An Anti-Locally-Nameless Approach to Formalizing Quantifiers

Olivier Laurent
Univ Lyon, EnsL, UCBL, CNRS, LIP
LYON, France
olivier.laurent@ens-lyon.fr

Abstract

We investigate the possibility of formalizing quantifiers in proof theory while avoiding, as far as possible, the use of true binding structures, α-equivalence or variable renamings. We propose a solution with two kinds of variables in terms and formulas, as originally done by Gentzen. In this way formulas are first-order structures, and we are able to avoid capture problems in substitutions. However at the level of proofs and proof manipulations, some binding structure seems unavoidable. We give a representation with de Bruijn indices for proof rules which does not impact the formula representation and keeps the whole set of definitions first-order.

CCS Concepts: • **Theory of computation → Proof theory**.

Keywords: quantifiers, proof assistant, formalization, proof theory, first-order logic, normalization

ACM Reference Format:
Olivier Laurent. 2021. An Anti-Locally-Nameless Approach to Formalizing Quantifiers. In *Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '21), January 18–19, 2021, Virtual, Denmark.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3437992.3439926

1 Introduction

The formalization, using proof assistants, of (syntactic) results in proof theory is something appealing since they often require a lot of (simple) cases to be considered. For example a proof of (strong) normalization often uses a clever measure on proofs, and then goes through checking that an important number of proof transformations (often described as local rewrite steps) makes the measure decrease. Once the measure is settled, it is interesting to use the machine to help dealing with the simple rewrite steps and to ensure that all cases have been covered. This works perfectly well for various propositional systems, but as soon as (first-order) quantifiers enter the picture, we have to face one of the nightmares of formalization: quantifiers are binders... The question of the formalization of binders does not yet have an answer which makes perfect consensus[1]. We are *not* going to address this general question. The main (more specific) problem addressed here is to try to find dedicated ways of formalizing *quantifiers* (and not arbitrary binders). The point is to rely on particular properties of quantifiers to lead to a simpler and convincing solution.

Concretely, a major trouble with many formalizations of binders is the notion of α-equivalence to be dealt with (or alternatively the use of some notion of variable renaming). We want to study how far it is possible to go in the formalization of proof theory while avoiding α-equivalence and variable renaming. Additionally we would like an approach formalizable with first-order constructs only, in order to be applicable in (a priori) any proof assistant, independently of the underlying meta-theory. Finally we would like our formalization to be close to what could be done on paper in order to be understandable by most logicians. This is particularly important in the case of the development of libraries supposed to be accessible to the widest possible audience.

Following these criteria, we want to avoid relying on advanced logical approaches which, while extremely interesting from the abstract point of view of formalization, are not versatile enough (their usability depends on the chosen proof assistant) and too far from the common practice of logicians on paper. This is for example the case of HOAS [16], nominal approaches [17], the ∇-quantifier [14], and even of de Bruijn representations [3] which do not need advanced logical principles but are considered unreadable by most people working usually on paper.

The reason for dealing with α-equivalence (or variable renaming) in the formalization of binders is to avoid variable-capture problems in substitutions [15]. A recurrent approach to control these phenomena without renamings, is to rely on two different kinds of variables. This has been proposed historically in major works on proof theory (Gentzen [6], Prawitz [18]) for quantifiers, but also more recently in proposals for formalizing general binders like in the locally

[1] https://www.seas.upenn.edu/%7eplclub/poplmark/ [1]

nameless (*lnl*) [2] and locally named (*ln*) [13] representations.

This two-kinds-of-variables approach has been successfully applied in the formalization of proof theory on machine (see for example [4, 10]). Our representation of terms and formulas will be very close to those works. However in [4, 10], proofs are mainly used as static objects. Almost no proof transformation is considered. If such transformations were required (as in syntactic proof theory we want to address), they would have to face some difficulties. This is testified by the fact that the weakening lemma (possibly one of the simplest proof transformations one can imagine) requires some non trivial work for the systems they consider (we will come back to this point in Section 3.1). While not modifying the representation of formulas and sequents, our proposal is to introduce a restricted use of de Bruijn indices in the construction of proof rules. We show that:

- without a full management of indices as in de Bruijn representations for general binders,
- with limited impact on the representation of formulas with respect to paper notations,
- while dealing with first-order structures only,

we obtain a good compromise between complexity and readability for dealing with general syntactic proof transformations appearing in corner-stone results of proof theory, such as cut elimination and proof normalization for example.

Related Works. As already mentioned, approaches based on specific logical constructions [14, 16, 17] provide very nice formalizations of binders and quantifiers but are dependent on the available meta-theory (thus not suited for arbitrary proof assistants). Moreover, in the context of proposing libraries for proof theory in proof assistants, not all users would accept to dig into such specific constructions.

Let us now focus on approaches which are mostly expressible with first-order structures. The de Bruijn indices approach [3] has the interesting properties of being very general and of requiring only first-order constructs in the meta-theory. Various formalizations of predicate logic follow this methodology by using indices both in formulas and in the definition of generalization rules [5, 8, 9]. The main defects are the induced manipulations of indices to be dealt with, in particular for substitutions. Moreover formulas written this way are often difficult to read by people used to paper notations with names. For example, the formula $\forall x.\exists y.(Pxy \to \forall z.Pxz)$ is represented as $\underline{\forall}.\exists.(P\underline{1}0 \to \forall.P\underline{2}0)$. In particular different occurrences of the same variable (the underlined ones) may be given different names (here 1 and 2). A similar remark applies to the locally nameless (lnl) approaches [2] which use indices in formulas for bound variables.

In trying to use names but avoiding α-equivalence and variable renamings, the natural proposal is the locally named approach (ln) [13]. It has already been used for the formalization of predicate logic [4, 10]. Two kinds of names are used, and no particular properties of the names used for free variables (also called parameters) are required. Up to an appropriate policy regarding free names in substituting terms (they have to be parameters), a renaming-free notion of substitution can be used without generating captures. Our representation of terms and formulas is simply the particular choice of the ln approach with $\mathbb{N}$ as underlying name-set for free variables. Since it is the opposite choice with respect to the lnl approach which uses $\mathbb{N}$ for bound variables and names for free ones, we call our approach the *anti-locally-nameless* (*alnl*) approach.

Table 1. Different Styles of Generalization Rules

$$\dfrac{\Gamma \vdash A \qquad x \text{ not free in } \Gamma}{\Gamma \vdash \forall x.A}\ \forall I$$

Kleene

$$\dfrac{\Gamma \vdash A[{}^{e}/_{x}] \qquad e \text{ not in } \Gamma, A}{\Gamma \vdash \forall x.A}\ \forall I$$

Gentzen

$$\dfrac{\Gamma\!\uparrow\, \vdash A}{\Gamma \vdash \forall.A}\ \forall I \qquad\qquad \dfrac{\Gamma\!\uparrow\, \vdash A\!\uparrow\![{}^{0}/_{x}]}{\Gamma \vdash \forall x.A}\ \forall I$$

de Bruijn alnl

Let us now look at proofs and proof rules. We focus on an example where no capture problems happen. Here is a proof in sequent calculus of the sequent $\exists x.\forall y.Pxy \vdash \forall y.\exists x.Pxy$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P\underline{x}\overline{y} \vdash P\underline{x}\overline{y}}\ ax}{P\underline{x}\overline{y} \vdash \exists x.Px\overline{y}}\ \exists I}{\forall y.P\underline{x}y \vdash \exists x.Px\overline{y}}\ \forall L}{\exists x.\forall y.Pxy \vdash \exists x.Px\overline{y}}\ \underline{\exists L}}{\exists x.\forall y.Pxy \vdash \forall y.\exists x.Pxy}\ \overline{\forall I}$$

It uses formulas represented with a single kind of names. The generalization rules ($\forall I$ and $\exists L$) follow the Kleene-style presentation (see Table 1). Overline and underline markings are just given to help the reader follow variables, they are not part of the proofs.

