On the Herbrand content of LK

Bahareh Afshari Stefan Hetzl Graham E. Leigh
TU Wien, Austria
{bahareh.afshari,stefan.hetzl,graham.leigh}@tuwien.ac.at

We present a structural representation of the Herbrand content of LK-proofs with cuts of complexity prenex Π_2/Σ_2. The representation takes the form of a typed non-deterministic tree grammar $\mathscr{G}$ of order 2 which generates a finite language, $L(\mathscr{G})$, of first-order terms that appear in the Herbrand expansions obtained through cut-elimination. In particular, for every Gentzen-style reduction $\pi \rightsquigarrow \pi'$ between LK-proofs we study the induced grammars, respectively $\mathscr{G}$ and $\mathscr{G}'$, and classify the cases in which language equality, $L(\mathscr{G}) = L(\mathscr{G}')$, and language inclusion, $L(\mathscr{G}) \supseteq L(\mathscr{G}')$, hold.

1 Introduction

In classical first-order logic a proof can be considered as being composed of two layers: on the one hand the terms by which quantifiers are instantiated, and on the other hand, the propositional structure. This separation is most clearly illustrated by Herbrand's theorem [9, 3]: a formula is valid if and only if there is a finite expansion (of existential quantifiers to disjunctions and universal quantifiers to conjunctions of instances) which is a propositional tautology. Such Herbrand expansions can be transformed to and obtained from cut-free sequent calculus proofs in a quite straightforward way.

It is non-trivial to formally extend this separation to proofs with cuts. An approach which has been successful in this respect is the use of tree grammars, introduced in [10] for proofs with Π_1-cuts and extended to Π_2-cuts in [1, 2]. In this setting, a proof in sequent calculus induces a tree grammar which bears all instances of the end-sequent as well as the instantiation structure of the cuts without direct reference to the cut formulæ themselves: one obtains a Herbrand expansion by computing the language of the grammar.

In addition to the proof-theoretic interest behind an abstract representation of proofs with cut, proof grammars provide a number of applications. Motivated by the aim to structure and compress automatically generated proofs, an algorithm for cut-introduction based on proof grammars has been developed in [13, 12]. This method has been implemented and empirically evaluated with good results in [11]. An extension of these techniques to the case of proofs with Π_1-induction has led to a new technique for inductive theorem proving [6] which is currently being implemented. A final application of proof grammars is in the area of proof complexity, where lower bounds on the length of proofs with cuts (which are notoriously difficult to control) are obtained by transferring lower bounds on the size of the corresponding grammar [5, 4].

There are other formalisms which allow Herbrand expansions to be computed in a way that abstracts from the propositional structure. The historically first such formalism is Hilbert's ε-calculus [15]. In [7] Gerhardy and Kohlenbach adapt Shoenfield's variant of Gödel's Dialectica interpretation to a system of pure predicate logic. Recent work, related to proof nets, is that of Heijltjes [8] and McKinley [16], and a similar approach, in the formalism of expansion trees [17], can be found in [14]. What sets proof grammars apart from these formalisms is that they not only compute Herbrand expansions but provide a (well-understood) abstract description of its structure which is crucial for the applications mentioned above.

U. Kohlenbach, S. van Bakel, S. Berardi (Eds.): CL&C'16
EPTCS 213, 2016, pp. 1–10, doi:10.4204/EPTCS.213.1

In the present paper we provide an intermediate formalism between proof grammars and functional interpretations with the aim of studying the relationship between the two approaches. This intermediate formalism is presented as a grammar but instead of capturing the instantiation structure directly, it is given by a brief line-by-line definition on the proof, as functional interpretations usually are. The necessity for computing more than one witness (reflected by the case distinction constants in the Gerhardy–Kohlenbach version of the Dialectica interpretation [7]) is reflected by non-deterministic production rules in the grammar.

The main result we prove in this paper is stated below. Note that in the presence of Skolemisation it suffices to consider proofs with Σ_1 end sequents.

Theorem 1.1. *Let π be a proof of $\exists \boldsymbol{v} F$ with F quantifier-free in which cut-formulæ are prenex Π_2 or Σ_2. There exists an acyclic context-free grammar $\mathscr{G}$ such that $\bigvee_{\boldsymbol{t} \in L(\mathscr{G})} F(\boldsymbol{t})$ is valid. Moreover, $L(\mathscr{G})$ contains the Herbrand set extracted from any cut-free proof that can be obtained from π via a sequence of cut reductions (see Figure 2) that always reduces to the weak (quantifier) side of a cut before the strong side.*

More generally, $L(\mathscr{G})$ covers the Herbrand set of any cut-free proof obtained from π by a sequence of reductions fulfilling the following two restrictions.

1. *A contraction on a universally quantified Π_2 formula is reduced only when no other reduction rule is applicable (to this cut);*
2. *If two cuts are permuted in the form*

$$\dfrac{\dfrac{A,B,\Gamma \quad \bar{A},\Delta}{B,\Gamma,\Delta}\,\mathit{cut} \qquad \bar{B},\Lambda}{\Gamma,\Delta,\Lambda}\,\mathit{cut} \quad \rightsquigarrow \quad \dfrac{\dfrac{A,B,\Gamma \quad \bar{B},\Lambda}{A,\Gamma,\Lambda}\,\mathit{cut} \qquad \bar{A},\Delta}{\Gamma,\Delta,\Lambda}\,\mathit{cut}$$

then one of A and B is not Π_2.

