LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - University of Edinburgh

Proof Search in the $\lambda\Pi$-calculus

by

David Pym
Lincoln Wallen

Proof-search in the -calculus

LFCS Report Series ECS-LFCS-91-146

LFCS April 1991
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

To appear in: Logical Frameworks, G. Huet and G. Plotkin (editors), Cambridge University Press, 1991. Please refer to this publication.

Proof-search in the $\lambda\Pi$-calculus

David Pym
University of Edinburgh
Scotland, U.K.

Lincoln Wallen
University of Oxford
England, U.K.

We formulate a system for proof-search in the $\lambda\Pi$-calculus by exploiting a cut-elimination theorem and a subformula property. We modify this system to support the use of unification and prove that the modification reduces the search space uniformly. A permutation theorem is then exploited to reduce the search space further and a notion of *intrinsic well-typing* is introduced to ensure an adequate computational basis for proof-search.

1 Introduction

We present a series of proof systems for the $\lambda\Pi$-calculus: a theory of *first-order dependent function types* [7], [18], [11]. The systems are complete, but differ substantially as bases for algorithmic proof-search.

The first system, called N, is a natural deduction formulation of the $\lambda\Pi$-calculus. This system is also known as the type system of the (Edinburgh) Logical Framework or LF [11]. The main judgement of N is the typing assertion: $\Gamma \vdash M{:}A$, meaning that the term M has type A given the type assignments for free variables and constants recorded in Γ. This relation is decidable. A typical rule of N is the elimination rule for the dependent function type constructor, Π:

$$\Pi\mathrm{E}\quad \frac{\Gamma \vdash M{:}(\Pi x{:}A.B) \qquad \Gamma \vdash N{:}A}{\Gamma \vdash MN{:}B[N/x]}$$

where $B[N/x]$ denotes capture-avoiding substitution of N for free occurrences of x in B.[1]

In Section 3 we formulate a sound and complete Gentzen-style sequent calculus G and use a cut-elimination result for G to obtain a system for the semi-decidable relation of *inhabitation:* $\Gamma \Rightarrow A$, with the meaning $(\exists M)(\Gamma \vdash M{:}A)$. This system, called L, forms a foundation for proof-search in the $\lambda\Pi$-calculus since the judgements of L assert the *existence* of proofs of the judgements of N (as in the case of first-order logic [23]).

A typical rule of L is the Π-left rule, the counterpart of the Π-elimination rule of N:

$$\Pi l\quad \frac{\Gamma, z{:}B[N/x] \Rightarrow C}{\Gamma \Rightarrow C} \qquad \begin{array}{l}(\mathrm{a})\ w : (\Pi x{:}A.B) \in \Gamma \\ (\mathrm{b})\ z \notin \mathrm{Dom}(\Gamma) \\ (\mathrm{c})\ \mathrm{G}\backslash\mathrm{cut}\ \text{proves}\ \Gamma \vdash N{:}A.\end{array} \quad (1)$$

L is almost *logicistic*, in the sense of Gentzen, but there remains one localized appeal to an external notion, that notion being the system G\cut (*i.e.*, G without the cut rule) in side

[1] Readers unfamiliar with dependent types should read the construction $\Pi x{:}A.B$ as a (typed) universal quantifier such as $\forall x{:}A.B$; x is the bound variable, ranging over the type A, which may occur free in the term B.

condition (c) of the rule above. Indeed, with respect to the Π-type structure of terms in the language, **L** has a *subformula property* [10]. As a consequence, if the inference rules are used as *reduction operators* from conclusion to premisses then **L** induces a search space Sp(**L**) of derivations of a given sequent. A formalization of this notion of search space can be found in Section 4. Notice that if the Π*l* rule is used as a reduction, the choice of term N to use in the premiss is unconstrained by the conclusion of the rule. The sub*formula* property of the Π-types does not extend to a full sub*term* property (*cf.* the quantifier rules of the predicate calculus). The *axiom* sequent (or *closure* condition for the reduction system) is:

$$\Gamma, z{:}A, \Gamma' \Rightarrow A$$

i.e., the conclusion occurs as the type of a declaration in the context.

In Section 4 we formulate the system **U** which is is obtained from **L** by removing the appeal to G\cut in the Π*l* rule. The Π*l* rule of **U** is thus:

$$\Pi l \quad \frac{\Gamma, z{:}B[\alpha/x] \Rightarrow C}{\Gamma \Rightarrow C} \qquad \begin{array}{l}(a)\ w : (\Pi x{:}A.B) \in \Gamma \\ (b)\ z \notin \mathrm{Dom}(\Gamma).\end{array} \tag{2}$$

This rule introduces an *indeterminate* into the proof, denoted here by α. Indeterminates are distinct from the usual *eigenvariables* that appear in derivations.

A leaf $\Gamma \Rightarrow A$ of a **U**-derivation may be *closed* by an *instantiation:* a mapping from indeterminates to terms, such that for at least one declaration $z : B \in \Gamma$ we have $B\sigma \equiv A\sigma$. (Here $B\sigma$ denotes the image of B under the mapping σ *etc.*) The calculation of instantiations can be performed by a *unification algorithm* for the language. A suitable algorithm has been developed by the first author [24] based on a standard algorithm for simple type theory [14]. A similar algorithm has been developed, independently, by Elliott [8], [9].

A **U**-proof is a pair $\langle \psi, \sigma \rangle$ consisting of a **U**-derivation ψ and an instantiation σ such that $\psi\sigma$ (the image of ψ under σ) is an **L**-proof. Not every instantiation that closes all the leaves of a given **U**-derivation will yield an **L**-proof when so applied. It is sufficient to check that the instantiation can be *well-typed* in the derivation to ensure that the result is an **L**-proof. If Γ is the context and A the type of the indeterminate α when introduced into the **U**-derivation, the well-typing condition for α amounts to:

$$\Gamma\sigma \vdash \alpha\sigma{:}A\sigma$$

ensuring that side condition (c) of the Π*l* rule of **L** (1)—the condition omitted from the Π*l* rule of **U** (2)—is nevertheless satisfied in $\psi\sigma$. The unconstrained choice of term in the Π*l* rule of **L** is replaced by a highly constrained choice in **U**.[2] Since instantiations are only calculated by unifying declarations in the leaves of proofs to force the leaves to close, all such instantiations considered are *minimal* in a sense made precise in Section 4. It is

[2]The inference system that corresponds to the calculation of instantiations by unification does indeed have a *subterm property.* The soundness and completeness result for **U** is a form of *Herbrand Theorem* for the theory. The unification algorithm searches amongst the terms of a "Herbrand universe" defined by each leaf sequent. This aspect is not explored in detail in this paper, see [24].

then easy to prove that the search space Sp(U) induced by U is a subspace of Sp(L), but which nevertheless contains representatives for all L-proofs. This wholesale reduction in the search space is analogous to that obtained by Robinson in the context of the predicate calculus [26].

The well-typing of an instantiation depends on the structure of the derivation from which it is calculated. If it fails to be well-typed in that derivation it may be well-typed in some permutation of the derivation, since rule applications (or reductions) can sometimes be permuted whilst leaving the endsequent (*i.e.*, root) of the derivation and its leaves unchanged. The degree to which this can be done is summarized in the form of a *Permutation Theorem,* in the sense of Kleene [15] and Curry [6], and underlies the material of Sections 5 and 6.

In the final refinement of the paper, the well-typing condition on the instantiation in the notion of U-proof $\langle \psi, \sigma \rangle$ is weakened to require only that there exist a legal permutation of ψ, say ψ^*, in which the instantiation is well-typed; *i.e.*, $\psi^*\sigma$ is an L-proof. Of course the crucial computational question is whether the existence of at least one suitable ψ^*, given ψ and σ, can be determined as the search progresses. We show in Section 6 that this is indeed the case using a *reduction ordering:* a notion which was introduced by Bibel [4] for classical connectives and extended by the second author in [29] to various non-classical connectives.[3] In effect we consider two derivations to be equivalent with respect to a given instantiation if they are permutation variants of each other. The reduction ordering ensures that there is always at least one derivation in the equivalence class in which the instantiation is well-typed.

The reader may find it helpful to refer back to the overview given above to identify the motivation for various technicalities below.

This paper extends an earlier paper [25] in that we discuss the proof-theoretic foundation of the sequent calculus L and in that we provide a preliminary analysis of the notion of *search space* which allows us to prove certain *containment* results. We also provide a proof of the *Permutation Theorem* for the $\lambda\Pi$-calculus and provide examples to illustrate the important definitions.

2 $\lambda\Pi$-calculus: a theory of dependent function types

The $\lambda\Pi$-calculus is a language with entities on three levels: *objects, types and families of types,* and *kinds.* Objects are classified by types, types and families of types by kinds. The distinguished kind Type classifies the types; the other kinds classify functions f which yield a type $f(x_1)\ldots(x_n)$ when applied to objects $x_1, \ldots, x_n$ of certain types determined by the kind of f. Any function definable in the system has a type as domain, while its

[3]The condition is equivalent to an enhanced "occurs-check" in the unification algorithm if a suitable notion of *Skolem function* were introduced. The suitable notion is *not* the obvious one that the reader might suppose from experience of classical quantifiers, not least because the logic under investigation here is intuitionistic. In general, the theoretical diversion via Skolemization is unnecessary and can be difficult to justify semantically, even in simple type theory (*cf.* [19]). Our approach follows Herbrand's Theorem (which is finitary) rather than the Skolem-Herbrand-Gödel Theorem (which is not).

range can either be a type, if it is an object, or a kind, if it is a family of types. The $\lambda\Pi$-calculus is therefore predicative.

The theory we shall deal with is a formal system for deriving assertions of one of the following shapes:

$\vdash \Sigma$ sig	Σ is a signature
$\vdash_\Sigma \Gamma$ context	Γ is a context
$\Gamma \vdash_\Sigma K$ kind	K is a kind
$\Gamma \vdash_\Sigma A{:}K$	A has kind K
$\Gamma \vdash_\Sigma M{:}A$	M has type A

where the syntax is specified by the following grammar:

$$\begin{array}{lcl} \Sigma & ::= & \langle\rangle \mid \Sigma, c:K \mid \Sigma, c:A \\ \Gamma & ::= & \langle\rangle \mid \Gamma, x:A \\ K & ::= & \mathrm{Type} \mid \Pi x:A.K \\ A & ::= & c \mid \Pi x:A.B \mid \lambda x:A.B \mid AM \\ M & ::= & c \mid x \mid \lambda x:A.M \mid MN \end{array}$$

We let M and N range over expressions for objects, A and B for types and families of types, K for kinds, x and y over variables, and c over constants.