An alternative approach is to use two kinds of names (ln approach) and Gentzen-style generalization rules (Table 1):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P\underline{e'}\overline{e} \vdash P\underline{e'}\overline{e}}\ ax}{P\underline{e'}\overline{e} \vdash \exists x.Px\overline{e}}\ \exists I}{\forall y.P\underline{e'}y \vdash \exists x.Px\overline{e}}\ \forall L}{\exists x.\forall y.Pxy \vdash \exists x.Px\overline{e}}\ \underline{\exists L}}{\exists x.\forall y.Pxy \vdash \forall y.\exists x.Pxy}\ \overline{\forall I}$$

The de Bruijn approach would correspond to proving $\exists.\forall.P10 \vdash \forall.\exists.P01$ with the associated de Bruijn-style generalization rules (Table 1):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P\underline{0}\overline{1} \vdash P\underline{0}\overline{1}}\;ax}{P\underline{0}\overline{1} \vdash \exists.P0\overline{2}}\;\exists I}{\forall.P\underline{1}0 \vdash \exists.P0\overline{2}}\;\forall L}{\exists.\forall.P10 \vdash \exists.P0\overline{1}}\;\underline{\exists L}}{\exists.\forall.P10 \vdash \forall.\exists.P01}\;\overline{\forall I}$$

Note that, in the named proofs (the first two), all the occurrences of the same (free) variable (overlined occurrences or underlined occurrences) in the proof are given the same notation. Contrariwise, in the de Bruijn approach, the notations associated with a variable may vary inside proofs and even inside sequents or formulas. Our proposal lives in between. It relies on the new alnl-style generalization rules (Table 1) based on the fact the we use natural numbers for free variables:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P\underline{0}\overline{1} \vdash P\underline{0}\overline{1}}\;ax}{P\underline{0}\overline{1} \vdash \exists x.Px\overline{1}}\;\exists I}{\forall y.P\underline{0}y \vdash \exists x.Px\overline{1}}\;\forall L}{\exists x.\forall y.Pxy \vdash \exists x.Px\overline{0}}\;\underline{\exists L}}{\exists x.\forall y.Pxy \vdash \forall y.\exists x.Pxy}\;\overline{\forall I}$$

All occurrences of a given free variable inside a sequent share the same notation (a natural number). However the notation may vary along the proof (when crossing other generalization rules).

So far, the ln approach with Gentzen-style rules seems to win. A last level to consider concerns proof transformations such as weakening, substitution in proofs, cut elimination, etc. The de Bruijn management of variables makes these transformations definable with natural inductions, in particular when relying on parallel substitutions [5]. However things are much more difficult with the Gentzen-style rules. This starts with weakening which requires renaming manipulations to be proved [4, 10]. This justifies the alnl approach which is close to the de Bruijn one with respect to proof transformations but keeps a writing style closer to named notations for formulas and allows for a common notation for occurrences of a given variable inside a formula or a sequent (extending this to proofs does not seem compatible with a simple description of proof transformations).

Table 2 provides a summary of key ingredients of various typical formalizations of first-order logic we have been discussing (just a restricted choice of the existing work). The column “Name stability” describes how far it can be ensured that a given free variable keeps the same name (all over the proof, inside each sequent, or not even inside a formula).

Terminology and Content. We are in a context where the word “*first-order*” may refer to two different things: on one side, terminology from universal algebra, rewriting systems, etc. when talking about *first-order* signature, *first-order* language, etc., and on the other side terminology from logic as in *first-order* logic (as opposed to second-order or propositional logic). Similar problems arise with the use of the word “*term*”. In order to avoid confusion, we will try to stick to: *predicate logic* for first-order logic, *term* for the first-order terms used to build formulas in predicate logic, *first-order object* for a term generated by a first-order signature (*e.g.* we could say “let propositional formulas be the first-order objects generated by the first-order signature containing one unary symbol $\neg$ and two binary symbols $\wedge$ and $\vee$: $F ::= X|\neg F|F \wedge F|F \vee F$”).

In the whole paper, presented rules are for intuitionistic logic, but everything applies exactly in the same way to classical logic, linear logic, etc.

The first part of the paper (Sections 2 and 3) discusses difficulties and possible choices in trying to give a renaming-free representation of predicate logic. The second part (Sections 4 to 6) is the heart of the paper and gives the details of our alnl proposal.

In Section 2, we present terms and formulas built with two kinds of variables. We discuss the impact of avoiding variable renaming (or α-equivalence) on the formalization of quantifier rules. Section 3 addresses the additional difficulties coming from proof transformations and concludes with the need for some kind of binding structure in proofs. Section 4 presents the new anti-locally-nameless (alnl) approach which relies on an ln representation of formulas with two kinds of variables, and on a mix of the Gentzen-style and de Bruijn-style generalization rules. This gives a restricted use of de Bruijn indices at the level of rules and proofs to deal with eigenvariables. Adequacy of the proposed rules is justified in Section 5 through an explicit link with Hilbert system. Section 6 provides the details of a concrete application: normalization of natural deduction for the intuitionistic predicate calculus. The goal is to convince the reader that all the ingredients are there with no hidden part, and that the formalism is powerful enough to deal with real-size examples. Section 7 contains some comments on the `Coq` formalization of the results of the paper. This formalization is provided as an attached archive.

2 Formalization of Predicate Logic

2.1 Terms

Let us first consider a slight generalization of the usual notion of term based on *two* disjoint sets of variables $\mathcal{V}$ and $\mathcal{E}$. Usual terms can be recovered with the particular case $\mathcal{E} = \emptyset$. Given a first-order signature Σ, we build $\mathcal{E}$*-terms* in the usual way (first-order objects generated by Σ), except that we “inject” two kinds of variables:

Table 2. Comparison of Some Formalizations

Paper	Proof assistant	Variable kinds	Name stability	Generalization rule style	Most advanced proof transformation
FKS19 [4]	Coq[2]	2 (named + named)	proof	Gentzen	weakening with renaming
FKW20 [5]	Coq[3]	1 (de Bruijn)	none	de Bruijn	substitution
HvD19 [8]	Lean[4]	1 (de Bruijn)	none	de Bruijn	substitution
H02 [9]	Coq[5]	1 (de Bruijn)	none	de Bruijn	normalization
HKL17 [10]	Coq[6]	2 (named + named)	proof	Gentzen	weakening with renaming
O'C05 [15]	Coq[7]	1 (named)	proof	Hilbert	deduction theorem
here	Coq[8]	2 (named + de Bruijn)	sequent	alnl	normalization

- f-variables ("f" for formula) in $\mathcal{V}$ (a denumerable set, corresponding to usual variables, fixed in the whole paper) are denoted x, y, etc.
- e-variables ("e" for eigen) in a second set $\mathcal{E}$ (a countable set) are denoted e, e', e_1, etc. This notion of *eigenvariables* will be discussed later (see end of Section 2.3.2).

This means $\mathcal{E}$-*terms* are given by:

$$t ::= x \mid e \mid gt\dots t$$

The application of a function symbol g to its arguments should respect the arity of g. *Constant symbols* are the 0-ary function symbols. An $\mathcal{E}$-term is f-*closed* if it contains no f-variable. In particular, $\mathcal{E}$-terms reduced to one e-variable or to one constant symbol are f-closed. From the point of view of terms and formulas, $\mathcal{E}$ can be considered as a special set of constants (as in [10]).

The *substitution* $t[{}^{u}/_{x}]$ of an f-variable x by an $\mathcal{E}$-term u in an $\mathcal{E}$-term t is defined by induction on t:

$$\begin{aligned} x[{}^{u}/_{x}] &= u \\ y[{}^{u}/_{x}] &= y && \text{(if } x \neq y\text{)} \\ e[{}^{u}/_{x}] &= e \\ (gt_1\dots t_k)[{}^{u}/_{x}] &= g(t_1[{}^{u}/_{x}])\dots(t_k[{}^{u}/_{x}]) \end{aligned}$$

This is simply the usual definition of first-order substitution. Note that here, e-variables and constant symbols are handled in the same way.

2.2 Formulas

We consider $\mathcal{E}$-*formulas* as first-order objects built from the relation symbols of the first-order signature Σ, using first-order quantifiers $\forall$ and $\exists$, and some propositional connectives:

$$A ::= Pt\dots t \mid A \star A \mid \forall x.A \mid \exists x.A$$

where $\star$ denotes a generic binary propositional connective (and there can be more than one). The core of our development does not depend on a precise choice of propositional connectives. We will use $\wedge$ and $\to$ (with associated deduction rules) in practice for concrete examples. Note that, to be very precise, we consider here two-sorted constructions since the $\forall$ constructor takes an f-variable and a formula as arguments.

The notion of free or bound variable in a formula is not very important in our setting since we will avoid renaming. Note however that (if we refer to the usual notions of free and bound occurrences of a variable in a formula) an e-variable can only occur free in a formula, only f-variables can be bound. This comes from the key fact that quantifiers only act on f-variables. A formula with no free occurrence of f-variable is called f-*closed*.