2 The system LK

Traditionally LK is represented in two-sided sequent calculus. For notational simplicity however, we work in one-sided sequent calculus (Tait-style) with explicit weakening (w), contraction (c) and permutation (p) rules. Axioms and rules are laid out in Figure 1 and the cut reduction steps are presented in Figure 2. We generally leave applications of the permutation rule implicit: its only role is to facilitate defining the grammar in the next section.

Axioms:	$A,\bar{A}$ for A an atomic formula		
Inference rules:	$\vee\dfrac{A,B,\Gamma}{A\vee B,\Gamma}$	$\wedge\dfrac{A,\Gamma \quad B,\Delta}{A\wedge B,\Gamma,\Delta}$	
	$\forall\dfrac{A(v/\alpha),\Gamma}{\forall v A,\Gamma}$	$\exists\dfrac{A(v/t),\Gamma}{\exists v A,\Gamma}$	$\mathsf{cut}\dfrac{A,\Gamma \quad \bar{A},\Delta}{\Gamma,\Delta}$
	$\mathsf{w}\dfrac{\Gamma}{A,\Gamma}$	$\mathsf{c}\dfrac{A,A,\Gamma}{A,\Gamma}$	$\mathsf{p}\dfrac{\Gamma,B,A,\Delta}{\Gamma,A,B,\Delta}$

Figure 1: Axioms and rules of LK. The usual eigenvariable conditions for quantifier introduction apply.

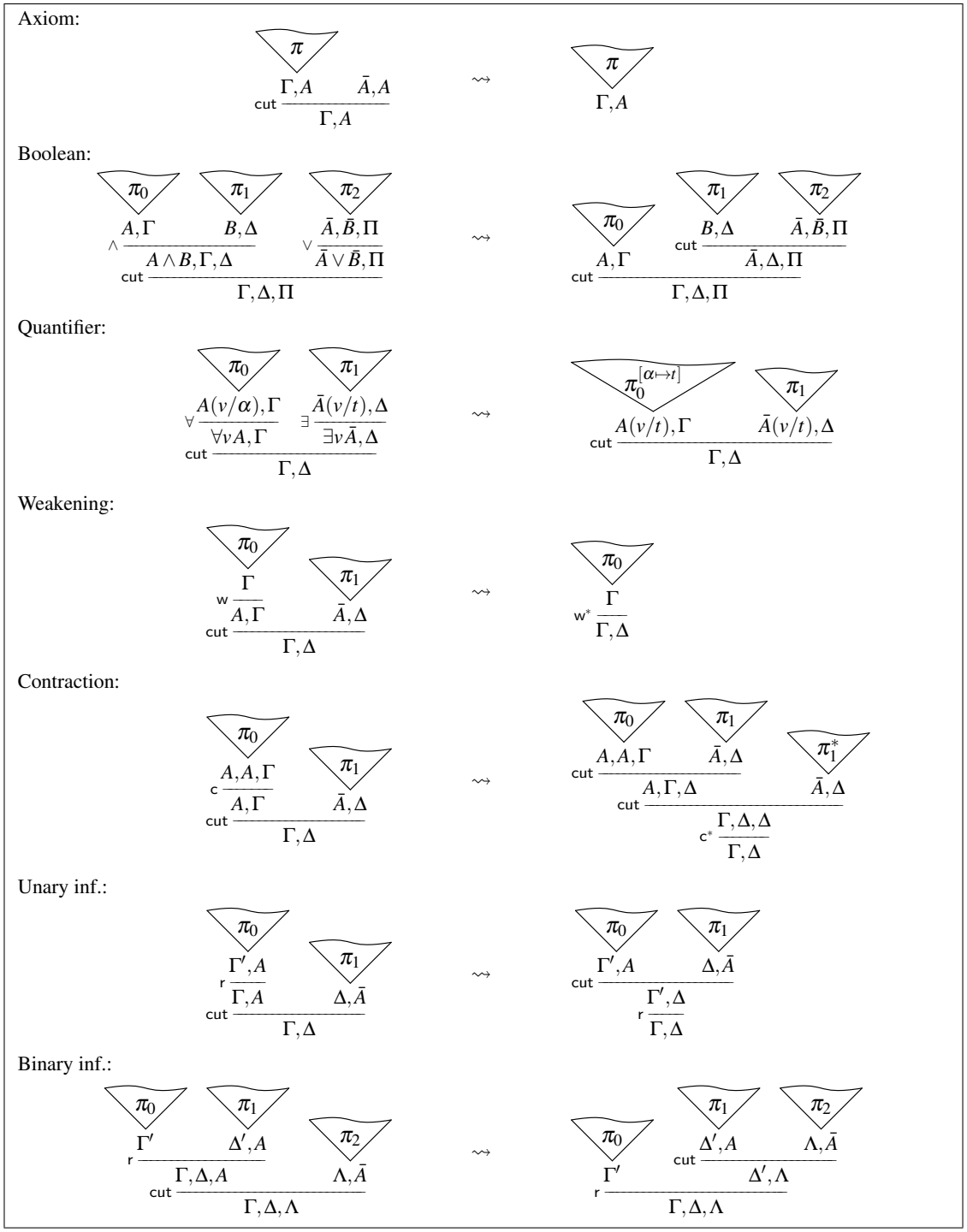

Figure 2: One-step cut reduction and permutation rules. In the final two reductions, r denotes respectively an arbitrary unary and binary rule.