We refer to the collection of variables declared in a context Γ as $\mathrm{Dom}(\Gamma)$. We assume α-conversion throughout. The inference rules of the $\lambda\Pi$-calculus appear in Table 1. We shall refer to this system as N to emphasize that it is a system of natural deduction.

A summary of the major metatheorems pertaining to N and its reduction properties are given by Theorem 2.1, [11], [27], [12], [2].

THEOREM 2.1 (THE BASIC METATHEORY OF THE $\lambda\Pi$-CALCULUS) *Let X range over basic assertions of the form $A{:}K$ and $M{:}A$. We write* N *proves $\Gamma \vdash_\Sigma X$ to denote the provability of the assertion $\Gamma \vdash_\Sigma X$ in the system* N.

1. *Thinning is an admissible rule: if* N proves $\Gamma \vdash_\Sigma X$ *and* N proves $\vdash_{\Sigma,\Sigma'} \Gamma, \Gamma'$ context, *then* N proves $\Gamma, \Gamma' \vdash_{\Sigma,\Sigma'} X$.

2. *Transitivity is an admissible rule: if* N proves $\Gamma \vdash_\Sigma M{:}A$ *and* N proves $\Gamma, x:A, \Delta \vdash_\Sigma X$, *then* N proves $\Gamma, \Delta[M/x] \vdash_\Sigma X[M/x]$.

3. *Uniqueness of types and kinds: if* N proves $\Gamma \vdash_\Sigma M{:}A$ *and* N proves $\Gamma \vdash_\Sigma M{:}A'$, *then* $A =_{\beta\eta} A'$, *and similarly for kinds.*

4. *Subject reduction: if* N proves $\Gamma \vdash_\Sigma M{:}A$ *and* $M \twoheadrightarrow_{\beta\eta} M'$, *then* N proves $\Gamma \vdash_\Sigma M'{:}A$, *and similarly for types.*

5. *All well-typed terms are strongly normalizing.*

6. *All well-typed terms are Church-Rosser.*

7. *Each of the five relations defined by the inference system of Table 1 is decidable, as is the property of being well-typed.*

8. *Predicativity: if* N proves $\Gamma \vdash_\Sigma M : A$ *then the type-free* λ*-term obtained by erasing all type information from* M *can be typed in the Curry type assignment system.*

9. *Strengthening is an admissible rule: if* N proves $\Gamma, x{:}A, \Gamma' \vdash_\Sigma X$ *and if* $x \notin \mathrm{FV}(\Gamma') \cup \mathrm{FV}(X)$ *then* N proves $\Gamma, \Gamma' \vdash_\Sigma X$.

A term is said to be *well-typed in a signature and context* if it can be shown to either be a kind, have a kind, or have a type in that signature and context. A term is *well-typed* if it is well-typed in some signature and context. The notion of $\beta\eta$-reduction, written $\rightarrow_{\beta\eta}$, can be defined both at the level of objects and at the level of types and families of types in the obvious way, for details see [12]. $M =_{\beta\eta} N$ iff $M \twoheadrightarrow_{\beta\eta} P$ and $N \twoheadrightarrow_{\beta\eta} P$ for some term P, where * denotes reflexive and transitive closure. For simplicity we shall write $\rightarrow_{\beta\eta}$ for $\twoheadrightarrow_{\beta\eta}$.

We stress that all five of the forms of assertion defined by the inference rules of Table 1 are decidable. We also stress that N is a system of first-order types in the following sense: the Π-type formation rule has the form:

$$\frac{\Gamma \vdash_\Sigma A{:}\mathrm{Type} \quad \Gamma, x{:}A \vdash_\Sigma B{:}\mathrm{Type}}{\Gamma \vdash_\Sigma \Pi x{:}A\,.\,B : \mathrm{Type}};$$

both A and B must be of kind Type (see also Rule 11 of Table 1). There are no variables of kind Type and consequently there are no higher-order types (of kind Type).

The $\lambda\Pi$-calculus may be conservatively extended with non-dependent function types, $A \rightarrow B$, denoting $\Pi x{:}A.B$ whenever x does not occur free in B. Similarly, we can write $A \rightarrow K$, for $\Pi x{:}A.K$ when x does not occur free in K. This extension is particularly valuable from the point of view of proof-search in the Gentzen-style system defined below, in that it allows us to separate out the propositional part of the logic from the quantificational part: the former gains a full analysis *within* the system while the latter (ΠI), as we have seen, requires external notions. Henceforth we shall use N to denote the system with explicit $\rightarrow$ constructions; *i.e.*, the rules of Table 2 are added to those of Table 1. The basic metatheoretic properties of Theorem 2.1 hold for this extended syntax.

In a more general setting, the $\lambda\Pi$-calculus is related to the logic $\lambda\mathbf{P}$ of Barendregt's cube, Barendregt *et al.* [3]. The principal differences are that (i) in the $\lambda\Pi$-calculus the signature is distinguished as an initial segment of the context in which all kind-declarations must occur and from which discharge is not permitted, and (ii) in the $\lambda\Pi$-calculus the η-rule is admitted. The presence of the η-rule increases the difficulty of the proof of Theorem 2.1, in particular the proof of the Church–Rosser property (Part 6) [27].

3 A metacalculus for N.

The inference rules of the system N represent a linearized natural deduction presentation of the $\lambda\Pi$-calculus. The system is said to be:

Valid Signature

$$\frac{}{\vdash \langle\rangle \text{ sig}} \tag{3}$$

$$\frac{\vdash \Sigma \text{ sig} \quad \vdash_\Sigma K \text{ kind} \quad c \notin \mathrm{Dom}(\Sigma)}{\vdash \Sigma, c : K \text{ sig}} \tag{4}$$

$$\frac{\vdash \Sigma \text{ sig} \quad \vdash_\Sigma A\text{: Type} \quad c \notin \mathrm{Dom}(\Sigma)}{\vdash \Sigma, c : A \text{ sig}} \tag{5}$$

Valid Context

$$\frac{\vdash \Sigma \text{ sig}}{\vdash_\Sigma \langle\rangle \text{ context}} \tag{6}$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad \Gamma \vdash_\Sigma A\text{: Type} \quad x \notin \mathrm{Dom}(\Gamma)}{\vdash_\Sigma \Gamma, x : A \text{ context}} \tag{7}$$

Valid Kinds

$$\frac{\vdash_\Sigma \Gamma \text{ context}}{\Gamma \vdash_\Sigma \text{Type kind}} \tag{8}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma K \text{ kind}}{\Gamma \vdash_\Sigma \Pi x : A.K \text{ kind}} \tag{9}$$

Valid Elements of a Kind

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad c : K \in \Sigma}{\Gamma \vdash_\Sigma c\text{:} K} \tag{10}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma B\text{: Type}}{\Gamma \vdash_\Sigma \Pi x : A.B\text{: Type}} \tag{11}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma B\text{:} K}{\Gamma \vdash_\Sigma \lambda x : A.B\text{:} \Pi x : A.K} \tag{12}$$

$$\frac{\Gamma \vdash_\Sigma B\text{:} \Pi x : A.K \quad \Gamma \vdash_\Sigma N\text{:} A}{\Gamma \vdash_\Sigma BN\text{:} K[N/x]} \tag{13}$$

$$\frac{\Gamma \vdash_\Sigma A\text{:} K \quad \Gamma \vdash_\Sigma K' \text{ kind} \quad K =_{\beta\eta} K'}{\Gamma \vdash_\Sigma A\text{:} K'} \tag{14}$$

Valid Elements of a Type

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad c : A \in \Sigma}{\Gamma \vdash_\Sigma c\text{:} A} \tag{15}$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad x : A \in \Gamma}{\Gamma \vdash_\Sigma x\text{:} A} \tag{16}$$

$$\Pi \quad \frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma M\text{:} B}{\Gamma \vdash_\Sigma \lambda x : A.M\text{:} \Pi x : A.B} \tag{17}$$

$$\Pi\mathrm{E} \quad \frac{\Gamma \vdash_\Sigma M\text{:} \Pi x : A.B \quad \Gamma \vdash_\Sigma N\text{:} A}{\Gamma \vdash_\Sigma MN\text{:} B[N/x]} \tag{18}$$

$$\frac{\Gamma \vdash_\Sigma M\text{:} A \quad \Gamma \vdash_\Sigma A'\text{: Type} \quad A =_{\beta\eta} A'}{\Gamma \vdash_\Sigma M\text{:} A'} \tag{19}$$

Table 1

Valid Kinds

$$\frac{\Gamma \vdash_\Sigma A\colon \text{Type} \quad \Gamma, x\colon A \vdash_\Sigma K \text{ kind} \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma A \to K \text{ kind}} \tag{20}$$

Valid Elements of a Kind

$$\frac{\Gamma \vdash_\Sigma A\colon \text{Type} \quad \Gamma, x\colon A \vdash_\Sigma B\colon K \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma \lambda x\colon A\,.\,B : A \to K} \tag{21}$$

$$\frac{\Gamma \vdash_\Sigma B\colon A \to K \quad \Gamma \vdash_\Sigma N\colon A}{\Gamma \vdash_\Sigma BN : K} \tag{22}$$

$$\frac{\Gamma \vdash_\Sigma A\colon \text{Type} \quad \Gamma, x\colon A \vdash_\Sigma B : \text{Type} \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma A \to B : \text{Type}} \tag{23}$$

Valid Elements of a Type

$$\to \text{I} \quad \frac{\Gamma \vdash_\Sigma A\colon \text{Type} \quad \Gamma, x\colon A \vdash_\Sigma M\colon B \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma \lambda x\colon A\,.\,M : A \to B} \tag{24}$$

$$\to \text{E} \quad \frac{\Gamma \vdash_\Sigma M\colon A \to B \quad \Gamma \vdash_\Sigma N\colon A}{\Gamma \vdash_\Sigma MN : B} \tag{25}$$

Table 2

- *Linearized* because the hypotheses are recorded in the context;
- *Natural deduction* because of the form of the application rules — they correspond to the implication-elimination rule of natural deduction formulations of intuitionistic logic [10], [23], [13].

In this section we introduce a system G for the $\lambda\Pi$-calculus in the style of Gentzen's sequent calculus LJ [10]. Via the formulae-as-types isomorphism between natural deduction proofs in intuitionistic propositional logic and the terms of the simply-typed lambda calculus [13], G provides a basis for a metacalculus for calculating the *normal* proofs of N.

