Proofs as Tree Languages

Stefan Hetzl

Proofs as Tree Languages

Stefan Hetzl
Laboratoire Preuves, Programmes et Systèmes (PPS)
Université Paris Diderot – Paris 7
175 Rue du Chevaleret, 75013 Paris, France
stefan.hetzl@pps.jussieu.fr

August 4, 2011

Abstract

In this work a new connection between proof theory and formal language theory is established. Cut-elimination in a class of first-order proofs is characterised by a class of tree languages in the following sense: the proof with cuts corresponds to a formal grammar, the cut-free proof to the language of the grammar and the process of cut-elimination to the computation of the language from the grammar. It is shown how to read off a grammar from a proof and vice versa how to construct a proof from a grammar such that this correspondence holds.

1 Introduction

Cut-elimination is one of the central techniques of proof theory. The cut-elimination theorem has first been proved for classical and intuitionistic first-order logic by Gentzen in his seminal work [11] where also the sequent calculus has been introduced. Since then, it has been generalised to many logics and calculi. The importance of cut-elimination for computational logic has various roots: via the Curry-Howard correspondence, cut-elimination and the related technique of normalisation correspond to the evaluation of terms in functional programming languages. In formalised mathematical proofs, the elimination of cuts has the effect of removing lemmas yielding constructive information about the theorem such as witnesses or bounds. Furthermore, a cut-elimination theorem for a logic has the cut-free completeness as a corollary which typically allows for efficient proof search procedures.

Formal language theory plays an important role in computer science with applications in various domains ranging from the specification of data formats to computer-aided verification. Traditionally, a formal language is defined to be a set of words. This notion can be generalised to considering a formal language to be a set of trees. Such tree languages posses a rich theory and many applications, see e.g. [4]. One particular application that has received considerable attention recently is the use of tree grammars for the compression of XML documents, see [20] for a survey.

The purpose of this paper is to develop a new connection between these two subjects. The basis for this connection is the following relation between cut-free proofs and tree languages: Herbrand's theorem, in its most simple version, states that a formula $\exists x\, A$ with A quantifier-free is valid iff there are terms $t_1, \ldots, t_n$ s.t. $\bigvee_{i=1}^{n} A[x\backslash t_i]$ is a propositional tautology. Such a disjunction of instances which is a tautology is called *Herbrand-disjunction*. It is quite straightforward

to observe that from a cut-free proof it is easy to read off an Herbrand-disjunction and vice versa, how to construct a cut-free proof from an Herbrand-disjunction. Furthermore, different proofs may have the same Herbrand-disjunctions, so "having the same Herbrand-disjunction" is an equivalence relation on the set of cut-free proofs of a theorem. Moreover, this equivalence relation is natural as all the information of a proof which is mathematically interesting (in the sense of proof analyses such as [16, 17], see also [15]) is contained in its Herbrand-disjunction. An instance $A[x\backslash t_i]$ is quantifier-free and hence without binding constructs. Therefore it can be regarded as a tree over the signature compromising in addition to the first-order signature also the propositional connectives. Identifying an Herbrand-disjunction $\bigvee_{i=1}^{n} A[x\backslash t_i]$ with the *set of instances* $\{A[x\backslash t_i] \mid 1 \leq i \leq n\}$ shows that an Herbrand-disjunction, and therefore a natural equivalence class of cut-free proofs, is nothing but a *finite tree language*. Note that Herbrand-disjunctions as well as this line of reasoning can be generalised to cover more complicated quantifier-prefixes, non-prenex formulas as well as full higher-order logic by using e.g. the expansion trees of [18].

In this paper, we will extend this connection to the class of proofs whose cuts are restricted to containing at most one quantifier each. While this is a significant restriction of the expressive power of the cuts, a natural extension to full first-order logic is suggested in the conclusion. We will show that a proof with such cuts corresponds to (and asymptotically has the same length size as) a tree grammar and that the language of this grammar is an Herbrand-disjunction. Consequently, cut-elimination corresponds to the computation of the language of such a grammar. It will turn out that the suitable kind of tree languages are *rigid tree languages*, a notion that has been introduced in [13, 14] for the purpose of verification of cryptographic protocols. Furthermore, we will describe a class of rigid tree grammars which corresponds exactly to the considered class of proofs in the sense that also a translation in the other direction, i.e. from grammars to proofs, is possible. Hence these results show that the combinatorial mechanism underlying the compression power of cuts is that of a particular kind of tree grammar. This uncovers a certainly unexpected similarity between cut-elimination and techniques used for XML compression.

In Section 2 we will develop rigid tree grammars and prove them equivalent to the rigid tree automata of [13, 14] as well as make several useful observations about these languages. Section 3 contains the proof-theoretic preliminaries leading to the definition of the grammar of a proof. Sections 4 and 5 are devoted to proving the main result of this paper: the language of the grammar of a proof is an Herbrand-disjunction. In Section 6 we show how to translate grammars into proofs.

2 Rigid Tree Languages

Tree languages are a natural generalisation of string languages to the case of trees and constitute a central notion in the theory of formal languages [10, 4]. A feature which is important for applications but not present in regular tree languages is the ability to carry out equality tests between subterms, for instance to recognise patterns of the form $f(x, x)$. There are several classes of tree automata providing this ability: some allow to specify local equality constraints as side conditions of transition rules by giving term positions explicitly, see [4] for a survey, while others consider global constraints specified via states. An important class of the latter kind are *tree automata with global equalities and disequalities* (TAGED) [6, 7, 8]. For our purposes it will turn out to be natural to work with *rigid tree automata* (RTA) that have been introduced in [13, 14] and are a subclass of TAGED (characterised by having minimal equality and disequality relations). In a rigid tree automaton, certain states are designated as *rigid* thus

demanding that only one term may appear on the different positions of this state.

Definition 1. A *tree automaton* on a signature Σ is a tuple $\langle Q, F, \Delta\rangle$ where Q is a finite set of state symbols, $F \subset Q$ is the set of final states and Δ a set of transition rules of the form: $f(q_1, \ldots, q_n) \to q$ where $f \in \Sigma$ and $q, q_1, \ldots, q_n \in Q$.

A *rigid tree automaton* on Σ is a tuple $\langle Q, R, F, \Delta\rangle$ where $\langle Q, F, \Delta\rangle$ is a tree automaton and $R \subseteq Q$ is the set of rigid states.

As usual a position is a list of natural numbers, $\mathrm{Pos}(t)$ is the set of positions of the term t, ε is the empty (root) position and concatenation of positions p_1 and p_2 is written as $p_1.p_2$. We write $\mathrm{Pos}(x, t)$ for the set of positions of the symbol x in t and we write $x \in t$ if $\mathrm{Pos}(x, t) \neq \emptyset$. A run of a tree automaton on a term t is a function $r : \mathrm{Pos}(t) \to Q$ s.t. for all $f \in \Sigma$ and all $p \in \mathrm{Pos}(f, t)$: $f(r(p.1), \ldots, r(p.n)) \to r(p) \in \Delta$ for all $p \in \mathrm{Pos}(t)$. A run of a rigid tree automaton on t is a run of the underlying tree automaton satisfying the additional *rigidity condition*: for all $p_1, p_2 \in \mathrm{Pos}(t)$: if $r(p_1) = r(p_2) \in R$ then $t|_{p_1} = t|_{p_2}$. $\mathcal{T}(\Sigma)$ denotes the set of ground terms over a signature Σ. The language of an automaton A in a state q is denoted as $L(A, q)$ and defined as the set of $t \in \mathcal{T}(\Sigma)$ s.t. there exists a run r on t with $r(\varepsilon) = q$. The language of A is defined as $L(A) = \bigcup_{q \in F} L(A, q)$.

Example 1. Let $\Sigma = \{0/0, s/1\}$. A simple pumping argument shows that the language $L = \{f(t, t) \mid t \in \mathcal{T}(\Sigma)\}$ is not regular. On the other hand, L is recognised by the rigid tree automaton $\langle Q, R, F, \Delta\rangle$ where $Q = \{q, q_r, q_f\}$, $R = \{q_r\}$, $F = \{q_f\}$ and $\Delta = \{0 \to q, 0 \to q_r, s(q) \to q, s(q) \to q_r, f(q_r, q_r) \to q_f\}$.

Our goal in this paper is the description of the computation of an Herbrand-disjunction during the process of cut-elimination. For this aim it is considerably more natural (and technically advantageous) to adopt the generative viewpoint of a grammar on a language. We will therefore first develop rigid tree grammars and prove them to be equivalent to the above rigid tree automata.

Definition 2. A *regular tree grammar* is a tuple $\langle \alpha, N, T, P\rangle$ composed of an axiom α, a set N of non-terminal symbols with arity 0 and $\alpha \in N$, a set T of terminal symbols with $T \cap N = \emptyset$ and a set P of production rules of the form $\beta \to t$ where $\beta \in N$ and $t \in \mathcal{T}(T \cup N)$.

A *rigid tree grammar* is a tuple $\langle \alpha, N, R, T, P\rangle$ where $\langle \alpha, N, T, P\rangle$ is a regular tree grammar and $R \subseteq N$ is the set of rigid non-terminal symbols.

The derivation relation $\to_G$ of a regular tree grammar G is defined for $s, t \in \mathcal{T}(T \cup N)$ as $s \to_G t$ if there is a production rule $\beta \to u$ and a position p s.t. $s|_p = \beta$ and t is obtained from s by replacing β at p by u. A derivation of a term $t \in \mathcal{T}(T)$ in a regular tree grammar is a list of terms $t_1, \ldots, t_n \in \mathcal{T}(T \cup N)$ s.t. $t_1 = \alpha, t_n = t$ and $t_i \to_G t_{i+1}$ for $i = 1, \ldots, n-1$. A derivation of t in a rigid tree grammar is a derivation in the underlying regular tree grammar satisfying the additional *rigidity condition*: if $t_i \to_G t_{i+1}$ and $t_j \to_G t_{j+1}$ are applications of productions rules at positions p_i, p_j with the same left-hand side $\beta \in R$, then $t|_{p_i} = t|_{p_j}$. We will often omit the subscript G of the arrow if the grammar is clear from the context. The language of a tree grammar $L(G)$ is the set of $t \in \mathcal{T}(T)$ that are derivable in G. A production whose left-hand side is β will often be called β-production. A first basic but useful observation about rigid tree grammars is the following

Lemma 1. Let $G = \langle \alpha, N, R, T, P\rangle$ be a rigid tree grammar and let $t \in L(G)$. Then there is a derivation $\alpha \to \cdots \to t$ which uses at most one β-production for each $\beta \in R$.

Proof. Suppose both $\beta \to s_1$ and $\beta \to s_2$ are used at positions p_1 an p_2 respectively. Then by the rigidity condition $t|_{p_1} = t|_{p_2}$ and we can replace the derivation at p_2 by that at p_1 (or the other way round). This transformation does not violate the rigidity condition because it only copies existing parts of the derivation. □

It is a well-known result that regular tree grammars and tree automata are equivalent in the sense that they describe the same class of tree languages. We will now show the analogous result for the rigid versions. The proof essentially follows the lines of that in [4] for the regular case, however it is more involved due to the necessity of respecting the rigidity condition.

Definition 3. A grammar is called *normalised* if every production rule has the form $\gamma \to a$ or $\gamma \to f(\gamma_1, \ldots, \gamma_n)$ for $a, f \in T$ and $\gamma, \gamma_1, \ldots, \gamma_n \in N$.

For a term t, $P \subseteq \mathrm{Pos}(t)$ and a substitution σ we write $t \circ_P \sigma$ for the term obtained from application of σ to the subterms of t at the positions in P.

Lemma 2. If G is a rigid tree grammar, then there is a normalised rigid tree grammar G^* s.t. $L(G) = L(G^*)$.

Proof. Let G be a rigid tree grammar which is not normalised. W.l.o.g. G has a rigid axiom α which does not appear on the right-hand side of any production and G does not contain productions of the form $\beta \to \beta$. Let $\beta \to f(t_1, \ldots, t_n)$ be a production rule where at least one t_i is not a non-terminal symbol. Define a new rigid tree grammar G' by adding new non-rigid non-terminal symbols $\beta_1, \ldots, \beta_n$ and replacing the production rule $\beta \to f(t_1, \ldots, t_n)$ by $\beta \to f(\beta_1, \ldots, \beta_n), \beta_1 \to t_1, \ldots, \beta_n \to t_n$. Then every G-derivation can obviously be transformed to a G'-deviation preserving rigidity. Derivations can also be transformed in the other direction: let $\alpha = t_0 \to_{G'} \cdots \to_{G'} t_n = t$. As t does not contain β_i, for every application of $\beta \to f(\beta_1, \ldots, \beta_n)$ at a position p there must be an application of $\beta_i \to t_i$ at $p.i$ later in the derivation. On the other hand, as β_i is a new symbol every application of $\beta_i \to t_i$ at $p.i$ must be preceded by an application of $\beta \to f(\beta_1, \ldots, \beta_n)$. By permuting independent derivation steps one can join the corresponding rule applications thus collapsing a G'-derivation to a G-derivation – again – preserving rigidity. By iterating this process one eventually obtains a rigid tree grammar whose only non-normalised production rules are of the form $\beta \to \gamma$ for non-terminals β, γ. The rest of this proof will be concerned with the removal of these production rules, distinguishing cases based on rigidity of β and γ. Let the *weight* of a grammar $\mathrm{w}(G)$ be defined as the sum of the lengths of all possible derivations of the form $\beta_1 \to \cdots \to \beta_k$ where $\beta_i \in N$ and $\beta_i \neq \beta_j$ for $i \neq j$.