We use α, β, etc. for free and v, w, etc. for bound variables. Upper-case Roman letters, A, B, etc. denote formulæ and upper-case Greek letters Γ, Δ, etc. range over *sequents*, namely finite sequences of formulæ. We write $\bar{A}$ to denote the dual of the formula A obtained by de Morgan laws. A *proof* is a finite binary tree labelled by sequents obtained from the axioms and rules of the calculus with the restriction that cuts apply to prenex Π_2/Σ_2 formulæ only. Without loss of generality, we assume all proofs are *regular*, namely strong quantifier inferences are associated unique eigenvariables. This is particularly relevant in the case of contraction reduction where a sub-proof is duplicated and eigenvariables renamed (expressed by annotating the proof in question by an asterisk) to maintain regularity. The *length* of a sequent Γ is denoted $|\Gamma|$ and we write $\pi \vdash \Gamma$ to express that π is a proof with end sequent Γ.

3 Proof grammars

To an LK-proof $\pi \vdash \Gamma$ we associate a typed non-deterministic tree grammar $\mathscr{G}_\pi$ (equivalently, an order-2 recursion scheme) with production rules that abstract the computation of Herbrand sets achieved through Gentzen-style cut-elimination.

Informally, $\mathscr{G}_\pi$ comprises rewrite rules for symbols $\sigma^i_{\pi'}$ where $\pi' \vdash \Gamma'$ is a sub-proof of π and $0 \leq i < |\Gamma'|$. The non-terminal $\sigma^i_{\pi'}$ is of function type (of order ≤ 2) with arity $|\Gamma'|$ returning a sequence of closed terms as witnesses for the weak quantifiers in the i-th formula in Γ'. The j-th argument of $\sigma^i_{\pi'}$ is interpreted as input for the strong quantifiers in the j-th formula in Γ' and is either a finite sequence (of determined length) or a function from first-order objects to sequences thereof, the case depending on the quantifier complexity of the corresponding formula. The type of the arguments to $\sigma^i_{\pi'}$ is independent of i.

As an example, consider a derivation $\pi \vdash A_0, A_1$ where A_0 and A_1 are prenex $\Sigma_2 \cup \Pi_2$ formulæ with m_0 and m_1 existential quantifiers respectively. The grammar contains two non-terminals associated to π, σ^0_π and σ^1_π, of type

$$\sigma^0_\pi : \tau_0 \to \tau_1 \to (\underbrace{o \times \cdots \times o}_{m_0}) \qquad\qquad \sigma^1_\pi : \tau_0 \to \tau_1 \to (\underbrace{o \times \cdots \times o}_{m_1})$$

where o denotes the type of first-order terms and τ_i is a type depending on the number of universal quantifiers in A_i. Given terms $T_0 : \tau_0$ and $T_1 : \tau_1$ the grammar rewrites the term $\sigma^i_\pi T_0 T_1$ to a sequence of first-order terms $\langle t_1, \ldots, t_{m_i} \rangle$ of length m_i to be interpreted as witnesses to the extensional quantifiers in A_i. The role of T_0 and T_1 is to provide input for the strong quantifiers (specifically, their corresponding eigenvariables) on which witnesses to the existential quantifiers may depend. For instance, if $A_i = \forall v \exists w_0 \exists w_1 B_i$ and B_i is quantifier-free for each i, then $\tau_0 = \tau_1 = o$, $m_0 = m_1 = 2$ and $\sigma^i_\pi T_0 T_1$ rewrites to pairs of the form $\langle r, s \rangle [\alpha \mapsto T_0][\beta \mapsto T_1]$ where r and s are first-order terms and α and β are the eigenvariables for the strong quantifier in A_0 and A_1 respectively. Higher-type terms arise in the case of sequents with Σ_2 formulæ. Suppose $A_0 = \exists v \forall w B_0$ where B_0 is quantifier-free and A_1 is as above. In this case T_0 has type $\tau_0 = (o \to o)$ and is utilised in generating input for the universal quantifier in A_0 modulo witnesses for the existential quantifier. If the final inference in π derives the sequent A_0, A_1 from $\pi_0 \vdash \forall w B_0(v/r), A_1$ then since B_0 is quantifier-free, the two non-terminals associated to π_0 are

$$\sigma^0_{\pi_0} : o \to o \to o \qquad\qquad \sigma^1_{\pi_0} : o \to o \to (o \times o).$$

The production rules corresponding to this inference are

$$\sigma^0_\pi T_0 T_1 \to \langle r \rangle \qquad\qquad \sigma^1_\pi T_0 T_1 \to \sigma^1_{\pi_0}(T_0 r) T_1.$$

The left-hand rule returns the term r as the (single) witness to the existential quantifier in A_0 whereas the right-hand rule records the fact that in a witness to the existential quantifier in A_1, any occurrence of the eigenvariable for the universal in A_0 will be substituted for $T_0 r$. In general, T_0 will itself contain non-terminals for other parts of the (wider) proof and will have been introduced through the nesting of non-terminals that occurs when passing through a cut rule (see Figure 3).

