Herbrand Disjunctions, Cut Elimination and Context-Free Tree Grammars

Bahareh Afshari, Stefan Hetzl, and Graham E. Leigh

Institute of Discrete Mathematics and Geometry
Vienna University of Technology
Wiedner Hauptstraße 8-10, 1040 Vienna, Austria
`{bahareh.afshari, stefan.hetzl, graham.leigh}@tuwien.ac.at`

Abstract

Recently a new connection between proof theory and formal language theory was introduced. It was shown that the operation of cut elimination for proofs in first-order predicate logic involving Π_1-cuts corresponds to computing the language of a particular class of regular tree grammars. The present paper expands this connection to the level of Π_2-cuts. Given a proof π of a Σ_1 formula with cuts only on Π_2 formulæ, we show there is associated to π a natural context-free tree grammar whose language is finite and yields a Herbrand disjunction for π.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.4.2 Grammars and Other Rewriting Systems, F.4.3 Formal Languages

Keywords and phrases Classical logic, Context-free grammars, Cut elimination, First-order logic, Herbrand's theorem, Proof theory

Digital Object Identifier 10.4230/LIPIcs.TLCA.2015.1

1 Introduction

The computational content of proofs is a central topic of proof theory. In intuitionistic first-order logic the existential witness property states that the provability of $\exists x F$ entails the existence of some ground term t such that $F(x/t)$ is provable. The analogue of this property in classical logic is Herbrand's theorem [12] (see also [5]). In its simplest version Herbrand's theorem states that if $\exists x F$ is valid and F quantifier-free there exist closed terms $t_1, \dots, t_k$ such that $\bigvee_{i=1}^{k} F(x/t_i)$ is a tautology. Such formulæ, quantifier-free disjunctions of instances, are hence also called Herbrand disjunctions. Herbrand's theorem applies not only to existential but arbitrary first-order formulæ, providing a tautology by replacing each non-prenex quantifier with a suitable finite disjunction or conjunction of instances. Provided one is willing to speak about provability instead of validity Herbrand's theorem even extends to classical higher-order logic, see for example [25].

A Herbrand disjunction can be read off directly from a cut-free proof though proofs with cut may be non-elementarily smaller than the shortest Herbrand disjunction [30, 26, 27]. Therefore, in general, cut elimination (or an equivalent normalisation process) is necessary in order to obtain a Herbrand disjunction. However, if one is only interested in the witness terms of a Herbrand disjunction and not in the complete cut-free proof then it would be desirable to circumvent the cumbersome process of cut elimination.

For instance, in [13, 14] it was shown that a proof $\pi \vdash \exists x F$ in which all cut formulæ have at most one quantifier induces a (totally rigid acyclic) tree grammar $\mathcal{G}_\pi$, of size no greater than the size of the proof, with a finite language that, when interpreted as a collection of witness terms, forms a Herbrand disjunction for $\exists x F$ (see Figure 2).

$$\dfrac{\begin{array}{cc}\dfrac{A(\alpha)}{\forall xA} & t_1, t_2, \dots, t_k \\ \vdots & \vdots \\ \Gamma, \forall xA & \Delta, \exists x\bar{A}\end{array}}{\Gamma, \Delta}\ \text{cut}$$

Figure 1 Π_1 cut.

$$\begin{tikzcd}
\pi \vdash \exists xF \arrow[r, "c.e."] \arrow[d, "\text{definition}"'] & \pi' \vdash_{cf} \exists xF \arrow[d, "\text{Herbrand extraction}"] \\
\mathcal{G}_\pi \text{ (of size } \leq |\pi|) \arrow[r, "\mathcal{L}"] & \mathcal{L}(\mathcal{G}_\pi) \supseteq \mathcal{H}(\pi')
\end{tikzcd}$$

$\mathcal{L}(\mathcal{G}_\pi)$: language of $\mathcal{G}_\pi$ $\quad\mathcal{H}(\pi')$: Herbrand set of π'

Figure 2 Proof grammars.

Grammars for proofs with only cuts of the form $\exists xA$ or $\forall xA$ with A quantifier free are remarkably simple. Let π be such a proof with end sequent $\exists xF$ where F is quantifier free. Suppose $u_1, u_2, \dots, u_m$ are the witnesses to the existential quantifier as they appear in π. The induced grammar $\mathcal{G}_\pi$ consists of the production rules i) $\sigma \to F(u_1) \mid F(u_2) \mid \dots \mid F(u_m)$ where σ is the starting symbol of the grammar, and ii) $\alpha \to t_1 \mid t_2 \mid \cdots \mid t_k$ for every Π_1 cut in π of the form given in Figure 1, where α is the eigenvariable of the cut and $t_1, t_2, \dots, t_k$ are witness terms of existential quantifier in the right subproof.

1.1 Contributions

In this paper we show how the correspondence between proofs and grammars can be extended to the level of Π_2 cuts. This class of cuts is particularly important for computational applications: a Π_2 formula can be read as a specification of a program and its proof as providing an (non-deterministic) algorithm in line with the Curry–Howard correspondence.

We consider Π_2-proofs in which all cut formulæ have at most one quantifier of each sort i.e. of the form $\forall x\exists yA$ or $\forall xA$ with A quantifier free. It turns out that these proofs correspond to (rigid) context-free tree grammars:

▶ **Theorem 1.** *Let π be a Π_2-proof of a Σ_1 formula F. There is an associated rigid context-free tree grammar $\mathcal{G}_\pi$ (whose number of production rules is bounded by the size of π) with a finite language yielding a Herbrand disjunction for F.*

In fact, if we consider proofs in which every universal introduction rule is immediately followed by a cut or an existential introduction (henceforth called *simple proofs*) we have

▶ **Theorem 2.** *Let $\pi_0, \pi_1, \dots, \pi_k$ be a sequence of simple Π_2-proofs such that π_{i+1} is obtained from π_i by an application of a cut reduction rule (as in Figure 4) to a subproof of π_i. For every $i \leq k$, $\mathcal{L}(\mathcal{G}_{\pi_0}) \supseteq \mathcal{L}(\mathcal{G}_{\pi_i})$. In particular, if the proof π_k contains only quantifier-free cuts $\mathcal{L}(\mathcal{G}_{\pi_0}) \supseteq \mathcal{H}(\pi_k)$.*

Rigid tree languages were first introduced in [21, 22] with applications to verification in mind. Later in [14] rigid regular grammars were defined by restricting the admissible derivations in the grammar with an equality constraint. In this paper we extend the notion of rigidity to context-free grammars. It plays an important role in obtaining a concise grammar for our proofs. For a Π_2-proof π, we prove the language of the induced rigid context-free grammar $\mathcal{G}_\pi$ has a bound double exponential in the size of π (see Theorem 10). This bound is optimal as it matches the blow up from cut elimination.

Structure of the article. In Section 2 we fix the calculus and cut reduction steps, and define the class of proofs under consideration. Section 3 develops the theory of rigid context-free tree grammars. In Section 4 we present the proof grammars induced by Π_2-proofs: an

Axioms: $A, \bar{A}$ for A an atomic formula

Inference rules:

$$\frac{\Gamma, A, B}{\Gamma, A \vee B}\,\vee \qquad \frac{\Gamma, A \qquad \Delta, B}{\Gamma, \Delta, A \wedge B}\,\wedge$$

$$\frac{\Gamma, A[x/\alpha]}{\Gamma, \forall x A}\,\forall \qquad \frac{\Gamma, A[x/s]}{\Gamma, \exists x A}\,\exists \qquad \frac{\Gamma, A \qquad \Delta, \bar{A}}{\Gamma, \Delta}\,\mathsf{cut}$$

$$\frac{\Gamma}{\Gamma, \Delta}\,\mathsf{w} \qquad \frac{\Gamma, \Delta, \Delta^*}{\Gamma, \Delta}\,\mathsf{c}$$

Figure 3 Axioms and rules of sequent calculus.

elementary grammar is given in Section 4.1 to motivate the definition laid out in Section 4.2. Section 5 establishes the preservation of the language of our proof grammars under transitive closure of cut reduction steps. In Section 6 we conclude by describing future work and potential applications of our results and techniques.

1.2 Related work

In [9] Gerhardy and Kohlenbach adapt Shoenfield's variant of Gödel's Dialectica interpretation to a system of pure predicate logic by explicitly adding decision-by-case constants to the target language. The resulting λ-term is first normalised and then used to directly read off a Herbrand disjunction. Heijltjes [10] and McKinley [24] study graphical formalisms of local reductions in classical first-order logic to derive a normal form corresponding to a cut-free proof from which a Herbrand disjunction is obtained. An approach similar to [10, 24] in the formalism of expansion trees [25] can be found in [19]. Historically, Hilbert's ε-calculus [20] is the first formalism which allows a step-wise computation of a Herbrand disjunction in a way that abstracts from the propositional layer of predicate logic.

Like the aforementioned formalisms, the results presented in this paper allow the computation of a Herbrand disjunction in a way that bypasses literal cut elimination. The novelty of our approach lies in the fact that this is achieved via formal grammars. On the one hand this has the consequence that standard problems from formal language theory assume a proof-theoretic meaning and hence standard algorithms can be used to solve the corresponding proof-theoretic problems. For example, an algorithm for solving the membership problem for an adequate class of grammars also solves the following proof-theoretic problem (for the corresponding class of proofs): given a proof π and a term t, is t a witness obtainable by cut-elimination from π? Usually, these 'induced' algorithms have a smaller asymptotic complexity (polynomial to at most exponential time; see e.g. [23, 22]) than the naive proof-theoretic algorithms which rely on explicitly computing the cut-free proof(s) (and need iterated-exponential time). On the other hand, the strong grip on the structure of a Herbrand disjunction afforded by a formal grammar opens the door to interesting theoretical and applied investigations (see Section 6).