In the setting of the formulae-as-types isomorphism, the $\to$E rule of the system N:

$$\to \text{E} \quad \frac{\Gamma \vdash_\Sigma M : A \to B \quad \Gamma \vdash_\Sigma N\colon A}{\Gamma \vdash_\Sigma MN\colon B}$$

corresponds to the sequent calculus rule:

$$\supset \text{E} \quad \frac{\Gamma \vdash \phi \supset \psi \quad \Gamma \vdash \phi}{\Gamma \vdash \psi}$$

of intuitionistic propositional logic (in the obvious notation). In the system G the $\rightarrow$E rule is replaced by the $\rightarrow l$ rule:

$$\rightarrow l \ \frac{@{:}A \rightarrow B \in \Sigma \cup \Gamma \quad \Gamma \vdash_\Sigma N{:}A \quad \Gamma, y{:}B \vdash_\Sigma M{:}D}{\Gamma \vdash_\Sigma M[@N/y]{:}D}$$

for $y \notin \mathrm{FV}(D)$, which says that given some constant or variable $@{:}A \rightarrow B \in \Sigma \cup \Gamma$, some object N of type A in the context Γ, then given an object M which inhabits the type D in the context $\Gamma, y{:}B$, the rule $\rightarrow l$ constructs an object $M[@N/y]$ which inhabits the type D in the context Γ. Under the formulae-as-types isomorphism, the corresponding sequent calculus rule of intuitionistic propositional logic is

$$\frac{\Gamma \vdash \phi \quad \psi, \Delta \vdash \gamma}{\phi \supset \psi, \Gamma, \Delta \vdash \gamma}$$

see [13], Section 5.

In a similar way, the ΠE rule of the system N:

$$\Pi\mathrm{E} \ \frac{\Gamma \vdash_\Sigma M : \Pi x{:}A\,.\,B \quad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma MN{:}B[N/x]},$$

is replaced in the system G by the Πl rule:

$$\Pi l \ \frac{@{:}\Pi x{:}A.B \in \Sigma \cup \Gamma \ \ \Gamma \vdash_\Sigma N{:}A \ \ B[N/x] =_{\beta\eta} C \ \ \Gamma, y{:}C \vdash_\Sigma M{:}D}{\Gamma \vdash_\Sigma M[@N/y]{:}D},$$

for $y \notin \mathrm{FV}(D)$, which is understood in a manner similar to the $\rightarrow l$ rule.

Extension of the formulae-as-types correspondence to universal quantifiers and dependent types in the setting of *Generalized Type Systems* is reported in the work of Barendregt [3]. This work is due variously to Berardi, Terlouw, Geuvers and Barendsen. The extension of this correspondence to Gentzen-style Generalized Type Systems and sequent calculi is under investigation (*cf.* Martin-Löf [17]).

The system G includes the cut rule:

$$\frac{\Gamma, x{:}A \vdash_\Sigma X \quad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma X[N/x]}.$$

G is relatively sound and complete with respect to N.

PROPOSITION 3.1 (SOUNDNESS AND COMPLETENESS)

$$\mathbf{G} \text{ proves } \Gamma \vdash_\Sigma M{:}A \quad \textit{iff} \quad \mathbf{N} \text{ proves } \Gamma \vdash_\Sigma M{:}A.$$

A cut-elimination theorem can also be obtained; that is G\cut is relatively complete with respect to N provided we work with inhabiting objects in β-normal form.

THEOREM 3.2 (CUT-ELIMINATION) *Let* G\cut *be the system* G *without the cut rule and let* M *and* A *be* β*-normal forms. Then*

$$\mathrm{G\backslash cut} \text{ proves } \Gamma \vdash_\Sigma M{:}A \quad \textit{iff} \quad \mathbf{N} \text{ proves } \Gamma \vdash_\Sigma M{:}A.$$

While this might seem surprising at first sight (as we might expect completeness for *all* inhabiting objects by analogy with usual completeness of systems without cut) it actually corresponds to the fact that in such systems we do not get completeness for proofs, but rather that every proof in the system with cut *which is in normal form* corresponds to a proof in the system without cut. Indeed, the proof of the completeness of the system without cut may be considered to be analogous to that given by Prawitz [23] for proofs in normal form. To see this observe that the rules of G without the cut rule are such that only β-normal forms can be derived from β-normal forms. Note however that the cut rule is able to derive terms that are not β-normal forms from β-normal forms because the cut-term N might be a λ-abstraction.

The reader is referred to [24] for the proofs of Proposition 3.1 and Theorem 3.2. Where no confusion can arise, we omit "N proves", "G proves". "G\cut proves", *etc.*.

For proof-search we are not interested in the decidable typing assertions $\Gamma \vdash_\Sigma M{:}A$, in which the object M encodes the proof of the assertion, but rather in semi-decidable inhabitation assertions of the form $\Gamma \Rightarrow_\Sigma A$, which is to be read as "A is inhabited relative to the context Γ and signature Σ". Here, a proof remains to be calculated, and may not exist. This leads us to the following definition of sequent:

DEFINITION 3.3 (Sequent) A *sequent* is a triple $\langle \Sigma, \Gamma, A \rangle$, written $\Gamma \Rightarrow_\Sigma A$, where Σ is a signature, Γ a context and A a type (family). The intended interpretation of the sequent is the (meta)assertion:

$$(\exists M) \quad \mathrm{N} \text{ proves } \Gamma \vdash_\Sigma M{:}A. \qquad \square$$

There are two candidates for a calculus for manipulating such sequents. The first is obtained from the natural deduction system N by deleting inhabiting objects. This gives rise to a Π-elimination rule of the form:

$$\Pi\mathrm{E} \quad \frac{\Gamma \Rightarrow_\Sigma \Pi x{:}A\,.\,B}{\Gamma \Rightarrow_\Sigma B[N/x]} \qquad \text{(a) } \mathrm{N} \text{ proves } \Gamma \vdash_\Sigma N{:}A$$

which is similar to the usual natural deduction rule for quantifiers. The fact that the type A in the premiss is not a subformula of the conclusion means that a proof procedure based on such a calculus would have to invent the type.

The second candidate is obtained from the sequent calculus G\cut by deleting inhabiting objects. This gives rise to a Πl rule of the form:

$$\Pi l \quad \frac{\Gamma, y{:}D \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l} \text{(a) } @{:}\Pi x{:}A.B \in \Sigma \cup \Gamma \\ \text{(b) } y \notin \mathrm{Dom}(\Gamma) \\ \text{(c) G\cut proves } \Gamma \vdash_\Sigma N{:}A \\ \text{(d) } B[N/x] \rightarrow_{\beta\eta} D \end{array}$$

This rule is preferable to the elimination rule for proof-search because it restricts the non-determinism to the choice of object M, and such choices may be calculated by unification. Thus we define a semi-logicistic calculus L for deriving sequents as follows:

DEFINITION 3.4 (The system **L**)

$$Ax1 \qquad \Gamma, x{:}A, \Gamma' \Rightarrow_\Sigma A$$

$$Ax2 \qquad \Gamma \Rightarrow_{\Sigma, c:A, \Sigma'} A$$

$$\to r \qquad \frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma A \to B} \qquad \text{(a) } x \notin \mathrm{Dom}(\Gamma)$$

$$\Pi r \qquad \frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B} \qquad \text{(a) } x \notin \mathrm{Dom}(\Gamma)$$

$$\to l \qquad \frac{\Gamma \Rightarrow_\Sigma A \qquad \Gamma, y{:}B \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l}\text{(a) } @{:}A \to B \in \Sigma \cup \Gamma \\ \text{(b) } y \notin \mathrm{Dom}(\Gamma)\end{array}$$

$$\Pi l \qquad \frac{\Gamma, y{:}D \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l}\text{(a) } @{:}\Pi x{:}A.B \in \Sigma \cup \Gamma \\ \text{(b) } y \notin \mathrm{Dom}(\Gamma) \\ \text{(c) G\textbackslash cut proves } \Gamma \vdash_\Sigma M{:}A \\ \text{(d) } B[N/x] \to_{\beta\eta} D\end{array}$$

Here $B[N/x]$ denotes capture-avoiding substitution of N for x, and the conditions $x, y \notin \mathrm{Dom}(\Gamma)$ mean that x and y do not label any declaration in the context Γ. For simplicity and efficiency, we work exclusively with $\beta\eta$-normal forms, and for such terms syntactic identity ($\equiv$) is taken up to α-congruence (change of bound variable). As usual we refer to the variable x of the Πr rule as the *eigenvariable* of the inference. We can ensure that in any derivation eigenvariables occur only in sequents above the inference at which they are introduced. Both distinguished occurrences of A are said to be the *principal formula* of the $Ax1$ and $Ax2$ rules. $A \to B$ is the principal formula of the $\to r$ and $\to l$ rules and $\Pi x{:}A.B$ is the principal formula of the Πr and Πl rules. A and B are the *side formulae* of the $\to r$ and Πr rules, and A and $B[N/x]$ are the side formulae of the Πl rule. A, B and C are the side formulae of the $\to l$ rule. $@$ is the *principal atom* in the $\to l$ and Πl rules. **L**-*derivations* are trees of sequents regulated by the operational rules, and **L**-*proofs* are derivations whose leaves are axioms. □

In this paper, we restrict our attention to inhabitation assertions for types of kind Type.[4]

DEFINITION 3.5 (Well-formed sequent) A sequent $\Gamma \Rightarrow_\Sigma A$ is said to be *well-formed* just in case G\cut proves $\Gamma \vdash_\Sigma A$: Type. □

From Theorem 2.1 we have:

PROPOSITION 3.6 *The well-formedness problem for sequents is decidable.*

[4]This is a very natural restriction for the use made of the $\lambda\Pi$-calculus in the (Edinburgh) Logical Framework, *q.v.* [24].

For proof-search derivations are constructed from the root, or *endsequent*, toward the leaves in the spirit of Kleene [16] and systems of tableaux [28] (see Section 4). In support of this usage we have the following result:

PROPOSITION 3.7 (PYM, 1990) *For well-formed sequents* $\Gamma \Rightarrow_\Sigma A$,

$$\mathbf{L} \text{ proves } \Gamma \Rightarrow_\Sigma A \quad \textit{iff} \quad (\exists M) \quad \mathbf{G}\backslash\text{cut proves } \Gamma \vdash_\Sigma M{:}A. \quad \square$$

We revert to the system G\cut to decide if the endsequent is well-formed. If so, L may be used to prove inhabitation of A with respect to the context Γ. Moreover the inhabiting object M is obtained as the extract-object of the L-proof $\Gamma \Rightarrow_\Sigma A$, constructed by replacing the inhabiting objects in the proof, starting with the constants and variables at the leaves, in the manner of the system G\cut, *q.v.* [24]. L is not fully logicistic since an appeal is still made to G\cut for each application of the Πl rule (third side condition).