The *substitution* $A[{}^{u}/_{x}]$ of an f-variable x by an $\mathcal{E}$-term u in an $\mathcal{E}$-formula A is defined by induction on A:

$$\begin{aligned} (Pt_1\dots t_k)[{}^{u}/_{x}] &= P(t_1[{}^{u}/_{x}])\dots(t_k[{}^{u}/_{x}]) \\ (A \star B)[{}^{u}/_{x}] &= (A[{}^{u}/_{x}]) \star (B[{}^{u}/_{x}]) \\ (\forall y.A)[{}^{u}/_{x}] &= \forall y.(A[{}^{u}/_{x}]) && \text{(if } x \neq y\text{)} \\ (\forall x.A)[{}^{u}/_{x}] &= \forall x.A \\ (\exists y.A)[{}^{u}/_{x}] &= \exists y.(A[{}^{u}/_{x}]) && \text{(if } x \neq y\text{)} \\ (\exists x.A)[{}^{u}/_{x}] &= \exists x.A \end{aligned}$$

This definition of substitution is a *capture-allowing* substitution (for example $(\forall y.Px)[{}^{y}/_{x}] = \forall y.Py$), which differs from usual first-order substitution which would give something like: $(\forall x.A)[{}^{u}/_{x}] = \forall u.(A[{}^{u}/_{x}])$ or $(\forall x.A)[{}^{u}/_{x}] = \forall x.(A[{}^{u}/_{x}])$. The definition case $(\forall x.A)[{}^{u}/_{x}] = \forall x.A$ (and the same for $\exists$) is *the place* where $\forall$ keeps a binding flavor in formulas. This notion of substitution also differs from substitution up to α-renaming which would give for example $(\forall x.A)[{}^{u}/_{y}] = \forall z.(A[{}^{z}/_{x}][{}^{u}/_{y}])$ (with z fresh) [15].

What happens here with substitutions is the same as what happens with ln representations of binders [13].

Since this notion of substitution allows for f-variables capture, the logical meaning of a formula can be strongly altered by substitution: compare for example $\forall y.(Px \to Py)$ and $(\forall y.(Px \to Py))[{}^{y}/_{x}] = \forall y.(Py \to Py)$ (as opposed to

[2] https://www.ps.uni-saarland.de/extras/fol-undec/
[3] https://www.ps.uni-saarland.de/extras/fol-completeness/
[4] https://github.com/flypitch/flypitch/tree/1.2/src
[5] https://github.com/coq-contribs/prfx
[6] https://github.com/liganega/trace
[7] http://r6.ca/Goedel/Goedel20050512.tar.gz
[8] https://doi.org/10.1145/3410270

the capture-free version $\forall z.(Py \to Pz)$ with α-renaming). In order to ensure soundness of substitutions in a setting where we want to avoid renaming of variables, we thus have to check that no capture occurs every time we apply a substitution.

Capture can only occur with f-variables. It is thus interesting to remark that, thanks to the introduction of e-variables, the canonical representation of usual terms over the signature Σ by means of f-variables only (no e-variable at all), can also be replaced by the dual choice of using e-variables only. In this last case, the obtained term is always an f-closed $\mathcal{E}$-term (thus not subject to capture). We will see in particular that the set of terms is large enough to be able to require $\mathcal{E}$-terms used inside substitutions to always be f-closed. In this way, any capture problem is avoided: $(\forall y.(Px \to Py))[^e/_x] = \forall y.(Pe \to Py)$.

So far we have just extended the usual notions of terms and formulas with a new kind of variables (in fact behaving like constants at this level). Let us start now considering proof rules.

2.3 Quantifier Rules

We look at quantifier rules in a setting where formulas are first-order objects without quotienting them, nor being allowed to rename variables. We will mainly work with (sequent-based) natural deduction in the next sections. This is a choice of presentation, the sequent calculus case is very similar (possibly even simpler). Nevertheless for this first discussion, we look at rules from the two systems.

Let us split quantifier rules into *instantiation rules* which involve instantiating a variable in a formula by an arbitrary term, and *generalization rules* which relate a "generic" instance of a formula (typically with some freshness condition on a generic name) and its quantified form.

2.3.1 Instantiation Rules. An instantiation rule involves the substitution of a variable by an arbitrary term. This typically relates with the logical principle $(\forall x.A) \to A[^t/_x]$. In the world of sequent calculus and natural deduction (for the predicate calculus), three rules belong to this family: $\exists$ introduction right, $\forall$ elimination right, and $\forall$ introduction left.

$$\frac{\Gamma \vdash A[^t/_x]}{\Gamma \vdash \exists x.A}\,\exists I \qquad \frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A[^t/_x]}\,\forall E \qquad \frac{\Gamma, A[^t/_x] \vdash C}{\Gamma, \forall x.A \vdash C}\,\forall L$$

The first and the second are considered in natural deduction. The first and the third are considered in sequent calculus.

The key point in (formalizing) such rules is to define $A[^t/_x]$ in a meaningful way. That is in such a way that *variable capture is avoided.* Assuming that substitution is not allowed to do renaming (of bound variables), a meaningful application of the $\exists$ introduction right rule should have the shape:

$$\frac{\Gamma \vdash A[^t/_x] \qquad \text{no capture for } t \text{ in } A[^t/_x]}{\Gamma \vdash \exists x.A}\,\exists I$$

A typical logically-invalid application of the rules would be:

$$\frac{\dfrac{}{\forall x.x \le x \vdash \forall x.x \le x}\,ax}{\forall x.x \le x \vdash \exists y.\forall x.x \le y}\,\exists I$$

(while for any $x \in \mathbb{N}$, $x \le x$, it does not imply there exists a maximal element in $\mathbb{N}$, *i.e.* a y such that $x \le y$ for any $x \in \mathbb{N}$).

Note the conjunction of the "no renaming" hypothesis and of the "no capture" constraint has an impact on the provability of formulas with free variables. When trying to apply the rules described above: $Py \vdash \exists x.\exists y.Px$ is not derivable. Indeed if we try in sequent calculus for example:

$$\frac{\dfrac{\dfrac{}{Py \vdash Py}\,ax}{Py \vdash \exists y.P?}\,\exists I \qquad \text{no capture for } ? \text{ in } \exists y.P?}{Py \vdash \exists x.\exists y.Px}\,\exists I$$

We need "?" to be y in order to apply the top $(\exists I)$ rule, but then the bottom one involves capture.

By relying on e-variables, the solution to avoid captures will be to consider substitutions by f-closed terms only (see [4, 10, 13] for example):

$$\frac{\Gamma \vdash A[^t/_x] \qquad t \text{ f-closed}}{\Gamma \vdash \exists x.A}\,\exists I$$

(see Table 3 for the other instantiation rules).

It does not make the problem with free variables (mentioned above) disappear. For example, $Pt \to \exists x.Px$ is provable if and only if t is f-closed. This comes from the fact that the meaning of an $\mathcal{E}$-formula containing free f-variables is not immediate. One can argue (as in [10]) that such formulas are ill-formed, and one could restrict systems to f-closed formulas [13]. However ignoring this, makes the formalization much lighter, and it has no impact on the provability of f-closed formulas. The interested user could add everywhere the constraint that terms and formulas are f-closed with no major change, except multiple f-closedness checks

Table 3. Ln-Style Instantiation Rules

$$\frac{\Gamma \vdash A[^t/_x] \qquad t \text{ f-closed}}{\Gamma \vdash \exists x.A}\,\exists I$$

$$\frac{\Gamma \vdash \forall x.A \qquad t \text{ f-closed}}{\Gamma \vdash A[^t/_x]}\,\forall E$$

$$\frac{\Gamma, A[^t/_x] \vdash C \qquad t \text{ f-closed}}{\Gamma, \forall x.A \vdash C}\,\forall L$$

to deal with. This can also be done through dependent types as in [10] (in order to be compatible with (a priori) every proof-assistant, we prefer not to rely on such specific type constructions). The worried reader is encouraged to ignore such free f-variables and to refer to Proposition 5.4 for a comparison with the more standard setting given by Hilbert system (with only one kind of variables): in particular, provability is the same for closed formulas.

We take the opportunity of this discussion to address a similar question regarding a control over e-variables. The proposed rules allow us to derive $(\forall x.Px) \to \exists x.Px$ (independently of the signature Σ). One could argue that such a formula should not be provable for a signature containing no constant symbol. Standard solutions dealing with an explicit context for eigenvariables [14] would work perfectly here without interacting with what we discuss, so we make the choice of simplicity for our presentation.

2.3.2 Generalization Rules. A generalization rule allows us to relate provability for a quantified formula and a specific instance of it. Such rules usually work under a freshness hypothesis. It is used to ensure that the instance is generic enough for the reasoning on this instance not to depend on its specific value.

We consider three rules belonging to this family: $\forall$ introduction right, $\exists$ elimination right, and $\exists$ introduction left.