2 Proof calculus

We work with a Tait-style (one-sided) sequent calculus for first-order logic with explicit weakening and contraction rules. A *proof* is a finite tree obtained from the axioms and rules laid out in Figure 3. We use capital Greek letters Γ, Δ, etc. for multisets of formulæ, $\bar{A}$ to denote the dual of the formula A obtained by de Morgan laws, and $A[x/s]$ for the formula

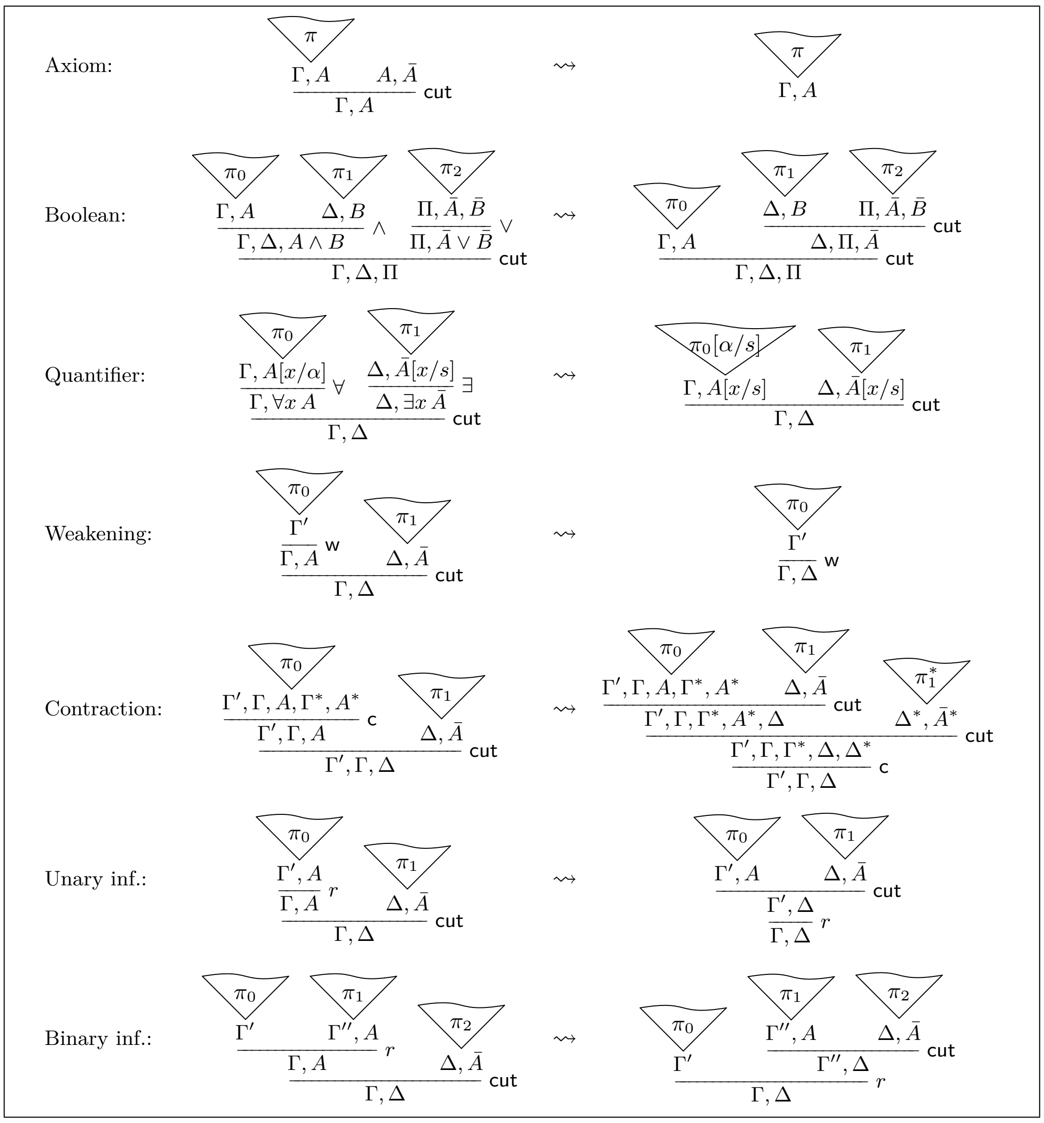


Figure 4 One-step cut reduction and permutation rules.

obtained from A by replacing x with the term s assuming this will not create any variable capture. Γ, Δ and Γ, A denote respectively the disjoint union of Γ, Δ and $\Gamma, \{A\}$.

In the ($\forall$) rule, α is called the *eigenvariable* and must not appear free in $\Gamma, \forall x A$; in ($\exists$) rule the term s is assumed to be free for x; and in the contraction rule (c), the set Δ^* denotes a renaming of Δ. Those formulæ which are explicitly mentioned in the premise of an inference rule are said to be *principal* in the rule applied, for example A and B are principal in ($\wedge$) rule, every formula from Δ^* is principal in (c), and there are no principal formulæ in the weakening rule (w).

We assume all proofs are *regular* namely, all quantifiers' eigenvariables are distinct and different from any free variables. We use the following naming convention for proofs throughout the paper: lower-case Greek letters α, β, γ, etc. represent eigenvariables; x, y for bound variable symbols; and π, π', etc. for proofs. $\pi[\alpha/s]$ is the result of replacing throughout the proof π each occurrence of the variable symbol α by the term s. We write $\pi \vdash \Gamma$ to express

π is a proof of Γ and $\mathrm{EV}(\pi)$ to denote the set of eigenvariables in π. For a position p in (the tree) π, $\pi|p$ denotes the subproof of π at position p with the convention that $\pi|\langle\rangle = \pi$, and $\pi|p0$ and $\pi|p1$ are respectively the left (or only) subproof and right subproof of $\pi|p$. We write $q < p$ if q is a proper prefix of p.

It is useful to isolate a particular class of Π_2-proofs:

▶ **Definition 3** (Simple Π_2-proof)**.** A *simple* Π_2*-proof* is a proof in which i) each sequent is a finite set of prenex Π_2 formulæ, ii) the conclusion is a set of closed Π_1 and Σ_1 formulæ, iii) cut formulæ feature at most one quantifier of each sort, and iv) every universally quantified formula appearing above a cut is principal in the inference directly after its introduction (either a cut or existential introduction).

Condition (ii) above is stipulated primarily for expository purposes. Note that every sequent can be transformed into a sequent containing only Σ_1 formulæ by Skolemization and prenexification. In contrast, it is impossible to Skolemize cut formulæ in a proof since the two (dual) occurrences of the cut formula would yield dual Skolemizations. Also note that condition (iv) does not restrict provability, as any proof satisfying (i)–(iii) can be modified using simple rule permutations to also satisfy (iv). This transformation will not increase the size (number of inference rules) of the proof.

Cut reduction and Herbrand sets

The standard cut reduction rules are given in Figure 4. For proofs π, π' we write $\pi \rightsquigarrow \pi'$ if π' is the result of applying one of the rules to π (and not to a subproof). Notice that in contraction reduction a subproof is duplicated and care is taken to rename the eigenvariables (expressed by annotating the proof/sequent/formula in question by an asterisk) to maintain regularity.

Let π be a cut-free proof of Γ and suppose $A \in \Gamma$ has the form $\exists x_1 \cdots \exists x_k B$ with B quantifier free. If $\bigvee_{(s_1,\dots,s_k)\in X} B[x_1/s_1,\dots,x_k/s_k]$ is the Herbrand disjunction for A read off from π, we call X the *Herbrand set of A in π* and define $\mathcal{H}(\pi, A) = X$. In addition, $\mathcal{H}(\pi) := \{\{A\} \times \mathcal{H}(\pi, A) \mid A \in \Gamma \cap \Sigma_1\}$.

Running example

We consider a formal proof of the pigeonhole principle for two boxes via the infinite pigeonhole principle. The question of the computational content of this proof is attributed to G. Stolzenberg in [6]. A variety of analytic methods have since been applied to this proof [11, 33, 3, 2] and its generalisations [29, 28]. Despite its relatively short and symmetric nature it allows us to adequately demonstrate grammars for proofs with Π_2 cuts.

Let $f\colon \mathbb{N} \to \{0, 1\}$ be a total Boolean function, let I_i express that there are infinitely many $m \in \mathbb{N}$ for which $f(m) = i$ and T express that there exists $m < n$ such that $f(m) = f(n)$. A consequence of the law of excluded middle is $I_0 \vee I_1$. Moreover I_i implies T for each i: assuming I_i, there exists $m \geq 0$ and $n \geq m + 1$ for which $f(m) = f(n) = i$. Combining these observations we conclude that T holds.

