma \triangleright B[t/x]}{\Sigma;\Gamma \longrightarrow \sigma \triangleright \exists_\tau x.B}\ \exists\mathcal{R}$$

Fig. 2. The intuitionistic rules of $FO\lambda$.

3. AN INTUITIONISTIC LOGIC WITH ∇

We now consider Gentzen's LJ calculus [Gentzen 1969] with the addition of global and local signatures and ∇. Besides this new quantifier, the other logical connectives are $\bot$, $\top$, $\wedge$, $\vee$, $\supset$, $\forall_\tau$, and $\exists_\tau$ (again, the type τ does not contain o): their inference rules are given in Figure 2. Notice that no inference rule in Figure 2 requires non-empty local signatures: as a result, if all the local signatures in sequents in a derivation built from those rules are set to empty, the resulting derivation is a standard derivation in intuitionistic logic.

The interaction between the global and local signatures and the universal and existential quantifiers needs some explanations. In the rule for $\forall\mathcal{L}$ (and, dually, for $\exists\mathcal{R}$), the quantifier appears in the scope of the global signature Σ and the local signature σ. This quantifier can be instantiated (reading the rule bottom-up) with a term built from variables in both of these signatures. Similarly, in the rule for $\forall\mathcal{R}$ (and, dually, for $\exists\mathcal{L}$), the quantifier appears in the scope of the global signature Σ and the local signature σ. This quantifier can be instantiated (reading the rule bottom-up) with an eigenvariable whose intended range is over all terms built from variables in Σ and σ. Since, however, the eigenvariable h is stored in the global scope, its dependency on σ would be forgotten unless we employ some particular encoding technique. For this purpose, we use *raising* [Miller 1992]: to denote a variable of type τ_0 that can range over Σ and over the variables in $\sigma = (x_1 : \tau_1, \ldots, x_n : \tau_n)$ $(n \geq 0)$, we can use instead the term $(hx_1 \ldots x_n)$ where the variable h ranges over Σ only (the dependency on σ can be forgotten). Of course, the type of h will be $\tau_1 \to \cdots \to \tau_n \to \tau_0$ instead of simply τ_0. In the

$$\begin{array}{ll} \nabla x \neg Bx \equiv \neg \nabla x Bx & \nabla x(Bx \wedge Cx) \equiv \nabla x Bx \wedge \nabla x Cx \\ \nabla x(Bx \vee Cx) \equiv \nabla x Bx \vee \nabla x Cx & \nabla x(Bx \supset Cx) \equiv \nabla x Bx \supset \nabla x Cx \\ \nabla x \forall y Bxy \equiv \forall h \nabla x Bx(hx) & \nabla x \exists y Bxy \equiv \exists h \nabla x Bx(hx) \\ \nabla x \forall y Bxy \supset \forall y \nabla x Bxy & \nabla x. \top \equiv \top, \quad \nabla x. \bot \equiv \bot \end{array}$$

Fig. 3. Some theorems of $FO\lambda^{\nabla}$.

$$\begin{array}{lll} \nabla x \nabla y Bxy \supset \nabla z Bzz & (1) & \nabla x Bx \supset \exists x Bx & (2) \\ \forall y \nabla x Bxy \supset \nabla x \forall y Bxy & (3) & \forall x Bx \supset \nabla x Bx & (4) \\ \exists x Bx \supset \nabla x Bx & (5) & B \supset \nabla x B & (6) \\ \nabla x Bx \supset \forall x Bx & (7) & \nabla x \nabla y Bxy \supset \nabla y \nabla x Bxy & (8) \end{array}$$

Fig. 4. Some non-theorems of $FO\lambda^{\nabla}$. Here, B denotes some uninterpreted formula or abstraction over a formula. In (6), x is not free in B.

inference rules of Figure 2, we write $(h\sigma)$ to denote $(hx_1 \dots x_n)$.

For the sake of consistency with a naming convention from the papers [McDowell 1997; McDowell and Miller 2000], we shall refer to the inference system defined with just the rules in Figure 2 as $FO\lambda$ (mnemonic for a "first-order logic for λ-expressions"). The proof system resulting from the addition of the rules for ∇ (Figure 1) is called $FO\lambda^{\nabla}$.

Figure 3 lists some theorems of $FO\lambda^{\nabla}$ involving ∇. In general, we use $\neg C$ to abbreviate $C \supset \bot$ and $B \equiv C$ to abbreviate $(B \supset C) \wedge (C \supset B)$. As a result of these equivalences, ∇ can alway be given atomic scope within formulas (with the simple cost of raising the quantified variables in its scope). Figure 4 lists some non-theorems of $FO\lambda^{\nabla}$ involving ∇. In the next section we will extend the core logic with a proof theoretic notion of definition. In this extension, we will be able to prove certain instances of the last three of these non-theorems (see the end of Section 7.2). The first five will not be provable in the extension, and it seems important that they are not provable. For example, in the non-theorem (4), $\forall_\tau B \supset \nabla_\tau B$, if τ is empty then the statement would not be expected to hold and hence we do not accept it in the core logic.

4. INTRODUCTION RULES FOR DEFINITIONS

Introduction rules are, generally, restricted to logical connectives and quantifiers. The recent development of a proof theoretic notion of *definitions* [Hallnäs and Schroeder-Heister 1991; Schroeder-Heister 1992; Girard 1992; McDowell and Miller 2000] provides left and right introduction rules also for non-logical predicate symbols, provided that they are "defined" appropriately. Given certain restrictions on the syntax of definitions, a logic with such definition introduction rules can enjoy cut-elimination. In this section, we take the treatment of definitions from [McDowell 1997; McDowell and Miller 2000] and extend it to handle the extension of local signatures.

Definition 4.1. A *definitional clause* is written $\forall \bar{x}.p\,\bar{t} \stackrel{\triangle}{=} B$, where p is a predicate constant, every free variable of the formula B is also free in at least one term in the list $\bar{t}$ of terms, and all variables free in $p\,\bar{t}$ are contained in the list $\bar{x}$ of variables. The atomic formula $p\,\bar{t}$ is called the *head* of the clause, and the formula B is called

the *body*. The symbol $\triangleq$ is used simply to indicate a definitional clause: it is not a logical connective. A *definition* is a (perhaps infinite) set of definitional clauses. The same predicate may occur in the head of multiple clauses of a definition: it is best to think of a definition as a mutually recursive definition of the predicates in the heads of the clauses.

In this paper we shall assume that all predicate constants are defined since this addresses the applications we wish to illustrate. See [Schroeder-Heister 1994] for other approaches to treating undefined predicates. We shall also use the convention that when displaying definition clauses, tokens with an initial uppercase letter will be assumed to be universally quantified with outermost scope.

Although predicates are defined via mutual recursion, circularities through implications (negations) must be avoided. To do this, we stratify definitions by first associating to each predicate p a natural number $\mathrm{lvl}(p)$, the *level* of p. The notion of level is generalized to formulas as follows.

Definition 4.2. Given a formula B, its *level* $\mathrm{lvl}(B)$ is defined as follows:

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

For every definitional clause $\forall \bar{x}.p\,\bar{t} \triangleq B$, we shall require that $\mathrm{lvl}(B) \leq \mathrm{lvl}(p)$. This requirement allows us to prove cut-elimination for intuitionistic logic extended with definitions (see [McDowell and Miller 2000] and Section 7).

Definition rules involve the use of substitutions. We recall some basic definitions related to substitutions. A *substitution* θ is a mapping (with application written in postfix notation) from variables to terms, such that the set $\{x \mid x\theta \neq x\}$ is finite. Substitutions must preserve types, that is, given a substitution θ and a variable $x : \tau$, the result of applying θ to x, $x\theta$, is of type τ. Although substitutions are extended to mappings from terms to terms, generic judgments to generic judgments, etc., when we refer to the domain and the range of a substitution, we refer to those sets defined on this most basic function. A substitution is extended to a function from terms to terms in the usual fashion. Composition of substitutions is defined as $x(\theta \circ \sigma) = (x\theta)\sigma$, for all variable x. Two substitutions θ and σ are considered equal if for all variables x, $x\sigma =_{\beta\eta} x\theta$ (equal modulo $\beta\eta$-conversion). The empty substitution is written as ϵ. The application of a substitution θ to a generic judgment $x_1, \ldots, x_n \triangleright B$, written as $(x_1, \ldots, x_n \triangleright B)\theta$, is $y_1, \ldots, y_n \triangleright B'$, if $(\lambda x_1 \ldots \lambda x_n.B)\theta$ is equal (modulo λ-conversion) to $\lambda y_1 \ldots \lambda y_n.B'$. If Γ is a multiset of generic judgments, then $\Gamma\theta$ is the multiset $\{J\theta \mid J \in \Gamma\}$. Finally, if Σ is a signature then $\Sigma\theta$ is the signature that results from removing from Σ the variables in the domain of θ and adding the variables that are free in the range of θ.

The introduction of a defined atom may take place in the context of a local signature. To account for this, we again use the technique of raising to code this dependency by introducing the notion of "raised" definition clause.

$$\frac{\Sigma; \Gamma \longrightarrow \mathcal{B}\theta}{\Sigma; \Gamma \longrightarrow \mathcal{A}}\ def\mathcal{R},\ \text{where } \mathrm{dfn}(\epsilon, \mathcal{A}, \theta, \mathcal{B}) \qquad \frac{\{\Sigma\rho; \mathcal{B}\theta, \Gamma\rho \longrightarrow \mathcal{C}\rho \mid \mathrm{dfn}(\rho, \mathcal{A}, \theta, \mathcal{B})\}}{\Sigma; \mathcal{A}, \Gamma \longrightarrow \mathcal{C}}\ def\mathcal{L}$$

Fig. 5. The definition introduction rules

Definition 4.3. Let $\forall_{\tau_1} x_1 \ldots \forall_{\tau_n} x_n.H \triangleq B$ be a definition clause and consider a list of variables $y_1, \ldots, y_m$ of types $\alpha_1, \ldots, \alpha_m$, respectively. A *raised definition clause* with respect to the signature $\{y_1 : \alpha_1, \ldots, y_m : \alpha_m\}$ is defined as

$$\forall h_1 \ldots \forall h_n.\bar{y} \triangleright H\theta \triangleq \bar{y} \triangleright B\theta$$

where θ is the substitution $[(h_1\ \bar{y})/x_1, \ldots, (h_n\ \bar{y})/x_n]$ and h_i, for every $i \in \{1, \ldots, n\}$, is of type $\alpha_1 \to \ldots \to \alpha_m \to \tau_i$.

Raised definition clauses can be seen as definitions for *atomic judgments* (i.e., judgments which contain no occurrences of logical constants) and a definition clause is just a concise way of representing a family of definition clauses for atomic judgments. Raising a definition in this manner is similar to $\forall$-lifting [Paulson 1989; Miller 1992].

The following relation is useful for presenting the introduction rules for defined atomic judgments.

Definition 4.4. The four-place relation $\mathrm{dfn}(\rho, \mathcal{A}, \theta, \mathcal{B})$ holds for the atomic judgment $\mathcal{A}$, the judgment $\mathcal{B}$, and the substitutions ρ and θ if there is a raised clause $\forall h_1 \ldots \forall h_n.\mathcal{H} \triangleq \mathcal{B}$ in the given definition such that $\mathcal{A}\rho = \mathcal{H}\theta$.

Obviously, for the relation $\mathrm{dfn}(\rho, \mathcal{A}, \theta, \mathcal{B})$ to hold, given a raised clause $\mathcal{H} \triangleq \mathcal{B}$, the judgments $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{H}$ must share the same local signature (up to α-conversion).

The right and left rules for atoms are given in Figure 5. Specifying a set of sequents as the premise should be understood to mean that each sequent in the set is a premise of the rule. Notice that in the $def\mathcal{L}$ rule, the free variables of the conclusion can be instantiated in the premises. In particular, a variable in Σ could possibly be replaced by several new variables.

These rules for definitions add considerable expressive power to intuitionistic logic. For example, $def\mathcal{R}$ is essentially the *backchaining* rule on closed atoms found in logic programming, while $def\mathcal{L}$ is essentially a case analysis on how an atom can be proved and can be used to establish *finite failure*. Together, these two rules can be used to encode simulation and bisimulation in certain abstract transition systems [McDowell et al. 2003]. Other uses involve reasoning about computational systems [McDowell and Miller 2002].

The rule $def\mathcal{L}$ may have an infinite number of premises since the unification of simply typed λ-terms may return infinitely many unifiers and since the domains of the substitutions ρ and θ may include variables which are not free in $\mathcal{A}$ and $\mathcal{B}$. The latter may introduce "noise" into proof search since they can insert eigenvariables of any types in the premises. The presence of this noise does not, however, affect provability. To see why, consider the definition $eq\ X\ X \triangleq \top$ where $eq : i \to i \to o$ for some base type i, and the sequent $\{z : i\}; eq\ z\ z \longrightarrow \exists_\alpha y.\top$ where α is some base type different from i. Assuming no other constants, this sequent should not be provable since there is no closed term of type α. One can, however, introduce

(during proof search) new eigenvariables of type α via *def*$\mathcal{L}$. By applying *def*$\mathcal{L}$, we can substitute any term into z and we are allowed to introduce new variables. In particular, among these premises are the premises $\{f : \alpha \to i, x : \alpha\}; \longrightarrow \exists_\alpha y.\top$, obtained via $[(f\,x)/z]$, and $\{z : i\}; \longrightarrow \exists_\alpha.\top$, via the empty substitution. The first sequent is provable, using $\exists\mathcal{R}$ with x, but the second sequent is not and hence the original sequent is not provable.

It is possible to have a finite number of premises and reduce the noises in *def*$\mathcal{L}$, provided that we restrict the definitions to have only a finite number of clauses and to restrict the use of *def*$\mathcal{L}$ to those judgments $\mathcal{A}$ such that for every raised definition clause there is a finite, complete set of unifiers (CSU) [Huet 1975] of $\mathcal{A}$ and the head of the clause. Then the following inference rules can be shown interadmissible with *def*$\mathcal{L}$:

$$\frac{\{\Sigma\theta; \mathcal{B}\theta, \Gamma\theta \longrightarrow \mathcal{C}\theta \mid \theta \in CSU(\mathcal{A}, \mathcal{H}) \text{ for some clause } \forall\bar{h}[\mathcal{H} \stackrel{\triangle}{=} \mathcal{B}]\}}{\Sigma; \mathcal{A}, \Gamma \longrightarrow \mathcal{C}}\ \textit{def}\mathcal{L}_{csu}.$$

This rule is originally due to [Eriksson 1991] and is also used in [McDowell and Miller 2000]. The proof of its interadmissibility with *def*$\mathcal{L}$ follows the same outline as the one in [McDowell and Miller 2000]. The meta-theoretic analysis of definitions (see Section 7) is more naturally addressed using *def*$\mathcal{L}$ while the presentation of examples (see Sections 5 and 6) is more natural using *def*$\mathcal{L}_{csu}$.