Let $G = \langle \alpha, N, R, T, P \rangle$ be a rigid tree grammar and $\beta \to \gamma$ a production with γ non-rigid, let $\gamma \to s_1, \ldots, \gamma \to s_k$ be all γ-productions and define $G' = \langle \alpha, N, R, T, P' \rangle$ where

$$P' = (P \setminus \{\beta \to \gamma\}) \cup \{\beta \to s_i \mid 1 \leq i \leq k, s_i \neq \beta\}.$$

Let $\alpha = t_0 \to_G \cdots \to_G t_n = t$. If $\beta \to \gamma$ is applied at some position $p \in \mathrm{Pos}(t)$, there must be an application of one of the $\gamma \to s_i$ at position p later and by permuting independent steps we can collapse $\beta \to \gamma \to s_i$ to $\beta \to s_i$ to obtain a G'-derivation of t. If some $\delta \in R$ is used at a certain position in this G'-derivation then it is used at the same position in the G-derivation so the rigidity condition is preserved. To obtain a G-derivation from a G'-derivation we replace applications of $\beta \to s_i$ by $\beta \to \gamma \to s_i$. The rigidity condition is preserved as γ is not rigid, hence $L(G) = L(G')$. Every derivation $\beta_1 \to \cdots \to \beta_k$ with $\beta_i \in N$ and $\beta_i \neq \beta_j$ for $i \neq j$ in G' has a corresponding (and longer) one in G obtained from replacing $\beta \to s_i$ by $\beta \to \gamma \to s_i$ but $\beta \to \gamma$ does no longer exist in G' so $\mathrm{w}(G') < \mathrm{w}(G)$.

If $\beta \to \gamma$ is a production rule where β is non-rigid let $\alpha_1 \to s_1, \ldots, \alpha_k \to s_k$ be all productions that contain β on their right-hand side and define G' by changing the productions P to

$$P' = (P \setminus \{\beta \to \gamma\}) \cup \{\alpha_i \to s_i \circ_P [\beta \backslash \gamma] \mid 1 \leq i \leq k, P \subseteq \mathrm{Pos}(\beta, s_i), s_i \circ_P [\beta \backslash \gamma] \neq \alpha_i\}.$$

Let $\alpha = t_0 \to \cdots \to t_n = t$ be a G-derivation. If $\beta \to \gamma$ is applied at a position p then (as $\beta \neq \alpha$ because β is non-rigid) a production $\alpha_i \to s_i$ at or above p is applied before it in the G-derivation. After permutation of independent steps, replace applications of $\alpha_i \to s_i$ and all applications of $\beta \to \gamma$ to this copy of s_i by an application of $\alpha_i \to s_i \circ_P [\beta \backslash \gamma]$ to obtain a G'-derivation. If a non-terminal δ is used at a certain position in the G'-derivation it is used at the same position in the G-derivation so the rigidity condition transfers. For the other direction one obtains a G-derivation from a G'-derivation from replacing $\alpha_i \to s_i \circ_P [\beta \backslash \gamma]$ by $\alpha_i \to s_i$ followed by copies of $\beta \to \gamma$. Now if $\delta \in R$ is used at a certain position in the G-derivation then $\delta \neq \beta$ by assumption so δ is used at the same position in the G'-derivation and rigidity transfers. As above it is easy to check that $\mathrm{w}(G') < \mathrm{w}(G)$.

These transformations decrease the weight, so we eventually arrive at a grammar G all of whose non-normalised productions are of the form $\beta \to \gamma$ for both β, γ rigid; let $\{\beta_i \to \gamma_i \mid 1 \leq i \leq m\}$ be these productions. If $\alpha = t_0 \to \cdots \to t_n = t$ is a G-derivation using $\beta \to \gamma$ then by Lemma 1 we can assume that it uses $\beta \to \gamma$ on all β-positions; let $\{p_1, \ldots, p_n\}$ be the set of β-positions of this derivation. The set $M \subseteq \{1, \ldots, m\}$ of indices of the $\beta_i \to \gamma_i$ used in the derivation is non-branching in the sense that it does not contain i, j with $\beta_i = \beta_j$ but $\gamma_i \neq \gamma_j$. Furthermore it is also acyclic in the sense that there are no $i_1, \ldots, i_k \in M$ s.t. $\gamma_{i_j} = \beta_{i_{j+1}}$ for $j \in \{1, \ldots, k-1\}$ and $\gamma_{i_k} = \beta_{i_1}$ because cycles can trivially be eliminated.

For such a non-branching and acyclic M denote with β^* the image of the non-terminal β under exhaustive application of the productions $\{\beta_i \to \gamma_i \mid i \in M\}$. This image is well-defined (by M being acyclic) and unique (by M being non-branching). Define $G_M = \langle \alpha, N', R', T, P' \rangle$ by

$$\begin{aligned} N' &= N \setminus \{\beta_i \mid i \in M\}, \\ R' &= N' \cap R, \text{ and} \\ P' &= (P \setminus \{\beta_i \to \gamma_i \mid 1 \leq i \leq m\})[\beta_i \backslash \beta_i^*]_{i \in M}. \end{aligned}$$

Let $\alpha = t_0 \to \cdots \to t_n = t$ be a G-derivation and M its acyclic and non-branching index set, then we obtain a G_M-derivation from replacing β_i by β_i^* everywhere hence removing the applications of the $\beta_i \to \gamma_i$. The rigidity condition is still fulfilled as using some $\delta \in R$ at a position in the G_M-derivation implies that it is used at the same position in the G-derivation. For the other direction, we translate a G_M-derivation to a G-derivation by inserting $\beta_i \to \gamma_i$-applications. The rigidity condition for the β_i is fulfilled because they are newly introduced at only the positions of the rigid β_i^* which ensure equality of the subterms at these positions.

Define G' as disjoint union of the G_M for all acyclic and non-branching M joined by a single axiom. Then G' is normalised and $L(G') = L(G)$. □

Theorem 1. A set of terms is language of a rigid tree grammar iff it is language of a rigid tree automaton.

Proof. It is straightforward to translate between normalised rigid tree grammars and rigid tree automata by reversing the direction of the arrows, translating non-terminal symbols as states and the axiom as final state. The result then follows from Lemma 2. □

Definition 4. A rigid tree grammar $\langle \alpha, N, R, T, P \rangle$ is called *totally rigid* if $N = R$.

Totally rigid tree grammars will simply be written as $\langle \alpha, R, T, P\rangle$. If $G = \langle \alpha, R, T, P\rangle$ is totally rigid and $t_0 = \alpha \to \cdots \to t_n = t \in \mathcal{T}(T)$ is a derivation in G we can by Lemma 1 assume that for every non-terminal at most one production rule is applied. A set S of production rules induces an order $<_S$ on the non-terminals as transitive closure of $\alpha <_S \beta$ if $\alpha \to t \in S$ and $\beta \in t$. Given a derivation in G, the order induced by its set S of production rules is acyclic. For suppose it would be cyclic, then either because it permits a derivation $\alpha \to \cdots \to \alpha$ which can be removed or because one of the form $\alpha \to \cdots \to s$ where $\alpha \neq s$ but $\alpha \in s$. But in this case the derivation cannot finish with a term built from terminals only due to the rigidity of α.

So while every single derivation induces an acyclic order, this is not necessarily the case for the whole set of productions of a grammar as can be seen in the simple example

$$\alpha \to h(\beta)|h(\gamma),\ \beta \to f(\gamma)|a,\ \gamma \to g(\beta)|b$$

by deriving $h(f(b))$ and $h(g(a))$.

Definition 5. A grammar is called *acyclic* if the order induced by the set of its productions is.

So an acyclic totally rigid grammar is a totally rigid grammar with a uniform order. Such grammars are central for this paper. They allow the following simple description of their language in terms of substitutions:

Lemma 3. If G is totally rigid and acyclic, then up to renaming of the non-terminals $G = \langle \alpha_0, \{\alpha_0, \ldots, \alpha_n\}, T, P\rangle$ with $L(G) = \{\alpha_0[\alpha_0 \backslash t_0] \cdots [\alpha_n \backslash t_n] \mid \alpha_i \to t_i \in P\}$.

Proof. Acyclicity permits a renaming of non-terminals s.t. $\alpha_i >_P \alpha_j$ implies $i > j$. The right-to-left inclusion is obvious. For the left-to-right inclusion, let $\alpha_0 = s_0 \to \ldots \to s_n = s \in \mathcal{T}(T)$ be a derivation in G. By Lemma 1 we can assume that for each i at most one production whose left-hand side is α_i is applied, say $\alpha_i \to t_i$. By acyclicity we can rearrange the derivation so that $\alpha_i \to t_i$ is only applied after $\alpha_j \to t_j$ if $i > j$. For those α_i which do not appear in the derivation we can insert any substitution without changing the final term so we obtain $s = \alpha_0[\alpha_0 \backslash t_0] \cdots [\alpha_n \backslash t_n]$. □

We finish this section on tree languages with basic observations that will be useful later on.

Lemma 4. If a rigid tree grammar G' is obtained from another rigid tree grammar G by deletion of production rules, then $L(G') \subseteq L(G)$.

Proof. Every G'-derivation is a G-derivation. □

Lemma 5. If a rigid tree grammar G' is obtained from another rigid tree grammar G by identifying two rigid non-terminals, then $L(G') \subseteq L(G)$.

Proof. Let β and γ be two non-terminals in G and call $(\beta\gamma)$ the new non-terminal in G' which replaces all occurrences of both β and γ. Let $\alpha \to \cdots \to t$ be a G'-derivation then by Lemma 1 we can assume that only one $(\beta\gamma)$-rule is used in it, say $(\beta\gamma) \to s$. If $s = (\beta\gamma)$ we obtain a G-derivation by dropping all applications of $(\beta\gamma) \to (\beta\gamma)$. If $s \neq (\beta\gamma)$, then $(\beta\gamma) \notin s$ as $(\beta\gamma) \in s$ would contradict rigidity. Then either $\beta \to s$ or $\gamma \to s$ must be a production in G so $\alpha \to \cdots \to t$ can be transformed to a derivation in G. □

Lemma 6. Let $G = \langle \alpha, N, R, T, P\rangle$ be a rigid tree grammar, $\beta \in R$, $\beta \to t_1, \ldots, \beta \to t_n \in P$, $\gamma \notin N$ and $G' = \langle \alpha, N \cup \{\gamma\}, R \cup \{\gamma\}, T, P'\rangle$ where $P' = (P \setminus \{\beta \to t_1, \ldots, \beta \to t_n\}) \cup \{\beta \to \gamma, \gamma \to t_1, \ldots, \gamma \to t_n\}$. Then $L(G) = L(G')$.