The following formalises the above argument. The formal language, Σ, consist of two unary function symbols f, s, one binary function symbol M, a constant symbol 0 and a binary relation $\leq$. We make the following definitions and abbreviations:

- $T = \exists x \exists y (x < y \wedge \mathsf{f}x = \mathsf{f}y)$,
- $I_i = \forall x \exists y (x \leq y \wedge \mathsf{f}y = \underline{i})$ for $i \in \{0, 1\}$ where $\underline{0} = 0$ and $\underline{1} = \mathsf{s}0$,
- $\Gamma = \{\overline{\forall x \forall y (x \leq \mathsf{M}xy \wedge y \leq \mathsf{M}xy)}, \overline{\forall x (\mathsf{f}x = 0 \vee \mathsf{f}x = \mathsf{s}0)}\}$,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, I_0^{\alpha,\mathsf{M}\alpha\hat\alpha}, I_1^{\hat\alpha,\mathsf{M}\alpha\hat\alpha}}{\Gamma, I_0^{\alpha}, I_1^{\hat\alpha}}\exists^*}{\Gamma, I_0^{\alpha}, I_1}\forall \qquad \dfrac{\Delta, T_{\hat\beta,\hat\beta'}, \bar I_1^{0,\hat\beta}, \bar I_1^{\mathsf{s}\hat\beta,\hat\beta'}}{\Delta, T, \bar I_1}\exists^*}{\Gamma,\Delta,T,I_0^{\alpha}}\mathsf{cut}}{\Gamma,\Delta,T,I_0}\forall \qquad \dfrac{\Delta, T_{\beta,\beta'}, \bar I_0^{0,\beta}, \bar I_0^{\mathsf{s}\beta,\beta'}}{\Delta, T, \bar I_0}\exists^*}{\Gamma,\Delta,T}\mathsf{cut,c}$$

Figure 5 Simple Π_2-proof π_∞ of pigeonhole principle. The inference rules labelled $\exists^*$ represent finitely many existential introduction rules (the order of which is unimportant).

- $\Delta = \{\overline{\forall x \forall y \forall z (x = y \wedge z = y \to x = z)}, \overline{\forall x \forall y (sx \le y \to x < y)}\}$,
- I_i^s and $I_i^{s,t}$ denote, respectively, $\exists y (s \le y \wedge \mathsf{f} y = \underline{i})$ and $(s \le t \wedge \mathsf{f}t = \underline{i})$,
- $T_{s,t}$ denotes $(s < t \wedge \mathsf{f}s = \mathsf{f}t)$.

The intended interpretation of the symbols is: f represents the (arbitrary) function f, s the successor function on $\mathbb{N}$, $\le$ the standard ordering and M the binary max function.

A formal proof of the pigeonhole principle is given by the simple Π_2-proof in Figure 5 which we name π_∞. For brevity only eigenvariables and witnesses of the quantifiers and instances of the existential formula T are displayed in π_∞. The proof uses about 50 application of the axioms and rules of the calculus but the only cuts in π_∞ are the two Π_2 cuts shown in the figure. Two normal forms of the proof (of size $\sim$200) have been computed in a case study [33] from which one can read off the Herbrand set $\{(0,1),(1,2),(2,3),(0,2),(1,3)\}$ (up to interpretation of the logical symbols) relative to the formula T. Once we have introduced our grammars we shall see how this Herbrand set can be directly computed from π_∞.

3 Context-free tree grammars

In [14] and elsewhere a refinement of regular tree grammars was studied that mimics the construction of terms appearing in cut-elimination for first-order logic with Π_1 cuts. These grammars were called *rigid* tree grammars and are equivalent to the notion of rigid automata introduced and explored in [21, 22]. In this section we provide a generalisation to the class of context-free tree grammars corresponding to Π_2-proofs.

Given a ranked alphabet Σ, we let $\mathrm{Terms}(\Sigma)$ denote the set of terms in the simply-typed λ-calculus built from Σ. For a $T \in \mathrm{Terms}(\Sigma)$ we write $\mathrm{Pos}(T)$ for the set of positions in term (tree) T. For $p \in \mathrm{Pos}(T)$, $T|p$ is the subterm of T at position p.

A *context-free tree grammar* (CFG) is a tuple $\mathcal{G} = \langle N, \Sigma, \sigma, \mathrm{Pr}\rangle$ where N is a set of typed *non-terminals* of order at most 1, $\sigma \in N$ is a designated *start symbol* (of base-type ι), Σ is a ranked alphabet, called *terminals*, disjoint from N, and Pr consists of pairs (a, T) (called *production rules* and written $a \to T$) where $a \in N$ and $T \in \mathrm{Terms}(\Sigma \cup N)$ that has the same type as a. Given a CFG $\mathcal{G}$ we assume $\mathcal{G} = \langle N_\mathcal{G}, \Sigma_\mathcal{G}, \sigma_\mathcal{G}, \mathrm{Pr}_\mathcal{G}\rangle$. If the set $N_\mathcal{G}$ of non-terminals contains only symbols of order 0, $\mathcal{G}$ is a *regular tree grammar.*

Let d be a sequence $\langle \rho_i, p_i\rangle_{i<k}$ of pairs of production rules of a CFG $\mathcal{G}$ and positions, and S and T terms. We call d a *derivation from S to T*, written $d\colon S \to T$, if there exist terms $(N_i)_{i \le k}$ such that $N_0 = S$, $N_k = T$, and for each $0 \le i < k$,

1. ρ_i is a production rule of $\mathcal{G}$ and $p_i \in \mathrm{Pos}(N_i)$,
2. For $\rho_i = (a \to S)$, we have $N_i|p_i = a$ and $N_{i+1} = N_i[p_i/S]$, namely the result of replacing the sub-term of N_i at position p_i by S (renaming bound variables if necessary).

The sequence of terms $(N_i)_{i\leq k}$ is uniquely determined by d and S, whence we may write $d(i)$ for N_i. The *length* of d, $\mathrm{lh}(d)$, is k. We say T *is derivable from* S if there exists a derivation $d\colon S \to T$. $\mathcal{G}$ is *acyclic* if for every non-terminal $a \in N_{\mathcal{G}}$ and every derivation $d\colon a \to T$ with $\mathrm{lh}(d) > 0$, a does not appear in T.

The *language* of a CFG $\mathcal{G}$ is defined as $\mathcal{L}(\mathcal{G}) = \{T \in \mathrm{Terms}(\Sigma_{\mathcal{G}}) \mid \exists d\colon \sigma_{\mathcal{G}} \to T\}$. When comparing languages of CFGs it is convenient to work modulo β-convertibility. Thus for grammars $\mathcal{G}$, $\mathcal{H}$, we write $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{H})$ to express that for every $S \in \mathcal{L}(\mathcal{G})$ there exists a β-equivalent $T \in \mathcal{L}(\mathcal{H})$.

3.1 Rigidity

Let $\mathcal{G}$ be a CFG, suppose $\lhd$ is a transitive binary relation on $N_{\mathcal{G}}$, and R is a designated set of non-terminals. A derivation $d = \langle (a_i \to S_i), p_i\rangle_{i<\mathrm{lh}(d)}\colon S \to T$ induces a natural equivalence relation on the positions in T corresponding to connectedness in parse trees: for $j_0, j_1 < \mathrm{lh}(d)$, let $j_0 \sim_d j_1$ iff there exist $i_0 < j_0, j_1$ such that

1. $p_{i_0} \leq p_{j_0}, p_{j_1}$,
2. $a_{j_0} = a_{j_1} \in R$,
3. for every $k \in \{0,1\}$ and $i_0 < i < j_k < \mathrm{lh}(d)$, if $p_i \leq p_{j_k}$ then $a_{j_k} \not\lhd a_i$.

In other words, two occurrences of a non-terminal $a \in R$ in (the natural tree representation of) the derivation d are considered connected if there is no non-terminal of higher priority between them and their closest common ancestor.

Of particular interest to us is the class of derivations that respect their own $\sim$ relation.

▶ **Definition 4.** Let $\mathcal{G}$ be a CFG and suppose $\lhd$ is a transitive ordering on $N_{\mathcal{G}}$ and $R \subseteq N_{\mathcal{G}}$. A derivation $d = \langle \rho_i, p_i\rangle_{i<k}\colon S \to T$ in $\mathcal{G}$ is *rigid with respect to* $(\lhd, R)$ if for every $i, j < k$, $i \sim_d j$ implies $T|p_i = T|p_j$.

A *rigid context-free tree grammar* is a structure $\mathcal{G} = \langle N_{\mathcal{G}}, R_{\mathcal{G}}, \lhd_{\mathcal{G}}, \Sigma_{\mathcal{G}}, \sigma_{\mathcal{G}}, \mathrm{Pr}_{\mathcal{G}}\rangle$ such that $\langle N_{\mathcal{G}}, \Sigma_{\mathcal{G}}, \sigma_{\mathcal{G}}, \mathrm{Pr}_{\mathcal{G}}\rangle$ is a CFG, $R_{\mathcal{G}} \subseteq N_{\mathcal{G}}$ and $\lhd_{\mathcal{G}}$ is a transitive order on $N_{\mathcal{G}}$. $R_{\mathcal{G}}$ is the set of *rigid* non-terminals of $\mathcal{G}$ and $\lhd_{\mathcal{G}}$ is the *priority ordering of* $\mathcal{G}$. If $R_{\mathcal{G}} = N_{\mathcal{G}}$ then $\mathcal{G}$ is called *totally rigid*.