The proof system that arises from adding together the inference rules in Figures 2 and 5 is called $FO\lambda^{\Delta}$. If we add to $FO\lambda^{\Delta}$ the rules in Figure 1, the resulting proof system is called $FO\lambda^{\Delta\nabla}$ (pronounced "fold nabla"). It is this logic that will involve us for the remainder of this paper.

Definition clauses that are similar to Horn clauses are important in our investigation. In particular, an hc-*goal* (named for Horn clauses) is a formula built from the base set of logic connectives $\top$, $\wedge$, $\vee$, and $\exists$. An $\mathsf{hc}^{\forall}$-*goal* is a formula built from these connectives and $\forall$; an hc^{∇}-*goal* is a formula built from the base set and ∇; and an $\mathsf{hc}^{\forall\nabla}$-*goal* is a formula admitting the base set as well as both $\forall$ and ∇. A definition is an hc-*definition* (resp., $\mathsf{hc}^{\forall}$-*definition*, hc^{∇}-*definition*, and $\mathsf{hc}^{\forall\nabla}$-*definition*) if the body of all of its clauses are hc-*goals* (resp., $\mathsf{hc}^{\forall}$-*goals*, hc^{∇}-*goals*, and $\mathsf{hc}^{\forall\nabla}$-*goals*). Notice that all of these kinds of definitions are trivially stratifiable. Numerous interesting computer science motivated specifications are examples of $\mathsf{hc}^{\forall}$-definitions: we consider in more detail two such examples in Sections 5 and 6.

5. EXAMPLE: THE π-CALCULUS

Operational semantics of specification languages or programming languages are often given using inference rules, following the small-step approach (a.k.a., structured operational semantic) or big-step approach (a.k.a. natural semantics). Frequently, the proper specification of such semantics includes abstractions over names that are used for such things as nonces in security protocols [Cervesato et al. 1999], locations for reference cells [Chirimar 1995; Miller 1996], or new communication channels [Milner et al. 1992]. One declarative way to capture these features in the inference rule setting is to employ scoped (eigen)variables. Given the logic $FO\lambda^{\nabla}$, we now have the ability to scope variables within sequents either globally via $\forall$ or locally via ∇. We illustrate these choices with a specification of the π-calculus.

$$\frac{}{\tau.P \xrightarrow{\tau} P}\tau \qquad \frac{P \xrightarrow{A} Q}{[X = X]P \xrightarrow{A} Q}\text{match} \qquad \frac{P \xrightarrow{H} M}{[X = X]P \xrightarrow{H} M}\text{match}$$

$$\frac{P \xrightarrow{A} R}{P + Q \xrightarrow{A} R}\text{sum} \qquad \frac{Q \xrightarrow{A} R}{P + Q \xrightarrow{A} R}\text{sum} \qquad \frac{P \xrightarrow{H} M}{P + Q \xrightarrow{H} M}\text{sum} \qquad \frac{Q \xrightarrow{H} N}{P + Q \xrightarrow{H} N}\text{sum}$$

$$\frac{P \xrightarrow{A} P'}{P \mid Q \xrightarrow{A} P' \mid Q}\text{par} \qquad \frac{Q \xrightarrow{A} Q'}{P \mid Q \xrightarrow{A} P \mid Q'}\text{par}$$

$$\frac{P \xrightarrow{H} M}{P \mid Q \xrightarrow{H} \lambda n(Mn \mid Q)}\text{par} \qquad \frac{Q \xrightarrow{H} N}{P \mid Q \xrightarrow{H} \lambda n(P \mid Nn)}\text{par}$$

$$\frac{\nabla n(Mn \xrightarrow{A} M'n)}{\nu n.Mn \xrightarrow{A} \nu n.M'n}\text{res} \qquad \frac{\nabla n(Mn \xrightarrow{H} Sn)}{\nu n.Mn \xrightarrow{H} \lambda m\ \nu n.(Snm)}\text{res} \qquad \frac{\nabla y(My \xrightarrow{\uparrow Xy} M'y)}{\nu y.My \xrightarrow{\uparrow X} M'}\text{ open}$$

$$\frac{}{out\ X\ Y\ P \xrightarrow{\uparrow XY} P}\text{ out} \qquad \frac{P \xrightarrow{\downarrow X} M \qquad Q \xrightarrow{\uparrow X} N}{P \mid Q \xrightarrow{\tau} \nu n.(Mn \mid Nn)}\text{ close} \qquad \frac{P \xrightarrow{\uparrow X} M \qquad Q \xrightarrow{\downarrow X} N}{P \mid Q \xrightarrow{\tau} \nu n.(Mn \mid Nn)}\text{ close}$$

$$\frac{}{in\ X\ M \xrightarrow{\downarrow X} M}\text{ in} \qquad \frac{P \xrightarrow{\downarrow X} M \qquad Q \xrightarrow{\uparrow XY} Q'}{P \mid Q \xrightarrow{\tau} (MY) \mid Q'}\text{ com} \qquad \frac{P \xrightarrow{\uparrow XY} P' \qquad Q \xrightarrow{\downarrow X} N}{P \mid Q \xrightarrow{\tau} P' \mid (NY)}\text{ com}$$

Fig. 6. The rules for the (late) π-calculus.

Consider encoding π-calculus [Milner et al. 1992] using λ-tree syntax following [Miller and Palamidessi 1999; Miller and Tiu 2002]. Since we are focused here on abstractions in syntax, we shall deal with only *finite* π-calculus expression; that is, expressions without ! or defined constants. Extending this work to infinite process expressions should be possible by adding induction (as in [McDowell et al. 2003]) or co-induction (as in [Momigliano and Tiu 2003; Tiu 2004b]) to our proof system. We shall require three primitive syntactic categories: n for channels, p for processes, and a for actions. The output prefix is the constructor *out* of type $n \to n \to p \to p$ and the input prefix is the constructor *in* of type $n \to (n \to p) \to p$: the π-calculus expressions $\bar{x}y.P$ and $x(y).P$ are represented as $(out\ x\ y\ P)$ and $(in\ x\ \lambda y.P)$, respectively. We use $\mid$ and $+$, both of type $p \to p \to p$ and written as infix, to denote parallel composition and summation, and ν of type $(n \to p) \to p$ to denote restriction. The π-calculus expression $(x)P$ will be encoded as $\nu\lambda n.P$, which itself is abbreviated as simply $\nu x.P$. The match operator, $[\cdot = \cdot]\cdot$ is of type $n \to n \to p \to p$. When τ is written as a prefix, it has type $p \to p$. When τ is written as an action, it has type a. The symbols $\downarrow$ and $\uparrow$, both of type $n \to n \to a$, denote the input and output actions, respectively, on a named channel with a named value: e.g., $\downarrow xy$ denotes the action of inputing y on channel x.

We use two predicates to encode the one-step transition semantics for the π-calculus. The predicate $\cdot \xrightarrow{\cdot} \cdot$ of type $p \to a \to p \to o$ encodes transitions

$$
\begin{array}{lrcl}
\text{(res)} & \nu n.Mn \xrightarrow{A} \nu n.M'n & \triangleq & \nabla n(Mn \xrightarrow{A} M'n) \\
\text{(res)} & \nu y.My \xrightarrow{\uparrow X} M' & \triangleq & \nabla y(My \xrightarrow{\uparrow Xy} M'y) \\
\text{(in)} & in\ X\ M \xrightarrow{\downarrow X} M & \triangleq & \top \\
\text{(com)} & P \mid Q \xrightarrow{\tau} P' \mid (N\ Y) & \triangleq & \exists x.P \xrightarrow{\uparrow xY} P' \wedge Q \xrightarrow{\downarrow x} N
\end{array}
$$

Fig. 7. Corresponding definition clauses

involving free values and the predicate $\cdot \stackrel{\cdot}{\longrightarrow} \cdot$ of type $p \to (n \to a) \to (n \to p) \to o$ encodes transitions involving bound values. Figure 6 (taken from [Miller and Tiu 2002]) contains the inference rules specifying the late version of the transitions for the π-calculus [Milner et al. 1992]. In these rules, capital letters (possibly primed) are used to denote schema variables for inference rules. We adopt the following typing convention for these scheme variables: $X, Y : n$, $A : a$, $H : n \to a$, $P, Q, R, P', Q' : p$, $M, N, M', N' : n \to p$, and $S : n \to n \to p$. These inference rules can trivially be written as definition clauses: a few such clauses are presented in Figure 7. Here, schema variables are universally quantified (implicitly) at the top-level of such clauses. Notice that the complicated side conditions in the original specification of π-calculus are no longer present, as they are now treated directly and declaratively by the meta-logic. For example, the side condition that $x \neq y$ in the open rule is implicit, since x is outside the scope of y and therefore cannot be instantiated with y.

To illustrate the expressiveness that the ∇ quantifier adds to logic, consider the following presentations of the transition system for the π-calculus. Let $\mathcal{L}$ be the complete definition for the one step transitions for the π-calculus. Clearly, $\mathcal{L}$ is an hc^{∇}-definition. Let $\mathcal{L}'$ be the result of replacing all occurrences of ∇ in $\mathcal{L}$ with $\forall$. Furthermore, let $\mathcal{L}''$ be the result of replacing all occurrences of the symbol $\triangleq$ in the definition clauses of $\mathcal{L}'$ by reverse implication: thus, $\mathcal{L}''$ is a set of formulas and is not a definition. Finally, assume that we are only interested in computing the one-step transitions of the late π-calculus, that is, proving atomic formulas such as $P \xrightarrow{A} P'$ or $P \xrightarrow{H} P'$ (let B range over such atomic formulas).

As we shall see in Section 7.2, when we restrict ourselves to Horn definitions (no implications and, hence, no negations in the body of definitions), then it is not possible to distinguish between uses of ∇ and $\forall$ in the body of clauses. In particular, Proposition 7.10 implies that $\cdot;\cdot \longrightarrow B$ is provable in $FO\lambda^{\Delta\nabla}$ using definition $\mathcal{L}$ if and only if $\cdot;\cdot \longrightarrow B$ is provable in $FO\lambda^{\Delta}$ using definition $\mathcal{L}'$. Furthermore, a cut-free proof of $\cdot;\cdot \longrightarrow B$ in $FO\lambda^{\Delta}$ using definition $\mathcal{L}'$ does not contain occurrences of $def\mathcal{L}$, and, as a result, the definition mechanism itself can be replaced: the sequent $\cdot;\cdot \longrightarrow B$ is provable in $FO\lambda^{\Delta}$ with the definition $\mathcal{L}'$ if and only if the sequent $\Sigma; \mathcal{L}'' \longrightarrow B$ is provable in $FO\lambda$. Thus, to compute with this specification of one-step transitions for the π-calculus, ∇ and definitions do not add expressive power and only a standard logic programming language, such as λProlog, is needed to automate proof search.

To illustrate what expressive power is contributed by both ∇ and definitions in a proof system, we will need to consider the problem of dealing with *negative*

information about transitions in the π-calculus. Such information is often needed when proving simulation of processes, e.g., in showing that a process can make certain transitions and *no more*. We shall see that the encoding of the restriction operator using the $\forall$-quantifier is not appropriate in this case while the use of ∇ is appropriate.

Consider the process $\nu y.[x = y]\bar{x}y.0$. This process cannot make any transition since the bound variable y denotes a name different from x. We would therefore expect that the following is provable.

$$\forall x \forall z \forall Q \forall \alpha.[(\nu y.[x = y](out\ x\ z\ 0) \xrightarrow{\alpha} Q) \supset \bot]$$

If we had used $\forall$ in encoding restriction (that is, in the premises of inference rules res and open in Figure 6), attempting to prove the above formula would have reduced to attempting to prove the sequent

$$\{x, z, Q, \alpha\}; \forall y.([x = y](out\ x\ z\ 0) \xrightarrow{\alpha} Q) \longrightarrow \bot.$$

The only applicable rule (given the cut-elimination result in Corollary 7.6) is $\forall\mathcal{L}$, followed by $def\mathcal{L}_{csu}$. For the sequent to be provable, y would have to be instantiated with some term t such that t is distinct from all possible instantiation of x, so that the $def\mathcal{L}_{csu}$ rule will produce an empty premise. More precisely, suppose we instantiate y with some constant a different from x, then we are left with proving the sequent

$$\{x, z, Q, \alpha\}; ([x = a](out\ x\ z\ 0) \xrightarrow{\alpha} Q) \longrightarrow \bot.$$

However, we see that no matter with which closed term a we choose to instantiate y, applying $def\mathcal{L}_{csu}$ to this sequent will result in a premise in which x is identified with a, that is,

$$\{z, \alpha, Q\}; ((out\ a\ z\ 0) \xrightarrow{\alpha} Q) \longrightarrow \bot$$

via the unifier $\{a/x\}$. Applying another $def\mathcal{L}_{csu}$ to this sequent leaves us with the sequent $.;. \longrightarrow \bot$, which is clearly not provable. Hence, the scoping of variables at the object-level is lost at the meta-level. Fortunately, this scoping constraint is captured precisely by ∇, as it is shown in the derivation in Figure 8. The success of the topmost instance of $def\mathcal{L}_{csu}$ depends on the failure of the unification problem $\lambda w.x = \lambda w.w$. Notice that the scoping of object variables is maintained at the meta-level by the separation of (global) eigenvariables and (locally bound) generic variables. The "newness" of w is internalized as λ-abstraction and hence it is not subject to any instantiation.

A more complete picture of the differences between ∇ and $\forall$ is illustrated in the definition clause for simulation in Figure 9. Notice the when checking simulation for bounded inputs, the $\forall$ quantifier is used while for bounded outputs, the ∇ quantifier is used.

In the following illustration, we shall use the original syntax of the π-calculus for readability purpose: when we mix that original syntax with logic, we will assume that the reader encodes it directly into logic following the encoding mentioned above.