4 Unification and search

In this section we introduce a new calculus U which allows unification to be used to calculate terms for use with Πl inferences. The calculation is constrained by the propositional structure of the derivation in such a way that only terms that are *relevant* to the formation of a proof (rather than a mere derivation) are considered (*cf.* [26]). The search space of U is then shown to be a proper subspace of the search space of L containing representatives for all proofs in the latter. U-based search is therefore a *complete* and *uniform* improvement over L-based search.

We introduce a new syntactic class of variables called *indeterminates*, denoted by lowercase Greek letters α, β, *etc.*, and extend the syntactic category of objects to include them thus:

$$\textit{Objects} \quad M \ ::= \ c \mid \alpha \mid x \mid \lambda x{:}A.M \mid MN.$$

Notice that indeterminates cannot appear λ-bound. By virtue of this extension, entities of all syntactic classes may now contain indeterminates as subterms. When we wish to emphasize that a syntactic entity does not contain any indeterminates we shall refer to it as being *ground*.

We define U by dropping the axiom schemata of L and modifying the Πl rule as follows:

DEFINITION 4.1 (U-derivation) The rules of U consist of the $\rightarrow r$, $\rightarrow l$ and Πr rules of L, together with the rule

$$\Pi l \quad \frac{\Gamma, z{:}B[\alpha/x] \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l}(\text{a}) \ @ : \Pi x{:}A.B \in \Sigma \cup \Gamma \\ (\text{b}) \ z \notin \text{Dom}(\Gamma).\end{array}$$

U-*derivations* are trees regulated by the above rules such that the sequent at the root of the tree is well-formed. In applications of the Πl rule we call Γ the *typing context* of α and A the *type* of α. We shall use $\text{Ind}(\psi)$ and $\text{Eig}(\psi)$ to denote the indeterminates and eigenvariables respectively of a derivation ψ. $\square$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, p{:}-, y{:}B(\alpha), s{:}E, q{:}C(\beta) \Rightarrow_\Sigma C(y)}{\Gamma, p{:}-, y{:}B(\alpha), s{:}E \Rightarrow_\Sigma C(y)}\,\Pi l_2}{\Gamma, p{:}-, y{:}B(\alpha) \Rightarrow_\Sigma E \to C(y)}\,{\to} r}{\Gamma, p{:}- \Rightarrow_\Sigma (\Pi y{:}B(\alpha).(E \to C(y)))}\,\Pi r \qquad \Gamma, p{:}-, r{:}D(\alpha) \Rightarrow_\Sigma D(a)}{\Gamma, p{:}(\Pi y{:}B(\alpha).(E \to C(y))) \to D(\alpha) \Rightarrow_\Sigma D(a)}\,{\to} l$$
$$\dfrac{\Gamma, p{:}(\Pi y{:}B(\alpha).(E \to C(y))) \to D(\alpha) \Rightarrow_\Sigma D(a)}{\Gamma \Rightarrow_\Sigma D(a)}\,\Pi l_1$$

$\Sigma = A{:}\mathrm{Type}, a{:}A, B{:}A \to \mathrm{Type}, b{:}B(a), C{:}B(a) \to \mathrm{Type}, D{:}A \to \mathrm{Type}, E{:}\mathrm{Type}$
$\Gamma = u{:}\Pi x{:}A.((\Pi y{:}B(x).(E \to C(y))) \to D(x)), \quad v{:}\Pi z{:}B(a).C(z)$

Typing context for α: Γ
Typing context for β: $\Gamma, p{:}(\Pi y{:}B(\alpha).(E \to C(y))) \to D(\alpha), y{:}B(\alpha), s{:}E$

Indeterminates: $\mathrm{Ind}(D1) = \{\alpha, \beta\}$
Eigenvariables: $\mathrm{Eig}(D1) = \{y\}$

Figure 1: **U**-derivation D1.

Indeterminates are, in a certain sense, bound within derivations like eigenvariables: their exact identity is irrelevant since each one may be indexed by the unique Πl inference that gives rise to it in a derivation. In the sequel we shall take identity of **U**-derivations up to change in such indeterminates (*cf.* pure variable proofs). Notice that we have not yet defined **U**-*proofs* since there are no axiom schemata. **U** thus consists of four operational rules; the Πl rule no longer contains an external appeal to G\cut or a choice of term, but is otherwise identical to the Πl rule of **L**.

EXAMPLE 4.2 Figure 1 contains an example of a **U**-derivation. □

DEFINITION 4.3 (Instantiation) An *instantiation* for a derivation ψ is a mapping from $\mathrm{Ind}(\psi)$ to *ground* objects. The (capture-avoiding) extension of instantiations to all of the constructs of the language is defined in the obvious way.[5] □

The following notion compensates for the absence of axiom schemata in **U**.

DEFINITION 4.4 (Closure) A sequent $\Gamma \Rightarrow_\Sigma A$ is said to be *closed under* an instantiation σ just in case $B\sigma \equiv A\sigma$ for some declaration $@{:}B \in \Sigma \cup \Gamma$. A **U**-derivation is said to be *closed under* σ just in case all of its leaf sequents are closed under σ. (Again, we work exclusively with $\beta\eta$-normal forms.) □

We are interested in instantiations that are *well-typed* in the following sense.

[5]Formally, we must ensure that any new variables generated during unification (*q.v.* [14], [24]) are distinct from the eigenvariables and indeterminates of the derivation.

DEFINITION 4.5 (Well-typing) An instantiation σ is said to be *well-typed* in a given U-derivation just in case for every indeterminate α of the derivation, with typing context Γ and type A, we have:

$$\mathbf{G}\backslash\text{cut proves } \Gamma\sigma \vdash_\Sigma \alpha\sigma{:}A\sigma. \quad \square$$

LEMMA 4.6 *If σ is a well-typed instantiation for* U*-derivation ψ with domain* $\mathrm{Ind}(\psi)$, *then $\psi\sigma$ is an* L*-derivation.*

PROOF. By induction on the structure of U-derivations. The well-typing condition ensures that the image under σ of each instance of a Πl rule of U in ψ satisfies the side conditions on the Πl rule of L. The remaining operational rules are common to the two systems. $\square$

We are now in a position to define a notion of proof for U.

DEFINITION 4.7 (U-proof) A U-*proof* is a pair $\langle\psi,\sigma\rangle$, where ψ is a U-derivation and σ an instantiation with domain $\mathrm{Ind}(\psi)$, such that (1) ψ is closed under σ, and (2) σ is well-typed in ψ. $\square$

THEOREM 4.8 *If $\langle\psi,\sigma\rangle$ is a* U*-proof, $\psi\sigma$ is an* L*-proof.*

PROOF. By the lemma above $\psi\sigma$ is an L-derivation; closure ensures that the leaves of $\psi\sigma$ are L-axioms.

We remark that it is immediate that an L-proof gives rise to a unique U-proof (up to change in indeterminates): this is the converse of Theorem 4.8.

EXAMPLE 4.9 The mappings $\theta_1 = \{\alpha \mapsto a, \beta \mapsto y\}$ and $\theta_2 = \{\alpha \mapsto a, \beta \mapsto b\}$ are instantiations for derivation (D1) of Figure 1. Both are well-typed since (for $i = 1, 2$) $\Gamma\theta_i \vdash_\Sigma \alpha\theta_i{:}A$ and $(\Gamma, p{:}-, y{:}B(\alpha), s{:}E)\theta_i \vdash_\Sigma \beta\theta_i{:}B(\alpha)\theta_i$. θ_1 closes the derivation whereas θ_2 does not. Consequently $\langle D1, \theta_1\rangle$ is a U-proof, $\langle D1, \theta_2\rangle$ is not. $\square$

We can use a unification algorithm to calculate instantiations when a putative leaf has been reached, then check the well-typing condition using G\cut. In this way the propositional structure of the U-derivation constrains the search for terms used at Πl inferences. The reader should note that the structure of the type of an atom, $@{:}A \in \Sigma \cup \Gamma$, is fixed by the (well-formed) endsequent and that this structure cannot be altered by subsequent instantiations of indeterminates: *i.e.*, the outermost connective of A, and of all of its sub*formulae*, is unchanged under the instantiation of indeterminates.

We now develop a simple argument that this use of U constitutes a uniform improvement over the standard use of L for proof search. The argument is relative to an arbitrary well-formed sequent $\Gamma \Rightarrow_\Sigma A$. Let Der(L) (resp. Der(U)) denote the (r.e.) set of L-derivations (resp. U-derivations) of the given sequent. There is a natural ordering $\sqsubseteq_{\mathrm{L}}$ (resp. $\sqsubseteq_{\mathrm{U}}$) on Der(L) (resp. on Der(U)) induced by the obvious notion of *subderivation*.

Searching for a proof of $\Gamma \Rightarrow_\Sigma A$ using L consists of *reducing* derivations one step at a time: *i.e.*, matching the leaf of a derivation with the conclusion of a rule in L and forming a new derivation with the premisses of the rule as new leaves. The original derivation is thus a subderivation of the new one.

DEFINITION 4.10 (Search space of **L**) The *search space* of **L**, denoted by Sp(**L**), is the ordered set $\langle \mathrm{Der}(\mathbf{L}), \sqsubseteq_{\mathbf{L}} \rangle$. □

Sp(**L**) may of course be viewed as a directed graph, the nodes being derivations and the arcs indicating one-step reductions.

Theorem 4.8 and its converse allow us to express the search space of **L** in terms of **U**-derivations and instantiations in preparation for comparison with the search space of **U**. Let D be the set of pairs $\langle \psi, \sigma \rangle$ such that (1) $\psi \in \mathrm{Der}(\mathbf{U})$, (2) σ is well-typed in ψ and (3) $\mathrm{Dom}(\sigma) = \mathrm{Ind}(\psi)$. Define $\sqsubseteq$ on $D \times D$ by: $\langle \psi, \sigma \rangle \sqsubseteq \langle \phi, \tau \rangle$ iff $\psi \sqsubseteq_{\mathbf{U}} \phi$ and $\sigma = \tau \restriction \mathrm{Ind}(\psi)$.

PROPOSITION 4.11 $\langle D, \sqsubseteq \rangle \cong \langle \mathrm{Der}(\mathbf{L}), \sqsubseteq_{\mathbf{L}} \rangle$.