Proof. Let $t \in L(G)$ then by Lemma 1 there is a G-derivation $\alpha \to \cdots \to t$ which uses at most one β-production. If it does not use a β-production at all, then $t \in L(G')$, so assume it uses $\beta \to s$ as only β-production. If $s \in \{t_1, \ldots, t_n\}$, obtain a G'-derivation by replacing all applications of $\beta \to s$ by $\beta \to \gamma \to s$. The rigidity of γ in the G'-derivation follows from $\gamma \notin N$ and the rigidity of β in the G-derivation. For the other direction, let $t \in L(G')$, then by Lemma 1 there is a G'-derivation $\alpha \to \cdots \to t$ which uses at most one γ-production. If it does not use a γ-production, then $t \in L(G)$, so assume it uses $\gamma \to t_i$. As γ is only introduced by $\beta \to \gamma$ we can permute independent steps and collapse $\beta \to \gamma \to t_i$ to $\beta \to t_i$ thereby obtaining a G-derivation which satisfies the rigidity condition. □

3 The Grammar of a Proof

Definition 6. A sequent is a pair of multisets of formulas. A proof is a tree that starts with sequents of the form $A \to A$ for an atomic formula A and is built up using the following rules.

$$\frac{\Gamma \to \Delta, A \quad \Pi \to \Lambda, B}{\Gamma, \Pi \to \Delta, \Lambda, A \wedge B}\ \wedge_{\mathrm{r}} \qquad \frac{A, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta}\ \wedge_{\mathrm{l}_1} \qquad \frac{B, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta}\ \wedge_{\mathrm{l}_2}$$

$$\frac{A, \Gamma \to \Delta \quad B, \Pi \to \Lambda}{A \vee B, \Gamma, \Pi \to \Delta, \Lambda}\ \vee_{\mathrm{l}} \qquad \frac{\Gamma \to \Delta, A}{\Gamma \to \Delta, A \vee B}\ \vee_{\mathrm{r}_1} \qquad \frac{\Gamma \to \Delta, B}{\Gamma \to \Delta, A \vee B}\ \vee_{\mathrm{r}_2}$$

$$\frac{\Gamma \to \Delta, A}{\neg A, \Gamma \to \Delta}\ \neg_{\mathrm{l}} \qquad \frac{A, \Gamma \to \Delta}{\Gamma \to \Delta, \neg A}\ \neg_{\mathrm{r}}$$

$$\frac{A[x\backslash t], \Gamma \to \Delta}{\forall x\, A, \Gamma \to \Delta}\ \forall_{\mathrm{l}} \qquad \frac{\Gamma \to \Delta, A[x\backslash\alpha]}{\Gamma \to \Delta, \forall x\, A}\ \forall_{\mathrm{r}} \qquad \frac{A[x\backslash\alpha], \Gamma \to \Delta}{\exists x\, A, \Gamma \to \Delta}\ \exists_{\mathrm{l}} \qquad \frac{\Gamma \to \Delta, A[x\backslash t]}{\Gamma \to \Delta, \exists x\, A}\ \exists_{\mathrm{r}}$$

$$\frac{\Gamma \to \Delta}{A, \Gamma \to \Delta}\ \mathrm{w}_{\mathrm{l}} \qquad \frac{\Gamma \to \Delta}{\Gamma \to \Delta, A}\ \mathrm{w}_{\mathrm{r}} \qquad \frac{A, A, \Gamma \to \Delta}{A, \Gamma \to \Delta}\ \mathrm{c}_{\mathrm{l}} \qquad \frac{\Gamma \to \Delta, A, A}{\Gamma \to \Delta, A}\ \mathrm{c}_{\mathrm{r}}$$

$$\frac{\Gamma \to \Delta, A \quad A, \Pi \to \Lambda}{\Gamma, \Pi \to \Delta, \Lambda}\ \mathrm{cut}$$

The quantifier rules are subject to the usual conditions:

1. t must not contain a bound variable,
2. α is called *eigenvariable* and must not occur in $\Gamma \cup \Delta \cup \{A\}$ (eigenvariable condition).

We consider $A \supset B$ to be an abbreviation of $\neg A \vee B$ and also allow free use of corresponding rule abbreviations $\supset_{\mathrm{l}}$ and $\supset_{\mathrm{r}}$. A proof is called *regular* if different strong quantifier inferences have different eigenvariables. Positive universal and negative existential quantifiers are called *strong*, the others are called *weak*.

Definition 7. A proof π is called *simple* if

1. π is regular,
2. the end-sequent is of the form $\Gamma \to \exists x\, A$ where A is quantifier-free and Γ consists only of formulas of the form $\forall x_1 \cdots \forall x_n\, B$ where B is quantifier-free, and
3. every cut-formula in π contains at most one quantifier.

The regularity of a proof is a useful and harmless assumption in the context of cut-elimination. Also the second condition does not present a severe restriction; it is a standard result that strong quantifiers can be removed from formulas and proofs by Skolemisation. Prenexifying and possibly moving the formulas to the left then yields an end-sequent as demanded. The only of the above restrictions that substantially decreases the scope of the present analysis and justifies the terminology "simple" is the third one concerning the use of quantifiers in cuts. The results in this paper can be extended to a slightly more general class of proofs where the existential prefix in the end-sequent is a block of quantifiers and also cut-formulas are allowed to contain an arbitrary number of quantifiers as long they do not contain both a strong and a weak quantifier. The class of simple proofs has been chosen here for expositional clarity.

Definition 8. Let π be a simple proof of $\Gamma \to \exists x\, A$ and ψ a subproof of π. The *Herbrand-set* $\mathrm{H}(\psi, \pi)$ of ψ w.r.t. π is defined as follows. If ψ is an axiom, then $\mathrm{H}(\psi, \pi) = \emptyset$. If ψ is of the form

$$\dfrac{\begin{matrix}(\psi')\\ \Pi \to \Lambda, A[x\backslash t]\end{matrix}}{\Pi \to \Lambda, \exists x\, A}\ \exists_{\mathrm{r}}$$

where the main formula $\exists x\, A$ is ancestor of the formula $\exists x\, A$ in the end-sequent, then $\mathrm{H}(\psi, \pi) = \mathrm{H}(\psi', \pi) \cup \{A[x\backslash t]\}$. If ψ ends with any other unary inference and ψ' is its immediate subproof then $\mathrm{H}(\psi, \pi) = \mathrm{H}(\psi', \pi)$. If ψ ends with a binary rule and ψ_1, ψ_2 are its immediate subproofs, then $\mathrm{H}(\psi, \pi) = \mathrm{H}(\psi_1, \pi) \cup \mathrm{H}(\psi_2, \pi)$. We write $\mathrm{H}(\pi)$ for $\mathrm{H}(\pi, \pi)$.

Example 2. Define the axioms $A_1 = P(a) \vee P(b)$, $A_2 = \forall x\,(P(x) \supset Q(f(x)))$ and $A_3 = \forall x \forall y\,(P(x) \supset Q(y) \supset R(g(x, y)))$ and the simple proof $\pi =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}(\pi_1)\\ A_1 \to P(a), P(b)\end{matrix}}{A_1 \to \exists x P(x), P(b)}\ \exists_{\mathrm{r}}}{A_1 \to \exists x P(x), \exists x P(x)}\ \exists_{\mathrm{r}}}{A_1 \to \exists x P(x)}\ \mathrm{c}_{\mathrm{r}} \qquad \dfrac{\dfrac{\dfrac{\begin{matrix}(\pi_2)\\ P(\alpha), A_2 \to Q(f(\alpha))\end{matrix}}{P(\alpha), A_2 \to \exists x Q(x)}\ \exists_{\mathrm{r}} \qquad \dfrac{\dfrac{\begin{matrix}(\pi_3)\\ P(\alpha), Q(\beta), A_3 \to R(g(\alpha, \beta))\end{matrix}}{P(\alpha), Q(\beta), A_3 \to \exists x R(x)}\ \exists_{\mathrm{r}}}{P(\alpha), \exists x Q(x), A_3 \to \exists x R(x)}\ \exists_{\mathrm{l}}}{P(\alpha), A_2, A_3 \to \exists x R(x)}\ \mathrm{c}_{\mathrm{l}}, \mathrm{cut}}{\exists x P(x), A_2, A_3 \to \exists x R(x)}\ \exists_{\mathrm{l}}}{A_1, A_2, A_3 \to \exists x R(x)}\ \mathrm{cut}$$

where π_1, π_2 and π_3 are the obvious cut-free proofs. Then $\mathrm{H}(\pi) = \{R(g(\alpha, \beta))\}$.

Definition 9. Let π be a proof and Q be a quantifier occurrence in π. Define a set of terms $\mathrm{t}(Q)$ associated with Q as follows: if Q occurs in the main formula of a weakening, then $\mathrm{t}(Q) = \emptyset$. If Q is introduced by a quantifier inference from a term t or a variable x, then $\mathrm{t}(Q) = \{t\}$ or $\mathrm{t}(Q) = \{x\}$ respectively. If Q occurs in the main formula of a contraction and Q_1, Q_2 are the two corresponding quantifiers in the auxiliary formulas of the contraction, then $\mathrm{t}(Q) = \mathrm{t}(Q_1) \cup \mathrm{t}(Q_2)$. In all other cases Q has exactly one immediate ancestor Q' and $\mathrm{t}(Q) = \mathrm{t}(Q')$.

Let π be a simple proof and c be a cut in π that contains a quantifier. Write Q_c for the strong occurrence of the quantifier in one cut-formula of c and Q'_c for the corresponding weak occurrence in the other cut-formula. For C being the set of cuts in π which contain a quantifier, define the base substitutions of π as $\mathrm{B}(\pi) := \bigcup_{c \in C} \{[\alpha\backslash t] \mid \alpha \in \mathrm{t}(Q_c), t \in \mathrm{t}(Q'_c)\}$.

Example 3. The proof π of Example 2 has $\mathrm{B}(\pi) = \{[\alpha\backslash a], [\alpha\backslash b], [\beta\backslash f(\alpha)]\}$.

Definition 10. For a simple proof π define the totally rigid grammar $\mathrm{G}(\pi) = \langle \varphi, N, T, P \rangle$ by

$$\begin{aligned} N &= \{\varphi\} \cup \mathrm{EV}(\pi),\\ T &= \Sigma(\pi) \cup \{\wedge, \vee, \neg\}, \text{ and}\\ P &= \{\varphi \to F \mid F \in \mathrm{H}(\pi)\} \cup \{\alpha \to t \mid [\alpha\backslash t] \in \mathrm{B}(\pi)\}. \end{aligned}$$

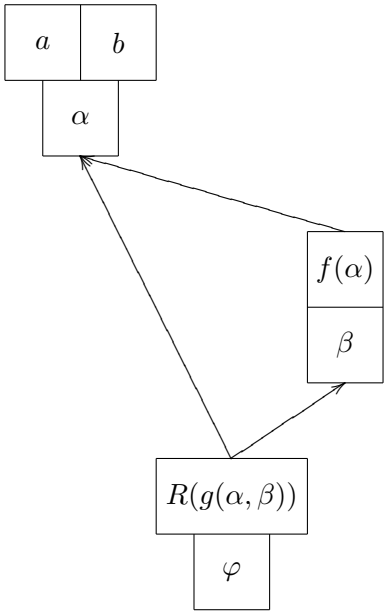

Figure 1: The Grammar of Example 4

where $\Sigma(\pi)$ is the signature of the proof π.

Let $|G|$ be the number of production rules of the grammar G and $|\pi|$ be the number of inferences in a proof π. From the above definition, it is easy to show that $|G| \leq |\pi|^2$. We will later (in Lemma 11) construct a G' with $|G'| \leq |\pi|$.

Example 4. The proof π of Example 2 has $\mathrm{G}(\pi) = \langle \varphi, N, T, P \rangle$ where $N = \{\varphi, \alpha, \beta\}$, $T = \{a, b, f, g, P, Q, R, \wedge, \vee, \neg\}$ and $P = \{\varphi \to R(g(\alpha, \beta)), \beta \to f(\alpha), \alpha \to a, \alpha \to b\}$, see also the diagram in Figure 1. The language of this grammar is $L(\mathrm{G}(\pi)) = \{R(g(a, f(a))), R(g(b, f(b)))\}$.

Lemma 7. If π is a simple proof, then $\mathrm{G}(\pi)$ is acyclic.

Proof. By induction on the number of cuts in the proof. The grammar of a cut-free proof is trivially acyclic. For the induction step, let ι be the lowest binary inference with subproofs π_1 and π_2 s.t. either 1. ι is a cut or 2. ι is not a cut but both π_1 and π_2 contain a cut. Let P_1 be the productions induced by the cuts in π_1, P_2 that induced by the cuts in π_2 and P that induced by cuts in π. In case 2, $<_P = <_{P_1} \cup <_{P_2}$ which is acyclic by induction hypothesis. In case 1, let P_ι be the productions induced by the cut ι, then $<^1_P = <^1_{P_1} \cup <^1_{P_2} \cup <^1_{P_\iota}$. By induction hypothesis, $<_{P_1}$ and $<_{P_2}$ are acyclic and as ι contains at most one quantifier, also $<_{P_\iota}$ is acyclic. Therefore, a cycle in $<_P$ must be of the form $\alpha_1 \leq_{P_1} \beta_1 <_{P_\iota} \alpha_2 \leq_{P_2} \beta_2 <_{P_\iota} \alpha_1$ where α_1, β_1 are eigenvariables of strong quantifier inferences in π_1 and α_2, β_2 of inferences in π_2. However, ι contains only one quantifier and depending on its polarity all substitutions in $\mathrm{B}(\iota)$ lead from π_1 to π_2 or from π_2 to π_1 but not both, so $<_P$ is acyclic. □

4 Some Algebraic Preliminaries

Before proving the main result, it will be useful to make some simple but rather abstract observations. We will work in a structure $(U, <^1, \sim)$ where $<^1$ is a binary relation on U and $\sim$

is an equivalence relation on U. The results of this section will only be applied to the situation $U = \mathrm{B}(\pi)$, $<^1$ being the one step scope order of Definition 13 and $[\alpha\backslash t] \sim [\beta\backslash s]$ iff $\alpha = \beta$. The author has found it useful to think about such structures in terms of diagrams such as the one depicted in Figure 1.