In the following, fix an LK-proof $\pi \vdash \Gamma$ in which all formulæ are prenex Π_2/Σ_2.

3.1 Terms and types

We expand first-order terms by a form of explicit substitution, resulting in *structured (first-order) terms*: every first-order term is a structured term, and if s and t are structured terms and α is a free-variable symbol then the expression $s[\alpha \mapsto t]$ is a structured term.

Let o denote the type of structured first-order terms and ε the unit type with a single element $\langle\rangle : \varepsilon$. o and ε are called *ground-types* and their elements *ground-terms*. We consider the explicit substitution constructors above as term building operations on both o and ε. A type hierarchy is formed by closing the ground-types under the usual pair-types and function-types: given $u, x : \rho$ and $u' : \rho'$ we have $u \star u' : \rho \times \rho'$ and $\lambda x u' : \rho \to \rho'$. To avoid unnecessary parenthesis the three binary (infix) operations are assumed to associate to the right. We define $o^0 = \varepsilon$ and $o^{k+1} = o \times o^k$. An element of a type of the form o^k is called a *sequence-term*. Given a (possibly empty) sequence $(u_i : o)_{i<k}$ of terms of type o, we write $\langle u_0, \ldots, u_{k-1}\rangle$ to abbreviate the sequence-term $u_0 \star \cdots \star u_{k-1} \star \langle\rangle$ of type o^k.

Rule of inference	Corresponding production rule(s)
$\pi \vdash \bar{A}, A$	$\sigma^i_\pi z_0 z_1 \to z_{1-i}$
$\forall \dfrac{\pi_0 \vdash A(v/\alpha), \Gamma}{\pi \vdash \forall v A, \Gamma}$	$\sigma^i_\pi (z_0 \star z_1)\boldsymbol{y} \to (\sigma^i_{\pi_0} z_1 \boldsymbol{y})[\alpha \mapsto z_0]$
$\exists \dfrac{\pi_0 \vdash A(v/r), \Gamma}{\pi \vdash \exists v A, \Gamma}$	$\sigma^i_\pi z\boldsymbol{y} \to \begin{cases} r \star (\sigma^0_{\pi_0}(z \cdot r)\boldsymbol{y}), & \text{if } i = 0, \\ \sigma^i_{\pi_0}(z \cdot r)\boldsymbol{y}, & \text{otherwise.} \end{cases}$
$\mathsf{cut} \dfrac{\pi_0 \vdash A, \Gamma \quad \pi_1 \vdash \bar{A}, \Delta}{\pi \vdash \Gamma, \Delta}$	$\sigma^i_\pi \boldsymbol{x}\boldsymbol{y} \to \begin{cases} \sigma^{i+1}_{\pi_0}((\sigma^0_{\pi_1} \circ_{\bar{A}} \sigma^0_{\pi_0})\boldsymbol{y}\boldsymbol{x})\boldsymbol{x}, & i < \lvert\Gamma\rvert, \\ \sigma^{i'+1}_{\pi_1}((\sigma^0_{\pi_0} \circ_A \sigma^0_{\pi_1})\boldsymbol{x}\boldsymbol{y})\boldsymbol{y}, & i \geq \lvert\Gamma\rvert,\ i' = i - \lvert\Gamma\rvert. \end{cases}$
$\mathsf{c} \dfrac{\pi_0 \vdash A, A, \Gamma}{\pi \vdash A, \Gamma}$	$\sigma^i_\pi z\boldsymbol{y} \to \begin{cases} \sigma^0_{\pi_0} zz\boldsymbol{y} \mid \sigma^1_{\pi_0} zz\boldsymbol{y}, & \text{if } i = 0, \\ \sigma^{i+1}_{\pi_0} zz\boldsymbol{y}, & 0 < i < \lvert\Gamma\rvert. \end{cases}$
$\mathsf{w} \dfrac{\pi_0 \vdash \Gamma}{\pi \vdash A, \Gamma}$	$\sigma^i_\pi z\boldsymbol{y} \to \begin{cases} \mathsf{c}_A, & \text{for } i = 0, \\ \sigma^{i-1}_{\pi_0}\boldsymbol{y}, & \text{otherwise.} \end{cases}$
$\mathsf{p} \dfrac{\pi_0 \vdash \Gamma, B, A, \Delta}{\pi \vdash \Gamma, A, B, \Delta}$	$\sigma^i_\pi \boldsymbol{x} z_0 z_1 \boldsymbol{y} \to \begin{cases} \sigma^{i+1}_{\pi_0}\boldsymbol{x} z_1 z_0 \boldsymbol{y}, & \text{if } i = \lvert\Gamma\rvert, \\ \sigma^{i-1}_{\pi_0}\boldsymbol{x} z_1 z_0 \boldsymbol{y}, & \text{if } i = \lvert\Gamma\rvert + 1, \\ \sigma^{i}_{\pi_0}\boldsymbol{x} z_1 z_0 \boldsymbol{y}, & \text{otherwise.} \end{cases}$