Given a rigid CFG $\mathcal{G}$ and a derivation d in its underlying CFG, we call d *rigid* if it is rigid with respect to $(\lhd_{\mathcal{G}}, R_{\mathcal{G}})$. The *language* of a rigid CFG $\mathcal{G}$, $\mathcal{L}(\mathcal{G})$, is the collection of terms derivable from rigid derivations starting from $\sigma_{\mathcal{G}}$:

$$\mathcal{L}(\mathcal{G}) = \{T \in \mathrm{Terms}(\Sigma_{\mathcal{G}}) \mid \exists d\colon \sigma_{\mathcal{G}} \to T \text{ and } d \text{ is } (\lhd_{\mathcal{G}}, R_{\mathcal{G}})\text{-rigid}\}.$$

Note the language of a rigid CFG is a set of closed (well-typed) base-type λ-terms.

▶ **Example 5.** Let $\mathcal{G}$ be the rigid CFG with start symbol σ, non-terminals σ, α and γ, rigid non-terminals R, ordering $\lhd$, terminal symbols of appropriate arity, and production rules $\sigma \to f(\alpha,\gamma,\gamma)$, $\gamma \to g(\gamma) \mid a$, and $\alpha \to \gamma$. If $\gamma \notin R$ we have, unsurprisingly, $\mathcal{L}(\mathcal{G}) = \{f(g^m(a), g^n(a), g^o(a)) \mid m,n,o \geq 0\}$. Otherwise,

1. if $\gamma \lhd \gamma \lhd \alpha$, $\mathcal{L}(\mathcal{G}) = \{f(g^m(a), g^n(a), g^n(a)) \mid m,n \geq 0\}$;
2. if $\gamma \lhd \gamma \not\lhd \alpha$, $\mathcal{L}(\mathcal{G}) = \{f(g^m(a), g^m(a), g^m(a)) \mid m \geq 0\}$;
3. if $\gamma \not\lhd \gamma \not\lhd \alpha$, $\mathcal{L}(\mathcal{G}) = \{f(a,a,a)\}$.

In contrast to the grammar presented in Example 5, acyclic tree grammars must have a finite language. If the grammar is also totally rigid its language has size essentially double exponential in the size of the grammar.

▶ **Lemma 6.** *If $\mathcal{G}$ is an acyclic totally rigid CFG then* $|\mathcal{L}(\mathcal{G})| \leq |\mathrm{Pr}_{\mathcal{G}}|^{2^{|N_{\mathcal{G}}|-1}}$.

Proof. Suppose $\mathcal{G}$ satisfies the above requirements; let $m = |\mathrm{Pr}_{\mathcal{G}}|$ and $n = |N_{\mathcal{G}}| - 1$. Since $\mathcal{G}$ is acyclic we may further assume that $\lhd_{\mathcal{G}}$ is irreflexive. We argue, by induction on n, that the set $\{T \in \mathrm{Terms}(\Sigma_{\mathcal{G}}) \mid \exists d\colon \sigma_{\mathcal{G}} \to T \text{ and } d \text{ is rigid}\}$ has size bounded by m^{2^n}.

The base case is $n = 0$. By the main assumption of the lemma every derivation $d\colon \sigma_{\mathcal{G}} \to T \in \mathrm{Terms}(\Sigma_{\mathcal{G}})$ has length 1, of which there are no more than m. For the induction step, suppose $n = n_0 + 1$. Let $N = N_{\mathcal{G}} \setminus \{a\}$ where $a \neq \sigma_{\mathcal{G}}$ is chosen such that $a \not\lhd_{\mathcal{G}} b$ for all $b \in N_{\mathcal{G}} \setminus \{\sigma_{\mathcal{G}}\}$. Suppose $d\colon \sigma_{\mathcal{G}} \to T \in \mathrm{Terms}(\Sigma_{\mathcal{G}})$ is a rigid derivation in $\mathcal{G}$. Since $\mathcal{G}$ is acyclic d can be re-ordered to have the form $d_0 d_1$ where $d_0\colon \sigma_{\mathcal{G}} \to S$ and $d_1\colon S \to T = S[a/S']$ for appropriate terms S and S', such that the non-terminal a is not rewritten in d_0 and not introduced by a production rule in d_1. The induction hypothesis implies there is no more than $m^{2^{n_0}}$ possibilities for each of S and S', whence there are $\leq m^{2^n}$ possibilities for T. ◀

4 Proof grammars

In this section we present rigid context-free grammars that arise from simple Π_2-proofs. We first define *elementary proof grammars* which are a natural extension of the rigid regular grammars arising from Π_1-proofs introduced in [14]. As we shall see, elementary proof grammars can have an infinite language and are not immediately suitable for producing Herbrand disjunctions. A proper form of *proof grammars* is then defined in Section 4.2 by adding further structure.

4.1 Elementary proof grammars

Let $\pi \vdash \Gamma$ be a simple Π_2-proof and Γ a set of closed Σ_1 and Π_1 formulæ. The *elementary grammar for* π is a rigid CFG, denoted $\mathcal{E}_\pi$, of the form $\langle N_\pi, R_\pi, \lhd_\pi, \Sigma_\pi, \sigma, \mathrm{Pr}_\pi \rangle$ where

- N_π consists of the eigenvariables appearing in π (of base-type) as well as a *starting symbol* σ and a symbol κ^p (of function type $\iota \to \iota$) for each position p in π at which the rule cut is applied;
- $R_\pi = \mathrm{EV}(\pi)$;
- Σ_π is the term language of first-order logic expanded by
  - a symbol $\tau_{i,F}$ of base-type for each formula $\forall x_0 \cdots x_k F \in \Gamma \cap \Pi_1$ and each $i \leq k$,
  - a symbol F of function type with k arguments for each $\exists x_0 \cdots \exists x_k F \in \Gamma \cap \Sigma_1$;
- $\lhd_\pi$ is the transitive ordering on non-terminals and Pr_π the set of production rules specified below.

Each cut occurring in π, as well as each quantified formula in the conclusion Γ, yields production rules in $\mathcal{E}_\pi$. In addition, the relative order of the quantifier introduction rules and each cut on a genuine Π_2 formula influence the rigidity ordering $\lhd_\pi$. We begin by specifying the rules induced by Γ.

For each formula $\exists x_0 \cdots \exists x_k F \in \Gamma$ with F quantifier-free, and each sequence of terms $\langle s_i \rangle_{i \leq k}$ that appear in π (together) as the witnesses of the sequence of existential quantifiers $\langle \exists x_i \rangle_{i \leq k}$, $\mathcal{E}_\pi$ features a production rule $\sigma \to \mathsf{F} s_0 \cdots s_k$. For each formula $\forall x_0 \cdots \forall x_k F \in \Gamma$ with F quantifier-free and each $i \leq k$, $\mathcal{E}_\pi$ contains the production rule $\alpha_i \to \tau_{i,F}$ where α_i is the (unique) eigenvariable for the quantifier $\forall x_i$ appearing in π. See Figure 6.

The remaining non-terminals attain their production rules based on the structure of the cut in which they are used. Let p be a position in π, A a quantifier-free formula, α, δ eigenvariables and s, t terms. Suppose $\pi|p$ has the form of either cuts given in Figure 7. In the proof on the left the two distinguished appearances of the formula $\exists y A[x/\alpha]$ as well as

$\pi \vdash \Gamma$	$\begin{array}{c}\dfrac{\Pi', F[x_0/s_0,\dots,x_k/s_k]}{\Pi', \exists x_k F[x_0/s_0,\dots,x_{k-1}/s_{k-1}]}\exists \\ \vdots \\ \hline \Gamma = \Pi, \exists x_0 \cdots \exists x_k F\end{array}$	$\begin{array}{c}\dfrac{\Pi', \forall x_{i+1}\cdots\forall x_k F[x_0/\alpha_0,\dots,x_i/\alpha_i]}{\Pi', \forall x_i \cdots \forall x_k F[x_0/\alpha_0,\dots,x_{i-1}/\alpha_{i-1}]}\forall \\ \vdots \\ \hline \Gamma = \Pi, \forall x_0 \cdots \forall x_k F\end{array}$
$\mathcal{E}_\pi$	$\sigma \to \mathsf{F} s_0 \cdots s_k$	$\alpha_i \to \tau_{i,F}$
$\mathcal{G}_\pi$	$\sigma \to \mathsf{F} s_0^\sigma \cdots s_k^\sigma$	$\alpha_i \to \tau_{i,F}$

Figure 6 Start and terminal production rules in $\mathcal{E}_\pi$ and $\mathcal{G}_\pi$.