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x.A}\ \forall I \qquad \frac{\Gamma \vdash \exists x.A \qquad \Gamma, A \vdash C}{\Gamma \vdash C}\ \exists E \qquad \frac{\Gamma, A \vdash C}{\Gamma, \exists x.A \vdash C}\ \exists L$$

The first and the second are considered in natural deduction. The first and the third are considered in sequent calculus. These are just skeletons of the rules. Real rules involve side conditions and possibly some renamings. Let us focus on the ($\forall I$) rule. A first presentation we can find in the literature is what we call *Kleene-style* [12] rule:

$$\frac{\Gamma \vdash A \qquad x \text{ not free in } \Gamma}{\Gamma \vdash \forall x.A}\ \forall I$$

It is a usual presentation of the rule in a context where formulas are considered up to α-equivalence. However if we refuse α-equivalence and variable renaming, this rule happens not to be powerful enough: $\forall x.\forall y.Px \vdash \forall y.Py$ is not provable (remark that all the involved formulas are closed, this is not about free variables). Indeed, in the sequent calculus, the two possible last rules are:

- $$\frac{\forall x.\forall y.Px \vdash Py \qquad y \text{ not free in } \forall x.\forall y.Px}{\forall x.\forall y.Px \vdash \forall y.Py}\ \forall I$$
  but $\forall x.\forall y.Px \vdash Py$ is not provable (same reason as for $Py \vdash \exists x.\exists y.Px$ above);
- $$\frac{\forall y.Pt \vdash \forall y.Py \qquad y \notin t}{\forall x.\forall y.Px \vdash \forall y.Py}\ \forall L$$
  but no proof of $\forall y.Pt \vdash \forall y.Py$ exists (if $y \notin t$) neither.

An alternative approach is the presentation of Gentzen [6]:

$$\frac{\Gamma \vdash A[^e/_x] \qquad e \text{ not in } \Gamma, A}{\Gamma \vdash \forall x.A}\ \forall I$$

In the original paper [6], two kinds of variables are considered, with no α-equivalence and no variable renaming. These variables are called free object variables (our e-variables) and bound object variables (our f-variables). In [6], only the second kind is subject to quantification, and formulas and terms are not allowed to contain free occurrences of these f-variables.

In these *Gentzen-style* rules, the e-variable e is called the *eigenvariable* of the rule. As testified by the necessity for a specific syntactic category of e-variables for representing them, we consider this notion as central in the formalization we consider. The status of eigenvariables is often unclear. They behave sometimes as constants, and sometimes as variables (see discussion in [14] for example). We will try to clarify the idea that f-variables behave as variables at the level of formulas (thus the name "f"-variable) and are thus the usual first-order variables for building terms and formulas, while e-variables behave as constants at the level of formulas but as variables at the level of proofs and correspond to Gentzen's eigenvariables (thus the name "e"-variable).

At this formula level, everything works as for locally named formalizations [13], which is perfectly fine since no α-equivalence is involved. Ln representations become more tricky when operations like β-reduction or α-equivalence have to be considered.

3 Proof Transformations

At this point our approach for the formalization of quantifiers is the same as in [10] (except that we do not reject free f-variables). In situations where only provability is important and no particular manipulations of proofs are involved, the formalization proposed so far works perfectly well and fulfills the initial requirements of a natural representation of formulas with quantifiers, involving no α-equivalence or variable renamings. The only surprising behavior comes from the possibility of free f-variables (which can be controlled if wanted, as discussed above).

We have presented the ingredients for dealing with proofs as static objects. However proof theory involves proof transformations making them dynamic. Important results such as normalization, cut elimination, etc. induce operations on proofs. As opposed to [10] we want to consider advanced *syntactic* manipulations of proofs as involved in *syntactic* cut-elimination proofs.

3.1 Weakening

A first simple example of proof transformation is weakening which extends the context of a proof (for systems in which it is not a primitive rule):

Lemma 3.1 (Weakening). *For any* Δ*, if* $\Gamma \vdash A$ *is provable then also* $\Gamma, \Delta \vdash A$*.*

In the setting described so far, this lemma is not immediate to get. Indeed if one tries to use a direct induction on the proof of $\Gamma \vdash A$, the generalization rules do not go through. One would like to apply the following transformation:

$$\dfrac{\overset{\vdots\,\pi}{\Gamma \vdash A[^e/_x]} \qquad e \notin \Gamma, A}{\Gamma \vdash \forall x.A}\,\forall I \quad \mapsto \quad \dfrac{\overset{\vdots\,\mathrm{IH}(\pi)}{\Gamma, \Delta \vdash A[^e/_x]} \qquad e \notin \Gamma, \Delta, A}{\Gamma, \Delta \vdash \forall x.A}\,\forall I$$

which is problematic in the case $e \in \Delta$. It requires a renaming property allowing to replace e by a fresh e' in the proof π of $\Gamma \vdash A[^e/_x]$ (see for example [10]).

3.2 Substitution

Syntactic normalization of proofs in natural deduction requires us to define a notion of substitution by a term at the level of proofs:

$$\dfrac{\dfrac{\overset{\vdots\,\pi}{\Gamma \vdash A[^e/_x]} \qquad e \notin \Gamma, A}{\Gamma \vdash \forall x.A}\,\forall I \qquad t \text{ f-closed}}{\Gamma \vdash A[^t/_x]}\,\forall E \quad \mapsto \quad \overset{\vdots\,\pi[^t/_e]}{\Gamma \vdash A[^t/_x]}$$

We can see that eigenvariables thus behave as constants in terms, formulas and sequents, but now also as variables in proofs, since they are the target of substitutions. The intended meaning of $\pi[^t/_e]$ is that *the* e-variable e coming from the $\forall$-introduction rule becomes substituted all over the proof. But if this e is used many times, things may become ambiguous, and we would like to avoid non-canonical choices. In the following two proofs (which differ only by some commutations of rules), e is the eigenvariable of the $(\forall I)$ rule, but it is not completely clear whether a substitution of e should involve the occurrence of e in Qe or not:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Pe \vdash Pe}\,ax}{\forall x.Px \vdash Pe}\,\forall L}{Qe, \forall x.Px \vdash Pe}\,wkL}{\forall y.Qy, \forall x.Px \vdash Pe}\,\forall L}{\forall y.Qy, \forall x.Px \vdash \forall x.Px}\,\forall I \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Pe \vdash Pe}\,ax}{\forall x.Px \vdash Pe}\,\forall L}{\forall x.Px \vdash \forall x.Px}\,\forall I}{Qe, \forall x.Px \vdash \forall x.Px}\,wkL}{\forall y.Qy, \forall x.Px \vdash \forall x.Px}\,\forall L$$

The general operation we look for, should have the shape:

$$\overset{\vdots\,\pi}{\Gamma \vdash A} \quad \mapsto \quad \overset{\vdots\,\pi[^t/_e]}{\Gamma[^t/_e] \vdash A[^t/_e]}$$

We will define formally this substitution operation $\pi[^t/_e]$ for proofs in Section 4.3. Note, in the general case, an additional difficulty comes with the fact that one should avoid capturing free variables in t. In our setting, as already mentioned (thanks to the distinction of the two kinds of variables) we can restrict ourselves to the case where t is f-closed. But this is not enough since, at the level of proofs, e-variables act as variables:

$$\dfrac{\overset{\vdots\,\pi}{\vdash Pe}}{\vdash \forall x.Px}\,\forall I \quad \overset{[^t/_e]}{\longmapsto} \quad \dfrac{\overset{\vdots\,\pi[^t/_e]}{\vdash Pt}}{\vdash \forall x.Px}\,???$$

$$\dfrac{\overset{\vdots\,\pi}{\vdash Qe'e}}{\vdash \forall x.Qe'x}\,\forall I \quad \overset{[^e/_{e'}]}{\longmapsto} \quad \dfrac{\overset{\vdots\,\pi[^e/_{e'}]}{\vdash Qee}}{\vdash \forall x.Qex}\,???$$

This would not happen if we would have required that e-variables used in a proof must all be eigenvariables, and of a unique rule. However duplication breaks this uniqueness property.

3.3 Duplication

Some proof transformations such as normalization require to duplicate a (sub) proof to build a new one. For example:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, A \vdash A}\,ax \qquad \dfrac{}{\Gamma, A \vdash A}\,ax}{\Gamma, A \vdash A \wedge A}\,\wedge I}{\Gamma \vdash A \to (A \wedge A)}\,\to I \qquad \overset{\vdots\,\pi}{\Gamma \vdash A}}{\Gamma \vdash A \wedge A}\,\to E$$

$$\mapsto \quad \dfrac{\overset{\vdots\,\pi}{\Gamma \vdash A} \qquad \overset{\vdots\,\pi}{\Gamma \vdash A}}{\Gamma \vdash A \wedge A}\,\wedge I$$

This shows that, without involving variable renaming, it is not possible to preserve uniqueness of eigenvariables. Losing this uniqueness could lead to problematic interactions between e-variables as seen above.

3.4 Binding Eigenvariables

This is the point where one must admit that generalization rules act as true binders on eigenvariables, and it is not possible to avoid it. In fact it is not really a surprise: the Curry-Howard correspondence tells us that the language of proofs and proof transformations is as expressive as the λ-calculus and β-reduction.