PROOF. Define a mapping from $\langle D, \sqsubseteq \rangle$ to $\langle \mathrm{Der}(\mathbf{L}), \sqsubseteq_{\mathbf{L}} \rangle$ by $\langle \psi, \sigma \rangle \mapsto \psi\sigma$. Theorem 4.8 and its converse ensure that this mapping is an isomorphism of the carrier sets, and it is easy to check that it preserves the order. □

At this point we could define the search space of **U** to be those pairs $\langle \psi, \sigma \rangle \in D$ such that ψ is closed under σ. Such a definition would yield the desired containment relation between the search spaces of the two calculi. However, we postpone the definition and develop a finer analysis of the structure of **U**-search.

DEFINITION 4.12 (Relevance) Let $\langle \psi, \sigma \rangle \in D$, $\alpha \in \mathrm{Dom}(\sigma) = \mathrm{Ind}(\psi)$ and $\tau = \sigma \restriction \mathrm{Dom}(\sigma) \setminus \{\alpha\}$. Let x be the eigenvariable introduced by the ΠI reduction that introduced α. Finally, let ϕ be the derivation obtained from ψ by deleting the ΠI reduction that introduced α.

(i) The component of σ, $\alpha \mapsto t$, is said to be *directly relevant* to $\langle \psi, \sigma \rangle$ if the number of closed leaves in $\psi\tau$ is strictly less than the number in $\psi\sigma$.

(ii) $\alpha \mapsto t$ is said to be *indirectly relevant* to $\langle \psi, \sigma \rangle$ if either (1) there is some $\beta \in \mathrm{Dom}(\tau)$, directly or indirectly relevant to $\langle \psi, \tau \rangle$, that is *not* well-typed in $\langle \phi, \tau \rangle$; or (2) x is either free in some type in ψ, or a principal atom in ψ.

(iii) The indeterminate α is said to be *relevant* if either it is directly relevant or it is indirectly relevant, and is said to be *irrelevant* if it is neither. □

Informally, $\alpha \in \mathrm{Dom}(\sigma)$ is relevant if either it or its associated eigenvariable contributes to either the closing a leaf of ψ, either directly via closure, or indirectly by contributing to the well-typing of a directly relevant component, or to the construction of ψ via an $\rightarrow I$ or a ΠI rule.

DEFINITION 4.13 (Minimality) $\langle \psi, \sigma \rangle \in D$ is said to be *propositionally minimal* just in case every $\alpha \in \mathrm{Dom}(\sigma)$ is relevant. □

EXAMPLE 4.14 Both components of θ_1 are directly relevant in $\langle D1, \theta_1 \rangle$. α is also indirectly relevant. α is directly relevant in θ_2, but β is irrelevant. θ_1 is therefore minimal, whereas θ_2 is not. □

While it is not true that every **L**-proof, seen as an element $\langle \psi, \sigma \rangle \in D$, has σ minimal, we have:

LEMMA 4.15 *If $\langle \psi, \sigma \rangle \in D$ and if $\psi\sigma$ is an* **L***-proof, there is some ϕ such that $\left\langle \phi, \sigma \restriction \mathrm{Ind}(\phi) \right\rangle$ is propositionally minimal.*

PROOF. Let $\langle \psi, \sigma \rangle$ be an **L**-proof that is not propositionally minimal. This arises because there is a leaf sequent of ψ in which there is an irrelevant indeterminate α. This indeterminate is introduced to ψ by a Πl reduction which also introduces an eigenvariable x to ψ. Since α is irrelevant, x does not contribute either to the closure of ψ or to its construction (as the principal atom of a left rule). Furthermore, α does not contribute to the closure of ψ. Therefore we can delete from ψ the Πl reduction that introduced α and x to obtain a derivation χ which is closed under the instantiation $\tau = \sigma \restriction \mathrm{Ind}(\chi)$ and is such that $\langle \chi, \tau \rangle \in D$. By deleting the Πl reductions associated with all such irrelevant indeterminates, and by restricting σ to the remaining indeterminates, we obtain a (closed) propositionally minimal $\left\langle \phi, \sigma \restriction \mathrm{Ind}(\phi) \right\rangle \in D$, as required. □

DEFINITION 4.16 (Search space of **U**) The *search space* of **U**, denoted Sp(**U**), is the set of propositionally minimal pairs $\langle \psi, \sigma \rangle \in D$ such that ψ is closed under σ, ordered by $\sqsubseteq$. □

Indeed, the space Sp(**U**) is now the quotient of the space Sp(**L**) by both closure and propositional minimality. In particular:

THEOREM 4.17 Sp(**U**) $\subset$ Sp(**L**). *Moreover,* Sp(**U**) *contains representatives for all* **L***-proofs.*

PROOF. By definition and Lemma 4.15. □

We should like to think that the search for a proof using **U** consists of reducing derivations one step at a time, as in **L**, but with the modified Πl rule and the calculation via unification of instantiations that close leaves. The above analysis of Sp(**U**) is inadequate in that the definition of the search space of **U** really fails to capture this intuition. The development of an adequate analysis of these notions constitutes a current (and substantial) research project and is related to the possibilities for incremental search afforded by the notion of intrinsic well-typing developed in the sequel (see Section 6).

5 Permutability of inferences and proof-search

The typing contexts of the indeterminates in a **U**-derivation depend on the structure of the derivation. For example, the **U**-derivation (D2) shown in Figure 2 is another derivation of the same endsequent as derived by (D1) (Figure 1). (D1) and (D2) differ in the order in which inference rules are applied.

$$\dfrac{\dfrac{\dfrac{\Gamma, p{:}-, q{:}C(\beta), y{:}B(\alpha), s{:}E \Rightarrow_\Sigma C(y)}{\Gamma, p{:}-, q{:}C(\beta), y{:}B(\alpha) \Rightarrow_\Sigma E \to C(y)}\to r \quad \Gamma, p{:}-, q{:}C(\beta) \Rightarrow_\Sigma (\Pi y{:}B(\alpha).(E \to C(y)))}{\Gamma, p{:}-, q{:}C(\beta), r{:}D(\alpha) \Rightarrow_\Sigma D(a)}\Pi r}{\Gamma, p{:}(\Pi y{:}B(\alpha).(E \to C(y))) \to D(\alpha), q{:}C(\beta) \Rightarrow_\Sigma D(a)}\to l$$

$$\dfrac{\Gamma, p{:}(\Pi y{:}B(\alpha).(E \to C(y))) \to D(\alpha) \Rightarrow_\Sigma D(a)}{\Gamma \Rightarrow_\Sigma D(a)}\Pi l_2 \Pi l_1$$

$\Sigma = A{:}\mathrm{Type}, a{:}A, B{:}A \to \mathrm{Type}, b{:}B(a), C{:}B(a) \to \mathrm{Type}, D{:}A \to \mathrm{Type}, E{:}\mathrm{Type}$
$\Gamma = u{:}\Pi x{:}A.((\Pi y{:}B(x).(E \to C(y))) \to D(x)), \quad v{:}\Pi z{:}B(a).C(z)$

Typing context for α: Γ
Typing context for β: $\Gamma, p{:}(\Pi y{:}B(\alpha).(E \to C(y))) \to D(\alpha)$

Indeterminates: $\mathrm{Ind}(D2) = \{\alpha, \beta\}$
Eigenvariables: $\mathrm{Eig}(D2) = \{y\}$

Figure 2: U-derivation D2.

The instantiation $\theta = \theta_1 = \{\alpha \mapsto a, \beta \mapsto y\}$ closes both derivations, but whereas θ is well-typed in (D1), it is not well-typed in (D2) since

$$\Gamma, p{:}(\Pi y{:}B(\alpha).(E \to C(y))) \to D(\alpha)\ \not\vdash_\Sigma y{:}B(a).$$

Our final refinement is to alter the notion of U-proof so that existence of $\langle D2, \theta\rangle$ is sufficient to infer the existence of $\langle D1, \theta\rangle$, and hence to soundly terminate the search with success. That is, we investigate conditions under which rule instances may be *permuted* whilst leaving the endsequent and leaves of the derivation essentially unchanged. Implicitly we are defining an equivalence relation on derivations such that $\langle D2, \theta\rangle$ and $\langle D1, \theta\rangle$ are in the same equivalence class. The search strategy need construct *at most* one derivation from each class; the search being terminated if *at least* one member of the class is a proof.

The reader should be wary of supposing that the above problem can be overcome simply by defining a normal form for derivations in which inferences on the right are grouped as close to the root of the derivation as possible — in reduction terms: right reductions being preferred over left reductions.[6] Whilst this normal form certainly quotients the set of derivations, we obtain below (implicitly) a *stronger* equality on derivations and hence gain a smaller quotient space. Notice that both (D1) and (D2) are in this normal form. The relative order of Πl inferences is in fact important.[7]

Let ψ be a U-derivation of a given endsequent and let $\mathcal{F}_\psi$ denote the collection of inferences that comprise ψ. We use $\mathcal{F}_\psi(\Xi) \subseteq \mathcal{F}_\psi$ to denote the inferences of a given type

[6]Such a strategy is complete, in that it allows a proof to be found for any provable sequent, and finds proofs whose extract-objects (*q.v.* Note 8) are long $\beta\eta$-normal forms, see [24].

[7]Recall that in the LF, left Πs are used to encode *both* the universal *and* existential quantifiers of encoded logics [11].

Ξ, for Ξ one of the following: $\rightarrow l$, $\rightarrow r$, Πl or Πr. Let σ be an instantiation for ψ.

DEFINITION 5.1 The following binary relations are defined on $\mathcal{F}_\psi$:

(i) $R <_\psi R'$ iff a side formula of R is the principal formula of R' [8];

(ii) $R \ll_\psi R'$ iff R occurs below R' in ψ;

(iii) $R \sqsubset_\sigma R'$ iff the indeterminate or eigenvariable introduced by R is a free variable of $\alpha\sigma$, where α is the indeterminate introduced by R'. □

EXAMPLE 5.2 Consider the derivations (D1) and (D2) of Figures 1 and 2.

(i) We have

$$\mathcal{F}_1 = \mathcal{F}_2 = \{\Pi l_1, \Pi l_2, \Pi r, \rightarrow r, \rightarrow l\}.$$

Notice that an inference in a derivation is determined by its form (Πl, *etc.*) and its principal formula occurrence in the derivation. We have numbered the Πl inferences in each derivation to distinguish them. The above identification is useful here since (D1) arises from a permutation of the inferences of (D2). It will always be clear which derivation we have in mind when we refer to Πl_1 *etc.*.

(ii) Next, we have

$$<_1 = <_2 = < = \{(\Pi l_1, \rightarrow l), (\rightarrow l, \Pi r), (\Pi r, \rightarrow r)\}.$$

This should be expected from the subformula property of the system U given that $\mathcal{F}_1 = \mathcal{F}_2$.

$$\begin{aligned} \ll_1 &= \{(\Pi l_1, \rightarrow l), (\rightarrow l, \Pi r), (\Pi r, \rightarrow r), (\rightarrow r, \Pi l_2)\}^+ \\ \ll_2 &= \{(\Pi l_1, \Pi l_2), (\Pi l_2, \rightarrow l), (\rightarrow l, \Pi r), (\Pi r, \rightarrow r)\}^+, \end{aligned}$$

where $+$ indicates transitive closure. These orderings are just the tree orderings on the derivations.