Subproof $\pi\|p$	$\dfrac{\dfrac{\begin{array}{c}\dfrac{\Pi'', A[x/\alpha, y/s]}{\Pi'', \exists y A[x/\alpha]}\exists \\ \vdots \\ \Pi', \exists y A[x/\alpha]\end{array}}{\Pi', \forall x \exists y A}\forall \qquad \begin{array}{c}\dfrac{\dfrac{\Delta', \bar{A}[x/t, y/\beta]}{\vdash \Delta', \forall y \bar{A}[x/t]\ (\dagger)}\forall}{\Delta', \exists x \forall y \bar{A}}\exists \\ \vdots \\ \Delta, \exists x \forall y \bar{A}\end{array}}{\Pi, \Delta}\text{cut}$	$\dfrac{\dfrac{\Pi', A[x/\alpha]}{\Pi', \forall x A}\forall \qquad \begin{array}{c}\dfrac{\Delta', \bar{A}[x/t]}{\Delta', \exists x \bar{A}}\exists \\ \vdots \\ \Delta, \exists x \bar{A}\end{array}}{\Pi, \Delta}\text{cut}$
Rules in $\mathcal{E}_\pi$	$\alpha \to t \qquad \beta \to \kappa^p t \qquad \kappa^p \to \lambda\alpha\, s$	$\alpha \to t$
Rules in $\mathcal{G}_\pi$	$\alpha \to \lambda x_1 \cdots \lambda x_{k_\alpha}\, t^\alpha$ $\beta \to \lambda x_1 \cdots \lambda x_{k_\beta}.\ \kappa^p x_1 \cdots x_{k_\alpha} t^\beta$ $\kappa^p \to \lambda x_1 \cdots \lambda x_{k_\alpha+1}\, s^\alpha$	$\alpha \to \lambda x_1 \cdots \lambda x_{k_\alpha}\, t^\alpha$

Figure 7 Internal production rules in $\mathcal{E}_\pi$ and $\mathcal{G}_\pi$.

the two distinguished occurrences of $\exists x \forall y \bar{A}$ are assumed to be on the same trace, as are, in the right proof, the two occurrences of $\exists x \bar{A}$. The grammar includes the production rules

$$\alpha \to t \qquad\qquad \beta \to \kappa^p t \qquad\qquad \kappa^p \to \lambda\alpha\, s$$

and we set $a \lhd_\pi \alpha \lhd_\pi \kappa^p \lhd_\pi b$ for each $a \in \mathrm{EV}(\pi|q_\alpha)$ and $b \in \mathrm{EV}(\pi|q_\beta)$ where q_α and q_β are, respectively, the positions in π at which the variables α and β were eliminated (that is $p0$ and the position marked by ($\dagger$) in Figure 7). In the scenario pictured on the right in which the cut is performed on a Π_1 formula, only a single production rule for α is needed.

▶ **Example 7** (Elementary proof grammar for π_∞)**.** Let $\mathcal{E}_{\pi_\infty} = \langle N_\infty, R_\infty, \lhd_\infty, \Sigma_\infty, \sigma, \mathrm{Pr}_\infty\rangle$. The non-terminals of $\mathcal{E}_{\pi_\infty}$ comprise: starting symbol σ; rigid non-terminals: α, $\hat{\alpha}$, β, β', $\hat{\beta}$ and $\hat{\beta}'$; non-rigid non-terminals: κ^0 for the topmost cut on I_1 and κ ($=\kappa^{\langle\rangle}$) for the lower cut on I_0. In Σ_∞ we have the term symbols and also terminal symbols for each formula in the conclusion (note there are no universals in the conclusion of π_∞). The induced priority ordering is $\hat{\alpha} \lhd_\infty \kappa^0 \lhd_\infty \{\hat{\beta}', \hat{\beta}\} \lhd_\infty \alpha \lhd_\infty \kappa^{\langle\rangle} \lhd_\infty \{\beta', \beta\}$.

In this example we will focus only on the highlighted formula T and the corresponding symbol T of type $\iota \to \iota \to \iota$ in Σ_∞. Pr_∞ has the following rules. The production rules on the left (right) hand side are read from the cut on I_0 (I_1).

$$\begin{array}{llll}
\sigma \to \mathsf{T}\beta\beta' & & \sigma \to \mathsf{T}\hat{\beta}\hat{\beta}' & \\
\alpha \to \mathsf{0} \mid \mathsf{s}\beta & \beta \to \kappa\mathsf{0} & \hat{\alpha} \to \mathsf{0} \mid \mathsf{s}\hat{\beta} & \hat{\beta} \to \kappa^0\mathsf{0} \\
\kappa \to \lambda\alpha.\ \mathsf{M}\alpha\hat{\alpha} & \beta' \to \kappa(\mathsf{s}\beta) & \kappa^0 \to \lambda\hat{\alpha}.\ \mathsf{M}\alpha\hat{\alpha} & \hat{\beta}' \to \kappa^0(\mathsf{s}\hat{\beta})
\end{array}$$

In general, the production rules of the elementary proof grammar $\mathcal{E}_\pi$ may be cyclic and therefore permit infinite derivations. In the case of $\mathcal{E}_{\pi_\infty}$, for example, this is demonstrated by the sequence of non-terminals $\beta, \kappa, \hat{\alpha}, \hat{\beta}, \kappa^0, \alpha, \beta$.

To avoid enumerating unnecessary terms into the language of the grammar certain derivations should be disallowed. While it is possible to provide a characterisation of the derivations that yield Herbrand sets, the work is beyond the scope of this paper and will not be presented here. Instead, in the following section, we define acyclic proof grammars for which standard (rigid) derivations suffice. This is achieved by raising the types of non-terminals to make the necessary dependencies between non-terminals explicit.

4.2 Typed proof grammars

As before, let $\pi \vdash \Gamma$ be a simple Π_2-proof and Γ a set of closed prenex Σ_1 and Π_1 formulæ. The *(proof) grammar for* π is a rigid CFG denoted by $\mathcal{G}_\pi$. $\mathcal{G}_\pi = \langle N_\pi, R_\pi, \lhd_\pi, \Sigma_\pi, \sigma, \mathrm{Pr}_\pi \rangle$ has the same definition as the elementary grammar but with two essential differences:

1. Eigenvariables are no longer necessarily base-type symbols and their type depends on their relationship to other eigenvariables;
2. Production rules are modified accordingly by type-raising operations.

Types of non-terminals in $\mathcal{G}_\pi$

The type of a given non-terminal will be determined by the relation of its position in π to other eigenvariables in π. This relation will be defined as a well-founded ordering $\prec_\pi$ on the elements of N_π and formalises the (potential) dependency of one non-terminal on another. As well as fixing the type of a non-terminal, the ordering can be seen as the basis of the priority ordering $\lhd_\pi$ used in recognising rigid derivations.

For each position p in π, if $\pi|p$ has the form

$$\dfrac{\dfrac{\Pi', A[x/\alpha]}{\Pi', \forall x A}\,\forall \qquad \Delta, \exists x \bar{A}}{\pi|p \vdash \Pi, \Delta}\,\mathrm{cut}$$

we set $\alpha \prec_\pi \kappa^p$ and $\alpha \prec_\pi a$ for every $a \in \mathrm{EV}(\pi|p0) \cup \{\kappa^q \mid p0 \leq q\}$. Notice that for $a, b \in \mathrm{EV}(\pi)$, $a \prec_\pi b$ implies $b \lhd_\pi a$, and if we write $\prec_a$ for the set $\{b \in N_\pi \mid b \prec_\pi a\}$, $\prec_a$ is a set of eigenvariables linearly ordered by $\prec_\pi$. Moreover, for α appearing as above $\prec_{\kappa^p} = \prec_\alpha \cup \{\alpha\}$.

The type of a non-terminal is chosen to be its order-type in $\prec_\pi$. Let $k_a = |\prec_a|$. The *type* of $a \in N_\pi$ is that of a function over the base-type taking k_a arguments. So in particular σ is of base-type, as is any eigenvariable relating to a universal quantifier in the conclusion.

▶ **Example 8** (Types of non-terminals in π_∞). In $\mathcal{G}_{\pi_\infty}$ we have the same set of non-terminals symbols as in $\mathcal{E}_{\pi_\infty}$ but they are now assigned the following types. The ordering $\prec_\infty$ on N_∞ gives $\prec_\infty = \{(\alpha, \hat{\alpha}), (\hat{\alpha}, \kappa^0), (\alpha, \kappa^0), (\alpha, \hat{\beta}), (\alpha, \hat{\beta}'), (\alpha, \kappa^{\langle\rangle})\}$ so α, β, β' and σ all have base-type, $\hat{\alpha}$, $\hat{\beta}$, $\hat{\beta}'$ and $\kappa^{\langle\rangle}$ have type $\iota \to \iota$ and κ^0 has type $\iota \to \iota \to \iota$. Note, the ordering $\lhd_\infty$ specifying rigidity is unchanged from $\mathcal{E}_{\pi_\infty}$.

Production rules in $\mathcal{G}_\pi$

As mentioned earlier, the production rules of $\mathcal{G}_\pi$ have the form as in $\mathcal{E}_\pi$. We now explain how the change in the type of non-terminals is to be taken into account. This is achieved by means of a type-lifting operation that either lifts an occurrence of a non-terminal to its appropriate type or replaces it with a variable symbol in case it should be abstracted.

For each $\gamma \in \mathrm{EV}(\pi)$, let $\gamma_1 \prec_\pi \cdots \prec_\pi \gamma_{k_\gamma+1} = \gamma$ be the sequence of eigenvariables enumerating the elements of $\{\xi \mid \xi \preceq_\pi \gamma\}$. Also let $\{x_i\}$ be a fresh set of variable symbols.