We use letters $X, Y, \ldots$ for subsets of U, letters $\mathfrak{X}, \mathfrak{X}_1, \ldots$ for subsets of the power set of U and the letters $C, D, \ldots$ for $\sim$-classes. We write $x <_X y$ if there are $x_1 = x, x_2, \ldots, x_n = y \in X$ with $x_i <^1 x_{i+1}$ for $1 \leq i < n$. We write $x < y$ for $x <_U y$. For $X \subseteq U$ define $X \uparrow x = \{y \in X \mid x <_X y\}$ which for the case $x \notin X$ entails $X \uparrow x = \emptyset$. For $X \subseteq U$ define $X^* = \{Y \subseteq X \mid$ for every $C \in {}^X/\!\sim$ there is exactly one $x \in C \cap Y\}$ which entails $\emptyset^* = \{\emptyset\}$. For a set $\mathfrak{X}$ of subsets of U we define $\mathfrak{X} \uparrow x := \{X \uparrow x \mid x \in X \in \mathfrak{X}\}$. We will often use sets of the form $X^* \uparrow x = \{X_0 \uparrow x \mid x \in X_0 \in X^*\}$.

Example 5. Let $U = \{[\alpha\backslash a], [\alpha\backslash b], [\beta\backslash f(\alpha)]\} = \mathrm{B}(\pi)$ where π is the proof of Example 2, let $[\beta\backslash f(\alpha)] <^1 [\alpha\backslash a]$ and $[\beta\backslash f(\alpha)] <^1 [\alpha\backslash b]$ be all pairs in the $<^1$-relation and let $[\alpha\backslash a] \sim [\alpha\backslash b]$ be the only pair in the equivalence relation $\sim$. Then $U^* \uparrow \alpha = \{\{[\alpha\backslash a]\}, \{[\alpha\backslash b]\}\}$ and $U^* \uparrow \beta = \{\{[\beta\backslash f(\alpha)], [\alpha\backslash a]\}, \{[\beta\backslash f(\alpha)], [\alpha\backslash b]\}\}$.

Lemma 8 (Monotonicity)**.** If $X_1 \subseteq X_2$, then $X_1 \uparrow x \subseteq X_2 \uparrow x$.

Proof. If $y \in X_1 \uparrow x$ then there are $x_1 = x, \ldots, x_n = y \in X_1$ with $x_i <^1 x_{i+1}$ and the x_i also being in X_2 we have $y \in X_2 \uparrow x$. □

X and Y are called class-disjoint if for every $C \in {}^U/\!\sim$ we have $X \cap C = \emptyset$ or $Y \cap C = \emptyset$.

Lemma 9 (Context)**.** If X, Y are class-disjoint and $x \in X$ with $(X \cup Y) \uparrow x = X \uparrow x$ then $(X \cup Y)^* \uparrow x = X^* \uparrow x$.

Proof. First note that due to class-disjointness we have

$$(X \cup Y)^* \uparrow x = \{(X_0 \cup Y_0) \uparrow x \mid x \in X_0 \cup Y_0, X_0 \in X^*, Y_0 \in Y^*\}$$

but $(X \cup Y) \uparrow x = X \uparrow x$ implies $(X_0 \cup Y_0) \uparrow x = X_0 \uparrow x$ for all $X_0 \subseteq X, Y_0 \subseteq Y$ and therefore

$$= \{X_0 \uparrow x \mid x \in X_0 \in X^*\} = X^* \uparrow x.$$

□

For $X, Y \subseteq U$ we write $Y <^1 X$ if there are $x \in X, y \in Y$ s.t. $y <^1 x$ and $Y \not<^1 X$ if this is not the case. For sets X, Y write $X \uparrow Y$ for $\bigcup_{y \in Y} X \uparrow y$.

Definition 11. Let $X, Y \subseteq U$, a class $C \in {}^Y/\!\sim$ is called *entrance of Y for X* if i) $x <^1 z$ for all $x \in X$ and all $z \in C$ and ii) for all $x \in X, y \in Y$ with $x <_{X \cup Y} y$ there is a $z \in C$ s.t. $x <^1 z <_Y y$.

Lemma 10 (Splitting)**.** Let X, Y be class-disjoint with $Y \not<^1 X$, let $C \in {}^Y/\!\sim$ be the entrance of Y for X and $D \in {}^X/\!\sim$, let $X_0 \in X^*$ and $Y_0 \in Y^*$, then

$$(X_0 \cup Y_0) \uparrow D = X_0 \uparrow D \cup Y_0 \uparrow C.$$

Proof. As $D \in {}^X/\!\sim$ and $X_0 \in X^*$ we have $D \cap X_0 = \{x_0\} = D \cap (X_0 \cup Y_0)$ for some x_0, so it suffices to show

$$(X_0 \cup Y_0) \uparrow x_0 = X_0 \uparrow x_0 \cup Y_0 \uparrow C.$$

For the left-to-right inclusion let $z \in (X_0 \cup Y_0) \uparrow x_0$. If $z \in X_0$ then $x_0 <_{X_0} z$ because $Y \not<^1 X$ so $z \in X_0 \uparrow x_0$. If $z \in Y_0$ then $x <_{X_0 \cup Y_0} z$ implies $z \in Y_0 \uparrow C$ because C is entrance of Y for X. For the other direction, let $x \in X_0 \uparrow x_0$, then $x \in (X_0 \cup Y_0) \uparrow x_0$ by monotonicity. For $y \in Y_0 \uparrow C$ we have $z \in C$ with $z <_{Y_0} y$ and as C is entrance of Y for X we obtain $x_0 <^1 z <_{Y_0} y$. □

5 From Proofs to Tree Languages

The grammar of a proof is invariant under many proof transformations including most local cut-reduction rules. A notable exception are the duplications of proofs induced by the removal of contractions: such a transformation duplicates parts of the grammar. However – and this is the technical key point of this paper – this transformation can be done in a way which does not change the *language* of the grammar. To achieve this we will exercise a tight control over the removal of contractions by imposing an uppermost reduction strategy as well as a particular normal form for proofs (also for those containing cuts).

This normal form for simple proofs requires the cuts to be prenex. The transformation to prenex cuts will partially be the one used in [2] on first-order proofs as well as by [19, Theorem 5.13] in quantified propositional calculus, see also [5, Theorem VII.4.7]. In order to control the changes of the language of the grammar induced by this transformation we introduce a new dummy constant d which does not appear in the original proof and we write $\mathrm{G}_{\mathrm{nd}}(\pi)$ for $\mathrm{G}(\pi)$ without productions of the form $\beta \to d$.

In a proof where all formulas are prenex and without quantifier alternations, a weak quantifier inference ι can be shifted downwards until either: it reaches the premise sequent of a cut or the end-sequent where the quantifier is introduced, or: it reaches the premise sequent of a strong quantifier inference whose eigenvariable appears in the term of ι. Such a sequent will be called *blocking sequent for* ι.

Definition 12. A proof π is said to be in *normal form* if the following conditions are fulfilled:

1. π is simple.
2. The dummy constant d appears only in a context of the form

$$\dfrac{\dfrac{\Gamma \to \Delta, A}{\Gamma \to \Delta, A \vee B[x\backslash d]}\,\vee_{\mathrm{r}_1}}{\Gamma \to \Delta, \exists x\,(A \vee B)}\,\exists_{\mathrm{r}} \qquad \text{or} \qquad \dfrac{\dfrac{\Gamma \to \Delta, A}{\Gamma \to \Delta, B[x\backslash d] \vee A}\,\vee_{\mathrm{r}_2}}{\Gamma \to \Delta, \exists x\,(B \vee A)}\,\exists_{\mathrm{r}}$$

   where $x \notin A$ and $\exists x\,(A \vee B)$ or $\exists x\,(B \vee A)$ respectively is ancestor of a cut-formula.
3. For every $\exists_{\mathrm{r}}$-inference ι: the only inferences between ι and its blocking sequent are other $\exists_{\mathrm{r}}$-inferences, contractions and $\vee_{\mathrm{r}}$-inferences deleting d.
4. All cut-formulas are of the form $\exists x\, A$ for some quantifier-free A and appear in a context of the form

$$\dfrac{\begin{array}{cc}(\psi_1) & \\ \Gamma \to \Delta, \exists x\, A & \dfrac{\begin{array}{c}(\psi_2)\\ A[x\backslash\alpha], \Pi \to \Lambda\end{array}}{\exists x\, A, \Pi \to \Lambda}\,\exists_{\mathrm{l}}\end{array}}{\Gamma, \Pi \to \Delta, \Lambda}\,\mathrm{cut}$$

   where ψ_1 contains at least one $\exists_{\mathrm{r}}$-inference whose main formula is ancestor of $\exists x\, A$ in the cut and two different such $\exists_{\mathrm{r}}$-inferences introduce different terms,

Lemma 11 (normal form)**.** If π is a simple proof then there is a proof π' in normal form and a totally rigid acyclic grammar G' with $|G'| \leq |\pi|$ and $L(\mathrm{G}_{\mathrm{nd}}(\pi')) = L(G') \subseteq L(\mathrm{G}(\pi))$.

Proof. For showing this lemma we will consider proofs using the inference rule

$$\frac{\Gamma \to \Delta, A \quad \exists x\, B, \Pi \to \Delta}{\Gamma, \Pi \to \Delta, \Lambda}\ \mathrm{p-cut}$$

where B is quantifier-free and A can be transformed to $\exists x\, B$ using the usual quantifier shifting rules. This inference rule is sound and the definition of the base substitutions extends naturally to simple proofs containing p-cuts because each p-cut still has one weak and one strong occurrence of a quantifier. Furthermore we use the notation $|\pi|_{\mathrm{w}}$ for the number of weak quantifier inferences in π.

First, define π_1 from π by shifting down weakenings as far as possible removing pairs of connected weakenings and contractions and deleting subproofs of binary inferences as in the replacement of

$$\dfrac{\dfrac{\overset{(\psi_1)}{\Gamma \to \Delta}}{\Gamma \to \Delta, A}\ \mathrm{w_r} \quad \overset{(\psi_2)}{A, \Pi \to \Lambda}}{\Gamma, \Pi \to \Delta, \Lambda}\ \mathrm{cut} \qquad \text{by} \qquad \dfrac{\overset{(\psi_1)}{\Gamma \to \Delta}}{\Gamma, \Pi \to \Delta, \Lambda}\ \mathrm{w^*}\ .$$

Then all weakenings in π_1 appear in a block above the end-sequent, $|\pi_1|_{\mathrm{w}} \leq |\pi|_{\mathrm{w}}$ and $L(\mathrm{G}(\pi_1)) \subseteq L(\mathrm{G}(\pi))$ by Lemma 4.

Each cut that contains a quantifier has a strong side (the side of the occurrence of the strong quantifier) and a weak side (defined analogously). Define π_2 from π_1 by replacing each cut

$$\frac{\overset{(\psi_1)}{\Gamma \to \Delta, A} \quad \overset{(\psi_2)}{A, \Pi \to \Lambda}}{\Gamma, \Pi \to \Delta, \Lambda}\ \mathrm{cut}$$

whose strong side is ψ_1 by

$$\dfrac{\dfrac{\overset{(\psi_2)}{A, \Pi \to \Lambda}}{\Pi \to \Lambda, \neg A}\ \neg_{\mathrm{r}} \quad \dfrac{\overset{(\psi_1)}{\Gamma \to \Delta, A}}{\neg A, \Gamma \to \Delta}\ \neg_{\mathrm{l}}}{\Gamma, \Pi \to \Delta, \Lambda}\ \mathrm{cut}\ .$$

Then $|\pi_2|_{\mathrm{w}} = |\pi_1|_{\mathrm{w}}$, $\mathrm{G}(\pi_2) = \mathrm{G}(\pi_1)$ and in π_2 the strong side of every cut is the right one.