It is important to note that in encoding late (bi)simulation, the free names in the processes being checked for (bi)similarity should be interpreted in such a way

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{x,z,Q,\alpha\};w \triangleright ([x=w](out\ x\ z\ 0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ def\mathcal{L}_{csu}}{\{x,z,Q,\alpha\};. \triangleright \nabla y.([x=y](out\ x\ z\ 0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ \nabla\mathcal{L}}{\{x,z,Q,\alpha\};. \triangleright (\nu y.[x=y](out\ x\ z\ 0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ def\mathcal{L}_{csu}}{\{x,z,Q,\alpha\};\longrightarrow . \triangleright (\nu y.[x=y](out\ x\ z\ 0) \xrightarrow{\alpha} Q) \supset \bot}\ \supset\mathcal{R}$$

Fig. 8. The proof of a negation.

$$\begin{aligned} sim\ P\ Q \stackrel{\triangle}{=}\ & \forall A \forall P'\ [(P \xrightarrow{A} P') \supset \exists Q'.(Q \xrightarrow{A} Q') \wedge sim\ P'\ Q'] \wedge \\ & \forall X \forall P'\ [(P \xrightarrow{\downarrow X} P') \supset \exists Q'.(Q \xrightarrow{\downarrow X} Q') \wedge \forall w.sim\ (P'w)\ (Q'w)] \wedge \\ & \forall X \forall P'\ [(P \xrightarrow{\uparrow X} P') \supset \exists Q'.(Q \xrightarrow{\uparrow X} Q') \wedge \nabla w.sim\ (P'w)\ (Q'w)] \end{aligned}$$

Fig. 9. Definition of π-calculus simulation

that they are not subject to meta-level instantiation. One way of realizing this is to encode free names as some fixed non-logical constants of type n; another is to treat free names as ∇-quantified variables. These two approaches are equivalent, as far as the adequacy result for late bisimulation is concerned. However, the former is relatively simple to present, which is the reason we adopt this approach in the following discussion. The latter approach is more uniform, since newly generated free names and the existing free names are represented in the same way. This approach is also interesting in that it relates certain aspects of names to the way they are quantified, in particular, it is shown in [Tiu and Miller 2004] (where precise connections between this style specification and open and late bisimulations are given) that different ways of quantifying free names result in different bisimulation relations. Note that in encoding late (bi)simulation, free names in processes should not be interpreted as universally quantified variables, since otherwise we lose the adequacy of the encoding. For instance, $x|\bar{y}$ is simulated by $x.\bar{y}+\bar{y}.x$, but $x|\bar{x}$ is not simulated by $x.\bar{x}+\bar{x}.x$. However, if we interpret the free names x and y as universally quantified, then $\forall x \forall y.sim\ (x|\bar{y})\ (x.\bar{y} + \bar{y}.x)$ implies $\forall x.sim\ (x|\bar{x})\ (x.\bar{x} + \bar{x}.x)$.

Let us consider the following four π-calculus expressions. (Here we are using the usual abbreviations: when only a name, say d, is used as a prefix, it denotes the prefix $d(w)$, where w is vacuous in its scope; when the bar'ed name, say $\bar{d}$, is used as a prefix, it denotes the prefix $\bar{d}a$, where a is some fixed value; the expression $\bar{c}(y).P$ abbreviates $(y)\bar{c}y.P$; and when a prefix is written without a continuation, the continuation 0 is assumed. Thus, for example, $\bar{y} \mid d$ denotes $\bar{y}a.0 \mid d(w).0$.)

$$\begin{array}{ll} P_1 = c(y).(\bar{y} \mid d) & P_2 = c(y).((\bar{y}.d) + (d.\bar{y})) \\ P_3 = \bar{c}(y).(\bar{y} \mid d) & P_4 = \bar{c}(y).((\bar{y}.d) + (d.\bar{y})) \end{array}$$

The process P_2 is simulated by P_1 but the converse is not true since after P_1 performs an $(\downarrow cd)$, it is possible for the resulting process to take a τ step. The sequence of actions $(\downarrow cd)$ and τ is not possible with P_2. The processes P_3 and P_4 do, however, simulate each other (they are, in fact, bisimilar). The only difference between these pairs of processes is, of course, that the first is prefixed with a

bounded input prefix while the second is prefixed with a bounded output prefix. These different bounded prefixes are handled in the simulation definition in Figure 9 using, in one case, $\forall$ and the other case ∇.

For example, consider proving the sequent

$$\cdot;\cdot \longrightarrow sim\ (c(y).(\bar{y} \mid d))\ (c(y).((\bar{y}.d) + (d.\bar{y}))),$$

which, as we discussed above, should not be provable. Here, the free names c and d are encoded as non-logical constants c and d of type n. We argue informally why the above sequent has no proof (for the formal statements and proofs of the adequacy of the more general encoding of bisimulation, see [Tiu 2004b]). The attempt to prove this sequent reduces (via $def\mathcal{R}$, $\forall\mathcal{R}$, and $\supset\mathcal{R}$) to needing to prove the three sequents (1-3) in Figure 10. By Corollary 7.8 (see Section 7), if a sequent with an atom on the left has a proof, it has a proof with an instance of the $def\mathcal{L}_{csu}$ rule that introduces that atom. Thus, we can conclude that sequents (1) and (3) are trivially provable since the required unification problem in $def\mathcal{L}_{csu}$ fails for all clauses in the definition. The second sequent is the consequence of a non-trivial occurrence of the $def\mathcal{L}_{csu}$ rule, giving rise to the need to prove sequent (4) in Figure 10 (here, the variable X is instantiated to c and M is instantiated to $\lambda y.(\bar{y} \mid d)$). Proving this requires making the appropriate substitution for N (obvious) and then proving the sequent

$$\cdot;\cdot \longrightarrow \forall w.sim\ (\bar{w} \mid d)\ ((\bar{w}.d) + (d.\bar{w}))$$

Similarly to our first step, proving this reduces to the three sequents (5), (6), and (7). Both sequents (6) and (7) have simple proofs (which we leave as an exercise to the reader). A proof of (5) using $def\mathcal{L}_{csu}$ has two premises: one with A instantiated to τ, w to d, and P to $0 \mid 0$, and one with A instantiated to $\uparrow wa$ and P to $0 \mid d$ (w is not instantiated). The first of these premise sequents is

$$\cdot;\cdot \longrightarrow \exists Q[((\bar{d}.d) + (d.\bar{d})) \stackrel{\tau}{\longrightarrow} Q \wedge sim\ (0 \mid 0)\ Q]$$

This is not provable since there is no τ transition from $((\bar{d}.d) + (d.\bar{d}))$. As a result, since this sequent is not provable we may conclude that the original sequent is not provable. The reason for this failure is also clear from this attempt of a proof construction: although both P_1 and P_2 make an initial input step, the first of the resulting pair of processes can make a τ step but the second cannot.

Turning to the case of expressions P_3 and P_4, consider proving the sequent

$$\cdot;\cdot \longrightarrow sim\ (\bar{c}(y).(\bar{y} \mid d))\ (\bar{c}(y).((\bar{y}.d) + (d.\bar{y}))),$$

which, as we discussed above, should be provable. A proof attempt of this sequent proceeds similar to the previous example, yielding the sequent (4′) in Figure 10. Proving this reduces to the three sequents (5′), (6′), and (7′): notice that w is not given global scope in the sequents but local scope and that the eigenvariables (H, M, Z, and S) are raised with respect to their counterparts in (5), (6), and (7). Sequents (6′) and (7′) are proved as in (6) and (7). In this case, however, a proof of (5′) using $def\mathcal{L}_{csu}$ has exactly one premise, where H is instantiated to $\lambda w. \uparrow wa$ and M to $\lambda w.0 \mid d$. The resulting sequent is

$$\cdot;\cdot \longrightarrow w \triangleright \exists Q[((\bar{w}.d) + (d.\bar{w})) \stackrel{\uparrow wa}{\longrightarrow} Q \wedge sim\ (0 \mid d)\ Q]$$

$$A, P; (c(y).(\bar{y} \mid d)) \xrightarrow{A} P \longrightarrow \exists Q[(c(y).(\bar{y}.d + d.\bar{y})) \xrightarrow{A} Q \wedge sim\ P\ Q] \quad (1)$$

$$X, M; (c(y).(\bar{y} \mid d)) \xrightarrow{\downarrow X} M \longrightarrow \exists N[(c(y).(\bar{y}.d + d.\bar{y})) \xrightarrow{\downarrow X} N \wedge \forall w.sim\ (Mw)\ (Nw)] \quad (2)$$

$$X, M; (c(y).(\bar{y} \mid d)) \xrightarrow{\uparrow X} M \longrightarrow \exists N[(c(y).(\bar{y}.d + d.\bar{y})) \xrightarrow{\uparrow X} N \wedge \nabla x.sim\ (Mx)\ (Nx)] \quad (3)$$

$$\cdot; \cdot \longrightarrow \exists N[(c(y).(\bar{y}.d + d.\bar{y})) \xrightarrow{\downarrow c} N \wedge \forall w.sim\ (\bar{w} \mid d)\ (Nw)] \quad (4)$$

$$w, A, P; (\bar{w} \mid d) \xrightarrow{A} P \longrightarrow \exists Q[(\bar{w}.d + d.\bar{w}) \xrightarrow{A} Q \wedge sim\ P\ Q] \quad (5)$$

$$w, X, M; (\bar{w} \mid d) \xrightarrow{\downarrow X} M \longrightarrow \exists N[(\bar{w}.d + d.\bar{w}) \xrightarrow{\downarrow X} N \wedge \forall u.sim\ (Mu)\ (Nu)] \quad (6)$$

$$w, X, M; (\bar{w} \mid d) \xrightarrow{\uparrow X} M \longrightarrow \exists N[(\bar{w}.d + d.\bar{w}) \xrightarrow{\uparrow X} N \wedge \nabla u.sim\ (Mu)\ (Nu)] \quad (7)$$

$$\cdot; \cdot \longrightarrow \exists N[(\bar{c}(y).(\bar{y}.d + d.\bar{y})) \xrightarrow{\uparrow c} N \wedge \nabla w.sim\ (\bar{w} \mid d)\ (Nw)] \quad (4')$$

$$H, M; w \triangleright (\bar{w} \mid d) \xrightarrow{(Hw)} (Mw) \longrightarrow w \triangleright \exists Q[(\bar{w}.d + d.\bar{w}) \xrightarrow{(Hw)} Q \wedge sim\ (Mw)\ Q] \quad (5')$$

$$Z, S; w \triangleright (\bar{w} \mid d) \xrightarrow{\downarrow(Zw)} (Sw) \longrightarrow w \triangleright \exists N[(\bar{w}.d + d.\bar{w}) \xrightarrow{\downarrow(Zw)} N \wedge \forall u.sim\ (Swu)\ (Nu)] \quad (6')$$

$$Z, S; w \triangleright (\bar{w} \mid d) \xrightarrow{\uparrow(Zw)} (Sw) \longrightarrow w \triangleright \exists N[(\bar{w}.d + d.\bar{w}) \xrightarrow{\uparrow(Zw)} N \wedge \nabla u.sim\ (Swu)\ (Nu)] \quad (7')$$

Fig. 10. Some sequents

This sequent, like all the remaining ones in this proof attempt, now has a simple proof.

Notice that although we have now encountered higher-order unification problems and higher-order substitutions, the unification problems generated from this particular example fall within L_λ-*unification* or *higher-order pattern unification* [Miller 1991; Nipkow 1993]. This subset of the unification of simply typed λ-terms has complexity similar to that of first-order unification, in that it is decidable and has most general unifiers when unifiers exist.

Certain substitution theorems are easy to prove from our specification of the π-calculus. For example, if the atomic formula $\nu n.Mn \xrightarrow{A} \nu n.Nn$ is provable from the definition in Figure 6, then it must be the case that (since there is only one way to prove this formula), we must have a proof of

$$\nabla n.Mn \xrightarrow{A} Nn.$$

Proposition 7.10 tells us that when we have a $\mathsf{hc}^{\forall\nabla}$-definition and a $\mathsf{hc}^{\forall\nabla}$-goal, interchanging ∇ and $\forall$ in the goal and the body of definition clauses does not affect provability. Thus, we conclude that we have a proof of $\forall n.Mn \xrightarrow{A} Nn$, and thus, for any particular t of type n, we know that $Mt \xrightarrow{A} Nt$.

The encoding of π-calculus above can also be extended to include the mismatch operator by using negation.

$$\frac{x = y \supset \bot \qquad P \xrightarrow{A} Q}{[x \neq y]P \xrightarrow{A} Q}\ \text{mismatch}$$

Operationally, mismatch is modeled as failure of unification at the logic level. Notice that the resulting definition is not Horn anymore since we have an implication in the body of the clause representing the above inference rule. As a consequence, Proposition 7.10 is not applicable to this definition and such substitution results as

$$
\begin{array}{rcl@{\qquad}rcl}
pv\ \hat{\top} & \triangleq & \top & & & \\
pv\ (G\ \&\ G') & \triangleq & pv\ G \wedge pv\ G' & & & \\
pv\ (\hat{\forall} G) & \triangleq & \nabla x.pv\ (Gx) & & & \\
pv\ (\hat{\exists} G) & \triangleq & \exists x.pv\ (Gx) & X = X & \triangleq & \top \\
pv\ A & \triangleq & \exists D.atom\ A \wedge prog\ D \wedge bc\ D\ A & atom\ (q\ X\ Y\ Z) & \triangleq & \top \\
bc\ A\ A & \triangleq & atom\ A & prog\ (\hat{\forall} X \hat{\forall} Y\ q\ X\ X\ Y) & \triangleq & \top \\
bc\ (G \Rightarrow D)\ A & \triangleq & bc\ D\ A \wedge pv\ G & prog\ (\hat{\forall} X \hat{\forall} Y\ q\ X\ Y\ X) & \triangleq & \top \\
bc\ (\hat{\forall} D)\ A & \triangleq & \exists t.\ bc\ D\ t\ A & prog\ (\hat{\forall} X \hat{\forall} Y\ q\ Y\ X\ X) & \triangleq & \top
\end{array}
$$

Fig. 11. Interpreter for an object-level logic and additional clauses.

mentioned above are either no longer true or require different proofs.

6. EXAMPLE: AN OBJECT-LOGIC ENCODING

Consider the problem of proving the formula

$$\forall u \forall v[q\ \langle u, t_1\rangle\ \langle v, t_2\rangle\ \langle v, t_3\rangle],$$

where q is a three place predicate, $\langle\cdot,\cdot\rangle$ is used to form pairs, t_1 and t_2 are some first-order terms, and the only assumptions for the predicate q are the (universal closure of the) three atomic formulas: $q\ X\ X\ Y$, $q\ X\ Y\ X$ and $q\ Y\ X\ X$. Clearly, this query succeeds only if terms t_2 and t_3 are equal [Miller and Tiu 2002]. One natural way to formalizing this reasoning involves first encoding provability of an object-level first-order logic in $FO\lambda^{\Delta\nabla}$ and then to reason directly on this encoding. Let obj be the type of object-level logic, let $\hat{\top} : obj$ be object-level true, let & and $\Rightarrow$ be object-level conjunction and implication (both at type $obj \to obj \to obj$), and let $\hat{\forall}$ and $\hat{\exists}$ be object-level quantifiers at type $(i \to obj) \to obj$ (for some fixed type i ranging over first-order object-level terms). To encode provability, we use two main predicates pv of type $obj \to o$ to indicate first-order provability and bc of type $obj \to obj \to o$ to specify "backchaining". The definition clauses on the left side of Figure 11 encodes provability for logic programming in a subset of first-order $\mathsf{hc}^{\forall}$ and is parametrized by the predicates $atom$ (describing object-level atomic formulas) and $prog$ (describing object-level logic programs clauses). The definition clauses on the right side of the figure contains encodes the object-level logic program we are considering here.

Notice that while the object-level logic here is $\mathsf{hc}^{\forall}$ (since our motivating example is concerned with the provability of a universally quantified formula), the meta-level definition is hc^{∇}.