Given $\alpha \in \mathrm{EV}(\pi) \cup \{\sigma\}$, we define an operation $S \mapsto S^\alpha$ on terms in $\Sigma_\pi \cup N_\pi$ that lifts the occurrence of non-terminals to their required type: we set $(ST)^\alpha = S^\alpha T^\alpha$, $(\lambda y.S)^\alpha = \lambda y.S^\alpha$, $c^\alpha = c$ for $c \in \Sigma_\pi$, and for $\gamma \in \mathrm{EV}(\pi) \cup \{\kappa^q \mid q \text{ a position in } \pi\}$,

$$\gamma^\alpha = \begin{cases} x_i, & \text{if } \gamma = \alpha_i \preceq_\pi \alpha, \\ \gamma\gamma_1^\alpha\gamma_2^\alpha \cdots \gamma_{k_\gamma}^\alpha, & \text{otherwise.} \end{cases}$$

The operation $S \mapsto S^\alpha$ replaces each non-terminal $\alpha_i \preceq_\pi \alpha$ by the variable x_i and to $\gamma \not\preceq_\pi \alpha$ makes explicit the dependence of γ on $\prec_\gamma$. For example, we have

$$\begin{aligned} \gamma^{\gamma_{k_\gamma}} &= \gamma x_1 x_2 \cdots x_{k_\gamma} \\ \gamma^\sigma &= \gamma\gamma_1(\gamma_2\gamma_1)(\gamma_3\gamma_1(\gamma_2\gamma_1)) \cdots (\gamma_{k_\gamma}\gamma_1(\gamma_2\gamma_1) \cdots (\gamma_{k_\gamma-1}\gamma_1 \cdots (\gamma_{k_\gamma-2}\gamma_1 \cdots))) \end{aligned}$$

Let p be a position in π, suppose $\pi|p$ has the form of either cuts in Figure 7. Then the grammar $\mathcal{G}_\pi$ includes the production rules

$$\alpha \to \lambda x_1 \cdots \lambda x_{k_\alpha}\, t^\alpha \qquad \beta \to \lambda x_1 \cdots \lambda x_{k_\beta}.\ \kappa^p x_1 \cdots x_{k_\alpha} t^\beta \qquad \kappa^p \to \lambda x_1 \cdots \lambda x_{k_\alpha+1}\, s^\alpha$$

Observe that $k_\alpha \le k_\beta$, so $\alpha_i = \beta_i$ for each $0 < i \le k_\alpha$ and the term in the central production rule is indeed closed. Also for α and κ^p above notice $k_{\kappa^p} = k_\alpha + 1$, hence the production rule for κ^p is well-typed. In the scenario pictured on the right side of Figure 7 in which the cut is performed on a Π_1 formula, naturally only the production rule for α is required.

Concerning the start symbol and formulæ in the conclusion Γ we add, in analogy to $\mathcal{E}_\pi$, production rules $\sigma \to \mathsf{F} s_0^\sigma \cdots s_k^\sigma$ and $\alpha_i \to \tau_{i,F}$ for appropriate formulæ F, terms $\langle s_i \rangle_{i \le k}$ and eigenvariable α_i. This completes the definition of $\mathcal{G}_\pi$.

▶ **Example 9** (Proof grammar for π_∞)**.** It is now possible to complete the definition of $\mathcal{G}_{\pi_\infty}$. As in Example 7 we will focus on parts of the grammar that are relevant to the formula T and the computation of its language, which we denote $\mathcal{L}(\mathcal{G}_\infty, T)$. The non-terminals and their types were described in the previous example. Notice for instance $\xi^\sigma = \xi$ for $\xi \in \{\alpha, \beta, \beta'\}$, $\xi^\sigma = \xi\alpha$ and $\xi^\alpha = \xi x_1$ for $\xi \in \{\kappa, \hat\alpha, \hat\beta, \hat\beta'\}$, and $(\mathsf{M}\alpha\hat\alpha)^{\hat\alpha} = \mathsf{M}x_1x_2$. The production rules of the grammar (relating to the formula T) are therefore

$$\begin{array}{llll} \sigma \to \mathsf{T}\beta\beta' & & \sigma \to \mathsf{T}(\hat\beta\alpha)(\hat\beta'\alpha) & \\ \alpha \to 0 \mid \mathsf{s}\beta & \beta \to \kappa 0 & \hat\alpha \to \lambda x.\ 0 \mid \lambda x.\ \mathsf{s}(\hat\beta x) & \hat\beta \to \lambda x.\ \kappa^0 x 0 \\ \kappa \to \lambda x.\ \mathsf{M}x(\hat\alpha x) & \beta' \to \kappa(\mathsf{s}\beta) & \kappa^0 \to \lambda x \lambda y.\ \mathsf{M}xy & \hat\beta' \to \lambda x.\ \kappa^0 x(\mathsf{s}(\hat\beta x)) \end{array}$$

The computation of the language of $\mathcal{L}(\mathcal{G}_\infty, T)$ is not complicated. Nevertheless, for space considerations it is necessary to make a few simplifications. In particular, in accordance with the informal proof presented in Section 2, various terms will be evaluated according to the intended semantics, so $\mathsf{s}S$ will be written as $S+1$ and $\mathsf{M}ST$ will be replaced by $\max\{S, T\}$. Also, $\mathsf{T}ST$ will be presented as $\mathsf{T}(S, T)$. In addition, implicit β-conversion of terms will be performed as this has no effect on rigidity of the considered derivations.

We begin by calculating the terms derivable from $\hat\beta$ and $\hat\beta'$ (with implicit β-conversion):

$$\hat\beta x \to \kappa^0 x 0 \to \mathsf{M}x0 = x \qquad \hat\beta' x \to^* \mathsf{M}x(\mathsf{s}(\hat\beta x)) \to^* \mathsf{M}x(\mathsf{s}(\mathsf{M}x0)) = x+1$$

Regarding $\hat\alpha$, modulo β-conversion it is easy to see we have $\hat\alpha x \to^* 0 \mid x+1$. Thus we can also compute the derivations starting from β, β' and α:

$$\begin{array}{ll} \beta \to^* \mathsf{M}0(\hat\alpha 0) \to^* 0 \mid 1 & \alpha \to^* 0 \mid 1 \mid 2 \\ \beta' \to^* \mathsf{M}(\mathsf{s}\beta)(\hat\alpha(\mathsf{s}\beta)) \to \beta+1 \mid \beta+2 & \end{array}$$

Thus $\mathsf{T}(\beta, \beta+1)$ and $\mathsf{T}(\beta, \beta+2)$ are the two terms derivable from $\mathsf{T}(\beta, \beta')$ without rewriting β. Combining these with the terms obtained from β above yields a total of eight derivations in the underlying non-rigid grammar. However, as $\kappa, \beta', \hat{\alpha}$ are the sole non-terminals used in deriving β and $\beta \not\lhd \kappa, \beta', \hat{\alpha}$, only four of the derivations are rigid, leaving

$$\sigma \to \mathsf{T}(\beta, \beta') \to^* \mathsf{T}(0,1) \mid \mathsf{T}(1,2) \mid \mathsf{T}(0,2) \mid \mathsf{T}(1,3)$$

Concerning the terms derivable from σ via $\mathsf{T}(\hat{\beta}\alpha, \hat{\beta}'\alpha)$, rigid derivations yield

$$\sigma \to \mathsf{T}(\hat{\beta}\alpha, \hat{\beta}'\alpha) \to^* \mathsf{T}(0,1) \mid \mathsf{T}(1,2) \mid \mathsf{T}(2,3)$$

Thus we conclude $\mathcal{L}(\mathcal{G}_\infty, T) = \{\mathsf{T}(0,1), \mathsf{T}(1,2), \mathsf{T}(2,3), \mathsf{T}(0,2), \mathsf{T}(1,3)\}$. The reader may check that $\Gamma \cup \Delta \cup \{T_{m,n} \mid \mathsf{T}(m,n) \in \mathcal{L}(\mathcal{G}_\infty, T)\}$ is derivable.

▶ **Theorem 10** (Language bound). *Let π be a simple Π_2-proof. The number of production rules in $\mathcal{G}_\pi$ is bounded by the number of quantifier inferences in π and $|\mathcal{L}(\mathcal{G}_\pi)| < 2^{2^{|\pi|}}$.*

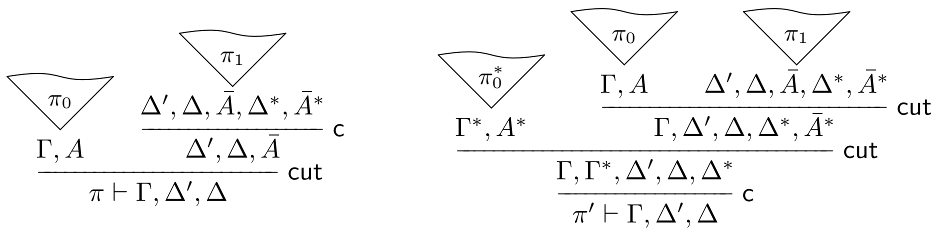
Proof. Let $\mathcal{G}' = \langle N_\pi, N_\pi, \lhd_\pi, \Sigma_\pi, \sigma, \mathrm{Pr}_\pi \rangle$ be the modification of $\mathcal{G}_\pi$ in which all non-terminals are marked as rigid. We observe $\mathcal{L}(\mathcal{G}') = \mathcal{L}(\mathcal{G}_\pi)$. Moreover, a study of paths in π reveals that $\mathcal{G}'$ is acyclic, whereby $|\mathcal{L}(\mathcal{G}_\pi)| \leq |\mathrm{Pr}_\pi|^{2^{|N_\pi|}}$ by Lemma 6. Let $k = \lfloor \frac{|\mathrm{Pr}_\pi|}{2} \rfloor$. Then $N_\pi + k < |\pi|$ and so $|\mathcal{L}(\mathcal{G}_\pi)| \leq 2^{2^{|N_\pi|+k}} < 2^{2^{|\pi|}}$ as required. ◀

5 Language containment

▶ **Lemma 11** (Local reduction). *If $\pi \rightsquigarrow \pi'$ is a local one-step cut reduction between regular simple Π_2-proofs then $\mathcal{L}(\mathcal{G}_{\pi'}) \subseteq \mathcal{L}(\mathcal{G}_\pi)$.*

Proof. We present the argument for two of the interesting cases, the case of Contraction Reduction and Quantifier Reduction; the remaining cases follow by a simple argument mirroring that of the former case.