Define π_3 from π_2 by replacing each cut

$$\frac{\overset{(\psi_1)}{\Gamma \to \Delta, A} \quad \overset{(\psi_2)}{A, \Pi \to \Lambda}}{\Gamma, \Pi \to \Delta, \Lambda}\ \mathrm{cut}$$

by

$$\frac{\overset{(\psi_1)}{\Gamma \to \Delta, A} \quad \dfrac{\overset{(\psi_2')}{B[x\backslash\alpha], \Pi \to \Delta}}{\exists x\, B, \Pi \to \Delta}\ \exists_{\mathrm{l}}}{\Gamma, \Pi \to \Delta, \Lambda}\ \mathrm{p-cut}$$

where ψ_2' is obtained from ψ_2 by leaving out all strong quantifier inferences that introduce the quantifier in A and identifying their eigenvariables. Then $|\pi_3|_{\mathrm{w}} = |\pi_2|_{\mathrm{w}}$ and by Lemma 5: $L(\mathrm{G}(\pi_3)) \subseteq L(\mathrm{G}(\pi_2))$. Defining $G' = \mathrm{G}(\pi_3)$ we obtain $|G'| \leq |\pi_3|_{\mathrm{w}} \leq |\pi|$ and $L(G') \subseteq L(\mathrm{G}(\pi))$.

It remains to construct π'. To that aim, we will first define π_4 from π_3 by prenexifying the weak side of a cut through the introduction of additional cuts. Apply the following transformation top-down to the whole proof: Suppose $Qx\,A$ is a ancestor of a cut-formula on the weak side of the cut and auxiliary formula of an inference that is about to destroy its prenex property, for example $Qx = \exists x$ and the inference being $\neg_\mathrm{l}$,

$$\dfrac{\overset{(\psi)}{\Gamma \to \Delta, \exists x\, A}}{\neg\exists x\, A, \Gamma \to \Delta}\ \neg_\mathrm{l}\,,$$

then replace the above inference by

$$\dfrac{\overset{(\psi)}{\Gamma \to \Delta, \exists x\, A} \qquad \dfrac{\dfrac{\dfrac{\overset{(\eta)}{A[x\backslash\alpha] \to A[x\backslash\alpha]}}{A[x\backslash\alpha], \neg A[x\backslash\alpha] \to}\ \neg_\mathrm{l}}{A[x\backslash\alpha], \forall x\, \neg A \to}\ \forall_\mathrm{l}}{\exists x\, A, \forall x\, \neg A \to}\ \exists_\mathrm{l}}{\forall x\, \neg A, \Gamma \to \Delta}\ \mathrm{cut}$$

where α is a new eigenvariable. Similarly, replace inferences of the form

$$\dfrac{\overset{(\psi)}{\Gamma \to \Delta, \exists x\, A}}{\Gamma \to \Delta, \exists x\, A \vee B}\ \vee_{\mathrm{r}_1} \qquad \text{by} \qquad \dfrac{\overset{(\psi)}{\Gamma \to \Delta, \exists x\, A} \qquad \dfrac{\dfrac{\dfrac{\overset{(\eta)}{A[x\backslash\alpha] \to A[x\backslash\alpha]}}{A[x\backslash\alpha] \to A[x\backslash\alpha] \vee B}\ \vee_{\mathrm{r}_1}}{A[x\backslash\alpha] \to \exists x\, (A \vee B)}\ \exists_\mathrm{r}}{\exists x\, A \to \exists x\, (A \vee B)}\ \exists_\mathrm{l}}{\Gamma \to \Delta, \exists x\, (A \vee B)}\ \mathrm{cut} \qquad \text{and}$$

$$\dfrac{\overset{(\psi_1)}{\Gamma \to \Delta, \exists x\, A} \qquad \overset{(\psi_2)}{\Pi \to \Lambda, B}}{\Gamma, \Pi \to \Delta, \Lambda, \exists x\, A \wedge B}\ \wedge_\mathrm{r} \quad \text{by} \quad \dfrac{\overset{(\psi_2)}{\Pi \to \Lambda, B} \qquad \dfrac{\overset{(\psi_1)}{\Gamma \to \Delta, \exists x\, A} \qquad \dfrac{\dfrac{\dfrac{\overset{(\eta)}{A[x\backslash\alpha] \to A[x\backslash\alpha]} \qquad \overset{(\eta)}{B \to B}}{A[x\backslash\alpha], B \to A[x\backslash\alpha] \wedge B}\ \wedge_\mathrm{r}}{A[x\backslash\alpha], B \to \exists x\, (A \wedge B)}\ \exists_\mathrm{r}}{\exists x\, A, B \to \exists x\, (A \wedge B)}\ \exists_\mathrm{l}}{B, \Gamma \to \Delta, \exists x\, (A \wedge B)}\ \mathrm{cut}}{\Gamma, \Pi \to \Delta, \Lambda, \exists x\, (A \wedge B)}\ \mathrm{cut}\,.$$

Such a transformation replaces a set of production rules $\{\beta \to t_1, \dots \beta \to t_n\}$ by $\{\beta \to \alpha, \alpha \to t_1, \dots \alpha \to t_n\}$ where α is a new non-terminal, so by Lemma 6 the language is not changed. The symmetric cases for $Qx = \forall x$ are treated analogously.

Furthermore, replace inferences of the form

$$\dfrac{\overset{(\psi)}{\Gamma \to \Delta, B}}{\Gamma \to \Delta, \exists x\, A \vee B}\ \vee_{\mathrm{r}_2} \qquad \text{by} \qquad \dfrac{\overset{(\psi)}{\Gamma \to \Delta, B} \qquad \dfrac{\dfrac{\overset{(\eta)}{B \to B}}{B \to A[x\backslash d] \vee B}\ \vee_{\mathrm{r}_2}}{B \to \exists x\, (A \vee B)}\ \exists_\mathrm{r}}{\Gamma \to \Delta, \exists x\, (A \vee B)}\ \mathrm{cut}\,.$$

This transformation introduces productions of the form $\beta \to d$ which leave G_nd unchanged. Again, the symmetric case for the universal quantifier is treated analogously. So we have obtained a proof π_4 with $L(\mathrm{G}_\mathrm{nd}(\pi_4)) = L(G')$.

Finally, we shift down $\exists_{\mathrm{r}}$-inferences together with contractions having their main formulas as auxiliary formulas and with the corresponding $\vee_{\mathrm{r}}$-inferences that delete d until there are no other inferences between the $\exists_{\mathrm{r}}$-inference and its blocking sequent. If one quantifier is introduced from the same term t two times we replace

$$\dfrac{\dfrac{\dfrac{(\psi)}{\Gamma \to \Delta, A[x\backslash t], A[x\backslash t]}\,\exists_{\mathrm{r}}}{\Gamma \to \Delta, A[x\backslash t], \exists x\, A}\,\exists_{\mathrm{r}}}{\Gamma \to \Delta, \exists x\, A, \exists x\, A}\,\mathrm{c}_{\mathrm{r}} \qquad \text{by} \qquad \dfrac{\dfrac{(\psi)}{\Gamma \to \Delta, A[x\backslash t], A[x\backslash t]}\,\mathrm{c}_{\mathrm{r}}}{\Gamma \to \Delta, A[x\backslash t]}\,\exists_{\mathrm{r}}}{\Gamma \to \Delta, \exists x\, A}\,\exists_{\mathrm{r}} \quad .$$

The result of these transformations is a proof π' in normal form having $L(\mathrm{G}_{\mathrm{nd}}(\pi')) = L(G')$. $\square$

The proof transformations used for showing the above lemma illustrate an important point: on the level of the grammar one can carry out transformations, in particular simplifications, which are impossible on the level of the proof (i.e. they necessitate tricks like p-cuts and dummy constants). This is due to the fact that the equality of languages is sufficient for validity and local soundness criteria as in a formal proof are no longer needed. The small grammar G' can neither be read off from π nor from π'; both of them have larger grammars.

A proof in normal form exhibits a one-to-one correspondence between cuts and eigenvariables. Consequently, for an eigenvariable α we write c_α for the cut corresponding to it. The set of eigenvariables of a proof π is denoted as $\mathrm{EV}(\pi)$. For a proof π in normal form and $\alpha \in \mathrm{EV}(\pi)$, define $\mathrm{EV}_{\hat{\mathrm{L}}}(\pi, \alpha) = \{\beta \in \mathrm{EV}(\pi) \mid c_\beta \text{ is on the left above } c_\alpha\}$ and $\mathrm{EV}_{\mathrm{L}}(\pi, \alpha) = \{\alpha\} \cup \mathrm{EV}_{\hat{\mathrm{L}}}(\pi, \alpha)$. Define $\mathrm{EV}_{\mathrm{R}}(\pi, \alpha)$ and $\mathrm{EV}_{\hat{\mathrm{R}}}(\pi, \alpha)$ analogously for the right side of c_α. Given a set V of variables and a set S of unary substitutions, define $S_V = \{\sigma \in S \mid \mathrm{dom}(\sigma) \subseteq V\}$. Write $\mathrm{B}_V(\pi)$ for $(\mathrm{B}(\pi))_V$ and define $\mathrm{B}_X(\pi, \alpha) = \mathrm{B}_{\mathrm{EV}_X(\pi,\alpha)}(\pi)$ for $X \in \{\mathrm{L}, \mathrm{R}, \hat{\mathrm{L}}, \hat{\mathrm{R}}\}$.

During the elimination of contractions, proof parts are duplicated and the eigenvariables introduced there are copied hence renamed. The final value of an eigenvariable develops gradually through such renamings. It will turn out to be very useful to *name an eigenvariable by the current state of development of its final value*. To that aim, we label an eigenvariable α of the original proof π by a set of substitutions $S \subseteq \mathrm{B}(\pi)$ which is denoted as α_S. By convention we identify α and $\alpha_\emptyset$. An indexed variable α_k with label S is written as $\alpha_{k,S}$.

A variable-renaming is a substitution whose range contains only variables. For a set of variables V, a variable-renaming ρ is called V-fresh if $\mathrm{rge}(\rho) \cap V = \emptyset$. For a substitution σ, a variable-renaming ρ is called σ-fresh if $\mathrm{rge}(\rho) \cap (\mathrm{dom}(\sigma) \cup \mathrm{rge}(\sigma)) = \emptyset$ and for a set S of substitutions ρ is called S-fresh if it is σ-fresh for all $\sigma \in S$. Define $S^R = \{[x\rho\backslash t\rho] \mid [x\backslash t] \in S, \rho \in R\}$ for a set of S-fresh variable renamings R and a set of substitutions S. For a variable-renaming ρ we write $\mathrm{B}^\rho(\pi)$ for $(\mathrm{B}(\pi))^\rho$. For a variable-renaming ρ, a set of unary substitutions S and a set of variables V we write S^ρ_V for $(S_V)^\rho$, so $\mathrm{B}^\rho_V(\pi)$ is $((\mathrm{B}(\pi))_V)^\rho$.

Definition 13 (scope order)**.** For a proof π in normal form and $[\alpha\backslash t], [\beta\backslash s] \in \mathrm{B}(\pi)$ we write $[\alpha\backslash t] <^{\mathrm{o}^1} [\beta\backslash s]$ if the $\exists_{\mathrm{r}}$-inference of t is above the $\exists_{\mathrm{l}}$-inference of β. We furthermore write $<^{\mathrm{o}}$ for the transitive and $\leq^{\mathrm{o}}$ for the reflexive and transitive closure of $<^{\mathrm{o}^1}$.

Note that $[\alpha\backslash t] <^{\mathrm{o}} [\beta\backslash s_1]$ iff $[\alpha\backslash t] <^{\mathrm{o}} [\beta\backslash s_2]$ so we often just write $[\alpha\backslash t] <^{\mathrm{o}} \beta$. Furthermore, $[\alpha\backslash t] <^{\mathrm{o}^1} [\beta\backslash s]$ is necessary (although not sufficient) for $\beta \in t$. Hence the scope order includes the order induced by the productions of the grammar as in Section 2 and is equal to it if every term introduced by a weak quantifier inference contains all variables introduced underneath it by strong quantifier inferences. For technical reasons (the applicability of Lemma 10) it is more

convenient to work with the scope order in this section. We apply the notation and results of Section 4 to the set $\mathrm{B}(\pi)$ with order $<^{\mathrm{o}^1}$ and equivalence relation $[\alpha\backslash t] \sim [\beta\backslash s]$ iff $\alpha = \beta$; the upset formation $\uparrow$ as well as the choice-operation $*$ have the meaning defined in Section 4. We write $\mathrm{B}^*(\pi)$ for $(\mathrm{B}(\pi))^*$. For a set of unary substitutions A we define the variable renaming $\rho_A = [\alpha\backslash\alpha_{A\uparrow\alpha}]_{\alpha\in A}$. As we will see later, this renaming plays an important role because it labels α with the partial value given by A. A cut c in a proof π in normal form is called *active* if its cut-formula $\exists x\, A$ is introduced from more than one term in the left subproof. The following lemma is the central technical result of this paper.