Even if so far we have worked precisely to avoid it, we thus have to make a choice of formalization for binders... A priori any choice would do the job, with all advantages and drawbacks each choice may have. Since we focused from the beginning on considering first-order objects (in order in particular to be compatible with any proof assistant) and on avoiding on-the-fly renamings, let us try to go on in this way by relying on de Bruijn indices.

However our use of de Bruijn indices will be restricted to the proof construction level with a smaller impact on formulas than the full de Bruijn approach [5, 9].

4 A Mixed Gentzen-de Bruijn Approach

Let us present now the exact ingredients of our alnl proposal. Terms and formulas are those defined in Sections 2.1 and 2.2, with the particular choice $\mathcal{E} = \mathbb{N}$. That is we focus on $\mathbb{N}$-terms and $\mathbb{N}$-formulas. As a consequence we will denote e-variables by n, m, etc.

In this setting, a prototype generalization rule would thus look like:

$$\dfrac{\vdash A[{}^{0}/_{x}]}{\vdash \forall x.A}\ \forall I$$

In this way no particular choice of an e has to be done: we choose the canonical 0. But we then have to make it different from other eigenvariables already used. Following the de Bruijn policy, the natural number representation of an eigenvariable will correspond to the number of generalization rules to be crossed downwards from the place it occurs to the generalization rule where it is introduced[9]. This means we need to lift indices up when crossing a generalization rule upwards:

$$\dfrac{\Gamma\!\uparrow\ \vdash A\!\uparrow\![{}^{0}/_{x}]}{\Gamma \vdash \forall x.A}\ \forall I \qquad \dfrac{\Gamma\!\uparrow, A\!\uparrow\![{}^{0}/_{x}] \vdash C\!\uparrow}{\Gamma, \exists x.A \vdash C}\ \exists L$$

$$\dfrac{\Gamma \vdash \exists x.A \qquad \Gamma\!\uparrow, A\!\uparrow\![{}^{0}/_{x}] \vdash C\!\uparrow}{\Gamma \vdash C}\ \exists E$$

Note there is no need for a freshness condition anymore since the lifting guarantees there is no possible clash between 0 and previously used names. This will make weakening easy for example (see Lemma 6.1 below).

The lifting operation $A\!\uparrow$ on $\mathbb{N}$-formulas, which increments all e-variables by 1, is a particular case of a simultaneous mapping function applied on e-variables (see Sections 4.2 and 4.3).

Example 4.1. In the following proof, we give the same color (and $\overline{()}$ or $\underline{()}$ decoration for B&W printing) to occurrences of e-variables which correspond to the same $\forall$-introduction rule (the natural number itself may vary through the proof, because of the lifting policy of de Bruijn indices, but not inside a sequent):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x.\forall y.Pxy \vdash \forall x.\forall y.Pxy}\ ax}{\forall x.\forall y.Pxy \vdash \forall y.P\underline{0}y}\ \forall E}{\forall x.\forall y.Pxy \vdash P\underline{0}\overline{1}}\ \forall E}{\forall x.\forall y.Pxy \vdash \underline{\forall y}.Py\overline{0}}\ \forall I \qquad \dfrac{\dfrac{\dfrac{}{\forall x.\forall y.Pxy \vdash \forall x.\forall y.Pxy}\ ax}{\forall x.\forall y.Pxy \vdash \forall y.P\overline{0}y}\ \forall E}{\forall x.\forall y.Pxy \vdash P\overline{00}}\ \forall E}{\dfrac{\forall x.\forall y.Pxy \vdash (\forall y.Py\overline{0}) \wedge P\overline{00}}{\forall x.\forall y.Pxy \vdash \overline{\forall}x.((\forall y.Pyx) \wedge Pxx)}\ \forall I}\ \wedge I$$

[9]This happens to be simpler than with full de Bruijn approaches.

Table 4. Alnl-Style Natural Deduction

$$\dfrac{}{\Gamma, A, \Delta \vdash A}\ ax$$

$$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \longrightarrow B}\ {\longrightarrow} I \qquad \dfrac{\Gamma \vdash A \longrightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ {\longrightarrow} E$$

$$\dfrac{\Gamma\!\uparrow\ \vdash A\!\uparrow\![{}^{0}/_{x}]}{\Gamma \vdash \forall x.A}\ \forall I \qquad \dfrac{\Gamma \vdash \forall x.A \qquad t \text{ f-closed}}{\Gamma \vdash A[{}^{t}/_{x}]}\ \forall E$$

$$\dfrac{\Gamma \vdash A[{}^{t}/_{x}] \qquad t \text{ f-closed}}{\Gamma \vdash \exists x.A}\ \exists I$$

$$\dfrac{\Gamma \vdash \exists x.A \qquad \Gamma\!\uparrow, A\!\uparrow\![{}^{0}/_{x}] \vdash C\!\uparrow}{\Gamma \vdash C}\ \exists E$$

4.1 Natural Deduction

From now on, we will focus on natural deduction with just one binary connective: $\longrightarrow$. Terms are those defined in Section 2.1 with e-variables being natural numbers. Formulas are $\mathbb{N}$-formulas with $\longrightarrow$ as unique propositional connective:

$$A ::= Pt \dots t \mid A \longrightarrow A \mid \forall x.A \mid \exists x.A$$

Sequents of intuitionistic natural deduction are $\Gamma \vdash A$ where Γ is a list of formulas. Rules of natural deduction are presented in Table 4 and follow the discussions above.

4.2 Parallel Substitution of Indices

Let us, just for this section, go back to the case of terms parameterized by an arbitrary set $\mathcal{E}$ for e-variables. Given a function r from a set $\mathcal{E}$ to $\mathcal{E}'$-terms, we define the *parallel substitution* operation[10] $t[r]$, mapping $\mathcal{E}$-terms to $\mathcal{E}'$-terms:

$$\begin{aligned} x[r] &= x \\ e[r] &= r(e) \\ (gt_1 \dots t_k)[r] &= g(t_1[r]) \dots (t_k[r]) \end{aligned}$$

Note this endows (_)-terms with a structure of monad over the category Set. We will not use this explicitly, but some related lemmas appear below.

For formulas, we have:

$$\begin{aligned} (Pt_1 \dots t_k)[r] &= P(t_1[r]) \dots (t_k[r]) \\ (A \longrightarrow B)[r] &= (A[r]) \longrightarrow (B[r]) \\ (\forall x.A)[r] &= \forall x.(A[r]) \\ (\exists x.A)[r] &= \exists x.(A[r]) \end{aligned}$$

Note the straightforward definition cases for $\forall$ and $\exists$.

[10]We are in debt to the anonymous referee who suggested us to use this general operation to subsume lifting and substitution of indices.

Lemma 4.2 (Composition)**.**

$$t[r][s] = t[e \mapsto r(e)[s]]$$
$$A[r][s] = A[e \mapsto r(e)[s]]$$

Proof. By induction on t for the first statement and then, using it, by induction on A for the second statement. □

4.3 Substitution in Proofs

We go back to the specific case $\mathcal{E} = \mathbb{N}$ and we denote by S the successor function: $S(n) = n+1$, seen as a function from $\mathbb{N}$ to $\mathbb{N}$-terms. Given an $\mathbb{N}$-formula A, $A\uparrow$ is defined by: $A\uparrow = A[S]$. And given $r : \mathbb{N} \to \mathbb{N}$-term, we define $\Uparrow r : \mathbb{N} \to \mathbb{N}$-term by:

$$\Uparrow r(n) = \begin{cases} 0 & \text{if } n = 0 \\ r(n-1)\uparrow & \text{otherwise} \end{cases}$$

Lemma 4.3.

$$t\uparrow[\Uparrow r] = t[r]\uparrow$$
$$A\uparrow[\Uparrow r] = A[r]\uparrow$$

Proof. By Lemma 4.2, using: for all n, $(n+1)[\Uparrow r] = r(n)\uparrow$. □

A function $r : \mathbb{N} \to \mathbb{N}$-term is called f-*closed* if, for any $n \in \mathbb{N}$, $r(n)$ is an f-closed $\mathbb{N}$-term.