Given the definition in Figure 11, the following query encodes our intended theorem about object-level provability.

$$\forall x, y, z[pv\ (\hat{\forall} u \hat{\forall} v[q\ \langle u, x\rangle\ \langle v, y\rangle\ \langle v, z\rangle]) \supset y = z]$$

Attempting a proof of this formula leads to the following sequent (after applying some right rules and a pair of $def\mathcal{L}_{csu}$ and $\nabla\mathcal{L}$ rules):

$$X, Y, Z; (s, r) \triangleright pv\ (q\ \langle s, X\rangle\ \langle r, Y\rangle\ \langle r, Z\rangle) \longrightarrow \triangleright Y = Z.$$

A series of $def\mathcal{L}_{csu}$ rules will now need to be applied in order to work through

the encoding of the object-level interpreter. In the end, three separate unification problems will be attempted, one for each of the three ways to prove the predicate q. In particular, the $def\mathcal{L}_{csu}$ rule will attempt to unify $\lambda s\lambda r.(q\ \langle s, X\rangle\ \langle r, Y\rangle\ \langle r, Z\rangle)$ with each of the following three terms:

$$\lambda s\lambda r.(q\ (X'\ s\ r)\ (X'\ s\ r)\ (Y'\ s\ r))$$
$$\lambda s\lambda r.(q\ (X'\ s\ r)\ (Y'\ s\ r)\ (X'\ s\ r))$$
$$\lambda s\lambda r.(q\ (Y'\ s\ r)\ (X'\ s\ r)\ (X'\ s\ r))$$

The first two unification problems fail and hence the corresponding occurrences of $def\mathcal{L}_{csu}$ succeed. The third of these unification problems is solvable, however, with X' instantiated to $\lambda s\lambda r.\langle r, Z\rangle$, Y' instantiated to $\lambda s\lambda r.\langle s, Z\rangle$, Y instantiated to Z (or vice versa), and X uninstantiated. As a result, this third premise is the sequent $\cdot;\cdot \longrightarrow Y = Y$, which is provable using $def\mathcal{R}$.

The more common approach to encoding object-logic provability into a meta-logic uses the meta-level universal quantifier instead of the ∇ for the clause encoding the provability of object-level universal quantification: that is, the clause

$$pv\ (\hat{\forall} x.G\ x) \stackrel{\triangle}{=} \forall x[pv\ (G\ x)].$$

is used instead. In this case, attempting a proof of this formula reduces to an attempt to prove the sequent

$$X, Y, Z; \triangleright pv\ (q\ \langle s_1, X\rangle\ \langle s_2, Y\rangle\ \langle r, Z\rangle) \longrightarrow \triangleright Y = Z,$$

and were s_1 and s_2 are two terms. To complete the proof, these two terms must be chosen to be different. While this sequent can be proved, doing so requires the assumption that there are two such distinct terms (the domain is non-empty and not a singleton). Our encoding using ∇ allows the (meta-level) proof to be completed in a more natural and "internal" way without this "external" assumption.

The encoding of $\hat{\forall}$ using ∇ reflects the intensional use of object-logic eigenvariables in object-logic proofs for universally quantified goals; that is, (object-logic) eigenvariables are not instantiated in the proofs. The encoding using ∇, however, does come with a price: the extensional aspect of $\hat{\forall}$ — that is, if $\hat{\forall} G$ is provable then $G\,t$ is provable for every closed term t — cannot be directly proved in $FO\lambda^{\Delta\nabla}$. That is, the formula $\forall G.\ pv\ (\hat{\forall} G) \supset \forall t.\ pv\ (G\,t)$ is not a theorem in $FO\lambda^{\Delta\nabla}$, although it is valid as a meta-level observation about the encoding of the object-logic provability. Recall that the definition for the object-logic in Figure 11 is hc^{∇}, hence by Proposition 7.10, if we are only concerned with proving positive goals (no implication), ∇ and $\forall$ can be interchanged. As a consequence, if $pv\ (\hat{\forall} G)$ is provable in $FO\lambda^{\Delta\nabla}$, then $\nabla x.\ pv\ (G\,x)$ is provable, and by interchanging ∇ with $\forall$, we have $\forall x.\ pv\ (G\,x)$ is provable, and hence $pv\ (G\,t)$ is also provable.

7. META THEORY

We now present some meta-theoretic results concerning the logic $FO\lambda^{\Delta\nabla}$. The main such result is, of course, that it satisfied cut-elimination.

7.1 Cut Elimination

The proof of cut-elimination for $FO\lambda^{\Delta\mathbb{N}}$ that we present here is similar to the one given by Gentzen [Gentzen 1969] in that the main induction involves the heights of

proofs and an additional measure involving the level of cut formulas. The stratification of definitions makes sure that the level of cut formulas does not increase when permuting up cut over definition rules, while other measures decrease. Central to the proof is the following substitution lemma about $FO\lambda^{\Delta\nabla}$ proofs: if $\Sigma; \Gamma \longrightarrow \mathcal{C}$ has a proof and θ be a substitution, then there is a derivation of $\Sigma\theta; \Gamma\theta \longrightarrow \mathcal{C}\theta$ such that certain measures are not increased. The precise statement will follow.

We define several measures on derivation that are needed to show termination of cut reduction.

Definition 7.1. Given a derivation Π with premise derivations $\{\Pi_i\}_i$, the height of the derivation Π, denoted by $\mathrm{ht}(\Pi)$, is $\mathrm{lub}(\{\mathrm{ht}(\Pi_i)\}_i) + 1$, where $\mathrm{lub}(S)$ is the least upper bound of the set S. The measure $\mathrm{def}(\Pi)$ which indicates the depth of applications of *def*$\mathcal{L}$ rule is defined as follows.

$$\mathrm{def}(\Pi) = \begin{cases} \mathrm{lub}(\{\mathrm{def}(\Pi_i)\}_i) + 1, & \text{if } \Pi \text{ ends with a } \mathit{def}\mathcal{L} \text{ rule} \\ \mathrm{lub}(\{\mathrm{def}(\Pi_i)\}_i), & \text{otherwise.} \end{cases}$$

Similary, the depth of $c\mathcal{L}$ rules is defined as

$$\mathrm{contr}(\Pi) = \begin{cases} \mathrm{lub}(\{\mathrm{contr}(\Pi_i)\}_i) + 1, & \text{if } \Pi \text{ ends with a } c\mathcal{L} \text{ rule} \\ \mathrm{lub}(\{\mathrm{contr}(\Pi_i)\}_i), & \text{otherwise.} \end{cases}$$

Note that given the possible infinite branching of *def*$\mathcal{L}$ rule, the measures defined above can, in general, be ordinals. Therefore in proofs involving induction on those measures, transfinite induction is needed. In the following inductive proofs, we often do case analyses on the last rule of a derivation. In such situation, the inductive cases for both successor ordinals and limit ordinals are basically covered by the case analyses on the inference figures involved, and we shall not make explicit use of transfinite induction.

LEMMA 7.2. *Let* Π *be a derivation of* $\Sigma; \Gamma \longrightarrow \mathcal{C}$*. Then there is a derivation* Π' *of* $\Sigma, x; \Gamma \longrightarrow \mathcal{C}$*, where* $x \notin \Sigma$*, such that* $\mathrm{ht}(\Pi') \leq \mathrm{ht}(\Pi)$*,* $\mathrm{def}(\Pi') \leq \mathrm{def}(\Pi)$ *and* $\mathrm{contr}(\Pi') \leq \mathrm{contr}(\Pi)$*.*

PROOF. By induction on $\mathrm{ht}(\Pi)$. □

LEMMA 7.3. *Let* Π *be a derivation of* $\Sigma; \Gamma \longrightarrow \mathcal{C}$ *and* θ *be a substitution. Then there is a derivation* $\Pi\theta$ *of* $\Sigma\theta; \Gamma\theta \longrightarrow \mathcal{C}\theta$ *such that* $\mathrm{ht}(\Pi\theta) \leq \mathrm{ht}(\Pi)$*,* $\mathrm{def}(\Pi\theta) \leq \mathrm{def}(\Pi)$ *and* $\mathrm{contr}(\Pi\theta) \leq \mathrm{contr}(\Pi)$*.*

PROOF. By induction on $\mathrm{ht}(\Pi)$. Most cases follow immediately from induction hypothesis. We show the interesting cases involving *def*$\mathcal{L}$ and *def*$\mathcal{R}$. Suppose Π ends with the *def*$\mathcal{L}$ rule

$$\frac{\left\{\begin{array}{c}\Pi^{(\rho,\mathcal{B})} \\ \Sigma\rho; \mathcal{B}\gamma, \Gamma'\rho \longrightarrow \mathcal{C}\rho\end{array}\right\}_{\mathrm{dfn}(\rho,\mathcal{A},\gamma,\mathcal{B})}}{\Sigma; \mathcal{A}, \Gamma' \longrightarrow \mathcal{C}}\ \mathit{def}\mathcal{L},$$

and suppose that $\mathrm{dfn}(\rho', \mathcal{A}\theta, \gamma', \mathcal{B})$ holds for some ρ' and γ'. We have $(\mathcal{A}\theta)\rho' = \mathcal{H}\gamma'$, given a raised definition clause $\forall \bar{y}.[\mathcal{H} \triangleq \mathcal{B}]$, where $\bar{y}$ are chosen to be distinct from the variables in Σ and the variables free in the range of θ. Then, obviously,

$\mathrm{dfn}(\theta \circ \rho', \mathcal{A}, \gamma', \mathcal{B})$ holds as well. Therefore we construct $\Pi\theta$ as the derivation

$$\frac{\left\{\begin{array}{c}\Pi^{(\theta\circ\rho',\mathcal{B})} \\ \Sigma\theta\rho'; \mathcal{B}\gamma', \Gamma'\theta\rho' \longrightarrow \mathcal{C}\theta\rho'\end{array}\right\}_{\mathrm{dfn}(\rho',\mathcal{A}\theta,\gamma',\mathcal{B})}}{\Sigma\theta; \mathcal{A}\theta, \Gamma'\theta \longrightarrow \mathcal{C}\theta}\ def\mathcal{L}.$$

Otherwise, suppose Π ends with the $def\mathcal{R}$ rule

$$\frac{\begin{array}{c}\Pi' \\ \Sigma; \Gamma \longrightarrow \mathcal{B}\rho\end{array}}{\Sigma; \Gamma \longrightarrow \mathcal{A}}\ def\mathcal{R},$$

where $\mathcal{A}$ and $\mathcal{B}$ are the judgments, and $\mathrm{dfn}(\epsilon, \mathcal{A}, \rho, \mathcal{B})$ holds for a given raised definition clause $\forall\bar{y}.[\mathcal{H} \triangleq \mathcal{B}]$. By Definition 4.4, this means $\mathcal{A} = \mathcal{H}\rho$. Obviously, $\mathcal{A}\theta = (\mathcal{H}\rho)\theta$ and therefore $\mathrm{dfn}(\epsilon, \mathcal{A}\theta, \rho \circ \theta, \mathcal{B})$ holds as well. We can then construct $\Pi\theta$ as the derivation

$$\frac{\begin{array}{c}\Pi'\theta \\ \Sigma\theta; \Gamma\theta \longrightarrow \mathcal{B}\rho\theta\end{array}}{\Sigma\theta; \Gamma\theta \longrightarrow \mathcal{A}\theta}\ def\mathcal{R},$$

where $\Pi'\theta$ is obtained from Π' by inductive hypothesis.

Since each transformation step from Π to $\Pi\theta$ does not introduce extra applications of rules, $\mathrm{ht}(\Pi\theta)$, $\mathrm{def}(\Pi\theta)$ and $\mathrm{contr}(\Pi\theta)$ are less than or equal to $\mathrm{ht}(\Pi)$, $\mathrm{def}(\Pi)$ and $\mathrm{contr}(\Pi)$, respectively. They can be smaller than the corresponding measures of Π because in the case of $def\mathcal{L}$ there could be fewer premises. □

In proving cut-elimination, we use a more general form of cut rule, called the *multicut* rule,

$$\frac{\Delta_1 \longrightarrow B_1 \quad \ldots \quad \Delta_n \longrightarrow B_n \quad B_1, \ldots, B_n, \Gamma \longrightarrow C}{\Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow C}\ mc \qquad (n \geq 0).$$

This generalization is due to Slaney [Slaney 1989], and it is used to simplify the presentation of the cut-elimination proof.

We associate a measure to a derivation ending with *mc* and show that the measure decreases as we permute up the *mc* rule. The general cut-elimination theorem is proved by successively removing the topmost cut instances. The measure involves a multiset as one of its component. We use $\uplus$ to denote multiset union.

Definition 7.4. Let Ξ be the following derivation ending with a multicut rule:

$$\frac{\begin{array}{c}\Pi_1 \\ \Sigma; \Delta_1 \longrightarrow \mathcal{B}_1\end{array} \quad \cdots \quad \begin{array}{c}\Pi_n \\ \Sigma; \Delta_n \longrightarrow \mathcal{B}_n\end{array} \quad \begin{array}{c}\Pi \\ \Sigma; \mathcal{B}_1, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{array}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc.$$

Assume also that the proofs $\Pi_1, \ldots, \Pi_n, \Pi$ are (multi)cut-free. We define a measure $\mu(\Xi)$ to be the tuple

$$\langle \max\{\mathrm{lvl}(\mathcal{B}_1), \ldots, \mathrm{lvl}(\mathcal{B}_n)\}, \mathrm{def}(\Pi), \mathrm{contr}(\Pi), \sum \mid \mathcal{B}_i \mid, \mathcal{M}(\Xi)\rangle\ ,$$

where $\mathcal{M}(\Xi)$ is the multiset $\{\mathrm{ht}(\Pi_1), \ldots, \mathrm{ht}(\Pi_n), \mathrm{ht}(\Pi)\}$ and $\mid \mathcal{B}_i \mid$ is the number of occurrences of logical connectives in $\mathcal{B}_i$. The ordering on the measure μ is defined lexicographically on the ordering of its components.

THEOREM 7.5. *Let Ξ be a derivation of $\Sigma; \Gamma \longrightarrow \mathcal{C}$ ending with a multicut, which is the only cut in the derivation. Then there exists a cut-free derivation of the same sequent.*

PROOF. Let Ξ be the derivation

$$\frac{\Sigma; \Delta_1 \overset{\Pi_1}{\longrightarrow} \mathcal{B}_1 \quad \cdots \quad \Sigma; \Delta_n \overset{\Pi_n}{\longrightarrow} \mathcal{B}_n \quad \Sigma; \mathcal{B}_1, \ldots, B_n, \Gamma \overset{\Pi}{\longrightarrow} \mathcal{C}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc.$$

If $n = 0$, Ξ reduces to the premise derivation Π.