(iii) Finally, we have

$$\sqsubset_\theta \;=\; \{(\Pi r, \Pi l_2)\}.$$

□

$\ll_\psi$ decomposes into sixteen subrelations: $\ll_\psi^{\Xi,\Omega} \subseteq \mathcal{F}_\psi(\Xi) \times \mathcal{F}_\psi(\Omega)$ for Ξ, Ω amongst $\rightarrow l$, $\rightarrow r$, Πl and Πr. $\ll_\psi$ and its subrelations are called the *skeletal orderings* of the derivation ψ. Notice also that $<_\psi$ is a subrelation of $\ll_\psi$.

[8] More accurately: iff a side formula of R is a "descendent" of the principal formula of R; we distinguish the "occurrences" of a formula in a derivation [16].

DEFINITION 5.3 (Reduction ordering) The *reduction ordering* induced by a U-derivation ψ and a instantiation σ is defined by:

$$\lhd_{\psi,\sigma} \;=_{\text{def}}\; \left(<_\psi \cup \prec_\psi \cup \sqsubset_\sigma\right)^+,$$

where the relation $\prec_\psi$ is defined by:

$$\prec_\psi \;=_{\text{def}}\; \ll_\psi \setminus \bigcup_{\Xi \in \mathcal{F}_\psi} \left(\ll_\psi^{\Pi l,\Xi} \cup \ll_\psi^{\Xi,\Pi l}\right).$$

□

EXAMPLE 5.4 We have that

$$\begin{aligned}
\bigcup_{\Xi \in \mathcal{F}_1} \ll_1^{\Pi l,\Xi} &= \{(\Pi l_1, \Pi l_2), (\Pi l_1, \rightarrow l), (\Pi l_1, \Pi r), (\Pi l_1, \rightarrow r)\} \\
\bigcup_{\Xi \in \mathcal{F}_2} \ll_2^{\Xi,\Pi l} &= \{(\Pi l_1, \Pi l_2)\}.
\end{aligned}$$

We leave it to the reader to verify that, as expected,

$$\bigcup_{\Xi \in \mathcal{F}_1} \left(\ll_1^{\Pi l,\Xi} \cup \ll_1^{\Xi,\Pi l}\right) = \bigcup_{\Xi \in \mathcal{F}_2} \left(\ll_2^{\Pi l,\Xi} \cup \ll_2^{\Xi,\Pi l}\right),$$

$$\prec_1 = \prec_2 = \prec \;=\; \{(\rightarrow l, \Pi r), (\Pi r, \rightarrow r)\}^+$$

and

$$\begin{aligned}
\lhd_{1,\theta} = \lhd_{2,\theta} = \lhd_\theta &= (< \cup \prec \cup \sqsubset_\theta)^+ \\
&= \{(\Pi l_1, \rightarrow l), (\rightarrow l, \Pi r), (\Pi r, \rightarrow r), (\Pi r, \Pi l_2)\}^+.
\end{aligned}$$

Reduction and skeletal orderings can be pictured as DAGs as in Figure 3 □

The presence of a relation in $\lhd_{\psi,\sigma}$ indicates that that relationship between specific inferences may not be altered by permutation. Consequently, the definition of $\prec_\psi$ fixes the relative positions of all rule applications in ψ *except* Πl rule applications (since they are removed from $\ll_\psi$ to form $\prec_\psi$).

DEFINITION 5.5 (i) (Compatibility) A derivation is said to be *compatible* with an instantiation just in case the reduction ordering induced is irreflexive.

(ii) (Degree) The *degree* of a compatible derivation is the number of pairs of inferences in the derivation whose skeletal order is inconsistent with the reduction ordering. That is, $\langle R, R' \rangle$ for which both $R \ll_\psi R'$ (R is below R' in ψ) and $R' \lhd_{\psi,\sigma} R$. If a derivation is compatible with an instantiation with degree n, we say it is *n-compatible.* □

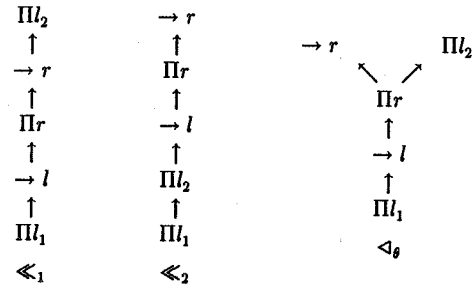

Figure 3: $\ll_i$ and $\lhd_\theta$ as DAGs for $\langle Di, \theta \rangle$, i=1,2.

EXAMPLE 5.6 From the DAGs in Figure 3 it is easy to see that (D1) is 0-compatible and (D2) 2-compatible with $\lhd_\theta$. In the latter case the Πl_2 inference is below two inferences which precede it in $\lhd_\theta$. □

THEOREM 5.7 (PERMUTATION THEOREM)

If ψ is compatible with σ, then there is a 0-compatible U-derivation ψ^* of the same endsequent. Moreover, if ψ is closed under σ, so is ψ^*. We say that ψ^* is a *permutation* of ψ.

PROOF. By induction on the degree of ψ. We interchange Πl inferences with other inferences to reduce the degree. One must check that the leaves of the permuted derivation contain the same declarations (though in a different order).

In order to simplify the presentation of the argument we distinguish two cases:

1. In which we consider the exchange of the Πl rule with the binary rule $\to l$;
2. In which we consider the exchange of the Πl rule with each of the unary rules $\to r$, Πr and Πl itself.

The first case divides into two subcases (we give the details of just one case). Suppose that R' is $\to l$ and R is Πl. Here we must assume that the Πl occurs on one of the branches of the proof, and then add an instance of Πl with the same indeterminate to the other branch immediately above the $\to l$. This addition introduces no new inconsistencies and preserves closure. If we did not introduce this copy of the inference, the reordering of these two rules would introduce many new inconsistencies, thereby destroying our inductive argument. Consequently, we may assume that, locally, the original derivation tree is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma D}{\Gamma \Rightarrow_\Sigma D} \qquad \dfrac{\Gamma, y{:}E, x{:}C(\alpha) \Rightarrow_\Sigma A}{\Gamma, y{:}E \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A},$$

where there is some $@ : D \to E \in \Sigma \cup \Gamma$ and some $@' : \Pi z : B . C(z) \in \Sigma \cup \Gamma$. (The substitution ordering, $\sqsubset_\sigma$, ensures that $@' \neq y$.) Locally, the reordered derivation is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma D \qquad \Gamma, x{:}C(\alpha), y{:}E \Rightarrow_\Sigma D}{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A}.$$

We must show that if each branch of the original derivation leads to a closed leaf under σ, then each branch of the reordered derivation leads to a closed leaf under σ. It is immediate that this holds for the left hand branch of the derivation, since the reordering leaves the bottom, left hand sequent unchanged. As for the right hand branch, we note that the components of the context of the bottom, right hand sequent are altered only in order. Therefore the right hand branch of the reordered derivation is closed under σ. Since $\sqsubset_\sigma$ is a subrelation of $\lhd_{\psi,\sigma}$ and because $\lhd_{\psi,\sigma}$ is irreflexive, y is not a subterm of $C(\alpha)$. Therefore the reordering of the context is consistent with the reordered derivation and the degree of the reordered derivation is one less than that of the original derivation.

The exchange of ΠI with the unary operators divides into five subcases (we give the details of the proof in just one case). Suppose that R' is ΠI and R is ΠI. Locally, the original derivation is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}C(\alpha), y{:}D(\beta) \Rightarrow_\Sigma A}{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A},$$

for suitable instances of ΠI. Locally, the reordered derivation, which has degree one less than that of the original derivation, is of the form:

$$\dfrac{\dfrac{\Gamma, y{:}D(\beta), x{:}C(\alpha) \Rightarrow_\Sigma A}{\Gamma, y{:}D(\beta) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A}.$$

Once again we note that the components of the bottom sequent have altered only in order, so that if the original derivation is closed under σ, then so is the reordered derivation. Since $\sqsubset_\sigma$ is a subrelation of $\lhd_{\psi,\sigma}$ and because $\lhd_{\psi,\sigma}$ is irreflexive, x is not a subterm of $D(\beta)$. Therefore the reordering of the context is consistent with the reordered derivation and the degree of the reordered derivation is one less than that of the original derivation.

This completes the proof. □

We have as a corollary to the construction performed in the proof of the Permutation Theorem:

COROLLARY 5.8 $\lhd_{\psi^*,\sigma} = \lhd_{\psi,\sigma}$.

PROOF. The only relationships changed in the permutation are those excluded from $\lhd_{\psi,\sigma}$. □

We have already verified this for our example derivations.

The following lemma shows that the 0-compatibility of a derivation and instantiation is a necessary condition for the well-typing of the latter in the former.

LEMMA 5.9 *If σ is well-typed in ψ, ψ is 0-compatible with σ.*

PROOF. An inconsistency between the skeletal ordering of ψ and the reduction ordering arises from an inconsistency between $\sqsubset_\sigma$ and the skeletal ordering. Since $\sqsubset_\sigma \subseteq \mathcal{F}_\psi(\Xi) \times \mathcal{F}_\psi(\Pi l)$, it must arise from a Πl inference, introducing α say, being nearer the root of the derivation than a Πl or Πr inference that gives rise to a variable, v say, free in $\alpha\sigma$. But then v cannot be declared in the typing context of α since it is introduced above α, and therefore σ is not well-typed, contradicting our hypothesis. Therefore there can be no inconsistencies and ψ is 0-compatible with σ. □

6 Incremental search

From a computational point of view testing for compatibility is a simple matter given a derivation and an instantiation: it is an acyclicity check in a directed graph. Compatibility is not, however, a *sufficient* test for well-typing. The Permutation Theorem gives us the existence of 0-compatible derivations in which we might test for well-typing of the instantiation incrementally (*i.e.*, as it is found) but this involves repeatedly constructing permutations using the constructive proof of the theorem. This is inelegant and computationally expensive.

Another alternative would be to ignore well-typing until a closed, compatible derivation and instantiation have been found, and then use the Permutation Theorem once and check well-typing. We reject this option on the grounds that typing constraints reduce the search space of the unification algorithm drastically.