Lemma 4.4. *If r is f-closed,*

$$t[{}^u/_x][r] = t[r][{}^{u[r]}/_x]$$
$$A[{}^u/_x][r] = A[r][{}^{u[r]}/_x]$$

Proof. By induction on t for the first statement and then, using it, by induction on A for the second statement. □

An important point is that f-closed functions act not only on terms and formulas but also on proofs:

$$\begin{array}{c}\vdots\,\pi \\ \Gamma \vdash A\end{array} \quad\mapsto\quad \begin{array}{c}\vdots\,\pi[r] \\ \Gamma[r] \vdash A[r]\end{array} \qquad \text{(for } r \text{ f-closed)}$$

The definition of $\pi[r]$ is given by induction on π for all r. The key cases are:

$$\pi = \dfrac{\begin{array}{c}\vdots\,\pi_1 \\ \Gamma\uparrow \vdash A\uparrow[{}^0/_x]\end{array}}{\Gamma \vdash \forall x.A}\,\forall I \quad\mapsto\quad \dfrac{\begin{array}{c}\vdots\,\pi_1[\Uparrow r] \\ \Gamma[r]\uparrow \vdash A[r]\uparrow[{}^0/_x]\end{array}}{\Gamma[r] \vdash \forall x.(A[r])}\,\forall I \;= \pi[r]$$

$$\pi = \dfrac{\begin{array}{c}\vdots\,\pi_1 \\ \Gamma \vdash \forall x.A\end{array} \qquad t \text{ f-closed}}{\Gamma \vdash A[{}^t/_x]}\,\forall E$$

$$\mapsto \quad \dfrac{\begin{array}{c}\vdots\,\pi_1[r] \\ \Gamma[r] \vdash \forall x.(A[r])\end{array} \qquad t[r] \text{ f-closed}}{\Gamma[r] \vdash A[r][{}^{t[r]}/_x]}\,\forall E \;= \pi[r]$$

$$\pi = \dfrac{\begin{array}{c}\vdots\,\pi_1 \\ \Gamma \vdash A[{}^t/_x]\end{array} \qquad t \text{ f-closed}}{\Gamma \vdash \exists x.A}\,\exists I$$

$$\mapsto \quad \dfrac{\begin{array}{c}\vdots\,\pi_1[r] \\ \Gamma[r] \vdash A[r][{}^{t[r]}/_x]\end{array} \qquad t[r] \text{ f-closed}}{\Gamma[r] \vdash \exists x.(A[r])}\,\exists I \;= \pi[r]$$

$$\pi = \dfrac{\begin{array}{c}\vdots\,\pi_1 \\ \Gamma \vdash \exists x.A\end{array} \qquad \begin{array}{c}\vdots\,\pi_2 \\ \Gamma\uparrow, A\uparrow[{}^0/_x] \vdash C\uparrow\end{array}}{\Gamma \vdash C}\,\exists E$$

$$\mapsto \quad \dfrac{\begin{array}{c}\vdots\,\pi_1[r] \\ \Gamma[r] \vdash \exists x.(A[r])\end{array} \qquad \begin{array}{c}\vdots\,\pi_2[\Uparrow r] \\ \Gamma[r]\uparrow, A[r]\uparrow[{}^0/_x] \vdash C[r]\uparrow\end{array}}{\Gamma[r] \vdash C[r]}\,\exists E \;= \pi[r]$$

Note that $\pi[r]$ has the same size (number of rules) as π. The obtained proofs are well formed and with the appropriate conclusions, as shown by:

$$B\uparrow[\Uparrow r] = B[r]\uparrow \qquad \text{(Lemma 4.3)}$$
$$B[{}^t/_x][r] = B[r][{}^{t[r]}/_x] \qquad \text{(Lemma 4.4)}$$
$$B[{}^0/_x][\Uparrow r] = B[\Uparrow r][{}^{0[\Uparrow r]}/_x] = B[\Uparrow r][{}^0/_x] \qquad \text{(Lemma 4.4)}$$

Since de Bruijn indices allow us to identify uniquely the generalization rule which is associated with an occurrence of e-variable, we do not have problems anymore concerning uniqueness of the use of eigenvariables, and non-canonical targets of substitutions neither (see Sections 3.2 and 3.3). Indeed, given an occurrence of the natural number n in a proof, the unique instance of generalization rule it is related with, is obtained by crossing n generalization rules towards the root of the proof (or if there is less than n such rules, this occurrence of n is not the eigenvariable of any generalization rule in the proof).

We now go back to the definition of substitutions for proofs, which should allow us to do the following transformation in normalization:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi \\ \Gamma\uparrow \vdash A\uparrow[{}^0/_x]\end{array}}{\Gamma \vdash \forall x.A}\,\forall I \qquad t \text{ f-closed}}{\Gamma \vdash A[{}^t/_x]}\,\forall E \quad\mapsto\quad \begin{array}{c}\vdots\,??? \\ \Gamma \vdash A[{}^t/_x]\end{array}$$

Given an $\mathbb{N}$-term v, we consider the following function $v\Downarrow$ from $\mathbb{N}$ to $\mathbb{N}$-terms:

$$v\Downarrow(n) = \begin{cases} v & \text{if } n = 0 \\ n-1 & \text{otherwise} \end{cases}$$

which is f-closed if v is f-closed. So that:

$$\begin{array}{c}\vdots\,\pi \\ \Gamma \vdash A\end{array} \quad\mapsto\quad \begin{array}{c}\vdots\,\pi[v\Downarrow] \\ \Gamma[v\Downarrow] \vdash A[v\Downarrow]\end{array} \qquad \text{(for } v \text{ f-closed)}$$

Lemma 4.5.

$$t\!\uparrow\![v\!\Downarrow] = t$$
$$A\!\uparrow\![v\!\Downarrow] = A$$

Proof. By Lemma 4.2, using: for all n, $(n+1)[v\!\Downarrow] = n$. □

We need to check that the action of $v\!\Downarrow$ on formulas and sequents will be appropriate for substitution in proof normalization:

Lemma 4.6. *If π is a proof of $\Gamma\!\uparrow\; \vdash A\!\uparrow\![{}^{0}/_{x}]$ and t is an f-closed term, then $\pi[t\!\Downarrow]$ is a proof of $\Gamma \vdash A[{}^{t}/_{x}]$.*

Proof. By Lemmas 4.4 and 4.5 with t f-closed:

$$(\Gamma\!\uparrow\; \vdash A\!\uparrow\![{}^{0}/_{x}])[t\!\Downarrow] = \Gamma \vdash A[{}^{t}/_{x}]$$

□

5 Relation with Hilbert System

In order to compare the expressiveness of our representation of predicate logic with a more traditional one, let us consider here Hilbert system with $\forall$ and $\exists$ quantifiers for intuitionistic predicate logic.

5.1 Hilbert System

Terms and formulas are the usual ones, which correspond exactly to $\emptyset$-terms and $\emptyset$-formulas defined in Sections 2.1 and 2.2. All term variables are f-variables (*i.e.* coming from the given set $\mathcal{V}$). No quotient on formulas (like α-equivalence) is introduced.

The system is built from six axioms and two deduction rules (modus ponens and generalization). Axioms are:

$A \to B \to A$	
$(A \to B \to C) \to (A \to B) \to A \to C$	
$\forall x.A \to A[{}^{t}/_{x}]$	if no capture happens
$\forall x.(A \to B) \to A \to \forall x.B$	if x is not free in A
$A[{}^{t}/_{x}] \to \exists x.A$	if no capture happens
$\forall x.(A \to B) \to \exists x.A \to B$	if x is not free in B

A deduction of a formula F in Hilbert system is obtained inductively:

- by instantiating one of the axioms above with appropriate formulas A, B, and C to obtain F;
- or from a deduction of $A \to F$ and a deduction of A (this is the rule of *modus ponens*);
- or, if $F = \forall x.A$, from a deduction of A (this is the *generalization* rule).

5.2 Back and Forth with Natural Deduction

We denote by $\vdash_{\mathrm{H}} F$ the existence of a deduction of F in Hilbert system, and by $\vdash_{\mathrm{ND}} A$ the existence of a proof of the sequent $\vdash A$ in the natural deduction system of Table 4.

Lemma 5.1 (From Hilbert to Natural Deduction)**.** *If the free (f-)variables of F are among x_1, ..., x_n and $\vdash_H F$, then, for any* f-closed *terms t_1, ..., t_n, we have $\vdash_{ND} F[{}^{t_1}/_{x_1}, \dots, {}^{t_n}/_{x_n}]$.*

Note we must assume t_1, ..., t_n to be f-closed since $Py \to \exists x.Px$ is provable in Hilbert system but not with our formalization of natural deduction (see Section 2.3.1).

Lemma 5.2 (From Natural Deduction to Hilbert)**.** *If the e-variables of A are among e_1, ..., e_n and $\vdash_{ND} A$, then $\vdash_H A[{}^{x_1}/_{e_1}, \dots, {}^{x_n}/_{e_n}]$ as soon as the x_is are chosen in such a way that no capture happens in the substitution.*

Proposition 5.3 (Embedding Hilbert)**.** *If the free (f-)variables of F are among x_1, ..., x_n then:*

$$\vdash_H F \iff \vdash_{ND} F[{}^{1}/_{x_1}, \dots, {}^{n}/_{x_n}]$$

Proposition 5.4 (Embedding Natural Deduction)**.** *If A is* f-closed *and the e-variables of A are among e_1, ..., e_n, and if x_1, ..., x_n are distinct f-variables not occurring in A, then:*

$$\vdash_{ND} A \iff \vdash_H A[{}^{x_1}/_{e_1}, \dots, {}^{x_n}/_{e_n}]$$

We do not give the details of these proofs which, up to appropriate choices of fresh variables and some renaming manipulations, follow traditional patterns (details are available in the `Coq` formalization, see Section 7).