Figure 3: Production rules: $\boldsymbol{x}$ and $\boldsymbol{y}$ denote sequences of distinct variables of length $\lvert\Gamma\rvert$ and $\lvert\Delta\rvert$ respectively; the contraction rule is the only inference introducing non-determinism; c_k, $\circ_F$ and $z \cdot r$ are abbreviations for terms described in Section 3.2.

Let $\Gamma = \{A_0, \ldots, A_n\}$. The type of σ^i_π is given by

$$\sigma^i_\pi : \tau^*_{A_0} \to \cdots \to \tau^*_{A_n} \to \tau_{A_i}$$

where τ_F and τ^*_F are determined by the complexity of F:

- for $F = \forall v_1 \cdots \forall v_m \exists w_1 \cdots \exists w_n G$ with G quantifier-free,

$$\tau_F = o^n \qquad\qquad \tau^*_F = o^m;$$

- for $F = \exists v_1 \cdots \exists v_m \forall w_1 \cdots \forall w_n G$ with $n > 0$ and G quantifier-free,

$$\tau_F = o^m \qquad\qquad \tau^*_F = \underbrace{o \to \cdots \to o}_{m} \to o^n.$$

The *order* of a type ρ (and term of type ρ), $ord(\rho)$, is defined as usual: $ord(\varepsilon) = ord(o) = 0$, $ord(\rho \times \rho') = \max\{ord(\rho), ord(\rho')\}$ and $ord(\rho \to \rho') = \max\{ord(\rho)+1, ord(\rho')\}$. Thus for a proof $\pi \vdash \Gamma$ where Γ is a set of prenex Π_2 and Σ_2 formulæ, and for $i < |\Gamma|$, the order of σ^i_π is no greater than 2.

In the sequel we avoid explicit mention of types when they can be inferred from context.

3.2 Production rules

Let Σ be a finite set of (typed) variable symbols. A *structured λ-term over* Σ is a well-typed term constructed from ground-terms, variables and non-terminals via the term-forming operations described above in which any freely occurring variable is an element of Σ.

The production rules for non-terminals are determined by the final rule applied to the index proof and are presented in Figure 3. Each production rule has the form $u \to u'$ where the two terms are of the same ground-type and u' is a structured λ-term over the free variables in u. With the exception of the $\forall$ production rules (which are discussed below) u has the form $\sigma^i_\pi x_0 \cdots x_n$ which is often condensed to $\sigma^i_\pi \boldsymbol{x}$. Note that the contraction rule is the only inference rule that introduces non-determinism.

The presentation of the production rules includes the following abbreviations. The symbol c_A appearing in the production rule for weakening (w) denotes the sequence-term $\langle \mathsf{c}, \ldots, \mathsf{c} \rangle : o^k$ where $\mathsf{c} : o$ is some fixed constant symbol and k is the number of existential quantifiers in A.

The binary operation $\cdot$ appearing in the rule for existential quantifiers ($\exists$) extends term application to cases in which the first argument has type ε in a trivial way:

$$z \cdot x = \begin{cases} z, & \text{if } z : \varepsilon, \\ zx, & \text{otherwise.} \end{cases}$$

Its role is to compensate for the case that A is a Σ_1 formula whereby $\tau^*_{\exists vA} = \tau^*_A = \varepsilon$; otherwise $\tau^*_{\exists vA} = o \to \tau^*_A$ and so $z \cdot r : \tau^*_A$ as required.

In the production rules for cut (cut), the operation $\circ_F$ abbreviates combining two non-terminals depending on the quantifier complexity of F. Let $\pi_0 \vdash \Gamma$ and $\pi_1 \vdash \Delta$ be LK-proofs, and $i < |\Gamma|$, $j < |\Delta|$. For non-terminals $\sigma^0_{\pi_0}$ and $\sigma^0_{\pi_1}$ and variable sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ of length $|\Gamma| - 1$ and $|\Delta| - 1$ respectively, we define

$$(\sigma^0_{\pi_0} \circ_F \sigma^0_{\pi_1})\boldsymbol{x}\boldsymbol{y} = \begin{cases} \langle\rangle, & \text{if } F \text{ is quantifier-free} \\ \lambda z_0 \cdots \lambda z_m.\, \sigma^0_{\pi_0} \langle z_0, \ldots, z_m \rangle \boldsymbol{x}, & \text{if } F = \forall v_0 \cdots \forall v_m G \text{ and } G \in \Sigma_1, \\ \sigma^0_{\pi_0}(\lambda z_0 \cdots \lambda z_m.\, \sigma^0_{\pi_1} \langle z_0, \ldots, z_m \rangle \boldsymbol{y}) \boldsymbol{x}, & \text{if } F = \exists v_0 \cdots \exists v_m G \text{ and } G \in \Pi_1. \end{cases}$$

Observe that according to the typing introduced earlier, the production rules listed in Figure 3 are well-typed (by definition, the axiom case applies only when A is quantifier-free whence all relevant types are identical).