Lemma 12 (Contraction Lemma)**.** If π is a proof in normal form, then there is a proof π' in normal form without active cuts s.t. $\mathrm{H}(\pi') = \mathrm{H}(\pi)\{\rho_A \mid A \in \mathrm{B}^*(\pi)\}$ and $\mathrm{B}(\pi') = \bigcup_{A\in\mathrm{B}^*(\pi)} A^{\rho_A}$.

Proof. Let $\mathrm{EV}(\pi) = \{\alpha_1, \ldots, \alpha_n\}$ be ordered s.t. c_{α_i} is uppermost in $\{c_{\alpha_i}, \ldots, c_{\alpha_n}\}$. The following equalities hold for all $i \in \{1, \ldots, n\}$ and follow from simple properties of the scope order and the uppermost order of the c_{α_i}:

$$\mathrm{B}_{\{\alpha_1,\ldots,\alpha_i\}}(\pi)\uparrow\alpha_i \;=\; \mathrm{B}_{\mathrm{L}}(\pi,\alpha_i)\uparrow\alpha_i \tag{1}$$

$$\mathrm{B}_{\{\alpha_1,\ldots,\alpha_i\}}(\pi)\uparrow\alpha \;=\; \mathrm{B}_{\hat{\mathrm{R}}}(\pi,\alpha_{i+1})\uparrow\alpha \quad \text{for } \alpha \in \mathrm{EV}_{\mathrm{R}}(\pi,\alpha_{i+1}) \tag{2}$$

By Lemma 9 one obtains the $*$-ed version of these equalities. For $k \in \{0, \ldots, n\}$ let

$$R_k = \{\rho_A \mid A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_k\}}(\pi)\}.$$

We will show by induction on k that there is a proof π_k in normal form s.t.

1. the only active cuts in π_k are c_{α_j} for some $j > k$,
2. $\mathrm{EV}(\pi_k) = \mathrm{EV}(\pi)R_k$,
3. $\mathrm{H}(\pi_k) = \mathrm{H}(\pi)R_k$, and
4. $\mathrm{B}(\pi_k) = \bigcup_{A\in\mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_k\}}(\pi)}(A \cup \mathrm{B}_{\{\alpha_{k+1},\ldots,\alpha_n\}}(\pi))^{\rho_A}$.

The result then follows from letting $\pi' = \pi_n$.

If $k = 0$ then $\pi = \pi_0$ trivially satisfies the conditions. For the case $k+1$ we obtain π_k from induction hypothesis and observe that $\alpha_{k+1} \in \mathrm{EV}(\pi_k)$ and that $c_{\alpha_{k+1}}$ appears in a context of the form

$$\dfrac{(\psi_1)\quad \Gamma \to \Delta, \exists x\, A \qquad \dfrac{(\psi_2)\quad A[x\backslash\alpha_{k+1}], \Pi \to \Lambda}{\exists x\, A, \Pi \to \Lambda}\,\exists_{\mathrm{l}}}{\Gamma, \Pi \to \Delta, \Lambda}\,\mathrm{cut}(c_{\alpha_{k+1}}) \quad .$$

Let $\{t_1, \ldots, t_m\}$ be the terms inserted for $\exists x\, A$ in ψ_1, then $\{t_1, \ldots, t_m\} = \alpha_{k+1}\mathrm{B}_{\{\alpha_{k+1}\}}(\pi_k)$. We will first describe this set of terms by suitably chosen upsets in the base substitutions of the original proof π. By induction hypothesis we have

$$\mathrm{B}_{\{\alpha_{k+1}\}}(\pi_k) = \bigcup_{A\in\mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_k\}}(\pi)} \mathrm{B}^{\rho_A}_{\{\alpha_{k+1}\}}(\pi)$$

which together with (1) shows that the mapping

$$\tau : \mathrm{B}^*_{\mathrm{L}}(\pi,\alpha_{k+1})\uparrow\alpha_{k+1} \longrightarrow \{t_1, \ldots, t_m\},\ A \mapsto \alpha_{k+1}A\rho_A$$

is well-defined. It is also bijective; for surjectivity let $i \in \{1,\dots,m\}$, then there is $[\alpha_{k+1}\backslash t_i] \in \mathrm{B}_{\{\alpha_{k+1}\}}(\pi_k)$ hence there are $A \in \mathrm{B}^*_{\{\alpha_1,\dots,\alpha_k\}}(\pi)$ and $[\alpha_{k+1}\backslash s] \in \mathrm{B}_{\{\alpha_{k+1}\}}(\pi)$ s.t. $t_i = s\rho_A$. Let $A' = \{[\alpha_{k+1}\backslash s]\} \bigcup_{\alpha\in s} A \uparrow \alpha$ and observe that $A' \in \mathrm{B}^*_{\{\alpha_1,\dots,\alpha_{k+1}\}}(\pi) \uparrow \alpha_{k+1} = \mathrm{B}^*_{\mathrm{L}}(\pi,\alpha_{k+1}) \uparrow \alpha_{k+1}$ and $A \uparrow \alpha = A' \uparrow \alpha$ for all $\alpha \in s$ hence $t_i = \alpha_{k+1}A'\rho_{A'}$. For injectivity let $\{[\alpha_{k+1}\backslash s_1]\} \cup A_1, \{[\alpha_{k+1}\backslash s_2]\} \cup A_2 \in \mathrm{B}^*_{\mathrm{L}}(\pi,\alpha_{k+1}) \uparrow \alpha_{k+1}$ s.t. $s_1\rho_{A_1} = s_2\rho_{A_2}$. As the ρ_{A_i} only add labels and the s_i are label-free we have $s_1 = s_2$ hence $A_1 \uparrow \alpha = A_2 \uparrow \alpha$ for all $\alpha \in s_i$. If $\sigma \in A_i$ then there is $\alpha \in s_i$ with $\alpha \leq^{\mathrm{o}} \sigma$ so we can conclude $A_1 = A_2$.

Let $\mathrm{B}^*_{\mathrm{L}}(\pi,\alpha_{k+1}) \uparrow \alpha_{k+1} = \{C_1,\dots,C_m\}$ where the indices are obtained from τ and for $1 \leq i \leq m$ define a variable renaming

$$\delta_i = [\alpha_A\backslash\alpha_{A\cup C_i}]_{\alpha\in\mathrm{EV}_{\mathrm{R}}(\pi,\alpha_{k+1}),A\in\mathrm{B}^*_{\hat{\mathrm{R}}}(\pi,\alpha_{k+1})\uparrow\alpha}.$$

Let ψ_1' be the proof of $\Gamma \to \Delta, \exists x\, A, \dots, \exists x\, A$ obtained from ψ_1 by omitting all contractions on ancestors of $\exists x\, A$ that contain the existential quantifier and define π_{k+1} by replacing $c_{\alpha_{k+1}}$ by

$$\dfrac{\dfrac{\dfrac{\dfrac{(\psi_1')}{\Gamma \to \Delta, \exists x\, A, \dots, \exists x\, A} \quad \dfrac{\dfrac{(\psi_2\delta_1)}{A[x\backslash\alpha_{k+1,C_1}], \Pi \to \Lambda}}{\exists x\, A, \Pi \to \Lambda}\,\exists_{\mathrm{l}}}{\Gamma, \Pi \to \Delta, \Lambda, \exists x\, A, \dots, \exists x\, A}\,\mathrm{cut}}{\vdots} \quad \dfrac{\dfrac{(\psi_2\delta_m)}{A[x\backslash\alpha_{k+1,C_m}], \Pi \to \Lambda}}{\exists x\, A, \Pi \to \Lambda}\,\exists_{\mathrm{l}}}{\Gamma, \Pi, \dots, \Pi \to \Delta, \Lambda, \dots, \Lambda, \exists x\, A \qquad\qquad \Gamma, \Pi, \dots, \Pi \to \Delta, \Lambda, \dots, \Lambda}\,\mathrm{cut}}{\Gamma, \Pi \to \Delta, \Lambda}\,\mathrm{c}^*$$

where $\psi_2\delta_i$ is linked to t_i via a cut.

After this transformation each logical inference, each axiom and each cut in π_{k+1} has a unique ancestor in π_k and – by induction – in π. This ancestor relation extends naturally to auxiliary and main formulas of inferences. Observe that this proof transformation has the property that whenever an inference is above another one in one of the π_k then the same is true about their ancestors in π. Therefore we obtain

$$\mathrm{EV}(\psi_2) = \mathrm{EV}_{\mathrm{R}}(\pi,\alpha_{k+1})R_k$$

from the induction hypothesis which using (2) entails

$$\mathrm{EV}(\psi_2) = \{\alpha_A \mid \alpha \in \mathrm{EV}_{\mathrm{R}}(\pi,\alpha_{k+1}), A \in \mathrm{B}^*_{\hat{\mathrm{R}}}(\pi,\alpha_{k+1}) \uparrow \alpha\}.$$

For $\alpha_A \in \mathrm{EV}(\psi_2)$ and $i,j \in \{1,\dots,m\}$, the equality $\alpha_A\delta_i = \alpha_A\delta_j$ entails $C_i = C_j$ hence $i = j$ so the regularity of π_{k+1} follows from that of π_k. It is then easy to check that π_{k+1} is in normal form.

Let $A \in \mathrm{B}^*_{\hat{\mathrm{R}}}(\pi,\alpha_{k+1})$, $i \in \{1,\dots,m\}$, $C_i \subseteq C \in \mathrm{B}^*_{\mathrm{L}}(\pi,\alpha_{k+1})$, $A \cup C \subseteq A' \in \mathrm{B}^*_{\{\alpha_1,\dots,\alpha_{k+1}\}}(\pi)$. Let F be a main formula of an inference in π which *is ancestor* of one in ψ_2. If $\alpha \in F$ but $\alpha \notin \mathrm{EV}_{\mathrm{R}}(\pi,\alpha_{k+1})$ then $\alpha = \alpha_l$ for some $l > k+1$ hence $\alpha\rho_A\delta_i = \alpha$. If $\alpha \in \mathrm{EV}_{\mathrm{R}}(\pi,\alpha_{k+1})$, then $\alpha\rho_A\delta_i = \alpha_{A\uparrow\alpha\cup C_i}$, so $F\rho_A\delta_i = F[\alpha\backslash\alpha_{A\uparrow\alpha\cup C_i}]_{\alpha\in\mathrm{EV}_{\mathrm{R}}(\pi,\alpha_{k+1})}$. Now α_{k+1} is the entrance of $\mathrm{B}_{\mathrm{L}}(\pi,\alpha_{k+1})$ for $\mathrm{B}_{\hat{\mathrm{R}}}(\pi,\alpha_{k+1})$ and $\mathrm{B}_{\mathrm{L}}(\pi,\alpha_{k+1}) \not<^{\mathrm{o}} \mathrm{B}_{\hat{\mathrm{R}}}(\pi,\alpha_{k+1})$ because $c_{\alpha_{k+1}}$ is a purely existential cut, so by Lemma 10 $F\rho_A\delta_i = F[\alpha\backslash\alpha_{(A\cup C)\uparrow\alpha}]_{\alpha\in\mathrm{EV}_{\mathrm{R}}(\pi,\alpha_{k+1})}$ and from the context lemma we obtain

$$F\rho_A\delta_i = F\rho_{A'}. \tag{3}$$

Let G be a main formula of an inference in π which *is not ancestor* of one in ψ_2 and let $\alpha \in G$. Then $\alpha \notin \mathrm{EV}_{\mathrm{R}}(\pi, \alpha_{k+1})$ and $\alpha \leq^{\mathrm{o}}_{B_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)} \alpha_{k+1}$ is impossible because all cuts below $c_{\alpha_{k+1}}$ have index higher than $k+1$. Therefore

$$G\rho_A = G\rho_A\delta_i = G\rho_{A'} \tag{4}$$

by the context lemma. Summing up, (3) and (4) entail $HR_{k+1} = HR_k\{\delta_i \mid 1 \leq i \leq m\}$ for any formula H in π which together with the respective induction hypothesis shows 2. and 3.