We develop instead a computationally tractable test on a derivation and instantiation that, if passed, guarantees the well-typing of the instantiation in all 0-compatible permutations of the derivation. Our ability to define such a notion is a corollary of the strong normalization result for $\lambda\Pi$-calculus [11] with its attendant subformula property, just as the results obtained in [4] and [29] for other logics rely on metatheorems of this sort.

Henceforth we treat contexts as ordered structures or DAGs rather than sequences since the dependencies between declarations form such an order. Consequently the implicit union, denoted above by a comma, such as in "$\Gamma, x{:}A$", should be understood as an order preserving union of the order (DAG) Γ and the singleton order $x{:}A$. The latter will be higher in the resulting order than the declarations of the free variables in A, and incomparable with the other maximal elements of Γ. This assumption simplifies our discussion.

The following notions are introduced for a U-derivation ψ of a well-formed endsequent. Let $\mathrm{V}(\psi) = \mathrm{Ind}(\psi) \cup \mathrm{Eig}(\psi)$. We use u, v, w possibly subscripted to denote elements of $\mathrm{V}(\psi)$ and refer to them simply as "variables". Let $T(v)$ denote the typing expression for the variable v in ψ.

DEFINITION 6.1 (Intrinsic typing context) The *intrinsic typing context*, $I(v)$, for each variable $v \in \mathrm{V}(\psi)$, is defined inductively on the structure of the endsequent as follows:

$$I(v) \;=_{\mathrm{def}} \biguplus_{w \in \mathrm{FV}(T(v))} (I(w), w{:}T(w)),$$

where $\uplus$ denotes order-preserving union of orders. □

$I(v)$ is well-defined since the endsequent is well-formed. Indeed, we have:

LEMMA 6.2 $I(v) \vdash_\Sigma T(v)$: Type.

PROOF. By construction and the well-formedness of the endsequent. □

EXAMPLE 6.3 For our example derivations of Figures 1 and 2 we have $\mathrm{V}(D1) = \mathrm{V}(D2) = \{\alpha, y, \beta\}$. Moreover, since both derivations have the same endsequent we have for both:

$$\begin{array}{lll} T(\alpha) = A, & \mathrm{FV}(T(\alpha)) = \emptyset, & I(\alpha) = \emptyset; \\ T(y) = B(\alpha), & \mathrm{FV}(T(y)) = \{\alpha\}, & I(y) = \alpha{:}A; \\ T(\beta) = A, & \mathrm{FV}(T(\beta)) = \emptyset, & I(\beta) = \emptyset. \end{array}$$

Consequently,

$$\begin{array}{rcl} I(\alpha) \vdash_\Sigma T(\alpha)\text{: Type} & = & \vdash_\Sigma A\text{: Type}; \\ I(y) \vdash_\Sigma T(y)\text{: Type} & = & \alpha{:}A \vdash_\Sigma B(\alpha)\text{: Type}; \\ I(\beta) \vdash_\Sigma T(\beta)\text{: Type} & = & \vdash_\Sigma B(a)\text{: Type}. \end{array}$$

□

Let ψ be compatible with the ground instantiation σ. We give an inductive definition of the intrinsic typing context, $I_\sigma(v)$, and type, $T_\sigma(v)$, of a variable v of ψ under a compatible instantiation σ. The induction is on the (well-founded) reduction ordering $(\lhd_{\psi,\sigma})$ over the domain of σ.

DEFINITION 6.4 ($I_\sigma(v)$ and $T_\sigma(v)$) Base. For all $v \in \mathrm{V}(\psi)$, define $I_\epsilon(v) = I(v)$ and $T_\epsilon(v) = T(v)$. (ϵ is the empty instantiation.)

Step. Given $v \in \mathrm{V}(\psi)$, we assume that we have defined $I_\sigma(w)$ and $T_\sigma(w)$ for all $w \in \mathrm{V}(\psi)$ such that $w \lhd_{\psi,\sigma} v$ (Inductive Hypothesis). Define

$$D_0(v) = \begin{cases} I_\epsilon(v) \vdash_\Sigma T_\epsilon(v)\text{:Type}, & v \in \mathrm{Eig}(\psi); \\ I_\epsilon(v), \biguplus_{w \in \mathrm{FV}(v\sigma)} (I_\sigma(w), w{:}T_\sigma(w)) \vdash_\Sigma T_\epsilon(v)\text{:Type}, & v \in \mathrm{Ind}(\psi). \end{cases}$$

Now, let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be an enumeration of those indeterminates declared in the context of $D_0(v)$ such that if $\alpha_i \lhd_{\psi,\sigma} \alpha_j$ then $i < j$. By definition of $\lhd_{\psi,\sigma}$ and $I(v)$, we have $\alpha_i \lhd_{\psi,\sigma} v$ for $1 \leq i < n+1$. Define

$$D_{k+1}(v) \;=\; \mathrm{CUT}(I_\sigma(\alpha_k) \vdash_\Sigma \alpha_k\sigma{:}T_\sigma(\alpha_k) \;\;,\;\; D_k(v)), \qquad 0 \leq k < n.$$

If $D_n(v)$ is the assertion: $\Delta \vdash_\Sigma C$:Type, then define

$$\begin{aligned} I_\sigma(v) &=_{\text{def}} \Delta \\ T_\sigma(v) &=_{\text{def}} C. \quad \square \end{aligned}$$

The "CUT" operation in the above definition is the admissible rule of transitivity (see Theorem 2.1). That is, $D_{k+1}(v)$ is defined in terms of $D_k(v)$ by the following inference figure:

$$\frac{I_\sigma(\alpha_k) \vdash_\Sigma \alpha_k\sigma{:}T_\sigma(\alpha_k) \qquad D_k(v)}{D_{k+1}(v)} \text{ CUT.}$$

The cut rule is being used to effect substitution of the values (under σ) of indeterminates throughout the judgement starting from the "uninstantiated" intrinsic typing context and type. The definition is well-formed since the context of the left premiss of each cut is a subcontext of the right premiss. (This follows from the construction of $I(v)$.) The cut above then serves to eliminate the declaration $\alpha_k{:}T_\sigma(\alpha_k)$ from the context of $D_k(v)$, replacing α_k by $\alpha_k\sigma$ throughout the rest of the assertion.

The enumeration taken is irrelevant since independent cuts commute. Consider

$$\frac{I_\sigma(u_2) \vdash_\Sigma u_2\sigma{:}T_\sigma(u_2) \qquad \dfrac{I_\sigma(u_1) \vdash_\Sigma u_1\sigma{:}T_\sigma(u_1) \qquad D_k(v)}{D_{k+1}(v)}}{D_{k+2}(v)}$$

and

$$\frac{I_\sigma(u_1) \vdash_\Sigma u_1\sigma{:}T_\sigma(u_1) \qquad \dfrac{I_\sigma(u_2) \vdash_\Sigma u_2\sigma{:}T_\sigma(u_2) \qquad D'_k(v)}{D'_{k+1}(v)}}{D'_{k+2}(v)}.$$

In the first derivation $\alpha_k = u_1$ and $\alpha_{k+1} = u_2$. In the second, $\alpha_k = u_2$ and $\alpha_{k+1} = u_1$. If u_1 and u_2 are assumed independent (*i.e.*, unrelated via $\lhd_{\psi,\sigma}$), we have $u_i \notin \text{Dom}(I_\sigma(u_j))$, $i \neq j$. Hence substitution of the value $u_1\sigma$ for u_1 does not interfere with substitution of the value $u_2\sigma$ for u_2, and $D_{k+2} = D'_{k+2}$.

EXAMPLE 6.5 Recall, from our examples, that $\theta = \{\alpha \mapsto a, \beta \mapsto y\}$ and that $\alpha \lhd_\theta y \lhd_\theta \beta$. $\text{FV}(\alpha\theta) = \emptyset$, so

$$D_0(\alpha) \;=\; I_\epsilon(\alpha) \vdash_\Sigma T_\epsilon(\alpha)\text{: Type} = \vdash_\Sigma A\text{: Type}.$$

Thus, $I_\theta(\alpha) = \emptyset$ and $T_\theta(\alpha) = A$. For y we have

$$D_0(y) \;=\; I_\epsilon(y) \vdash_\Sigma T_\epsilon(y)\text{: Type} \;=\; \alpha{:}A \vdash_\Sigma B(\alpha)\text{: Type}.$$

$D_1(y)$ is defined by the inference figure:

$$\frac{I_\theta(\alpha) \vdash_\Sigma \alpha\theta{:}T_\theta(\alpha) \qquad D_0(y)}{D_1(y)} \;=\; \frac{\vdash_\Sigma a{:}A \qquad \alpha{:}A \vdash_\Sigma B(\alpha)\text{: Type}}{\vdash_\Sigma B(a)\text{: Type}}.$$

Consequently, $I_\theta(y) = \emptyset$ and $T_\theta(y) = B(a)$. $\mathrm{FV}(\beta\theta) = \{y\}$, so

$$\begin{aligned} D_0(\beta) &\doteq I_\epsilon(\beta) \vdash_\Sigma T_\epsilon(\beta)\text{: Type} \\ &= I_\theta(y), y{:}T_\theta(y) \vdash_\Sigma B(a)\text{: Type} \\ &= y{:}B(a) \vdash_\Sigma B(a)\text{: Type.} \end{aligned}$$

Consequently, $I_\theta(\beta) = y{:}B(a)$ and $T_\theta(\beta) = B(a)$. □

We can now state the desired well-typing condition for σ in ψ.

DEFINITION 6.6 (Intrinsic well-typing) σ is said to be *intrinsically well-typed* in ψ just in case for all indeterminates α of ψ, we have:

$$I_\sigma(\alpha) \vdash_\Sigma \alpha\sigma{:}T_\sigma(\alpha). \quad \square$$

The importance of the definition is summarized by:

PROPOSITION 6.7 *If σ is intrinsically well-typed in ψ, then it is intrinsically well-typed in all compatible permutations of ψ. In particular it is well-typed in 0-compatible permutations.*

PROOF. Reference to the definition will show that the intrinsic well-typing of σ in ψ does not depend on the Πl structure of ψ. (In fact we deliberately forbade such dependence by our definition of $\lhd_{\psi,\sigma}$.) Hence the conditions are unaffected by permutations allowed by the reduction ordering $\lhd_{\psi,\sigma}$, which is itself unaltered by permutation (Corollary 5.8). For a 0-compatible permutation ψ^* of ψ, the intrinsic typing context for a variable is a subcontext of the typing context in $\psi^*\sigma$. Since "Thinning" is admissible (Theorem 2.1), σ is well-typed in ψ^*. □

In a similar vein, we state without proof the following:

PROPOSITION 6.8 *If σ is well-typed in a 0-compatible derivation ψ, it is intrinsically well-typed in ψ.* □

EXAMPLE 6.9 Following through with our example we note that

$$\begin{aligned} I_\theta(\alpha) \vdash_\Sigma \alpha\theta{:}T_\theta(\alpha) &= \vdash_\Sigma a{:}A \\ I_\theta(\beta) \vdash_\Sigma \beta\theta{:}T_\theta(\beta) &= y{:}B(a) \vdash_\Sigma y{:}B(a). \end{aligned}$$

Hence θ is intrinsically well-typed in (D1) and, most importantly, also in (D2). □

We can now (re)define a computationally acceptable notion of U-proof that takes advantage of the Permutation Theorem.