Before we proceed with the definition of language it is important to address the universal introduction rule $\forall$. In its stated form, $\mathscr{G}_\pi$ is context-sensitive as the reduction depends on the form of at least one argument. A context-free grammar can be obtained by formulating the production rules using projection functions for pair-types:

$$\sigma^n_\pi z\boldsymbol{y} \to (\sigma^n_{\pi_0}(p_1 z)\boldsymbol{y})[\alpha \mapsto p_0 z]. \tag{1}$$

Doing so, however, will in general expand the language of the grammar: if T is a term that non-deterministically rewrites to $r_0 \star T_0$ and $r_1 \star T_1$ (and the four terms are pairwise distinct) then substituting T for z in (1) yields four combinations of terms, compared with just two available from the $\forall$ production rule. Nevertheless, this increase will be always finite.

3.3 Language

A *derivation* in the grammar is a sequence of structured λ-terms containing non-terminals obtained by applying a sequence of grammar production rules and β-reductions. For a well-typed term u (possibly containing non-terminals), $L(u)$ denotes the set of terms derivable from u to which no further rules may be applied. We write $u \sim u'$ if $L(u) = L(u')$.

Given a sequence-term $T = \langle t_1, \ldots, t_k \rangle : o^k$ of structured first-order terms, let T^* denote the result of evaluating all explicit substitutions occurring in T, forming a sequence of first-order terms. Fix a sequent $\Gamma = \exists \boldsymbol{v}_0 A_0, \ldots, \exists \boldsymbol{v}_k A_k$ of prenex Σ_1 formulæ wherein for each $i \leq k$, A_i is quantifier-free and $\exists \boldsymbol{v}_i$ abbreviates a block of existential quantifiers of length a_i. Let $\pi \vdash \Gamma$ be an LK-proof with cuts of complexity at most prenex Π_2/Σ_2. The *language* of π, denoted $L(\pi)$, is the set of pairs (i, T^*) such that $i \leq k$ and $T : o^{a_i}$ is a structured sequence-term free of non-terminal symbols derivable from the term $\sigma^i_\pi \langle\rangle \cdots \langle\rangle$. The next lemma demonstrates that the choice of starting symbol is canonical.

Lemma 3.1. *If $\pi \vdash A, \Gamma$ and A is prenex Σ_1 then for all terms $u_0, \ldots, u_{|\Gamma|}$ of the appropriate type we have $\sigma^i_\pi u_0 \cdots u_{|\Gamma|} \sim \sigma^i_\pi \langle\rangle u_1 \cdots u_{|\Gamma|}$.*

Since the production rules are naturally acyclic (rewriting a non-terminal σ^i_π introduces only non-terminals indexed by strict sub-proofs of π), we deduce

Lemma 3.2. *For any regular proof π, $L(\pi)$ is finite.*

As a consequence of Lemma 3.2 the language of a proof π can be viewed as inducing an *expansion* of its end-sequent, obtained by replacing each formula $\exists \boldsymbol{v}_i A_i$ by the corresponding disjunction $\bigvee \{A_i(\boldsymbol{t}) \mid (i, \boldsymbol{t}) \in L(\pi)\}$.

Theorem 3.3. *If $\pi \vdash \Gamma$ is an LK-proof of a prenex Σ_1 sequent in which all cuts are prenex Π_2 and Σ_2 formulæ then the expansion of Γ induced by $L(\pi)$ is a tautology.*

The proof of this theorem (and the more general statement in Theorem 1.1) is covered in Section 4 below by establishing that the language of a proof is preserved through most cut reduction steps. In the (base) case that all cuts in π are on quantifier-free formulæ, we observe that the grammar rules merely associate to each weak quantifier in the end-sequent the witnesses as they appear in π.

4 Language preservation

Let $\pi \rightsquigarrow \pi'$ express that π' is obtained from π by the application of a reduction rule in Figure 2 to a subproof of π. In the present section we determine for which reduction steps $\pi \rightsquigarrow \pi'$ we have: (i) *language inclusion*: $L(\pi) \supseteq L(\pi')$; and (ii) *language equality*: $L(\pi) = L(\pi')$. Language inclusion will suffice to derive the main theorem; equality allows a more fine-grained study of the Herbrand content of proofs as if π_0 and π_1 are proofs that can be connected by a sequence of forward and backward language-preserving reduction steps then $L(\pi_0) = L(\pi_1)$.