For $n > 0$ we specify the reduction relation based on the last rule of the premise derivations. If the rightmost premise derivation Π ends with a left rule acting on a cut formula B_i, then the last rule of Π_i and the last rule of Π together determine the reduction rules that apply. We classify these rules according to the following criteria: we call the rule an *essential* case when Π_i ends with a right rule; if it ends with a left rule, it is a *right-commutative* case; if Π_i ends with the *init* rule, then we have an *axiom* case. When Π 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 Π ends in a rule acting on a formula other than a cut formula, then we call this a *left-commutative* case. A *structural* case results when Π ends with a contraction or weakening on a cut formula. If Π ends with the *init* rule, this is also an axiom case. For simplicity of presentation, we always show $i = 1$ and we often abbreviate judgments like $\sigma \triangleright B$ and $\sigma \triangleright C$ as $\mathcal{B}$ and $\mathcal{C}$ when the local signature σ is irrelevant to the context of discussion.

Essential cases:

$\wedge\mathcal{R}/\wedge\mathcal{L}$. If Π_1 and Π are

$$\frac{\Sigma; \Delta_1 \overset{\Pi'_1}{\longrightarrow} \sigma \triangleright B'_1 \quad \Sigma; \Delta_1 \overset{\Pi''_1}{\longrightarrow} \sigma \triangleright B''_1}{\Sigma; \Delta_1 \longrightarrow \sigma \triangleright B'_1 \wedge B''_1}\ \wedge\mathcal{R} \qquad \frac{\Sigma; \sigma \triangleright B'_1, \ldots, \Gamma \overset{\Pi'}{\longrightarrow} \mathcal{C}}{\Sigma; \sigma \triangleright B'_1 \wedge B''_1, \ldots, \Gamma \longrightarrow \mathcal{C}}\ \wedge\mathcal{L}$$

then Ξ reduces to the derivation Ξ'

$$\frac{\Sigma; \Delta_1 \overset{\Pi'_1}{\longrightarrow} \mathcal{B}'_1 \quad \cdots \quad \Sigma; \Delta_n \overset{\Pi_n}{\longrightarrow} \mathcal{B}_n \quad \Sigma; \mathcal{B}'_1, \ldots, \mathcal{B}_n, \Gamma \overset{\Pi'}{\longrightarrow} \mathcal{C}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc.$$

The measure $\mu(\Xi)$ is smaller than $\mu(\Xi')$, since

$$\max\{\mathrm{lvl}(\mathcal{B}'_1), \mathrm{lvl}(\mathcal{B}_2), \ldots, \mathrm{lvl}(\mathcal{B}_n)\} \leq \max\{\mathrm{lvl}(\mathcal{B}_1), \ldots, \mathrm{lvl}(\mathcal{B}_n)\},$$

$$\mathrm{def}(\Pi') = \mathrm{def}(\Pi), \quad \mathrm{contr}(\Pi) = \mathrm{contr}(\Pi') \text{ and } \mid \mathcal{B}'_1 \mid < \mid \mathcal{B}_1 \mid .$$

Therefore we can apply the inductive hypothesis to Ξ' to obtain a cut free derivation. The case for the other $\wedge\mathcal{L}$ rule is symmetric.

$\vee\mathcal{R}/\vee\mathcal{L}$. If Π_1 and Π are

$$\frac{\Sigma; \Delta_1 \overset{\Pi'_1}{\longrightarrow} \sigma \triangleright B'_1}{\Sigma; \Delta_1 \longrightarrow \sigma \triangleright B'_1 \vee B''_1}\ \vee\mathcal{R}$$

$$\frac{\Sigma;\sigma \triangleright B_1', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C} \quad \Sigma;\sigma \triangleright B_1'', \mathcal{B}_2, \ldots, B_n, \Gamma \longrightarrow \mathcal{C}}{\Sigma;\sigma \triangleright B_1' \vee B_1'', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}} \vee\mathcal{L},$$

then Ξ reduces to a derivation Ξ'

$$\frac{\Sigma;\Delta_1 \longrightarrow \mathcal{B}_1' \quad \cdots \quad \Sigma;\Delta_n \longrightarrow \mathcal{B}_n \quad \Sigma;\mathcal{B}_1', \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}{\Sigma;\Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}} mc.$$

As in previous case, the size of cut formulas decreases, and therefore inductive hypothesis applies to the reduct Ξ'. The case for the other $\vee\mathcal{R}$ rule is symmetric.

$\supset\mathcal{R}/\supset\mathcal{L}$:. Suppose Π_1 and Π are

$$\frac{\Sigma;\sigma \triangleright B_1', \Delta_1 \longrightarrow \sigma \triangleright B_1''}{\Sigma;\Delta_1 \longrightarrow \sigma \triangleright B_1' \supset B_1''} \supset\mathcal{R}$$

$$\frac{\Sigma;\mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \sigma \triangleright B_1' \quad \Sigma;\sigma \triangleright B_1'', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}{\Sigma;\sigma \triangleright B_1' \supset B_1'', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}} \supset\mathcal{L}.$$

Let Ξ_1 be the derivation

$$\frac{\left\{\Sigma;\Delta_i \longrightarrow \mathcal{B}_i\right\}_{i\in\{2..n\}} \quad \Sigma;\mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{B}_1'}{\Sigma;\Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{B}_1'} mc.$$

The derivation Ξ_1 has a smaller size of cut formula than Ξ, while other measures remain non-increasing. Therefore, the inductive hypothesis can be applied to eliminate the multicut in Ξ_1. Let Ξ_1' denote the cut-free proof obtained by cut-elimination on Ξ_1 and let Ξ_2 be the derivation

$$\frac{\Sigma;\Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{B}_1' \quad \Sigma;\mathcal{B}_1', \Delta_1 \longrightarrow \mathcal{B}_1''}{\Sigma;\Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{B}_1''} mc.$$

The measure $\mu(\Xi_2)$ is strictly smaller than $\mu(\Xi)$ because

$$\mathrm{lvl}(\mathcal{B}_1') < \mathrm{lvl}(\mathcal{B}_1) \leq \max\{\mathrm{lvl}(\mathcal{B}_1), \ldots, \mathrm{lvl}(\mathcal{B}_n)\}.$$

Recall that $\mathrm{lvl}(B_1' \supset B_1'') = \max\{\mathrm{lvl}(B_1') + 1, \mathrm{lvl}(B_1'')\}$. Therefore, the multicut in Ξ_2 can be eliminated by inductive hypothesis to get a cut-free derivation Ξ_2'.

The derivation Ξ then reduces to the following derivation Ξ':

$$\frac{\Sigma;\ldots \longrightarrow \mathcal{B}_1'' \quad \left\{\Sigma;\Delta_i \longrightarrow \mathcal{B}_i\right\}_{i\in\{2..n\}} \quad \Sigma;\mathcal{B}_1'', \{\mathcal{B}_i\}_{i\in\{2..n\}}, \Gamma \longrightarrow \mathcal{C}}{\dfrac{\Sigma;\Delta_1, \ldots, \Delta_n, \Gamma, \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}{\Sigma;\Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\, c\mathcal{L}} mc.$$

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. Again, since the cut

formulas size decreases, we have $\mu(\Xi') < \mu(\Xi)$ and therefore inductive hypothesis can be applied to eliminate the multicut in Ξ'.

$\forall\mathcal{R}/\forall\mathcal{L}$. If Π_1 and Π are

$$\frac{\begin{array}{c}\Pi_1'\\ \Sigma, h; \Delta_1 \longrightarrow \sigma \triangleright B_1'[(h\ \sigma)/x]\end{array}}{\Sigma; \Delta_1 \longrightarrow \sigma \triangleright \forall_\tau x.B_1'}\ \forall\mathcal{R} \qquad \frac{\begin{array}{c}\Pi'\\ \Sigma, \sigma \vdash t : \tau \quad \Sigma; \sigma \triangleright B_1'[t/x], \ldots, \Gamma \longrightarrow \mathcal{C}\end{array}}{\Sigma; \sigma \triangleright \forall_\tau x.B_1', \ldots, \Gamma \longrightarrow \mathcal{C}}\ \forall\mathcal{L}$$

then Ξ reduces to the derivation Ξ'

$$\frac{\begin{array}{c}\Pi_1'[\lambda\sigma.t/h]\\ \Sigma; \Delta_1 \longrightarrow \sigma \triangleright B_1'[t/x]\end{array} \quad \left\{\begin{array}{c}\Pi_i\\ \Sigma; \Delta_i \longrightarrow \mathcal{B}_i\end{array}\right\}_{i\in\{2..n\}} \quad \begin{array}{c}\Pi'\\ \Sigma; \ldots \longrightarrow \mathcal{C}\end{array}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc.$$

The size of cut formulas decreases while other measures are non-increasing, therefore $\mu(\Xi') < \mu(\Xi)$ and the cut in Ξ' can be removed by induction hypothesis.

$\exists\mathcal{R}/\exists\mathcal{L}$. If Π_1 and Π are

$$\frac{\begin{array}{c}\Pi_1'\\ \Sigma, \sigma \vdash t : \tau \quad \Sigma; \Delta_1 \longrightarrow \sigma \triangleright B_1'[t/x]\end{array}}{\Sigma; \Delta_1 \longrightarrow \sigma \triangleright \exists_\tau x.B_1'}\ \exists\mathcal{R} \qquad \frac{\begin{array}{c}\Pi'\\ \Sigma, h; \sigma \triangleright B_1'[(h\ \sigma)/x], \ldots, \Gamma \longrightarrow \mathcal{C}\end{array}}{\Sigma; \sigma \triangleright \exists x_\tau.B_1', \ldots, \Gamma \longrightarrow \mathcal{C}}\ \exists\mathcal{L}$$

then Ξ reduces to the derivation Ξ'

$$\frac{\begin{array}{c}\Pi_1'\\ \Sigma; \Delta_1 \longrightarrow \sigma \triangleright B_1'[t/x]\end{array} \quad \left\{\begin{array}{c}\Pi_i\\ \Sigma; \Delta_i \longrightarrow \mathcal{B}_i\end{array}\right\}_{i\in\{2..n\}} \quad \begin{array}{c}\Pi'[\lambda\sigma.t/h]\\ \Sigma; \sigma \triangleright B_1'[t/x], \ldots \longrightarrow \mathcal{C}\end{array}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc.$$

As in the previous case, we can apply the induction hypothesis to remove the cut in Ξ'.

$\nabla\mathcal{R}/\nabla\mathcal{L}$. Suppose Π_1 and Π are

$$\frac{\begin{array}{c}\Pi_1'\\ \Sigma; \Delta_1 \longrightarrow (\sigma, y) \triangleright B_1'[y/x]\end{array}}{\Sigma; \Delta_1 \longrightarrow \sigma \triangleright \nabla x.B_1'}\ \nabla\mathcal{R} \qquad \frac{\begin{array}{c}\Pi'\\ \Sigma; (\sigma, y) \triangleright B_1'[y/x], \ldots, , \Gamma' \longrightarrow \mathcal{C}\end{array}}{\Sigma; \sigma \triangleright \nabla x.B_1', \ldots, \Gamma \longrightarrow \mathcal{C}}\ \nabla\mathcal{L}.$$

Then Ξ reduces to the derivation Ξ'

$$\frac{\begin{array}{c}\Pi_1'\\ \Sigma; \Delta_1 \longrightarrow (\sigma, y) \triangleright B_1'[y/x]\end{array} \quad \cdots \quad \begin{array}{c}\Pi'\\ \Sigma; (\sigma, y) \triangleright B_1'[y/x], \ldots \longrightarrow \mathcal{C}\end{array}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc.$$

The size of the cut formula decreases while other measures remain non-increasing, therefore the multicut in Ξ' can be eliminated by applying the inductive hypothesis.

def$\mathcal{R}$/*def*$\mathcal{L}$. Suppose Π_1 and Π are

$$\frac{\begin{array}{c}\Pi_1'\\ \Sigma; \Delta_1 \longrightarrow \mathcal{B}_1'\theta\end{array}}{\Sigma; \Delta_1 \longrightarrow \mathcal{B}_1}\ def\mathcal{R} \qquad \frac{\left\{\begin{array}{c}\Pi^{\rho,\mathcal{D}}\\ \Sigma\rho; \mathcal{D}\gamma, \mathcal{B}_2\rho, \ldots, \mathcal{B}_n\rho, \Gamma\rho \longrightarrow \mathcal{C}\rho\end{array}\right\}}{\Sigma; \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\ def\mathcal{L}.$$

By the *def*$\mathcal{R}$ rule in Π_1, $\mathrm{dfn}(\epsilon, \mathcal{B}_1, \theta, \mathcal{B}'_1)$ holds. Then Ξ reduces to Ξ'

$$\dfrac{\dfrac{\Pi'_1}{\Sigma; \Delta_1 \longrightarrow \mathcal{B}'_1\theta} \quad \left\{\dfrac{\Pi_i}{\Sigma; \Delta_i \longrightarrow \mathcal{B}_i}\right\}_{i\in\{2..n\}} \quad \dfrac{\Pi^{\epsilon,\mathcal{B}'_1}}{\Sigma; \mathcal{B}'_1\theta, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc.$$

By the definition of definition clause, we have $\mathrm{lvl}(\mathcal{B}'_1) \leq \mathrm{lvl}(\mathcal{B}_1)$, and therefore the maximum level of cut formulas is non-increasing. However, $\mathrm{def}(\Pi^{\epsilon,B'_1}) < \mathrm{def}(\Pi)$, therefore $\mu(\Xi') < \mu(\Xi)$ and inductive hypothesis can be applied to remove the multicut in Ξ'.

Left-commutative cases:

$\bullet\mathcal{L}/\circ\mathcal{L}$. Suppose Π ends with a left rule other than $c\mathcal{L}$ and $w\mathcal{L}$ acting on B_1, and Π_1 is

$$\dfrac{\left\{\dfrac{\Pi^i_1}{\Sigma'; \Delta^i_1 \longrightarrow B_1}\right\}}{\Sigma; \Delta_1 \longrightarrow B_1}\ \bullet\mathcal{L},$$

where $\bullet\mathcal{L}$ is any left rule except $\supset\mathcal{L}$, *def*$\mathcal{L}$, and Σ is a subset of Σ'. Then Ξ reduces to the derivation Ξ'

$$\dfrac{\left\{\dfrac{\Pi^i_1}{\Sigma'; \Delta^i_1 \longrightarrow \mathcal{B}_1} \quad \left\{\dfrac{\Pi'_j}{\Sigma'; \Delta_j \longrightarrow \mathcal{B}_j}\right\}_{j\in\{2..n\}} \quad \dfrac{\Pi'}{\Sigma'; \mathcal{B}_1, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\ mc\right\}}{\Sigma; \Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ \bullet\mathcal{L},$$

where Π'_j and Π' are obtained from Π_j and Π by applying Lemma 7.2. Let Ξ'_i be a premise derivation of Ξ'. Since for each Π^i_1, $\mathrm{ht}(\Pi^i_1) < \mathrm{ht}(\Pi_1)$ and since $\mathrm{ht}(\Pi'_j) \leq \mathrm{ht}(\Pi_j)$ and $\mathrm{ht}(\Pi') \leq \mathrm{ht}(\Pi)$, the multiset $\mathcal{M}(\Xi'_i)$ is strictly smaller (in the multiset ordering) than the multiset $\mathcal{M}(\Xi)$. Since other measures remain unchanged, $\mu(\Xi'_i) < \mu(\Xi)$ (and this applies to arbitrary premise derivations of Ξ') and therefore by induction hypothesis all the multicuts in Ξ' can be eliminated.