Suppose, to begin, that the reduction $\pi \rightsquigarrow \pi'$ is an instance of Contraction Reduction and that the cut formula is principal in the contraction. Thus π and π' can be assumed to take the form given below, where $A = \forall x B$ is Π_2 and π_0^* is a renaming of π_0 so that π' is regular.


π_0
π_1
$\Delta', \Delta, \bar{A}, \Delta^*, \bar{A}^*$
c
Γ, A
$\Delta', \Delta, \bar{A}$
cut
$\pi \vdash \Gamma, \Delta', \Delta$
π_0^*
π_0
π_1
Γ, A
$\Delta', \Delta, \bar{A}, \Delta^*, \bar{A}^*$
cut
Γ^*, A^*
$\Gamma, \Delta', \Delta, \Delta^*, \bar{A}^*$
cut
$\Gamma, \Gamma^*, \Delta', \Delta, \Delta^*$
c
$\pi' \vdash \Gamma, \Delta', \Delta$

Note that we may assume the contraction occurs in the 'right' sub-proof as π is simple and A is a universal formula.

We argue that every rigid derivation in $\mathcal{G}_{\pi'}$ starting from σ can be transformed into a rigid derivation in $\mathcal{G}_\pi$ beginning at σ. Consider the function $f\colon N_{\pi'} \to N_\pi$ defined by

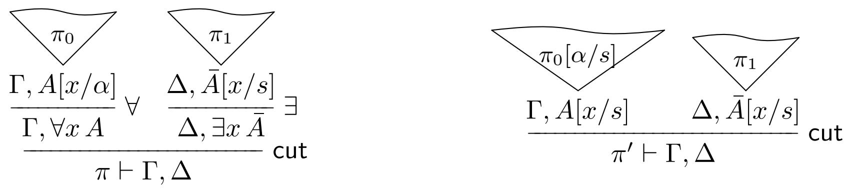
$$\begin{array}{lll}
f(\sigma) = \sigma & & f(\kappa^0) = f(\kappa^{01}) = \kappa^{\langle\rangle} \\
f(\gamma) = f(\gamma^*) = \gamma & \text{for } \gamma \in \mathrm{EV}(\pi_0) & f(\kappa^{00p}) = f(\kappa^{010p}) = \kappa^{0p} \\
f(\delta) = \delta & \text{for } \delta \in \mathrm{EV}(\pi_1) & f(\kappa^{011p}) = \kappa^{10p}
\end{array}$$

We observe that for all $a, b \in N_{\pi'}$ we have $a \prec_{\pi'} b$ iff $f(a) \prec_\pi f(b)$, thus f is type-preserving and uniquely extends to a function mapping terms in the language of $\Sigma_{\pi'} \cup N_{\pi'}$ to (well-typed) terms in $\Sigma_\pi \cup N_\pi$. Moreover, if $a \to S$ is a production rule in $\mathcal{G}_{\pi'}$, $f(a) \to f(S)$ is a production rule in $\mathcal{G}_\pi$. So f transforms derivations in the former grammar to derivations in the latter grammar; all that remains is to check the operation preserves rigidity.

Rigidity is not immediate as f is not injective. Indeed, let $d = \langle (a_i \to S_i), p_i \rangle_{i<\mathrm{lh}(d)} : \sigma \to S$ be a rigid derivation in $\mathcal{G}_{\pi'}$, let $d^f : \sigma \to f(S)$ be the derivation in $\mathcal{G}_\pi$ induced by f and suppose $j_0, j_1 < \mathrm{lh}(d)$ are such that $j_0 \not\sim_d j_1$ but $j_0 \sim_{d^f} j_1$, so $f(a_{j_0}) = f(a_{j_1}) \in R_{\mathcal{G}_\pi}$. Since f preserves the priority ordering, it follows that $a_{j_0} \neq a_{j_1}$ and so we may assume $a_{j_0} \in \mathrm{EV}(\pi_0)$ and $a_{j_1} \in \mathrm{EV}(\pi_0^*)$. But then d must utilise a production rule for a rigid non-terminal $\delta \in \mathrm{EV}(\pi_1)$ at a position q such that i) $q < p_{j_0}$ iff $q \not< p_{j_1}$ and ii) either $a_{j_0} \lhd \delta$ or $a_{j_1} \lhd \delta$, contradicting $j_0 \sim_{d^f} j_1$.

Before proceeding with the second case, we remark that in all the other local reduction steps, the natural choice of the 'renaming' function f is injective and preservation of rigidity is immediate.

Suppose now $\pi \rightsquigarrow \pi'$ is an instance of Quantifier Reduction. We may assume π and π' have the form below.



As in the previous case we define a function mapping rigid derivations in $\mathcal{G}_{\pi'}$ to rigid derivations in $\mathcal{G}_\pi$. Although every rigid non-terminal of $\mathcal{G}_\pi$ is a non-terminal in $\mathcal{G}_{\pi'}$, the non-terminals arising from $\mathrm{EV}(\pi_0)$ have a higher type than their counterpart in π' as we have $\alpha \prec_\pi \xi$ for every $\xi \in \mathrm{EV}(\pi_0)$. Notice, however, $k_\alpha = 0$ and $k_{\kappa^{\langle\rangle}} = 1$ in $\mathcal{G}_\pi$.

Let $d\colon \sigma \to S \in \mathcal{L}(\mathcal{G}_{\pi'})$ be a rigid derivation in $\mathcal{G}_{\pi'}$. As the formula $A[x/s]$ is Σ_1 it follows that d has one of following two forms.

In the first case, d is (up to simple renaming of non-terminals) a derivation in $\pi_0[\alpha/s]$ and $S \in \mathcal{L}(\mathcal{G}_{\pi_0[\alpha/s]})$. d induces a rigid derivation in $\mathcal{G}_{\pi_0}$ which when augmented by the production rule $\alpha \to s$ (present in $\mathcal{G}_\pi$) provides a derivation of S in $\mathcal{G}_\pi$.

In the second case, (a permutation of) d has the form $d_0 d_1 d_2$ where $d_0\colon \sigma \to S'$ is a derivation in $\mathcal{G}_{\pi_1}$; $d_1\colon S' \to S'[\beta/t^\beta]$ is a derivation using the single production rule $\beta \to t^\beta$ (note $k_\beta = 0$ in both $\mathcal{G}_{\pi'}$ and $\mathcal{G}_\pi$) where $\beta \in \mathrm{EV}(\pi_1)$ is the unique eigenvariable for the universal quantifier in $\bar{A}[x/s]$ (if there is one) in $\mathcal{G}_{\pi'}$; and $d_2\colon S'[\beta/t^\beta] \to S$ is a derivation in $\mathcal{G}_{\pi_0[\alpha/s]}$. We observe that the production rule $\beta \to t^\beta$ becomes, in $\mathcal{G}_\pi$, the derivation $\beta \to \kappa^{\langle\rangle} s \to (\lambda x_1 u^\alpha) s$ where $u[\alpha/s] = t$. Using these derivations in place of d_1 and making the appropriate modifications to the derivation d_2 yields a derivation $e\colon \sigma \to S''$ in $\mathcal{G}_\pi$ such that S'' β-reduces to S. In both cases rigidity is also easily checked. ◀

Using the previous lemma it is not difficult to establish that the language of proof grammars respect also non-local cut reductions, provided simplicity is maintained.

▶ **Theorem 12** (Global Reduction). *Suppose π and π' are regular simple Π_2-proofs such that π differs from π' only in its subproof at position p. If $\pi'|p \rightsquigarrow \pi|p$ then $\mathcal{L}(\mathcal{G}_\pi) \subseteq \mathcal{L}(\mathcal{G}_{\pi'})$.*

We now re-state and prove our main results from the introduction. The first of these is a restatement of Theorem 2 and follows from Theorem 12; the second is a generalisation of Theorem 1.

▶ **Theorem 13.** *Let $\langle\pi\rangle_{i\leq k}$ be a sequence of simple Π_2-proofs such that for each $i<k$, π_{i+1} is obtained by application of one of the reduction rules of Figure 4 to a sub-proof of π_i. Then for every term $T\in\mathcal{L}(\mathcal{G}_{\pi_k})$ there exists a term $S\in\mathcal{L}(\mathcal{G}_{\pi_0})$ that β-reduces to T.*

▶ **Theorem 14.** *Let $\pi\vdash\Gamma,\Delta$ be a Π_2-proof where $\Gamma\subseteq\Pi_1$ and $\Delta\subseteq\Sigma_1$ are sets of prenex formulæ and suppose $|\pi|$ denotes the number of inference rules occurring in π. There exists a totally rigid acyclic context-free tree grammar $\mathcal{G}$ such that i) $|\mathrm{Pr}_{\mathcal{G}}|\leq|\pi|$, ii) $|\mathcal{L}(\mathcal{G})|\leq 2^{2^{|\pi|}}$ and iii) there is a quantifier-free form of Γ, Γ', such that the formula*

$$\bigvee\Gamma'\vee\bigvee\{F[x_1/s_1,\dots,x_k/s_k]\mid(\exists x_1\cdots\exists x_kF)\in\Delta\wedge\mathsf{F}s_1\cdots s_k\in\mathcal{L}(\mathcal{G})\}\tag{1}$$

is a tautology.