DEFINITION 6.10 (U-proof) A U-*proof* is a pair $\langle\psi, \sigma\rangle$, where ψ is a U-derivation and σ an instantiation with domain $\mathrm{Ind}(\psi)$, such that (1) ψ is closed under σ; (2) ψ is compatible with σ, and (3) σ is intrinsically well-typed in ψ. □

This definition should be compared with Definition 4.7.

THEOREM 6.11 *For well-formed sequents* $\Gamma \Rightarrow_\Sigma A$,

$$\mathbf{U} \text{ proves } \Gamma \Rightarrow_\Sigma A \quad \text{iff} \quad \mathbf{L} \text{ proves } \Gamma \Rightarrow_\Sigma A.$$

PROOF. (Only if.) Suppose $\langle \psi, \sigma \rangle$ is a U-proof of $\Gamma \Rightarrow_\Sigma A$. The Permutability Theorem gives us a permutation ψ^* of ψ, closed under (hypothesis 1), and compatible with (hypothesis 2), the instantiation σ. Hypothesis (3), via Proposition 6.7, ensures that σ is well-typed in ψ^*. Hence $\psi^*\sigma$ is an L-proof.

(If.) Let $\langle \psi, \sigma \rangle$ be such that $\psi\sigma$ is an L-proof of $\Gamma \Rightarrow_\Sigma A$. By definition ψ is closed under σ and σ is well-typed in ψ. Compatibility follows from Lemma 5.9, and intrinsic well-typing from Proposition 6.8. □

7 Conclusions

Instantiations can be generated by a unification algorithm acting on putative axiom sequents. They can first be checked for compatibility (occurs-check) and then for intrinsic well-typing. The incremental nature of intrinsic typing means that the typing information can be used to constrain the unification algorithm. Newly calculated values for previously uninstantiated indeterminates may be used to eliminate those indeterminates from the typing contexts of the remaining ones. No permutations need be calculated. Note however that if we permit unification to calculate non-ground instantiations then we must modify the constructions of Section 6 to permit the inclusion of further proof-search.

We have been somewhat cautious in this development and allowed only Πl rules to migrate. As a consequence the basic structure of a derivation is largely fixed. The next step is to remove the ordering constraints induced by the propositional structure of the logic, perhaps using unification here as was done in [29] for first-order intuitionistic logic. The final result would be a *matrix method* in the style of Bibel [4] or Andrews [1].

Closure instantiations are calculated by unification. In general, the instantiations calculated by the unification algorithm introduce new variables (*i.e.*, variables that are not present in the original context) to the derivation[9]. However, we require only those substitutions which are well-typed instantiations (under some reordering), and so for a a given U-derivation ψ we accept (for further analysis) just those substitutions σ which do not introduce new variables. The unification algorithm of [24] is *sound* and *complete* for the calculation of such instantiations, subject to further proof-search.

We have considered ground endsequents and ground instantiations explicitly. The extension of U to non-ground endsequents is straightforward but involved. A non-ground sequent together with a *typing constraint* $\mathcal{T}$ for its indeterminates is considered to stand for the set of its well-formed ground instances. (The typing constraint consists of intrinsic typing contexts and types for each indeterminate occuring in the endsequent, ensuring that the mutual dependencies do not render any extension of the initial reduction ordering cyclic.) This determines a set $\mathcal{S}(\mathcal{T})$ of ground *answer instantiations*. Any non-ground instantiation calculated from a U-proof $\langle \psi, \sigma \rangle$ of the sequent determines a set of ground well-typed *extensions* $\mathcal{S}_\psi(\mathcal{T})$.

[9]Indeed, this is true of the basic algorithm for the simply-typed λ-calculus of [14]

We consider non-ground endsequents because they have an interesting logic programming interpretation. A sequent $\Gamma \Rightarrow_\Sigma A$ may be interpreted as a *logic program* in the following sense: Σ determines a language, Γ a list of *program clauses* and A a *query* written in the language Σ. Indeterminates correspond to *program variables* or *logic variables*; these correspond to the logical variables of the programming language Prolog [5]. The whole sequent represents a request to compute an instantiation σ for the indeterminates of the sequent such that σ renders the sequent $\Gamma\sigma \Rightarrow_\Sigma A\sigma$ L-provable. We are exploiting the fact that the underlying lambda calculus of the $\lambda\Pi$-calculus encodes a computation of type A; *i.e.*, a term M for which $\Gamma \vdash_\Sigma M{:}A$ is provable. A full discussion of this notion of logic programming, including both operational (as presented here) and model-theoretic semantics, may be found in [24] and in a forthcoming paper by the authors, where we also discuss the application of our techniques to a form of *resolution rule* which generalizes Paulson's higher-order resolution [21] to the $\lambda\Pi$-calculus. A notion of logic programming for the LF which is similar to the λProlog language of Miller and Nadathur [20] has been implemented in the Elf system by Pfenning [22].

Acknowledgements. The authors are grateful to Alan Bundy, Robert Harper, Furio Honsell, Gérard Huet, Gordon Plotkin, Randy Pollack, Anne Salvesen and the referees for helpful suggestions and comments.

References

[1] Andrews, P.B. Theorem-proving via general matings. *J. Assoc. Comp. Mach.* **28**(2) pp. 193–214, 1981.

[2] Avron, A., Honsell, F., Mason, I. Using Typed Lambda Calculus to Implement Formal Systems on a Machine. University of Edinburgh report ECS-LFCS-87-31, 1987.

[3] Barendregt, H. Introduction to Generalized Type Systems. *Proc. 3rd Italian Conference on Theoretical Computer Science.* World Scientific Publishing Co., Singapore, 1989.

[4] Bibel, W. Computationally Improved Versions of Herbrand's Theorem. In J. Stern, editor, Proc. of the Herbrand Symposium, *Logic Colloquium* '81, pp. 11-28, North-Holland, 1982.

[5] Clocksin, W.F., Mellish, C.S. Programming in Prolog, Springer-Verlag, 1984.

[6] Curry, H.B. The permutability of rules in the classical inferential calculus. *J. Symbolic Logic* **17**, pp. 245–248, 1952.

[7] van Daalen, D.T., The language theory of AUTOMATH. Ph.Dthesis, Technical University of Eindhoven, The Netherlands, 1980.

[8] Elliott, C. Higher-order Unification with Dependent Function Types. *Proc. 3rd International Conference on Rewriting Techniques and Applications*, Chapel Hill NC, 1989. Springer-Verlag Lecture Notes in Computer Science 355, N. Dershowitz Ed., pp. 121–136.

[9] Elliott, C. Extensions and Applications of Higher-order Unification. Ph.D thesis, Carnegie-Mellon University, 1990. Available as report CMU-CS-90-134.

[10] Gentzen, G. Untersuchungen über das logische Schliessen, *Mathematische Zeitschrift* **39** (1934) pp. 176–210, 405-431.

[11] Harper, R., Honsell, F., Plotkin, G. A Framework for Defining Logics. *Proc. Second Annual Symposium on Logic in Computer Science,* pp. 194–204. IEEE, 1987.

[12] Harper, R., Honsell, F., Plotkin, G. A Framework for Defining Logics. To appear in *J. Assoc. Comp. Mach.*.

[13] Howard, W.A. The formulae-as-types notion of construction. In: To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism (editors J.P. Seldin & J.R. Hindley). Academic Press, 1980.

[14] Huet, G. A Unification Algorithm for Typed λ-calculus. *Theor. Comp. Sci.* **1**(1975) pp. 27–57.

[15] Kleene, S.C. Permutability of inferences in Gentzen's calculi *LK* and *LJ. Memoirs of the American Mathematical Society* **10**, pp. 1–26, 1952.

[16] Kleene, S.C. Mathematical logic. Wiley and Sons, 1968.

[17] Martin-Löf, P. On the meanings of the logical constants and the justifications of the logical laws. Technical Report 2, Scuola di Specializziazione in Logica Matematica, Dipartimento di Matematica, Università di Siena, 1985.

[18] Meyer, A., Reinhold, M. 'Type' is not a type; preliminary report. *Proc. 13th ACM Symposium on the Principles of Programming Languages*, 1986.

[19] Miller, D. Proofs in higher-order logic. Ph.D thesis, Carnegie-Mellon University, Pittsburgh, USA, 1983. Available as report MS-CIS-83-37, University of Pennsylvania, 1983.

[20] Miller, D., Nadathur, G. Higher-order Logic Programming. Report MS-CIS-86-17, University of Pennsylvania, 1986.

[21] Paulson, L. Natural Deduction Proof as Higher-order Resolution. *J. Logic Programming* **3**, pp. 237–258, 1986.

[22] Pfenning, F. Elf: a language for logic definition and verifed metaprogramming. *Proc. Fourth Annual Symposium on Logic in Computer Science,* pp. 98-105. IEEE, 1989.

[23] Prawitz, D. Natural Deduction: A Proof-theoretical Study. Almqvist & Wiksell, Stockholm, 1965.

[24] Pym, D.J. Proofs, Search and Computation in General Logic. Ph.D thesis. University of Edinburgh, 1990. Available as report CST-69-90, Department of Computer Science, University of Edinburgh, 1990. (Also published as LFCS report ECS-LFCS-90-125.)

[25] Pym, D.J., Wallen, L.A. Investigations into Proof-search in a System of First-order Dependent Function Types. *Proc. 10th International Conference on Automated Deduction.* Kaiserslautern, FRG, July 1990. Lecture Notes in Artificial Intelligence 449, pp. 236–250, Springer-Verlag, 1990.

[26] Robinson, J. A machine-oriented logic based on the resolution principle. *J. Assoc. Comp. Mach.* **12**, pp. 23–41, 1965.

[27] Salvesen, A. Personal Communication. University of Edinburgh, 1990.

[28] Smullyan, R.M. First-order logic. *Ergebnisse der Mathematik*, Volume **43**, Springer Verlag, 1968.

[29] Wallen, L.A. Automated deduction in non-classical logics. MIT Press, 1989.