6 Normalization of Natural Deduction

All ingredients being settled, let us develop a complete concrete example, showing that we have enough material for a detailed syntactic proof of normalization of natural deduction for the intuitionistic predicate calculus (with one propositional connective: $\to$, and one quantifier: $\forall$). Presenting $\exists$ quantifiers as well would make the structure of the proof more complex without pointing out any specific novelty related with the representation of rules and quantifiers. Indeed normalization of natural deduction with $\exists$ quantifiers is known to require the introduction of commutative conversions which can be avoided if we restrict ourselves to $\to$ and $\forall$ (see Section 7 for additional comments).

We consider a *big-step* normalization statement: to each proof it is possible to associate a normal proof with the same conclusion sequent. A *normal proof* is a proof in which no introduction rule is followed by an elimination rule on the introduced connective. Such normal proofs are known to satisfy the sub-formula property [18].

Our proof mostly follows [11]. Normal proofs (*NF*) (or normal forms) can be described by a mutual induction with *neutral proofs* (*NE*):

$$NE ::= ax \mid {\to}E(NE, NF) \mid \forall E(NE)$$
$$NF ::= NE \mid {\to}I(NF) \mid \forall I(NF)$$

We use here the names of rules as constructors. For example the ${\to}E(NE, NF)$ entry means that if we have a neutral proof *NE* and a normal form *NF* then the proof obtained by adding an $({\to}E)$ rule is a neutral proof.

It is easy to check that, for any r f-closed, if *NF* (resp. *NE*) is a normal (resp. neutral) proof then $NF[r]$ (resp. $NE[r]$) as well.

Lemma 6.1 (Weakening). *If NF is a normal proof of $\Gamma, \Delta \vdash A$, for any Θ, there exists a normal proof of $\Gamma, \Theta, \Delta \vdash A$.*

Proof. Contrarily to what can happen with Kleene-style or Gentzen-style rules (see Section 3.1), a direct induction on NF (and NE) works here. The typical case is $(\forall I)$ where we apply the induction hypothesis with the list $\Theta\uparrow$:

$$\dfrac{\Gamma\uparrow, \Theta\uparrow, \Delta\uparrow \vdash A\uparrow[{}^0/_x]}{\Gamma, \Theta, \Delta \vdash \forall x.A}\ \forall I$$

□

The key lemma for normalization consists in substituting normal proofs in normal proofs. The normal form of a proof is then simple to obtain.

The *size* $|A|$ of a formula A is defined by counting connectives (the size of terms is ignored).

Lemma 6.2 (Substitution). *Given two normal proofs NF_1 of $\Gamma, \Delta \vdash A$ and NF_2 of $\Gamma, A, \Delta \vdash B$, there exists a normal proof $NF_2[{}^{NF_1}/_A]$ of $\Gamma, \Delta \vdash B$.*

Proof. We strengthen the statement by proving simultaneously that, if NF_2 is neutral and the size of A is strictly smaller than the size of B then $NF_2[{}^{NF_1}/_A]$ is neutral.

The proof goes by induction on the (lexicographically ordered) pair (size of A, size of NF_2). We use sizes rather than some structural induction because we have to move at some point from A to $A\uparrow$ (it preserves sizes but breaks the sub-formula relation). We focus on the key cases of last rule of NF_2:

- $(\forall I)$:

$$NF_2 = \dfrac{\begin{array}{c}\vdots\ NF_2' \\ \Gamma\uparrow, A\uparrow, \Delta\uparrow \vdash B\uparrow[{}^0/_x]\end{array}}{\Gamma, A, \Delta \vdash \forall x.B}\ \forall I$$

$$\mapsto \quad \dfrac{\begin{array}{c}\vdots\ NF_2'[{}^{NF_1[\mathrm{S}]}/_{A\uparrow}] \\ \Gamma\uparrow, \Delta\uparrow \vdash B\uparrow[{}^0/_x]\end{array}}{\Gamma, \Delta \vdash \forall x.B}\ \forall I \ = NF_2[{}^{NF_1}/_A]$$

  since $NF_1[\mathrm{S}]$ is a normal proof of $\Gamma\uparrow, \Delta\uparrow \vdash A\uparrow$, and $|A\uparrow| = |A|$.
- $(\forall E)$:

$$NF_2 = \dfrac{\begin{array}{c}\vdots\ NE_2 \\ \Gamma, A, \Delta \vdash \forall x.B\end{array} \qquad t \text{ f-closed}}{\Gamma, A, \Delta \vdash B[{}^t/_x]}\ \forall E$$

$$\mapsto \quad \begin{array}{c}\vdots\ ??? \\ \Gamma, \Delta \vdash B[{}^t/_x]\end{array}$$

  We apply the induction hypothesis to NE_2 to get a normal proof $NF' = NE_2[{}^{NF_1}/_A]$ of $\Gamma, \Delta \vdash \forall x.B$. If NF' is a neutral proof (in particular if $|A| < |\forall x.B|$), we apply a $(\forall E)$ rule to it and we obtain the appropriate neutral proof. Otherwise NF' ends with a $(\forall I)$ rule and its premise NF'' has conclusion $\Gamma\uparrow, \Delta\uparrow \vdash B\uparrow[{}^0/_x]$. We build $NF''[t\Downarrow]$ with conclusion $\Gamma, \Delta \vdash B[{}^t/_x]$ (see Lemma 4.6).
- $(\to E)$: By induction hypotheses, we have normal proofs NF' of $\Gamma, \Delta \vdash C \to B$ and NF'' of $\Gamma, \Delta \vdash C$. If NF' is a neutral proof (in particular if $|A| < |C \to B|$), we apply an $(\to E)$ rule and we obtain a neutral proof of $\Gamma, \Delta \vdash B$. Otherwise NF' ends with an $(\to I)$ rule and its premise NF''' has conclusion $\Gamma, \Delta, C \vdash B$. Since we can assume $|A| \geq |C \to B|$ and thus $|C| < |A|$, by induction hypothesis we obtain the required normal form with conclusion $\Gamma, \Delta \vdash B$.

□

Theorem 6.3 (Normalization). *If $\Gamma \vdash A$ is provable then it is provable by a normal proof.*

Proof. By induction on the proof π of $\Gamma \vdash A$. Except for the elimination rules, one can conclude by immediate application of the induction hypotheses. Let us now consider the two elimination rules:

- $(\to E)$: By induction hypotheses, we have two normal forms NF_1 and NF_2 with conclusions $\Gamma \vdash A \to B$ and $\Gamma \vdash A$. Either NF_1 is a neutral proof and we simply apply an $(\to E)$ rule with NF_2 to it, or NF_1 ends with an $(\to I)$ rule. Let NF_1' be the premise of this rule which has conclusion $\Gamma, A \vdash B$, we apply Lemma 6.2 to get a normal form $NF_1'[{}^{NF_2}/_A]$ with conclusion $\Gamma \vdash B$.
- $(\forall E)$: By induction hypothesis, we have a normal form NF with conclusion $\Gamma \vdash \forall x.A$ and we want to build a normal form with conclusion $\Gamma \vdash A[{}^t/_x]$. Either NF is a neutral proof and we simply apply a $(\forall E)$ rule to it, or NF ends with a $(\forall I)$ rule. Let NF' be the premise of this rule, $NF'[t\Downarrow]$ is the normal form we are looking for.

□

Since normal proofs have the sub-formula property [18], one can check that an f-closed formula (or sequent) can be proved using a proof involving f-closed formulas only.

7 Comments on the Coq Formalization

We describe here shortly the Coq formalization corresponding to the approach and results presented in the previous sections. It can be found in the associated archive:

https://doi.org/10.1145/3410270

It is developed with Coq 8.12.0 (see README.md for instructions).

The only important difference between the Coq development and the paper concerns the use of arity constraints from the signature Σ when building terms and formulas. To make things simpler in Coq, we consider that a copy of each function symbol and of each predicate symbol exists for each arity, to that $gt_1 \dots t_k$ is always well formed: it is then implicitly assumed that the k-ary version of g has been used.

In the whole formalization, the first-order signature Σ is considered as an abstract parameter through the types tatom (for function symbols) and fatom (for relation symbols).

Since we focus on proof transformations rather than dealing with provability properties, proofs are defined in Type. A definition in Prop, as often considered in Coq formalizations of predicate logic, corresponds to defining provability rather than proofs themselves.

By defining two specific tactics for unfolding inductions on terms and formulas, most of the results about substitutions, terms and formulas are proved in less than three lines, and a number of them by a single tactic call.