$\supset\mathcal{L}/\circ\mathcal{L}$. Suppose Π ends with a left rule other than $c\mathcal{L}$ and $w\mathcal{L}$ acting on B_1 and Π_1 is

$$\dfrac{\dfrac{\Pi'_1}{\Sigma; \Delta'_1 \longrightarrow \sigma \rhd D'_1} \quad \dfrac{\Pi''_1}{\Sigma; \sigma \rhd D''_1, \Delta'_1 \longrightarrow \mathcal{B}_1}}{\Sigma; \sigma \rhd D'_1 \supset D''_1, \Delta'_1 \longrightarrow \mathcal{B}_1}\ \supset\mathcal{L}.$$

Let Ξ_1 be

$$\dfrac{\dfrac{\Pi''_1}{\Sigma; \sigma \rhd D''_1, \Delta'_1 \longrightarrow \mathcal{B}_1} \quad \cdots \quad \dfrac{\Pi_n}{\Sigma; \Delta_n \longrightarrow \mathcal{B}_n} \quad \dfrac{\Pi}{\Sigma; \mathcal{B}_1, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}}{\Sigma; \sigma \rhd D''_1, \Delta'_1, \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc.$$

The multicut in Ξ_1 can be eliminated by inductive hypothesis since $\mathcal{M}(\Xi_1) < \mathcal{M}(\Xi)$ and other measures are equal. We let Ξ'_1 denote the resulting cut-free proof from

applying cut-elimination to Ξ_1. Then Ξ reduces to

$$\dfrac{\dfrac{\dfrac{\Pi_1'}{\Sigma; \Delta_1' \longrightarrow \sigma \triangleright D_1'}}{\Sigma; \Delta_1', \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \sigma \triangleright D_1'}\, w\mathcal{L} \quad \dfrac{\Xi_1'}{\Sigma; \sigma \triangleright D_1'', \Delta_1', \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}}{\Sigma; \sigma \triangleright D_1' \supset D_1'', \Delta_1', \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\, \supset \mathcal{L}.$$

def$\mathcal{L}/\circ\mathcal{L}$. If Π ends with a left rule other than $c\mathcal{L}$ and $w\mathcal{L}$ acting on B_1 and Π_1 is

$$\dfrac{\left\{\begin{array}{c}\Pi_1^{\rho,\mathcal{D}} \\ \Sigma\rho; \mathcal{D}\theta, \Delta_1'\rho \longrightarrow \mathcal{B}_1\rho\end{array}\right\}}{\Sigma; \mathcal{A}, \Delta_1' \longrightarrow \mathcal{B}_1}\, \textit{def}\mathcal{L}.$$

By the definition of *def*$\mathcal{L}$ rule, the relation $\mathrm{dfn}(\rho, \mathcal{A}, \theta, \mathcal{D})$ holds for a given raised definition clause $\forall \bar{x}.[\mathcal{H} \stackrel{\triangle}{=} \mathcal{D}]$ where $\bar{x}$ are chosen to be different from the variables in Σ. Then Ξ reduces to the derivation Ξ'

$$\dfrac{\left\{\begin{array}{c}\Pi_1^{\rho,\mathcal{D}} \quad \left\{\begin{array}{c}\Pi_j\rho \\ \Sigma\rho; \Delta_j\rho \longrightarrow \mathcal{B}_j\rho\end{array}\right\}_j \quad \Pi\rho \\ \Sigma\rho; \mathcal{D}\rho, \Delta_1'\rho \longrightarrow \mathcal{B}_1\rho \quad \Sigma\rho; \ldots \longrightarrow \mathcal{C}\rho \\ \hline \Sigma\rho; \mathcal{D}\theta, \Delta_1'\rho, \ldots, \Delta_n\rho, \Gamma\rho \longrightarrow \mathcal{C}\rho\end{array}\, mc\right\}}{\Sigma; \mathcal{A}, \Delta_1', \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\, \textit{def}\mathcal{L},$$

where j ranges over $\{2, \ldots, n\}$. Let Ψ be an arbitrary premise derivation of Ξ'. Since $\mathrm{ht}(\Pi_1^{\rho,\mathcal{D}}) < \mathrm{ht}(\Pi_1)$ and $\mathrm{ht}(\Pi\rho) \leq \mathrm{ht}(\Pi)$ and for each j, $\mathrm{ht}(\Pi^j\rho) \leq \mathrm{ht}(\Pi^j\rho)$, the multiset $\mathcal{M}(\Psi)$ is smaller than $\mathcal{M}(\Xi)$ and therefore induction hypothesis can be applied to eliminate the multicut in Ψ (and consequently, all multicuts in Ξ').

Right-commutative cases:

$-/\circ\mathcal{L}$. Suppose Π is

$$\dfrac{\left\{\begin{array}{c}\Pi^i \\ \Sigma'; \mathcal{B}_1, \ldots, \mathcal{B}_n, \Gamma^i \longrightarrow \mathcal{C}\end{array}\right\}}{\Sigma; \mathcal{B}_1, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\, \circ\mathcal{L},$$

where $\Sigma' \supseteq \Sigma$ and $\circ\mathcal{L}$ is any left rule other than $\supset\mathcal{L}$, *def*$\mathcal{L}$, acting on a judgment other than $\mathcal{B}_1, \ldots, \mathcal{B}_n$. Then Ξ reduces to the derivation Ξ'

$$\dfrac{\left\{\begin{array}{c}\Pi_1' \quad\quad \Pi_n' \quad\quad \Pi^i \\ \Sigma'; \Delta_1 \longrightarrow \mathcal{B}_1 \quad \cdots \quad \Sigma'; \Delta_n \longrightarrow \mathcal{B}_n \quad \Sigma'; \mathcal{B}_1, \ldots, \mathcal{B}_n, \Gamma^i \longrightarrow \mathcal{C} \\ \hline \Sigma'; \Delta_1, \ldots, \Delta_n, \Gamma^i \longrightarrow C\end{array}\, mc\right\}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow C}\, \circ\mathcal{L}.$$

The height of Π^i is smaller than the height of Π, therefore using the same argument as in the case *def*$\mathcal{L}/\circ\mathcal{L}$ we can eliminate the multicuts in Ξ'.

$-/\supset\mathcal{L}$. Suppose Π is

$$\dfrac{\dfrac{\Pi'}{\Sigma; \mathcal{B}_1, \ldots, \mathcal{B}_n, \Gamma' \longrightarrow \sigma \triangleright D'} \quad \dfrac{\Pi''}{\Sigma; \mathcal{B}_1, \ldots, \mathcal{B}_n, \sigma \triangleright D'', \Gamma' \longrightarrow \mathcal{C}}}{\Sigma; \mathcal{B}_1, \ldots, \mathcal{B}_n, \sigma \triangleright D' \supset D'', \Gamma' \longrightarrow \mathcal{C}}\, \supset\mathcal{L}.$$

Let Ξ_1 be

$$\frac{\overset{\Pi_1}{\Sigma;\Delta_1 \longrightarrow \mathcal{B}_1} \quad \cdots \quad \overset{\Pi_n}{\Sigma;\Delta_n \longrightarrow \mathcal{B}_n} \quad \overset{\Pi'}{\Sigma;\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma' \longrightarrow \sigma \triangleright D'}}{\Sigma;\Delta_1,\ldots,\Delta_n,\Gamma' \longrightarrow \sigma \triangleright D'}\ mc$$

and Ξ_2 be

$$\frac{\overset{\Pi_1}{\Sigma;\Delta_1 \longrightarrow \mathcal{B}_1} \quad \cdots \quad \overset{\Pi_n}{\Sigma;\Delta_n \longrightarrow \mathcal{B}_n} \quad \overset{\Pi''}{\Sigma;\mathcal{B}_1,\ldots,\mathcal{B}_n,\sigma \triangleright D'',\Gamma' \longrightarrow \mathcal{C}}}{\Sigma;\Delta_1,\ldots,\Delta_n,\sigma \triangleright D'',\Gamma' \longrightarrow \mathcal{C}}\ mc.$$

Then Ξ reduces to

$$\frac{\overset{\Xi_1}{\Sigma;\Delta_1,\ldots,\Delta_n,\Gamma' \longrightarrow \sigma \triangleright D'} \quad \overset{\Xi_2}{\Sigma;\Delta_1,\ldots,\Delta_n,\sigma \triangleright D'',\Gamma' \longrightarrow \mathcal{C}}}{\Sigma;\Delta_1,\ldots,\Delta_n,\sigma \triangleright D' \supset D'',\Gamma' \longrightarrow \mathcal{C}}\ \supset\mathcal{L}.$$

By similar arguments to the previous cases, i.e., the multiset of heights decreases in Ξ_1 and Ξ_2, the multicuts in Ξ' can be eliminated.

$-/def\mathcal{L}$. If Π is

$$\frac{\left\{\overset{\Pi^{\rho,\mathcal{D}}}{\Sigma\rho;\mathcal{B}_1\rho,\ldots,\mathcal{B}_n\rho,\mathcal{D}\theta,\Gamma'\rho \longrightarrow \mathcal{C}\rho}\right\}}{\Sigma;\mathcal{B}_1,\ldots,\mathcal{B}_n,\mathcal{A},\Gamma' \longrightarrow \mathcal{C}}\ def\mathcal{L}.$$

Then Ξ reduces to

$$\frac{\left\{\dfrac{\left\{\overset{\Pi_i\rho}{\Sigma\rho;\Delta_i\rho \longrightarrow \mathcal{B}_i\rho}\right\}_{i\in\{1..n\}} \quad \overset{\Pi^{\rho,\mathcal{D}}}{\Sigma\rho;\{\mathcal{B}_i\rho\}_{i\in\{1..n\}},\mathcal{D}\theta,\Gamma'\rho \longrightarrow \mathcal{C}\rho}}{\Sigma\rho;\Delta_1\rho,\ldots,\Delta_n\rho,\mathcal{D}\theta,\Gamma'\rho \longrightarrow \mathcal{C}\rho}\ mc\right\}}{\Sigma;\Delta_1,\ldots,\Delta_n,\mathcal{A},\Gamma' \longrightarrow \mathcal{C}}\ def\mathcal{L}.$$

Since $\mathrm{def}(\Pi^{\rho,\mathcal{D}}) < \mathrm{def}(\Pi)$, we can apply the inductive hypothesis to remove the multicuts.

$-/\circ\mathcal{R}$. If Π is

$$\frac{\left\{\overset{\Pi^i}{\Sigma';\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma^i \longrightarrow \mathcal{C}^i}\right\}}{\Sigma;\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}\ \circ\mathcal{R},$$

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

$$\frac{\left\{\dfrac{\overset{\Pi'_1}{\Sigma';\Delta_1 \longrightarrow \mathcal{B}_1} \quad \cdots \quad \overset{\Pi'_n}{\Sigma';\Delta_n \longrightarrow \mathcal{B}_n} \quad \overset{\Pi^i}{\Sigma';\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma^i \longrightarrow \mathcal{C}^i}}{\Sigma';\Delta_1,\ldots,\Delta_n,\Gamma^i \longrightarrow \mathcal{C}^i}\ mc\right\}}{\Sigma;\Delta_1,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\ \circ\mathcal{R}.$$

Here the derivation Π'_i is obtained from Π_i by Lemma 7.2 and hence $\mathrm{ht}(\Pi'_i) \leq \mathrm{ht}(\Pi_i) < \mathrm{ht}(\Pi)$. Therefore the multicuts in the reduct can be then eliminated by induction hypothesis.

Structural cases:

$-/c\mathcal{L}$. If Π is

$$\dfrac{\begin{array}{c}\Pi'\\ \Sigma; \mathcal{B}_1, \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{array}}{\Sigma; \mathcal{B}_1, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\ c\mathcal{L},$$

then Ξ reduces to

$$\dfrac{\begin{array}{ccccc}\Pi_1 & \left\{\begin{array}{c}\Pi_i\\ \Sigma; \Delta_i \longrightarrow \mathcal{B}_i\end{array}\right\}_{i\in\{1..n\}} & \Pi'\\ \Sigma; \Delta_1 \longrightarrow \mathcal{B}_1 & & \Sigma; \mathcal{B}_1, \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{array}}{\dfrac{\Sigma; \Delta_1, \Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ c\mathcal{L}}\ mc.$$

The measure $\mathrm{contr}(\Pi') < \mathrm{contr}(\Pi)$, while the maximum level of cut formulas does not change and $\mathrm{def}(\Pi) = \mathrm{def}(\Pi')$. Therefore $\mu(\Xi') < \mu(\Xi)$ and we can apply the inductive hypothesis to remove the multicut in the reduct.

$-/w\mathcal{L}$. If Π is

$$\dfrac{\begin{array}{c}\Pi'\\ \Sigma; \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{array}}{\Sigma; \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\ c\mathcal{L},$$

then Ξ reduces to

$$\dfrac{\begin{array}{cc}\left\{\begin{array}{c}\Pi_i\\ \Sigma; \Delta_i \longrightarrow \mathcal{B}_i\end{array}\right\}_{i\in\{2..n\}} & \begin{array}{c}\Pi'\\ \Sigma; \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{array}\end{array}}{\dfrac{\Sigma; \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}{\Sigma; \Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ w\mathcal{L}}\ mc.$$

The total size of cut formulas decreases in the reduct, therefore we can apply the inductive hypothesis to remove the multicut.

Axiom cases:

init$/-$. If Π_1 ends with the *init* rule, that is, $\mathcal{B}_1 \in \Delta_1$, then Ξ reduces to

$$\dfrac{\begin{array}{ccccc}\Pi_2 & & \Pi_n & & \Pi\\ \Sigma; \Delta_2 \longrightarrow \mathcal{B}_2 & \cdots & \Sigma; \Delta_n \longrightarrow \mathcal{B}_n & \Sigma; \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{array}}{\dfrac{\Sigma; \mathcal{B}_1, \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}{\Sigma; \Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ w\mathcal{L}}\ mc.$$

The size of cut formulas decreases, while other measures are non-increasing, therefore the multicut can be eliminated by induction hypothesis.

$-/$*init*. If Π ends with the *init* rule and $\mathcal{C}$ is a judgment in Γ, then Ξ reduces to

$$\dfrac{}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ init.$$

If Π ends with the *init* rule, but $\mathcal{C}$ is not a judgment in Γ, then $\mathcal{C}$ must be one of

the cut judgments, say $\mathcal{B}_1$. In this case Ξ reduces to

$$\dfrac{\begin{array}{c}\Pi_1\\ \Sigma; \Delta_1 \longrightarrow \mathcal{B}_1\end{array}}{\Sigma; \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{B}_1}\ w\mathcal{L}$$

□

The following corollary is the cut-elimination result for $FO\lambda^{\Delta\nabla}$ and it is proved by repeatedly removing uppermost cuts in a proof.