It remains to show 4. First we will establish

$$\mathrm{B}_{\{\alpha_{k+1}\}}(\pi_{k+1}) = \bigcup_{A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)} A^{\rho_A}_{\{\alpha_{k+1}\}}. \tag{5}$$

We start with the left-to-right inclusion: let $[\alpha_{k+1,C_i}\backslash t_i] \in \mathrm{B}(\pi_{k+1})$, then from τ and (1) we obtain a $C_i \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi) \uparrow \alpha_{k+1}$ with $t_i = \alpha_{k+1}C_i\rho_{C_i}$ and so $[\alpha\backslash s] \in C_i$ for some s. Let $C_i \subseteq A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)$, then $[\alpha_{k+1}\backslash s]^{\rho_A} = [\alpha_{k+1}\backslash s]^{\rho_{C_i}} = [\alpha_{k+1,C_i}\backslash t_i]$. For the other direction let $[\alpha_{k+1}\backslash s] \in A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)$ and let $i \in \{1,\ldots,m\}$ s.t. $C_i = A \uparrow \alpha_{k+1} \in \mathrm{B}^*_{\mathrm{L}}(\pi, \alpha_{k+1}) \uparrow \alpha_{k+1}$. Then $[\alpha_{k+1}\backslash s]^{\rho_A} = [\alpha_{k+1}\backslash s]^{\rho_{C_i}} = [\alpha_{k+1,C_i}\backslash t_i] \in \mathrm{B}(\pi_{k+1})$.

Let us show

$$\mathrm{B}_{\{\alpha\}}(\pi_{k+1}) = \bigcup_{A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)} A^{\rho_A}_{\{\alpha\}} \quad \text{for } \alpha \in \mathrm{EV}_{\hat{\mathrm{R}}}(\pi, \alpha_{k+1}) \tag{6}$$

next. If $[\alpha_{A\cup C_i}\backslash t'] \in \mathrm{B}(\pi_{k+1})$ then there is $[\alpha_A\backslash t] \in \mathrm{B}(\pi_k)$ with $t\delta_i = t'$ and $[\alpha\backslash s] \in A \in \mathrm{B}^*_{\hat{\mathrm{R}}}(\pi, \alpha_{k+1}) \uparrow \alpha$ with $\alpha\rho_A = \alpha_A$ and $s\rho_A = t$ by induction hypothesis and (2). Let $A \cup C_i \subseteq A' \in \mathrm{B}_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)$, then $[\alpha\backslash s] \in A'$ and by (3) we have $\alpha\rho_{A'} = \alpha\rho_A\delta_i = \alpha_{A\cup C_i}$ and $s\rho_{A'} = s\rho_A\delta_i = t'$. For the other direction let $A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)$, then there is $i \in \{1,\ldots,m\}$ and $A_0 \in \mathrm{B}_{\hat{\mathrm{R}}}(\pi, \alpha_{k+1})$ s.t. $C_i \cup A_0 \subseteq A$ and $[\alpha\backslash s] \in A_0$. By induction hypothesis $[\alpha\rho_{A_0}\backslash s\rho_{A_0}] \in \mathrm{B}(\pi_k)$ so $[\alpha\rho_{A_0}\delta_i\backslash s\rho_{A_0}\delta_i] \in \mathrm{B}(\pi_{k+1})$ and by (3) we have $\alpha\rho_{A_0}\delta_i = \alpha\rho_A$ and $s\rho_{A_0}\delta_i = s\rho_A$.

The equality

$$\mathrm{B}_{\{\alpha\}}(\pi_{k+1}) = \bigcup_{A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)} \mathrm{B}^{\rho_A}_{\{\alpha\}}(\pi) \quad \text{for } \alpha \in \{\alpha_1,\ldots,\alpha_k\} \setminus \mathrm{EV}_{\hat{\mathrm{R}}}(\pi, \alpha_{k+1}) \tag{7}$$

follows from induction hypothesis and the observation that $\mathrm{B}_\alpha(\pi) \not\leq^{\mathrm{o}} \alpha_{k+1}$ for such α. It remains to show

$$\mathrm{B}_{\{\alpha\}}(\pi_{k+1}) = \bigcup_{A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)} \mathrm{B}^{\rho_A}_{\{\alpha\}}(\pi) \quad \text{for } \alpha \in \{\alpha_{k+2},\ldots,\alpha_n\} \tag{8}$$

Let $[\alpha\backslash t'] \in \mathrm{B}(\pi_{k+1})$, then there is $[\alpha\backslash t] \in \mathrm{B}(\pi_k)$ where either (case i) t' is introduced by an $\exists_{\mathrm{r}}$-inference in one of the $\psi_2\delta_i$, then $t' = t\delta_i$ for some $i \in \{1,\ldots,m\}$ or otherwise (case ii) $[\alpha\backslash t]^{\rho_A} = [\alpha\backslash t]$ for all $A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)$ which directly entails the claim. In case i) we obtain $A \in \mathrm{B}^*_{\hat{\mathrm{R}}}(\pi, \alpha_{k+1})$ and $[\alpha\backslash s] \in \mathrm{B}(\pi)$ with $s\rho_A = t$ from induction hypothesis and (1). Let $A \cup C_i \subseteq A' \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)$ then $s\rho_{A'} = s\rho_A\delta_i = t'$ by (3). For the other direction let $A \in \mathrm{B}^*_{\{\alpha_1,\ldots,\alpha_{k+1}\}}(\pi)$, $[\alpha\backslash s] \in \mathrm{B}(\pi)$ and $A_0 \in \mathrm{B}_{\hat{\mathrm{R}}}(\pi, \alpha_{k+1})$, $i \in \{1,\ldots,m\}$ s.t. $A_0 \cup C_i \subseteq A$. If s is introduced by an $\exists_{\mathrm{r}}$-inference on the right above $c_{\alpha_{k+1}}$ in π then by (3) we have $s\rho_A = s\rho_{A_0}\delta_i$. By induction hypothesis and (2) we have $[\alpha\backslash s\rho_{A_0}] \in \mathrm{B}(\pi_k)$ which entails $[\alpha\backslash s\rho_{A_0}\delta_i] \in \mathrm{B}(\pi_{k+1})$. If s is not introduced on the right above $c_{\alpha_{k+1}}$ in π, then $s\rho_A = s\rho_{A_0}$ by (3). By induction hypothesis $[\alpha\backslash s\rho_{A_0}] \in \mathrm{B}(\pi_k)$ hence $[\alpha\backslash s\rho_{A_0}] \in \mathrm{B}(\pi_{k+1})$. □

Lemma 13. If π is a proof in normal form, then there is a proof π' in normal form without active cuts s.t. $L(\mathrm{G}(\pi')) = L(\mathrm{G}(\pi))$.

Proof. Let π' be the proof obtained in Lemma 12 having $\mathrm{H}(\pi') = \mathrm{H}(\pi)\{\rho_A \mid A \in \mathrm{B}^*(\pi)\}$ and $\mathrm{B}(\pi') = \bigcup_{A \in \mathrm{B}^*(\pi)} A^{\rho_A}$. Note that every substitution in $\mathrm{B}(\pi')$ is of the form $[\alpha\rho_A \backslash t\rho_A]$ for $[\alpha\backslash t] \in A \in \mathrm{B}^*(\pi)$. Let us first show that for every $\mathrm{G}(\pi')$-derivation $\varphi \to F'_1 \to \cdots \to F'_l$ there is a $\mathrm{G}(\pi)$-derivation $\varphi \to F_1 \to \cdots \to F_l$ and an $A \in \mathrm{B}^*(\pi)$ s.t. $F_i\rho_A = F'_i$ for $1 \leq i \leq l$ and every production $\gamma \to s$ used in $F_1 \to \cdots \to F_l$ satisfies $[\gamma\backslash s] \in A$. We proceed by induction on l: for $l = 1$ this follows immediately. Let $\varphi \to F'_1 \to \cdots \to F'_{l+1}$ be given and obtain A and $\varphi \to F_1 \to \cdots \to F_l$ from induction hypothesis. Let $\alpha\rho_{A'} \to t\rho_{A'}$ with $A' \in \mathrm{B}^*(\pi)$ be the production rule applied in $F'_l \to F'_{l+1}$. So $\alpha\rho_{A'}$ appears in $F'_l = F_l\rho_A$ hence $A' \uparrow \alpha = A \uparrow \alpha$. As $[\alpha\backslash t] \in A'$ we have $[\alpha\backslash t] \in A$ and $A \uparrow \beta = A' \uparrow \beta$ for all $\beta \in t$. Hence $\alpha\rho_A = \alpha\rho_{A'}$ and $t\rho_A = t\rho_{A'}$ so $\alpha\rho_A \to t\rho_A$ applied to F'_l gives F'_{l+1}. Applying $\alpha \to t$ at the same position to F_l leads to a formula F_{l+1} with $F_{l+1}\rho_A = F'_{l+1}$. By induction hypothesis the only α-production used in $F_1 \to \cdots \to F_l$ is $\alpha \to t$, and as all non-terminals of t are rigid, the rigidity condition on α is preserved.

For the other direction, suppose we are given a $\mathrm{G}(\pi)$-derivation. By Lemma 1 it induces a set $A \in \mathrm{B}^*(\pi)$ and by applying ρ_A can be converted to a $\mathrm{G}(\pi')$-derivation. Rigidity is preserved as ρ_A is an injective renaming of non-terminals. □

Lemma 14. If π is a proof of $\Gamma \to \exists x\, A$ in normal form without active cuts, then $\Gamma \to L(\mathrm{G}_{\mathrm{nd}}(\pi))$ is provable.

Proof. We will first produce a proof π' in normal form without active cuts that does no longer contain d. If d appears in π, then it does so in a context of the form

$$\dfrac{\dfrac{\dfrac{(\psi_1)}{\Pi \to \Lambda, A}}{\Pi \to \Lambda, A \vee B[x\backslash d]}\,\vee_{\mathrm{r}_1}}{\Pi \to \Lambda, \exists x\,(A \vee B)}\,\exists_{\mathrm{r}} \qquad \dfrac{\dfrac{(\psi_2)}{A \vee B[x\backslash\alpha], \Sigma \to \Theta}}{\exists x\,(A \vee B), \Sigma \to \Theta}\,\exists_{\mathrm{l}}}{\Pi, \Sigma \to \Lambda, \Theta}\,\mathrm{cut}$$

or in a symmetric variant. Let ψ'_2 be the proof of $A, \Sigma \to \Theta$ obtained from ψ_2 by replacing $A \vee B[x\backslash\alpha]$ by A until arriving at inferences of the form

$$\dfrac{\dfrac{(\xi_1)}{A, \Sigma_1 \to \Theta_1} \qquad \dfrac{(\xi_2)}{B[x\backslash\alpha], \Sigma_2 \to \Theta_2}}{A \vee B[x\backslash\alpha], \Sigma_1, \Sigma_2 \to \Theta_1, \Theta_2}\,\vee_{\mathrm{l}}$$

which are replaced by ξ_1. Let π_1 be the proof obtained from replacing the above cut by

$$\dfrac{\dfrac{(\psi_1)}{\Pi \to \Lambda, A} \qquad \dfrac{(\psi'_2)}{A, \Sigma \to \Theta}}{\Pi, \Sigma \to \Lambda, \Theta}\,\mathrm{cut}\,.$$

Repeating this transformation for all occurrences of d we obtain a proof π' which does not contain d anymore and by Lemma 4 satisfies $L(\mathrm{G}(\pi')) \subseteq L(\mathrm{G}(\pi))$ hence $L(\mathrm{G}(\pi')) \subseteq L(\mathrm{G}_{\mathrm{nd}}(\pi))$.

Let $\alpha \to t$ be any production in $\mathrm{G}(\pi')$, then $[\alpha\backslash t]$ is the only base substitution of c_α. We define a proof π'' as follows: we shift c_α upwards to the left (observing that inference permutations do not change the grammar) until we reach the $\exists_{\mathrm{r}}$-inference introducing t. The standard cut-reduction of the existential quantifier then yields the proof π'' with $\mathrm{H}(\pi'') = \mathrm{H}(\pi')[\alpha\backslash t]$ and

$\mathrm{B}(\pi'') = (\mathrm{B}(\pi') \setminus \{[\alpha\backslash t]\})^{[\alpha\backslash t]}$. A $\mathrm{G}(\pi'')$-derivation is then translated to a $\mathrm{G}(\pi')$-derivation by replacing applications of a production $\beta \to s[\alpha\backslash t]$ by $\beta \to s$ followed by a sufficient number of applications of $\alpha \to t$. Let p_1, p_2 be two positions of α in the $\mathrm{G}(\pi')$-derivation. As $\alpha \to t$ is the only α-production and all non-terminals in t are rigid, the rigidity condition is also fulfilled for α at p_1, p_2. For the other direction, observe that whenever α is introduced, say by a production $\beta \to s$ it must be followed by $\alpha \to t$ on all positions of α. After permutation of independent steps we can thus collapse $\beta \to s$ and the appropriate applications of $\alpha \to t$ to a single application of $\beta \to s[\alpha\backslash t]$ which preserves rigidity. We have obtained $L(\mathrm{G}(\pi'')) = L(\mathrm{G}(\pi'))$.