We then rely on these basic lemmas to prove more involved results with some automation (mostly by rewriting with tactics defined in term_tactics.v). The total number of lines for the tactics definitions is around 100. The total length of the three key files leading to the normalization proof of Section 6 (foterms.v, foformulas.v and nj1.v) is around 800 lines.

The formalization of formulas in foformulas.v is easily reusable for other logics since they are built on top of three abstract parameters for nullary propositional connectives (NCon), for binary propositional connectives (BCon) and for quantifiers (QCon). It is probably natural to consider unary propositional connectives as well, and this must be direct. We omit them because they were not required in the present work and moreover they could be encoded as a pair of a nullary connective and a binary connective by giving the nullary one as argument to the binary one.

The extension of natural deduction to existential quantifiers is presented in nj1_frlexs.v, together with the normalization proof. Existentials do not really impact the representation of formulas and proofs, but as already mentioned, normalization is a bit harder to prove since the existential elimination rules induce commutative steps in reduction. The sub-formula property is formalized as well.

Concerning the link with Hilbert system presented in Section 5, additional properties about free variables, capture checks, renamings, and substitutions (in particular iterated substitutions) are required. These kinds of manipulations are precisely supposed to be avoidable when simply manipulating natural deduction. These results are presented in foterms_ext.v and foformulas_ext.v. Then the results of Section 5 are formalized in files hilbert2nj.v, nj2hilbert.v and nj_vs_hilbert.v.

Additional related results are available at:

https://github.com/olaure01/quantifiers

This includes work on linear logic, second-order quantifiers, or proposals for the explicit management of arities from the signature Σ.

8 Conclusion

We have presented a general framework based on two kinds of variables for the formalization of quantifiers. It relies on first-order structures only. Our goal is to be as close as possible to what would be done on paper (the only difference here is the use of two kinds of variables as already done in [6, 18]), and to avoid α-renaming as far as possible. At the level of formulas, things look similar to the locally named approach [4, 10, 13], which is very natural to deal with in an α-equivalence-free setting. We choose $\mathbb{N}$ as the set of names for free variables, leading to an anti-locally-nameless representation of formulas. At the level of proofs, things look more like a locally nameless approach [2]: we never rename (quantified) f-variables and the renaming of eigenvariables is handled by the management of de Bruijn indices. These e-variables never being bound in formulas, the management of indices is lighter than in full de Bruijn representations. The only remaining defect of the use of de Bruijn indices is that different natural numbers at different positions in a proof may refer to the same eigenvariable. However, since this is not the case inside a sequent, it has a lower impact.

Everything can be adapted to *propositional* second-order logic, that is *first-order-free* second-order logic (also known as Girard's System F [7]). A typical definition of formulas for this system is:

$$A ::= P \mid X \mid n \mid A \to A \mid \forall X.A$$

where P comes from a given set of predicate symbols, X belongs to the set of second-order variables (f-variables), and n is a natural number (e-variables). However mixing first-order quantification *and* second-order quantification seems out of reach. The binding structures become richer and substitution in formulas already involves a general binding behavior:

$$(\underline{\forall} y.X\underline{y})[{}^{\overline{\forall} y.Px\overline{y}}/_{Xx}] = \underline{\forall} y.((\overline{\forall} y.Px\overline{y})[{}^{\underline{y}}/_{x}]) = \underline{\forall} y.\overline{\forall} y.P\underline{y}\overline{y}$$

(with troubles if $\underline{y} = \overline{y}$).

One of the main targets of application is the development of libraries for the formalization of meta-theoretical properties in proof theory with the hope to make it accessible to the largest audience. This should include people wanting to stick to traditional paper presentations of predicate logic. But also, users only interested in the propositional aspects should not be impacted with what happens for quantifiers. We plan in particular to introduce this representation of quantifiers in the `Yalla`[11] library which provides meta-theoretical results on the proof-theory of linear logic.

[11] https://perso.ens-lyon.fr/olivier.laurent/yalla/

Acknowledgments

This work was supported by the IRN Linear Logic, and by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

Many thanks to Damien Pous for various discussions around this work, and to the anonymous referees (also of previous versions) for their useful comments.

References

[1] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. 2005. Mechanized Metatheory for the Masses: The PoplMark Challenge. In *18th International Conference on Theorem Proving in Higher Order Logics (TPHOLs) (Lecture Notes in Computer Science, Vol. 3603)*, Joe Hurd and Thomas F. Melham (Eds.). Springer, 50–65. https://doi.org/10.1007/11541868_4

[2] Arthur Charguéraud. 2012. The Locally Nameless Representation. *Journal of Automated Reasoning* 49, 3 (2012), 363–408. https://doi.org/10.1007/s10817-011-9225-2

[3] Nicolaas G. de Bruijn. 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae (Proceedings)* 75, 5 (1972), 381–392. https://doi.org/10.1016/1385-7258(72)90034-0

[4] Yannick Forster, Dominik Kirst, and Gert Smolka. 2019. On synthetic undecidability in Coq, with an application to the entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, Assia Mahboubi and Magnus Myreen (Eds.). Association for Computing Machinery, 38–51. https://doi.org/10.1145/3293880.3294091

[5] Yannick Forster, Dominik Kirst, and Dominik Wehr. 2020. Completeness Theorems for First-Order Logic Analysed in Constructive Type Theory. In *Logical Foundations of Computer Science (LFCS) (Lecture Notes in Computer Science, Vol. 11972)*, Sergei Artemov and Anil Nerode (Eds.). Springer, 47–74. https://doi.org/10.1007/978-3-030-36755-8_4

[6] Gerhard Gentzen. 1969. Investigations into logical deductions. In *The Collected Works of Gerhard Gentzen*. North-Holland, 68–131. https://doi.org/10.1016/S0049-237X(08)70822-X

[7] Jean-Yves Girard. 1971. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In *Proceedings of the Second Scandinavian Logic Symposium (Studies in Logic and the Foundations of Mathematics, Vol. 63)*, J.E. Fenstad (Ed.). Elsevier, 63–92. https://doi.org/10.1016/S0049-237X(08)70843-7

[8] Jesse Michael Han and Floris van Doorn. 2019. A Formalization of Forcing and the Unprovability of the Continuum Hypothesis. In *10th International Conference on Interactive Theorem Proving (ITP 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 141)*, John Harrison, John O'Leary, and Andrew Tolmach (Eds.). Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 19:1–19:19. https://doi.org/10.4230/LIPIcs.ITP.2019.19

[9] Dimitri Hendriks. 2002. Proof Reflection in Coq. *Journal of Automated Reasoning* 29, 3–4 (2002), 277–307. https://doi.org/10.1023/A:1021923116629

[10] Hugo Herbelin, SunYoung Kim, and Gyesik Lee. 2017. Formalizing the meta-theory of first-order predicate logic. *J. Korean Math. Soc.* 54, 5 (Sept. 2017), 1521–1536. https://doi.org/10.4134/JKMS.J160546

[11] Felix Joachimski and Ralph Matthes. 2003. Short proofs of normalization for the simply-typed lambda-calculus, permutative conversions and Gödel's T. *Archive for Mathematical Logic* 42, 1 (2003), 59–87. https://doi.org/10.1007/s00153-002-0156-9

[12] Stephen Kleene. 1952. *Introduction to Metamathematics*. North-Holland.

[13] James McKinna and Robert Pollack. 1999. Some Lambda Calculus and Type Theory Formalized. *Journal of Automated Reasoning* 23, 3 (Nov. 1999), 373–409. https://doi.org/10.1023/A:1006294005493

[14] Dale Miller and Alwen Tiu. 2005. A proof theory for generic judgments. *ACM Transactions on Computational Logic* 6, 4 (2005), 749–783. https://doi.org/10.1145/1094622.1094628

[15] Russell O'Connor. 2005. Essential Incompleteness of Arithmetic Verified by Coq. In *18th International Conference on Theorem Proving in Higher Order Logics (Lecture Notes in Computer Science, Vol. 3603)*, Joe Hurd and Tom Melham (Eds.). Springer, 245–260. https://doi.org/10.1007/11541868_16

[16] Frank Pfenning and Conal Elliott. 1988. Higher-Order Abstract Syntax. In *Proceedings of the ACM SIGPLAN 1988 Conference on Programming Language Design and Implementation (PLDI)*, Richard Wexelblat (Ed.). Association for Computing Machinery, 199–208. https://doi.org/10.1145/53990.54010

[17] Andrew Pitts. 2003. Nominal logic, a first order theory of names and binding. *Information and Computation* 186, 2 (2003), 165–193. https://doi.org/10.1016/S0890-5401(03)00138-X

[18] Dag Prawitz. 1965. *Natural Deduction: A Proof-Theoretical Study*. Number 3 in Stockholm studies in philosophy. Almqvist and Wiksell.