COROLLARY 7.6. *Given a fixed stratified definition, a sequent has a proof in $FO\lambda^{\Delta\nabla}$ if and only if it has a cut-free proof.*

Cut-elimination and Lemma 7.3 can be used to show certain permutabilities of inference rules in cut-free proofs. We show the interesting case involving the *def*$\mathcal{L}$ rule.

PROPOSITION 7.7. *Let Π be a cut-free proof of the sequent $\Sigma; \mathcal{A}, \Gamma \longrightarrow \mathcal{C}$, where $\mathcal{A}$ is an atomic judgment. Then there exists a cut-free proof Π' of the same sequent whose last inference rule is an instance of* def$\mathcal{L}$ *applied to $\mathcal{A}$.*

PROOF. Let Ξ be the derivation

$$\dfrac{\left\{\begin{array}{c}\Psi\theta\\ \Sigma\theta; \mathcal{B}\theta, \Gamma\theta \longrightarrow \mathcal{C}\theta\end{array}\right\}_{\mathrm{dfn}(\theta,\mathcal{A},\rho,\mathcal{B})}}{\Sigma; \mathcal{A}, \Gamma \longrightarrow \mathcal{C}}\ def\mathcal{L}$$

where $\Psi\theta$ is the derivation

$$\dfrac{\dfrac{\dfrac{}{\Sigma\theta; \mathcal{B}\rho \longrightarrow \mathcal{B}\rho}\ init}{\Sigma\theta; \mathcal{B}\rho \longrightarrow \mathcal{A}\theta}\ def\mathcal{R} \qquad \begin{array}{c}\Pi\theta\\ \Sigma\theta; \mathcal{A}\theta, \Gamma\theta \longrightarrow \mathcal{C}\theta\end{array}}{\Sigma\theta; \mathcal{B}\rho, \Gamma\theta \longrightarrow \mathcal{C}\theta}\ mc.$$

The premise derivation $\Pi\theta$ in $\Psi\theta$ is obtained from Π by Lemma 7.3. The cut-free proof Π' is obtained by applying cut-elimination procedure on Ξ. Note that the last rule of Ξ is unchanged by cut-elimination and hence Π' ends with *def*$\mathcal{L}$. □

Since *def*$\mathcal{L}_{csu}$ is a special case of *def*$\mathcal{L}$, the above proposition holds as well if *def*$\mathcal{L}$ is replaced by *def*$\mathcal{L}_{csu}$.

COROLLARY 7.8. *Let Π be a cut-free proof of the sequent $\Sigma; \mathcal{A}, \Gamma \longrightarrow \mathcal{C}$, where $\mathcal{A}$ is an atomic judgment. Then there exists a cut-free proof Π' of the same sequent whose last inference rule is an instance of* def$\mathcal{L}_{csu}$ *applied to $\mathcal{A}$.*

7.2 Properties of ∇

We see from examples in Section 5 and Section 6 that ∇ and $\forall$ are significantly different when they are used negatively in a proof, i.e., when it appears to the left of certain implications in the proof. We shall now show that when definitions are essentially Horn clauses (recall that in Horn clauses, there are no occurrences of implication in the bodies of the clauses), the difference between ∇ and $\forall$ cannot actually be observed. In particular, we show that ∇ and $\forall$ can be interchanged

for $\mathsf{hc}^{\forall\nabla}$-definitions and $\mathsf{hc}^{\forall\nabla}$-goals without affecting provability. In proving this statement inductively we need a stronger hypothesis: that is, we can interchange the scope of variables in this case (either global or local) without affecting provability.

LEMMA 7.9. *Let* $\mathbf{D}$ *be an* $\mathsf{hc}^{\forall\nabla}$*-definition, and let* G *be an* $\mathsf{hc}^{\forall\nabla}$*-goal. The sequent* $\Sigma; . \longrightarrow (\sigma_1, x, \sigma_2) \triangleright G$ *is provable if and only if the sequent*

$$\Sigma, h; . \longrightarrow (\sigma_1\sigma_2) \triangleright G[(h\,\sigma_1)/x]$$

is provable. Moreover, given a derivation Π *of the first sequent, there is a derivation* Π' *of the second sequent such that* $\mathrm{ht}(\Pi') \leq \mathrm{ht}(\Pi)$*, and vice versa.*

PROOF. We show that given a derivation Π of one sequent, we can construct a derivation Π' of the other sequent by induction on $\mathrm{ht}(\Pi)$. In the transformation, there is no extra rules introduced, therefore $\mathrm{ht}(\Pi') \leq \mathrm{ht}(\Pi)$. We show here the non-trivial cases where the derivation Π ends with either $\forall\mathcal{R}$ or $def\mathcal{R}$.

Let Π be a derivation of $\Sigma; . \longrightarrow (\sigma_1, x, \sigma_2) \triangleright G$. Then we construct a derivation Π' of $\Sigma, h; . \longrightarrow (\sigma_1\sigma_2) \triangleright G[(h\,\sigma_1)/x]$ as follows. First, suppose that Π ends with $\forall\mathcal{R}$, that is,

$$\frac{\begin{array}{c}\Pi_1\\ \Sigma, f; . \longrightarrow (\sigma_1, x, \sigma_2) \triangleright G'[(f\,\sigma_1\,x\,\sigma_2)/y]\end{array}}{\Sigma; . \longrightarrow (\sigma_1, x, \sigma_2) \triangleright \forall y.G'}\ \forall\mathcal{R}.$$

Applying the substitution $[\lambda\sigma_1\lambda x\lambda\sigma_2.f'\,\sigma_1\,\sigma_2/f]$ to Π_1, where f' is a new eigenvariable, we obtain a derivation Ξ of $\Sigma, f'; . \longrightarrow (\sigma_1, x, \sigma_2) \triangleright G'[(f'\,\sigma_1\,\sigma_2)/y]$. By Lemma 7.3 substitution does not increase the height of derivation, therefore, the induction hypothesis can be applied to Ξ to get a derivation Ξ' of

$$\Sigma, h, f'; . \longrightarrow (\sigma_1\sigma_2) \triangleright G'[(h\,\sigma_1\,\sigma_2)/x, (f'\,\sigma_1\,\sigma_2)/y].$$

We can, therefore, take the following derivation as Π'

$$\frac{\begin{array}{c}\Xi'\\ \Sigma, h, f'; . \longrightarrow (\sigma_1\sigma_2) \triangleright G'[(h\,\sigma_1)/x, (f'\,\sigma_1\,\sigma_2)/y]\end{array}}{\Sigma, h; . \longrightarrow (\sigma_1\sigma_2) \triangleright \forall y.G'[(h\,\sigma_1)/x]}\ \forall\mathcal{R}.$$

Second, suppose that Π ends with $def\mathcal{R}$

$$\frac{\begin{array}{c}\Pi_1\\ \Sigma; . \longrightarrow (\sigma_1, x, \sigma_2) \triangleright D\theta\end{array}}{\Sigma; . \longrightarrow (\sigma_1, x, \sigma_2) \triangleright A}\ def\mathcal{R}$$

where $\forall w_1 \ldots w_n.[\sigma_1, x, \sigma_2 \triangleright H \stackrel{\triangle}{=} \sigma_1, x, \sigma_2 \triangleright D]$ is the raised definition clause matching $\sigma_1, x, \sigma_2 \triangleright A$, that is, $\lambda\sigma_1\lambda x\lambda\sigma_2.A =_{\beta\eta} (\lambda\sigma_1\lambda x\lambda\sigma_2.H)\theta$. We can assume without loss of generality that the substitution θ is of the form

$$\{\lambda\sigma_1\lambda x\lambda\sigma_2.t_1/w_1, \ldots, \lambda\sigma_1\lambda x\lambda\sigma_2.t_n/w_n\}.$$

Let us define a substitution γ as follows

$$\gamma = \{\lambda\sigma_1\lambda x\lambda\sigma_2.(u_1\,\sigma_1\sigma_2)/w_1, \ldots, \lambda\sigma_1\lambda x\lambda\sigma_2.(u_n\,\sigma_1\sigma_2)/w_n\}.$$

where $u_1, \ldots, u_n$ are new variables different from $\bar{w}$ and σ_1, x, σ_2. The corresponding raised definition clause for $\sigma_1\sigma_2 \triangleright A[(h\,\sigma_1)/x]$ is

$$\forall u_1 \ldots u_n.[\sigma_1\sigma_2 \triangleright H\gamma \stackrel{\triangle}{=} \sigma_1\sigma_2 \triangleright D\gamma].$$

It can be verified that the equation

$$(\lambda\sigma_1\lambda\sigma_2.A[(h\,\sigma_1)/x]) =_{\beta\eta} (\lambda\sigma_1\lambda\sigma_2.H\gamma)\rho$$

holds for $\rho = \{(\lambda\sigma_1\lambda\sigma_2.t_1[(h\,\sigma_1)/x])/u_1, \ldots, (\lambda\sigma_1\lambda\sigma_2.t_n[(h\,\sigma_1)/x])/u_n\}$.

Notice that $D\theta[(h\,\sigma_1)/x] =_{\beta\eta} D\gamma\rho$. Therefore, we construct Π' as the derivation

$$\frac{\begin{array}{c}\Pi'_1\\ \Sigma, h; . \longrightarrow (\sigma_1\sigma_2) \triangleright D\gamma\rho\end{array}}{\Sigma, h; . \longrightarrow \sigma_1\sigma_2 \triangleright A[(h\,\sigma_1)/x]}\ def\mathcal{R},$$

where Π'_1 is obtained by induction hypothesis.

Conversely, from the derivation Π' we construct the derivation of Π as follows. Let us assume that x is not in Σ. We first notice that the problem can be simplified by removing the dependency of h on σ_1; that is, by applying the substitution $[\lambda\sigma_1.x/h]$ to Π'. We can, therefore, suppose a simpler case where Π' is a derivation of $\Sigma, x; . \longrightarrow \sigma_1\sigma_2 \triangleright G$. We examine the following two non-trivial cases.

Suppose Π' ends with $\forall\mathcal{R}$

$$\frac{\begin{array}{c}\Pi_1\\ \Sigma, x, f; . \longrightarrow \sigma_1\sigma_2 \triangleright G'[(f\,\sigma_1\sigma_2)/y]\end{array}}{\Sigma, x; . \longrightarrow \sigma_1\sigma_2 \triangleright \forall y.G'}\ \forall\mathcal{R}.$$

Applying the substitution $[(\lambda\sigma_1\lambda\sigma_2.f'\,\sigma_1\,x\,\sigma_2)/f]$ to derivation Π_1, we get a derivation Π_2 of $\Sigma, x, f'; . \longrightarrow \sigma_1\sigma_2 \triangleright G'[(f'\,\sigma_1\,x\,\sigma_2)/y]$. The derivation Π is then

$$\frac{\begin{array}{c}\Pi'_2\\ \Sigma, f'; . \longrightarrow \sigma_1, x, \sigma_2 \triangleright G'[(f\,\sigma_1\,x\,\sigma_2)/y]\end{array}}{\Sigma; . \longrightarrow \sigma_1, x, \sigma_2 \triangleright \forall y.G'}\ \forall\mathcal{R},$$

where Π'_2 is obtained by applying the induction hypothesis to Π_2.

For the second case, suppose Π' ends with $def\mathcal{R}$

$$\frac{\begin{array}{c}\Pi_1\\ \Sigma, x; . \longrightarrow \sigma_1\sigma_2 \triangleright D\theta\end{array}}{\Sigma, x; . \longrightarrow \sigma_1\sigma_2 \triangleright A}\ def\mathcal{R}$$

where $\forall w_1 \ldots \forall w_n.[\sigma_1\sigma_2 \triangleright H \stackrel{\triangle}{=} \sigma_1\sigma_2 \triangleright D]$ is the matching definition clause, i.e., $\lambda\sigma_1\lambda\sigma_2.A =_{\beta\eta} (\lambda\sigma_1\lambda\sigma_2.H)\theta$. As in the previous case we can suppose that θ is of the form

$$\{(\lambda\sigma_1\lambda\sigma_2.t_1)/w_1, \ldots, (\lambda\sigma_1\lambda\sigma_2.t_n)/w_n\}.$$

The corresponding raised definition clause for the judgment $\sigma_1, x, \sigma_2 \triangleright A$ is

$$\forall u_1 \ldots \forall u_n.[\sigma_1\,x\,\sigma_2 \triangleright H\gamma \stackrel{\triangle}{=} \sigma_1\,x\,\sigma_2 \triangleright D\gamma],$$

where $\gamma = \{(\lambda\sigma_1\lambda\sigma_2.u_1\,\sigma_1\,x\,\sigma_2)/w_1, \ldots, (\lambda\sigma_1\lambda\sigma_2.u_n\,\sigma_1\,x\,\sigma_2)/w_n, \}$. Let ρ be the substitution

$$\{\lambda\sigma_1\lambda x\lambda\sigma_2.t_1, \ldots, \lambda\sigma_1\lambda x\lambda\sigma_2.t_n\}.$$

It can be verified that the equation $(\lambda\sigma_1\lambda x\lambda\sigma_2.A) =_{\beta\eta} (\lambda\sigma_1\lambda x\lambda\sigma_2.D\gamma)\rho$ holds, and

that $D\gamma\rho = D\theta$. Therefore, we construct Π as the derivation

$$\frac{\begin{array}{c}\Pi_1' \\ \Sigma; . \longrightarrow \sigma_1, x, \sigma_2 \triangleright D\theta\end{array}}{\Sigma; . \longrightarrow \sigma_1, x, \sigma_2 \triangleright A}\ def\mathcal{R}$$

where Π_1' is obtained from induction hypothesis. □

PROPOSITION 7.10. *Let* $\mathbf{D}$ *be an* $\mathsf{hc}^{\forall\nabla}$*-definition and let* $\mathbf{D}'$ *be the* $\mathsf{hc}^{\forall\nabla}$*-definition resulting from replacing some occurrences of* $\forall$ *and* ∇ *in the body of clauses of* $\mathbf{D}$ *with* ∇ *and* $\forall$*, respectively. Similarly, let* G *be an* $\mathsf{hc}^{\forall\nabla}$*-goal and let* G' *be the* $\mathsf{hc}^{\forall\nabla}$*-goal resulting from replacing some occurrences of* $\forall$ *and* ∇ *in* G *with* ∇ *and* $\forall$*, respectively. If the sequent* $\Sigma; \cdot \longrightarrow \sigma \triangleright G$ *is provable using definition* $\mathbf{D}$ *then the sequent* $\Sigma; \cdot \longrightarrow \sigma \triangleright G'$ *is provable using definition* $\mathbf{D}'$*.*

PROOF. Let Π be a derivation of $\Sigma; \cdot \longrightarrow \sigma \triangleright G$. We construct a derivation Π' of $\Sigma; \cdot \longrightarrow \sigma \triangleright G'$ by induction on the measure $\mathrm{ht}(\Pi)$. The non-trivial cases are when Π ends with the introduction rule for the connective being interchanged.