The structure of our proof grammars is such that to deduce inclusion or equality it suffices to analyse the reduction steps locally:

4.1 Cut permutation

We begin by considering the instances of the binary inference reduction that permute two cuts. Suppose $\pi \rightsquigarrow \pi'$ are the two proofs

$$\dfrac{\dfrac{\pi_0 \quad \pi_1}{F,G,\Gamma \qquad \bar{F},\Delta}\,\mathsf{cut} \qquad \pi_2}{\dfrac{G,\Gamma,\Delta \qquad \bar{G},\Lambda}{\pi \vdash \Gamma,\Delta,\Lambda}\,\mathsf{cut}} \qquad\qquad \dfrac{\dfrac{\pi_0 \quad \pi_2}{F,G,\Gamma \qquad \bar{G},\Lambda}\,\mathsf{cut} \qquad \pi_1}{\dfrac{F,\Gamma,\Lambda \qquad \bar{F},\Delta}{\pi' \vdash \Gamma,\Delta,\Lambda}\,\mathsf{cut}} \tag{2}$$

Lemma 4.1. *For π and π' in* (2)*, if either F or G is Σ_2 then $L(\pi) = L(\pi')$.*

Lemma 4.1 covers all cases of permuting two cuts that suffice for establishing Theorem 1.1. In the case both F and G are (genuine) Π_2 formulæ, the language of the grammars need not be preserved:

Lemma 4.2. *There are instantiations of π and π' in* (2) *such that $L(\pi)$ and $L(\pi')$ are incomparable.*

4.2 Contraction reduction

Consider the proofs

$$\dfrac{\dfrac{\dfrac{\pi_0}{F,F,\Gamma}\,\mathsf{c}}{F,\Gamma} \qquad \pi_1}{\pi \vdash \Gamma,\Delta}\,\mathsf{cut} \qquad\qquad \dfrac{\dfrac{\dfrac{\pi_0 \quad \pi_1}{F,F,\Gamma \qquad \bar{F},\Delta}\,\mathsf{cut} \qquad \pi_1^*}{F,\Gamma,\Delta \qquad \bar{F},\Delta}\,\mathsf{cut}}{\pi' \vdash \Gamma,\Delta}\,\mathsf{c}^* \tag{3}$$

Lemma 4.3. *For π and π' in* (3)*, if F is Σ_2 then $L(\pi) = L(\pi')$.*

As in the previous case, language inclusion does not hold in general when reducing a contraction. Specifically, if (i) F is a genuine Π_2 formula i.e. $F = \forall v_0 \cdots \forall v_k \exists w G$ for some Σ_1 formula G, and (ii) there are contractions on $\bar{F}$ in the subproof π_1 then the languages $L(\pi)$ and $L(\pi')$ can be incomparable. We do, however, have

Lemma 4.4. *For π and π' in* (3)*, if F is Π_2 and there are no contractions on $\bar{F}$ in the sub-proof π_1 then $L(\pi') \subseteq L(\pi)$.*

4.3 Quantifier reduction

Lemma 4.5. *For π and π' in* (4) *we have $L(\pi) = L(\pi')$.*

$$\dfrac{\dfrac{\pi_0}{F(v/\alpha),\Gamma}\,\forall}{\forall vF,\Gamma}\quad\dfrac{\dfrac{\pi_1}{\bar{F}(v/t),\Delta}\,\exists}{\exists v\bar{F},\Delta}\,\mathsf{cut}}{\pi \vdash \Gamma,\Delta} \qquad\qquad \dfrac{\dfrac{\pi_0^{[\alpha\mapsto t]}}{F(v/t),\Gamma}\quad\dfrac{\pi_1}{\bar{F}(v/t),\Delta}\,\mathsf{cut}}{\pi' \vdash \Gamma,\Delta} \tag{4}$$

4.4 Quantifier permutation

Consider permuting a universal quantifier with a cut:

$$\dfrac{\dfrac{\dfrac{\pi_0}{A(v/\alpha),\Gamma,F}\,\forall}{\forall vA,\Gamma,F}\quad\dfrac{\pi_1}{\Delta,\bar{F}}\,\mathsf{cut}}{\pi \vdash \forall vA,\Gamma,\Delta} \qquad\qquad \dfrac{\dfrac{\pi_0}{A(v/\alpha),\Gamma,F}\quad\dfrac{\pi_1}{\Delta,\bar{F}}\,\mathsf{cut}}{\dfrac{A(v/\alpha),\Gamma,\Delta}{\pi' \vdash \forall vA,\Gamma,\Delta}\,\forall} \tag{5}$$

Lemma 4.6. *For π and π' in* (5) *we have $L(\pi') \subseteq L(\pi)$.*

4.5 Remaining reductions

The remaining rules are straightforward to analyse and all induce language equality except for weakening reduction for which we have language inclusion.

5 Conclusion

To each proof in first order logic with prenex Π_2/Σ_2 cuts we associate a formal grammar abstracting the semantic aspect of cut elimination and classify the cut reduction and permutation rules according to whether or not the language of the grammar is preserved under these rules. The ultimate goal of the study is to extend this classification to arbitrary classes of cut-formulæ.

The grammars utilised in this paper have a number of advantages over previous language-theoretic approaches for proofs with Π_2/Σ_2 cuts [1, 2]. We can now deal with non-simple proofs (i.e. proofs admitting contractions on universal formulæ) as well as blocks of like quantifiers. Furthermore, unlike the grammars devised in [1, 2], derivations are not restricted by equality constraints (or rigidity requirements). As a result, preservation of language over the cut reduction steps reduces more or less to mere computation. Also notable is the fact that the production rules of the grammar given here are both acyclic and unidirectional relative to the associated proof: each production rule rewrites a non-terminal in favour of non-terminals labelled by strict sub-proofs. Finally, the new grammars make the appearance of non-confluence in cut-elimination more transparent.