Proof. By applying quantifiers inversion if necessary, π can be turned into a simple Π_2-proof π' without an increase in size. Let $\mathcal{G}=\langle N_{\pi'},N_{\pi'},\lhd_{\pi'},\Sigma_{\pi'},\sigma,\mathrm{Pr}_{\pi'}\rangle$ be the totally rigid grammar derived from $\mathcal{G}_{\pi'}$. Items (i) and (ii) follow from Theorem 10. Regarding (iii), let $\langle\pi_i\rangle_{i\leq k}$ be a reduction sequence of simple Π_2-proofs (for example any obtained from the standard cut elimination algorithms of [31, 32]) starting from $\pi_0=\pi'$ and leading to a cut-free proof π_k of Γ,Δ. By the previous theorem, $\mathcal{L}(\mathcal{G}_{\pi_k})\subseteq\mathcal{L}(\mathcal{G}_\pi)$.

Suppose Γ and Δ have the forms $\forall x_1\cdots\forall x_{k_0}G_0,\dots,\forall x_1\cdots\forall x_{k_m}G_m$ and $\exists x_1\cdots\exists x_{l_0}F_0$, $\dots$, $\exists x_1\cdots\exists x_{l_n}F_n$ respectively where $G_0,\dots,G_m,F_0\dots,F_n$ are quantifier-free. The Herbrand disjunction read from π_k is the formula

$$X=\bigvee_{j\leq m}G_j(\alpha_j^1,\dots,\alpha_j^{k_j})\vee\bigvee_{j\leq n}\bigvee\nolimits_{(s_1,\dots,s_{l_j})\in\mathcal{H}(\pi_k,F_j)}F_j(s_1,\dots,s_{l_j})$$

for appropriate choice of variables α_j^i. Since the formula in (1) is the result of replacing every α_j^i by τ_{i,G_j} in X we are done. ◀

6 Conclusion

This work provides an abstraction of proofs which focuses only on the aspects relevant to the extraction of Herbrand disjunctions. Compared to other approaches in the literature, including Herbrand nets [24], proof forests [10], expansion trees with cut [19] and functional interpretation [9], proof grammars offer a representation of Herbrand's theorem suitable for the following exploitation.

As carried out in [18], the result for Π_1-proofs can be strengthened to the following: let π' be any cut-free proof obtained from a Π_1-proof π via standard cut elimination, then $\mathcal{H}(\pi')\subseteq\mathcal{L}(\mathcal{G}_\pi)$. Moreover, if π' is obtained from π by non-erasing reduction (which corresponds to the λI-calculus, see [4, Section 9]) then we even have $\mathcal{H}(\pi')=\mathcal{L}(\mathcal{G}_\pi)$. Therefore all (possibly infinitely many) normal forms of the non-erasing reduction have the same Herbrand disjunction. This property of classical logic has been called *Herbrand-confluence* in [18] and provides a general way of defining the computational content of a classical proof in the sense that no witness is ruled out by a choice of reduction strategy. A deeper analysis of Π_2-proofs carried out in [1] has also yielded a Herbrand-confluence result analogous to [18].

A notable application of proof grammars is in *cut introduction*, which is motivated by the aim to structure and compress automatically generated analytic proofs. As shown in [17, 16], the arrows of Figure 2 can be inverted in the sense that a grammar can be computed from a Herbrand disjunction and that from such a grammar one can compute cut formulæ which realise the compression of the grammar. This method has been implemented and empirically evaluated with good results in [15]. An extension of these techniques to the case of proofs with Π_1-induction has led to a new technique for inductive theorem proving [8]. A natural continuation of the present work is to find an analogous characterisation for proofs with Π_2-induction as well as the development of techniques for the systematic introduction of Π_2 cuts.

An application of proof grammars to proof complexity consists in proving a lower bound on the length of proofs with cuts (which is notoriously difficult to control) by transferring a lower bound on the size of the corresponding grammar. This has been carried out for Π_1 cuts in [7] based on [14], and can potentially be extended to Π_2 cuts based on the results of this paper.

Acknowledgements. The authors' research was supported by the Wiener Wissenschafts-, Forschungs- und Technologiefonds (WWTF) project VRG12-04. The authors also wish to thank the anonymous referees for their helpful suggestions.

References

1 Bahareh Afshari, Stefan Hetzl, and Graham E. Leigh. On Herbrand confluence for first-order logic. Submitted, 2015.

2 Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Cut-Elimination: Experiments with CERES. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR) 2004*, volume 3452 of *Lecture Notes in Computer Science*, pages 481–495. Springer, 2005.

3 Franco Barbanera, Stefano Berardi, and Massimo Schivalocchi. "Classical" programming-with-proofs in λ_{PA}^{Sym}: An analysis of non-confluence. In Martín Abadi and Takayasu Ito, editors, *Theoretical Aspects of Computer Software*, volume 1281 of *Lecture Notes in Computer Science*, pages 365–390. Springer Berlin Heidelberg, 1997.

4 Hendrik Pieter Barendregt. *The Lambda Calculus*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1984.

5 Samuel R. Buss. On Herbrand's Theorem. In *Logic and Computational Complexity*, volume 960 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 1995.

6 Thierry Coquand. A semantics of evidence for classical arithmetic. *Journal of Symbolic Logic*, 60(1):325–337, 1995.

7 Sebastian Eberhard and Stefan Hetzl. Compressibility of finite languages by grammars. preprint, available at `http://www.logic.at/people/hetzl/research/`, 2015.

8 Sebastian Eberhard and Stefan Hetzl. Inductive theorem proving based on tree grammars. to appear in the Annals of Pure and Applied Logic, preprint available at `http://www.logic.at/people/hetzl/research/`, 2015.

9 Philipp Gerhardy and Ulrich Kohlenbach. Extracting Herbrand Disjunctions by Functional Interpretation. *Archive for Mathematical Logic*, 44:633–644, 2005.

10 Willem Heijltjes. Classical proof forestry. *Annals of Pure and Applied Logic*, 161(11):1346–1366, 2010.

11 Hugo Herbelin. *Séquents qu'on calcule.* PhD thesis, Université Paris 7, 1995.

12 Jacques Herbrand. *Recherches sur la théorie de la démonstration.* PhD thesis, Université de Paris, 1930.

13 Stefan Hetzl. On the form of witness terms. *Archive for Mathematical Logic*, 49(5):529–554, 2010.

14 Stefan Hetzl. Applying Tree Languages in Proof Theory. In Adrian-Horia Dediu and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications (LATA) 2012*, volume 7183 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2012.

15 Stefan Hetzl, Alexander Leitsch, Giselle Reis, Janos Tapolczai, and Daniel Weller. Introducing Quantified Cuts in Logic with Equality. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *Automated Reasoning - 7th International Joint Conference, IJCAR*, volume 8562 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2014.

16 Stefan Hetzl, Alexander Leitsch, Giselle Reis, and Daniel Weller. Algorithmic introduction of quantified cuts. *Theoretical Computer Science*, 549:1–16, 2014.

17 Stefan Hetzl, Alexander Leitsch, and Daniel Weller. Towards Algorithmic Cut-Introduction. In *Logic for Programming, Artificial Intelligence and Reasoning (LPAR-18)*, volume 7180 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2012.

18 Stefan Hetzl and Lutz Straßburger. Herbrand-Confluence for Cut-Elimination in Classical First-Order Logic. In Patrick Cégielski and Arnaud Durand, editors, *Computer Science Logic (CSL) 2012*, volume 16 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 320–334. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012.

19 Stefan Hetzl and Daniel Weller. Expansion trees with cut. preprint, available at `http://arxiv.org/abs/1308.0428`, 2013.

20 David Hilbert and Paul Bernays. *Grundlagen der Mathematik II.* Springer, 1939.

21 Florent Jacquemard, Francis Klay, and Camille Vacher. Rigid tree automata. In Adrian Horia Dediu, Armand-Mihai Ionescu, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications (LATA) 2009*, volume 5457 of *Lecture Notes in Computer Science*, pages 446–457. Springer, 2009.

22 Florent Jacquemard, Francis Klay, and Camille Vacher. Rigid tree automata and applications. *Information and Computation*, 209:486–512, 2011.

23 Markus Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.

24 Richard McKinley. Proof nets for Herbrand's Theorem. *ACM Transactions on Computational Logic*, 14(1):5:1–5:31, 2013.

25 Dale Miller. A Compact Representation of Proofs. *Studia Logica*, 46(4):347–370, 1987.

26 V.P. Orevkov. Lower bounds for increasing complexity of derivations after cut elimination. *Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta*, 88:137–161, 1979.

27 Pavel Pudlák. The Lengths of Proofs. In Sam Buss, editor, *Handbook of Proof Theory*, pages 547–637. Elsevier, 1998.

28 Diana Ratiu and Trifon Trifonov. Exploring the Computational Content of the Infinite Pigeonhole Principle. *Journal of Logic and Computation*, 22(2):329–350, 2012.

29 Monika Seisenberger. *On the Constructive Content of Proofs.* PhD thesis, Ludwig-Maximilians-Universität München, 2003.

30 Richard Statman. Lower bounds on Herbrand's theorem. *Proceedings of the American Mathematical Society*, 75:104–107, 1979.

31 G. Takeuti. *Proof Theory.* Dover Books on Mathematics Series. Dover Publications, Incorporated, 2013.

32 Anne S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2000.

33 Christian Urban. *Classical Logic and Computation.* PhD thesis, University of Cambridge, 2000.