Suppose $G = \forall x.H$, $G' = \nabla x.H'$ and Π ends with $\forall\mathcal{R}$.

$$\frac{\begin{array}{c}\Pi_1 \\ \Sigma, h; . \longrightarrow \sigma \triangleright H[(h\,\sigma)/x]\end{array}}{\Sigma; . \longrightarrow \sigma \triangleright \forall x.H}\ \forall\mathcal{R}$$

By Lemma 7.9 there is a derivation Π_1' of $\Sigma; . \longrightarrow (\sigma, x) \triangleright H$ such that $\mathrm{ht}(\Pi_1') \leq \mathrm{ht}(\Pi_1)$. We can, therefore, apply the induction hypothesis to Π_1' to get a derivation Π_2 of $\Sigma; . \longrightarrow (\sigma, x) \triangleright H'$. The derivation Π' is, therefore,

$$\frac{\begin{array}{c}\Pi_2 \\ \Sigma; . \longrightarrow (\sigma, x) \triangleright H'\end{array}}{\Sigma; . \longrightarrow \sigma \triangleright \nabla x.H'}\ \nabla\mathcal{R}$$

The case where $G = \nabla x.H$, $G' = \forall x.H'$ and Π ends with $\nabla\mathcal{R}$ is done analogously, since Lemma 7.9 works on both directions. □

As a consequence of this proposition, the difference between $\forall$ and ∇ (or, equivalently, between the global and local signatures of a sequent) cannot be seen if one is simply attempting to "evaluate" $\mathsf{hc}^{\forall}$ logical programs by determining the atoms that they can prove. To illustrate the difference between these two quantifiers, we need to consider goals and/or definitions that contain implications. We have done this in Section 5, for example, when we illustrated the differences between $\forall$ and ∇ with the specification of simulation in the π-calculus.

In Figure 4 we presented eight non-theorems of $FO\lambda^{\nabla}$ and claimed that, with certain restrictions, the last three are provable. For a fixed *noetherian* definition (see the following Definition), we claim the following: formula (8) is provable and if the definition is furthermore $\mathsf{hc}^{\forall\nabla}$ then formulas (6) and (7) are also provable. The fact that formula (8) is a theorem of $FO\lambda^{\Delta\nabla}$ for noetherian definitions follows from Proposition 7.12. The proof of the provability of formulas (6) and (7) follows by similarly structured proofs.

Definition 7.11. A definition $\mathbf{D}$ is *noetherian* if for every definition clause $\forall\bar{x}.[p\bar{t} \stackrel{\triangle}{=} B]$ in $\mathbf{D}$, it holds that $\mathrm{lvl}(p) > \mathrm{lvl}(B)$.

PROPOSITION 7.12. *Given a noetherian definition, the sequent*

$$\Sigma; \Gamma, \sigma \triangleright B \longrightarrow \sigma' \triangleright B,$$

where σ' is a permutation of σ, is provable in $FO\lambda^{\Delta\nabla}$.

PROOF. We construct a derivation of $\Gamma, \sigma \triangleright B \longrightarrow \sigma' \triangleright B$ inductively. The induction is on the level of B with subordinate induction on the size of B. We can assume without loss of generality that all predicates in the definition are assigned levels greater than 0 and recall that we require all predicates to be defined. The cases where B is a non-atomic formula are straightforward; we just apply the introduction rules for the outermost connective in B, coordinated between left and right rules. In the case where B is an atomic formula, suppose that $\mathrm{dfn}(\rho, \sigma \triangleright B, \theta, \sigma \triangleright D)$ holds for a clause $\forall h_1, \ldots, h_n.[\sigma \triangleright H \triangleq \sigma \triangleright D]$, that is, $(\lambda\sigma.B)\rho =_{\beta\eta} (\lambda\sigma.H)\theta$. Let δ be the substitution $\{(\lambda\sigma.h'_1\sigma')/h_1, \ldots, (\lambda\sigma.h'_n\sigma')/h_n\}$. It suffices to show that there is a substitution γ such that $\mathrm{dfn}(\epsilon, (\sigma' \triangleright B)\rho, \gamma, \sigma' \triangleright D\delta)$ holds for the raised clause $\forall h'_1, \ldots, h'_n.[\sigma \triangleright H\delta \triangleq \sigma \triangleright D\delta]$. The following substitution solves the matching:

$$\gamma(x) = \begin{cases} \lambda\sigma'.(h_i\theta)\ \sigma, \text{ if } x = h_i \text{ for some } i \in \{1, \ldots, n\}, \\ \theta(x), \text{ otherwise.} \end{cases}$$

□

We conjecture that if we incorporated into our proof system an appropriate induction inference rule, then the restriction of noetherian can be removed from Proposition 7.12 and from the claims made for formulas (6) and (7) of Figure 4.

8. RELATED WORK AND CONCLUSION

We have considered the approach to the specification of computation in which term-level and proof-level abstractions are used to encode abstractions both of the static structure of expressions (e.g., using meta-level λ-abstractions to encode the input prefix in the π-calculus) and the dynamic structure of computation (e.g., name generation as eigenvariables). While this style of syntactic representation has been successfully used to enumerate judgments about operational semantics and to encode object-logic provability, traditional proof-level abstractions (eigenvariables) seem inadequate when one wishes to reason about computation directly (as outlined in Section 1). We have explored a simple mechanism within sequent calculus to expand the notion of abstraction in the building of proofs. Our approach to the ∇ quantifier is thus not an attempt at a notion of name "freshness" or a semantics for "name generation".

It is natural to ask about possible connections between the ∇-quantifier and the new quantifier of Pitts and Gabbay [Gabbay and Pitts 2001; Pitts 2003]. Both are self dual and both have similar sets of applications in mind. The focus on ∇ has been proof theoretic while the work on Pitts and Gabbay has been model theoretic. More concretely, while ∇ neither implies nor is implied by $\forall$ or $\exists$, the quantifier of Pitts and Gabbay is entailed by $\forall$ and entails $\exists$. In Pitts and Gabbay, the domain of quantification is fixed to a certain denumerably infinite set of names, while the ∇ quantifier works at any type. In their recent paper [Gabbay and Cheney 2004], Gabbay and Cheney provide some initial connections between these two quantifiers.

Pursuing such a connection might help provide a model theoretic semantics for ∇ and for $FO\lambda^{\Delta\nabla}$ more generally.

Formal logic has also been used as a framework for meta-theoretic reasoning about dependent type systems. The closest such work to $FO\lambda^{\Delta\nabla}$ is probably Schürmann's $\mathcal{M}_2^+$ logic for reasoning about the LF system [Schürmann 2000]. The logic $\mathcal{M}_2^+$ allows richer definitions of object-systems, compared to $FO\lambda^{\Delta\nabla}$, since they are not subject to the stratification using levels. Instead, definitions in $\mathcal{M}_2^+$ are stratified using something called "regular worlds assumption". Translated to our setting, this feature would permit, in particular, some unstratifiable definitions. Possible connections between these two systems is left for future work.

To work with interesting examples, an implementation of $FO\lambda^{\Delta\nabla}$ is needed. An outline for such implementation is discussed in [Tiu and Miller 2004] and a prototype implementation of a subset of $FO\lambda^{\Delta\nabla}$ following this outline has been built using the functional language Standard ML [Tiu 2004a]. Using this prover, we were able to write the specifications of the transition system of π-calculus and the open bisimulation relation given in [Tiu and Miller 2004] and automatically check for open bisimilarity for finite processes. The Isabelle theorem prover might also provide a setting for building an interactive theorem prover given the work reported in [Momigliano et al. 2002].

A natural next step involves adding directly to $FO\lambda^{\Delta\nabla}$ both induction and co-induction. A preliminary step in that direction appears in [Tiu 2004b] and follows the earlier work on induction in the $FO\lambda^{\Delta\mathbb{N}}$ logic [McDowell and Miller 2000]. Closely related work involving induction and co-induction but without ∇ in appears in [Momigliano and Tiu 2003].

Acknowledgments An earlier version of this paper appeared as [Miller and Tiu 2003]. The authors wish to thank Catuscia Palamidessi for valuable discussions regarding our π-calculus examples, Frank Pfenning for his comments on the general project of this paper, Gopalan Nadathur for his comments a draft of this paper, and an anonymous reviewer of this paper for many useful comments and suggestions. This work has been supported in part by NSF grant CCR-9912387 and the ACI grants GEOCAL and Rossignol. The second author gratefully acknowledges support from LIX at École polytechnique.

REFERENCES

Cervesato, I., Durgin, N. A., Lincoln, P. D., Mitchell, J. C., and Scedrov, A. 1999. A meta-notation for protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop — CSFW'99*, R. Gorrieri, Ed. IEEE Computer Society Press, Mordano, Italy, 55–69.

Cervesato, I. and Pfenning, F. 1996. A linear logic framework. In *Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, New Brunswick, New Jersey, 264–275.

Chirimar, J. 1995. Proof theoretic approach to specification languages. Ph.D. thesis, University of Pennsylvania.

Church, A. 1940. A formulation of the simple theory of types. *J. of Symbolic Logic 5*, 56–68.

Eriksson, L.-H. 1991. A finitary version of the calculus of partial inductive definitions. In *Proc. of the Second International Workshop on Extensions to Logic Programming*, L.-H. Eriksson, L. Hallnäs, and P. Schroeder-Heister, Eds. LNAI, vol. 596. Springer-Verlag, 89–134.

Felty, A. and Miller, D. 1988. Specifying theorem provers in a higher-order logic programming language. In *Ninth International Conference on Automated Deduction*. Springer-Verlag, Argonne, IL, 61–80.

Gabbay, M. J. and Cheney, J. 2004. A sequent calculus for nominal logic. In *Proc. 19th IEEE Symposium on Logic in Computer Science (LICS 2004)*. 139–148.

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

Gentzen, G. 1969. Investigations into logical deductions. In *The Collected Papers of Gerhard Gentzen*, M. E. Szabo, Ed. North-Holland Publishing Co., Amsterdam, 68–131.

Girard, J.-Y. 1992. A fixpoint theorem in linear logic. Email to the linear@cs.stanford.edu mailing list.

Hallnäs, L. and Schroeder-Heister, P. 1991. A proof-theoretic approach to logic programming. II. Programs as definitions. *Journal of Logic and Computation 1,* 5 (October), 635–660.

Huet, G. 1975. A unification algorithm for typed λ-calculus. *Theoretical Computer Science 1*, 27–57.

McDowell, R. 1997. Reasoning in a logic with definitions and induction. Ph.D. thesis, University of Pennsylvania.

McDowell, R. and Miller, D. 2000. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science 232*, 91–119.

McDowell, R. and Miller, D. 2002. Reasoning with higher-order abstract syntax in a logical framework. *ACM Transactions on Computational Logic 3,* 1 (January), 80–136.

McDowell, R., Miller, D., and Palamidessi, C. 2003. Encoding transition systems in sequent calculus. *Theoretical Computer Science 294,* 3, 411–437.

Miller, D. 1989. Lexical scoping as universal quantification. In *Sixth International Logic Programming Conference*. MIT Press, Lisbon, Portugal, 268–283.

Miller, D. 1991. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation 1,* 4, 497–536.

Miller, D. 1992. Unification under a mixed prefix. *J. of Symbolic Computation 14,* 4, 321–358.

Miller, D. 1993. The π-calculus as a theory in linear logic: Preliminary results. In *Proceedings of the 1992 Workshop on Extensions to Logic Programming*, E. Lamma and P. Mello, Eds. Number 660 in LNCS. Springer-Verlag, Bologna, Italy, 242–265.

Miller, D. 1996. Forum: A multiple-conclusion specification language. *Theoretical Computer Science 165,* 1 (Sept.), 201–232.

Miller, D. 2000. Abstract syntax for variable binders: An overview. In *Computational Logic - CL 2000*, J. Lloyd and et. al., Eds. Number 1861 in LNAI. Springer, 239–253.

Miller, D. and Palamidessi, C. 1999. Foundational aspects of syntax. In *ACM Computing Surveys Symposium on Theoretical Computer Science: A Perspective*, P. Degano, R. Gorrieri, A. Marchetti-Spaccamela, and P. Wegner, Eds. Vol. 31. ACM.

Miller, D. and Tiu, A. 2002. Encoding generic judgments. In *Proceedings of FSTTCS*. Number 2556 in LNCS. Springer-Verlag, 18–32.

Miller, D. and Tiu, A. 2003. A proof theory for generic judgments: An extended abstract. In *Proc. 18th IEEE Symposium on Logic in Computer Science (LICS 2003)*. IEEE, 118–127.

Milner, R., Parrow, J., and Walker, D. 1992. A calculus of mobile processes, Part I. *Information and Computation 100,* 1 (September), 1–40.

Momigliano, A., Ambler, S., and Crole, R. 2002. A hybrid encoding of Howe's method for establishing congruence of bisimilarity. In *LFM'02*. ENTCS, vol. 70.2. Springer-Verlag.

Momigliano, A. and Tiu, A. 2003. Induction and co-induction in sequent calculus. In *Proceedings of TYPES 2003 Workshop*. LNCS, vol. 3085. Springer, 293 – 308.

Nipkow, T. 1993. Functional unification of higher-order patterns. In *Proc. 8th IEEE Symposium on Logic in Computer Science (LICS 1993)*, M. Vardi, Ed. IEEE, 64–74.

Paulson, L. C. 1989. The foundation of a generic theorem prover. *Journal of Automated Reasoning 5*, 363–397.

Pfenning, F. and Elliott, C. 1988. Higher-order abstract syntax. In *Proc. of the ACM-SIGPLAN Conf. on Prog. Language Design and Implementation*. ACM Press, 199–208.

Pfenning, F. and Rohwedder, E. 1992. Implementing the meta-theory of deductive systems. In *Proceedings of the 1992 Conference on Automated Deduction*. Number 607 in LNCS. Springer, 537–551.

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

Schroeder-Heister, P. 1992. Cut-elimination in logics with definitional reflection. In *Non-classical Logics and Information Processing*, D. Pearce and H. Wansing, Eds. LNCS, vol. 619. Springer, 146–171.

Schroeder-Heister, P. 1994. Cut elimination for logics with definitional reflection and restricted initial sequents. In *Proceedings of the Post-Conference Workshop of ICLP 1994 on Proof-Theoretic Extensions of Logic Programming*.

Schürmann, C. 2000. Automating the meta theory of deductive systems. Ph.D. thesis, Carnegie Mellon University.

Slaney, J. 1989. Solution to a problem of Ono and Komori. *J. of Philosophic Logic 18*, 103–111.

Tiu, A. 2004a. Level 0/1 prover: A tutorial. Available via the web and from Tiu.

Tiu, A. 2004b. A logical framework for reasoning about logical specifications. Ph.D. thesis, Pennsylvania State University.

Tiu, A. and Miller, D. 2004. A proof search specification of the π-calculus. In *Proceedings of the 3rd Workshop on the Foundations of Global Ubiquitous Computing*.

Received November 2003; first revision August 2004; accepted November 2004.