Acknowledgements The authors' research was supported by the Wiener Wissenschafts-, Forschungs- und Technologiefonds (WWTF), project no. VRG12-04. The authors wish to thank the anonymous referees for their helpful comments and suggestions.

References

[1] Bahareh Afshari, Stefan Hetzl & Graham E. Leigh (2015): *Herbrand disjunctions, cut elimination and context-free tree grammars*. In Thorsten Altenkirch, editor: *13th International Conference on Typed Lambda Calculi and Applications (TLCA 2015)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 38, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 1–16, doi:10.4230/LIPIcs.TLCA.2015.1.

[2] Bahareh Afshari, Stefan Hetzl & Graham E. Leigh (2016): *Herbrand confluence for first-order proofs with Π_2-cuts*. In Dieter Probst & Peter Schuster, editors: *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, Ontos Mathematical Logic, De Gruyter, Berlin, Boston. isbn: 978-1-5015-0262-0. Preprint avaiable at `http://dmg.tuwien.ac.at/afshari/publications.html`.

[3] Samuel R. Buss (1995): *On Herbrand's theorem*. In: *Logic and Computational Complexity*, *Lecture Notes in Computer Science* 960, Springer, pp. 195–209, doi:10.1007/3-540-60178-3_85.

[4] Sebastian Eberhard & Stefan Hetzl: *On the compressibility of finite languages and formal proofs*. In preparation, preprint available at `http://www.logic.at/people/hetzl/research/`.

[5] Sebastian Eberhard & Stefan Hetzl (2015): *Compressibility of finite languages by grammars*. In Jeffrey Shallit & Alexander Okhotin, editors: *Descriptional Complexity of Formal Systems (DCFS) 2015*, *Lecture Notes in Computer Science* 9118, Springer, pp. 93–104, doi:10.1007/978-3-319-19225-3_8.

[6] Sebastian Eberhard & Stefan Hetzl (2015): *Inductive theorem proving based on tree grammars*. *Annals of Pure and Applied Logic* 166(6), pp. 665 – 700, doi:10.1016/j.apal.2015.01.002.

[7] Philipp Gerhardy & Ulrich Kohlenbach (2005): *Extracting Herbrand disjunctions by functional interpretation*. *Archive for Mathematical Logic* 44, pp. 633–644, doi:10.1007/s00153-005-0275-1.

[8] Willem Heijltjes (2010): *Classical proof forestry*. *Annals of Pure and Applied Logic* 161(11), pp. 1346–1366, doi:10.1016/j.apal.2010.04.006.

[9] Jacques Herbrand (1930): *Recherches sur la théorie de la démonstration*. Ph.D. thesis, Université de Paris.

[10] Stefan Hetzl (2012): *Applying tree languages in proof theory*. In Adrian-Horia Dediu & Carlos Martín-Vide, editors: *Language and Automata Theory and Applications (LATA) 2012*, *Lecture Notes in Computer Science* 7183, Springer, pp. 301–312, doi:10.1007/978-3-642-28332-1_26.

[11] Stefan Hetzl, Alexander Leitsch, Giselle Reis, Janos Tapolczai & Daniel Weller (2014): *Introducing quantified cuts in logic with equality*. In Stéphane Demri, Deepak Kapur & Christoph Weidenbach, editors: *Automated Reasoning - 7th International Joint Conference, IJCAR*, *Lecture Notes in Computer Science* 8562, Springer, pp. 240–254, doi:10.1007/978-3-319-08587-6_17.

[12] Stefan Hetzl, Alexander Leitsch, Giselle Reis & Daniel Weller (2014): *Algorithmic introduction of quantified cuts*. *Theoretical Computer Science* 549, pp. 1–16, doi:10.1016/j.tcs.2014.05.018.

[13] Stefan Hetzl, Alexander Leitsch & Daniel Weller (2012): *Towards algorithmic cut-introduction*. In Andrei Bjørner, Nikolaj & Voronkov, editor: *Logic for Programming, Artificial Intelligence and Reasoning (LPAR-18)*, *Lecture Notes in Computer Science* 7180, Springer, pp. 228–242, doi:10.1007/978-3-642-28717-6_19.

[14] Stefan Hetzl & Daniel Weller (2013): *Expansion trees with cut*. Preprint available at `http://arxiv.org/abs/1308.0428`.

[15] David Hilbert & Paul Bernays (1939): *Grundlagen der Mathematik II*. Springer.

[16] Richard McKinley (2013): *Proof nets for Herbrand's theorem*. *ACM Transactions on Computational Logic* 14(1), pp. 5:1–5:31, doi:10.1145/2422085.2422090.

[17] Dale Miller (1987): *A compact representation of proofs*. *Studia Logica* 46(4), pp. 347–370, doi:10.1007/BF00370646.