By iterating this procedure we obtain a proof π^* with $L(\mathrm{G}(\pi^*)) = L(\mathrm{G}(\pi')) \subseteq L(\mathrm{G}_{\mathrm{nd}}(\pi))$ all of whose cuts are quantifier-free. Therefore $\mathrm{B}(\pi^*) = \emptyset$, i.e. there are no B-productions in $\mathrm{G}(\pi^*)$ and using the standard proof of the mid-sequent theorem (i.e. inference permutations) one obtains a proof of $\Gamma \to \mathrm{H}(\pi^*)$ i.e. $\Gamma \to L(\mathrm{G}(\pi^*))$ from which by adding weakenings one obtains a proof of $\Gamma \to L(\mathrm{G}_{\mathrm{nd}}(\pi))$. □

We can now prove the main results of this paper.

Theorem 2. If π is a simple proof of $\Gamma \to \exists x\, A$ then there is a totally rigid acyclic grammar G' with $|G'| \leq |\pi|$ and $L(G') \subseteq L(\mathrm{G}(\pi))$ s.t. $\Gamma \to L(G')$ is provable.

Proof. From Lemma 11 we obtain a proof π_1 in normal form and a grammar G' satisfying $|G'| \leq |\pi|$ and $L(G') = L(\mathrm{G}_{\mathrm{nd}}(\pi_1)) \subseteq L(\mathrm{G}(\pi))$. Applying Lemma 13 to π_1 we obtain a proof π_2 in normal form without active cuts satisfying $L(\mathrm{G}(\pi_2)) = L(\mathrm{G}(\pi_1))$ hence $L(\mathrm{G}_{\mathrm{nd}}(\pi_2)) = L(\mathrm{G}_{\mathrm{nd}}(\pi_1)) = L(G')$. Applying Lemma 14 to π_2 then shows that $\Gamma \to L(G')$ is provable. □

Corollary 1. If π is a simple proof of $\Gamma \to \exists x\, A$ then $\Gamma \to L(\mathrm{G}(\pi))$ is provable.

Proof. Append weakenings to the proof of $\Gamma \to L(G')$ obtained from Theorem 2. □

Example 6. Applying Corollary 1 to the proof π of Example 2 shows that

$$A_1, A_2, A_3 \to R(g(a, f(a))), R(g(b, f(b)))$$

is provable.

The following corollary shows how the logical restriction on the cut-formulas (of having at most one quantifier) induces a combinatorial restriction on the Herbrand-disjunctions obtainable from simple proofs.

Corollary 2. If π is a simple proof of $\Gamma \to \exists x\, A$, then there are variables $x_1, \ldots, x_n$ and sets of terms $U_1, \ldots, U_n$ s.t.

$$\Gamma \to \bigvee_{(u_1,\ldots,u_n)\in U_1\times\cdots\times U_n} A[x_1\backslash u_1]\cdots[x_n\backslash u_n]$$

is provable and $\sum_{i=1}^{n} |U_i| \leq |\pi|$.

Proof. By Theorem 2 there is G' with $|G'| \leq |\pi|$ and $\Gamma \to L(G')$ being provable. By Lemma 3 we can write $L(G')$ using sets U_i as above. □

Example 7. Applying Corollary 2 to the proof π of Example 2 yields the representation

$$A_1, A_2, A_3 \to \bigvee_{(u_1,u_2,u_3)\in U_1\times U_2\times U_3} R(x)[x\backslash u_1][\beta\backslash u_2][\alpha\backslash u_3]$$

where $U_1 = \{g(\alpha, \beta)\}$, $U_2 = \{f(\alpha)\}$ and $U_3 = \{a, b\}$.

6 From Tree Languages to Proofs

Already the results in [12] show that a simple proof induces an acyclic regular(!) tree grammar whose finite language is an Herbrand-disjunction. So what have we gained from the strengthening of this result by adding total rigidity? On the one hand, we have gained an exponent: we have seen in Lemma 3 that in the totally rigid case the size of the language is bound by n^n, but:

Lemma 15. There is an acyclic regular tree grammar G with $2n$ productions and $L(G) = n^{n^n}$.

Proof. Let f be an n-ary function symbol, then the productions

$$\alpha_0 \to f(\alpha_1, \ldots, \alpha_1), \ldots, \alpha_{n-1} \to f(\alpha_n, \ldots, \alpha_n)$$

create an tree with n^n leaves. Let $c_1, \ldots, c_n$ be terminal symbols, then by adding the productions

$$\alpha_n \to c_1, \ldots, \alpha_n \to c_n$$

we obtain the desired grammar G. □

However there is another – more fundamental – motivation for this result: in this section we show that the compression power of simple proofs corresponds **exactly** to that of totally rigid acyclic grammars. Given such a grammar G we will obtain a simple proof π that induces G. As $L(\mathrm{G}(\pi))$ is a set of formulas but $L(G)$ is a set of terms (which do not necessarily represent formulas), we cannot expect to obtain $\mathrm{G}(\pi) = G$. The closest possible connection is to wrap up the term language of G in some new unary predicate symbol R_1.

Theorem 3. For every totally rigid acyclic tree grammar $G = \langle \alpha_1, R, T, P \rangle$ there is a simple proof π with $\mathrm{G}(\pi) = \langle \alpha_0, R \cup \{\alpha_0\}, T, P \cup \{\alpha_0 \to R_1(\alpha_1)\} \rangle$.

Proof. By Lemma 3 we can assume that $G = \langle \alpha_1, \{\alpha_1, \ldots, \alpha_n\}, T, P \rangle$ s.t. α_i depends only on α_j with $j > i$. The proof π is defined in the language $T \cup \{R_i \mid 1 \leq i \leq n\}$ where the R_i are unary predicate symbols with intended interpretation "being reachable from the non-terminal α_i". For each rule $\alpha_i \to t$ define the formula

$$\varphi_{\alpha_i \to t} \;=\; \forall x_{i+1} \cdots \forall x_n \; (\; R_{i+1}(x_{i+1}) \supset \cdots R_n(x_n) \supset R_i(t[\alpha_j \backslash x_j]_{j=i+1}^n) \;).$$

For each non-terminal α_i with rules $\alpha_i \to t_1, \ldots, \alpha_i \to t_m$ define the formula

$$\varphi_i \;=\; \bigvee_{j=1}^{m} \varphi_{\alpha_i \to t_j}$$

and the proof $\psi_i =$

$$\dfrac{\cdots \quad \dfrac{\dfrac{R_i(t_j) \to R_i(t_j)}{R_i(t_j) \to \exists x\, R_i(x)}\,\exists_{\mathrm{r}}}{\varphi_{\alpha_i \to t_j}, R_{i+1}(\alpha_{i+1}), \ldots, R_n(\alpha_n) \to \exists x\, R_i(x)}\,\forall_{\mathrm{l}}^*, \supset_{\mathrm{l}}^* \quad \cdots}{\varphi_i, R_{i+1}(\alpha_{i+1}), \ldots, R_n(\alpha_n) \to \exists x\, R_i(x)}\,\vee_{\mathrm{l}}^* .$$

Now define proofs $\pi_i : \varphi_1, \ldots, \varphi_i, R_{i+1}(\alpha_{i+1}), \ldots, R_n(\alpha_n) \to \exists x\, R_1(x)$ for $i \in \{0, \ldots, n\}$ and $\pi_i' : \varphi_1, \ldots, \varphi_i, \exists x\, R_{i+1}(x), R_{i+2}(\alpha_{i+2}), \ldots, R_n(\alpha_n) \to \exists x\, R_1(x)$ for $i \in \{0, \ldots, n-1\}$ by

$$\pi_i' \;=\; \dfrac{\begin{matrix}(\pi_i)\\ \varphi_1, \ldots, \varphi_i, R_{i+1}(\alpha_{i+1}), \ldots, R_n(\alpha_n) \to \exists x\, R_1(x)\end{matrix}}{\varphi_1, \ldots, \varphi_i, \exists x\, R_{i+1}(x), R_{i+2}(\alpha_{i+2}), \ldots, R_n(\alpha_n) \to \exists x\, R_1(x)}\,\exists_{\mathrm{l}}$$

and

$$\pi_0 \;=\; \frac{R_1(\alpha_1) \to R_1(\alpha_1)}{R_1(\alpha_1), \ldots, R_n(\alpha_n) \to \exists x\, R_1(x)}\; \mathrm{w}^*_{\mathrm{l}}, \exists_{\mathrm{r}}$$

and

$$\pi_{i+1} \;=\; \frac{\overset{(\psi_{i+1})}{\varphi_{i+1}, R_{i+2}(\alpha_{i+2}), \ldots, R_n(\alpha_n) \to \exists x\, R_{i+1}(x)} \qquad (\pi'_i)}{\varphi_1, \ldots, \varphi_{i+1}, R_{i+2}(\alpha_{i+2}), \ldots, R_n(\alpha_n) \to \exists x\, R_1(x)}\; \mathrm{cut}\;.$$

Then it is easy to verify that $\pi = \pi_n : \varphi_1, \ldots, \varphi_n \to \exists x\, R_1(x)$ has the desired grammar. □

It is worthwhile to describe the models of the end-sequent $\varphi_1, \ldots, \varphi_n \to \exists x\, R_1(x)$ of the proof constructed above: in a structure that satisfies φ_i at least one of the α_i-productions must be true. A structure satisfying $\varphi_1, \ldots, \varphi_n$ hence makes one production of every non-terminal true and therefore one formula from $L(\mathrm{G}(\pi))$. The disjunction of the formulas in $L(\mathrm{G}(\pi))$ is therefore true in every model of $\varphi_1, \ldots, \varphi_n$ and vice versa: every disjunction of instances of $\exists x\, R_1(x)$ which is true in all models of $\varphi_1, \ldots, \varphi_n$ must contain $L(\mathrm{G}(\pi))$.

7 Conclusion

We have shown that the compression power of proofs where each cut contains at most one quantifier corresponds exactly to that of totally rigid acyclic tree grammars. This work constitutes a proof-of-concept result for a new connection between proof theory and formal language theory arising from exact characterisations of classes of proofs by classes of grammars.

A side effect of Corollary 1 is to illustrate how little information about a proof is necessary for computing an Herbrand-disjunction: the current instances of the end-formula together with the base substitutions suffice, not even the cut-formulas are necessary.

The following directions are left for future work.

7.1 Extensions

The strategy of describing a cut-free proof by a tree language is applicable to any system that possesses an Herbrand-like theorem, i.e. even full higher-order logic as in [18]. The difficulty consists in finding an appropriate type of grammars. For full first-order logic (including quantifier alternations in the cuts) it would be natural to consider a stack of tree grammars, each layer of which corresponds to one layer of quantifiers in the cuts and to prove an analogous result by iterating Lemma 12. An extension to arithmetic would require – in addition – a mechanism for recursion in the grammars to model induction.

7.2 Confluence

One of the original motivations for describing cut-elimination by grammars was to characterise the set of cut-free proofs that can be obtained from a single proof by cut-elimination without any restriction of the reduction strategy [12]; let $\to$ denote this cut-elimination relation. The author conjectures that the result of [12] can be extended to the rigid tree grammars presented in this paper, i.e.

Conjecture 1. If $\pi \to \pi'$ and π' is cut-free, then $\mathrm{H}(\pi') \subseteq L(\mathrm{G}(\pi))$.

In fact, the behaviour of rigid tree grammars under the cut-reduction steps suggests the following stronger version. Let $\rightarrow^-$ be $\rightarrow$ without deletion of subproofs (e.g. in a purely multiplicative calculus: without the reduction of weakening).

Conjecture 2. If $\pi \rightarrow^- \pi'$, then $L(\mathrm{G}(\pi)) = L(\mathrm{G}(\pi'))$.

This conjecture entails a confluence result by applying it to two arbitrary $\rightarrow^-$-normal forms π'_1 and π'_2 yielding $\mathrm{H}(\pi'_1) = L(\mathrm{G}(\pi)) = \mathrm{H}(\pi'_2)$. Confluence properties of classical logic have been the subject of many papers. This conjecture would be an important contribution to that discussion: it states that the first-order level is confluent (though highly redundant [1]) and that the different normal forms arise only from different choices taken at the propositional level how to (partially) remove this redundancy.

7.3 Algorithmic Cut-Introduction

Computer-generated proofs are typically analytic, i.e. in the case of first-order logic can be written as Herbrand-disjunctions. Due to their shorter length it is natural to try to find proofs with cut algorithmically. One approach is to abbreviate a given cut-free proof by the introduction of cuts, see e.g. [9, 23]. With respect to general-purpose data compression, such a procedure has the algorithmic advantage of also reducing the runtime of operations, for example proof checking, which is important in applications such as proof-carrying code. Cut-introduction (as any compression) exploits redudancy which is present in the Herbrand-disjunction. A fundamental problem of this approach is to understand which type of redudancy can be removed by which cuts. The characterisation results of this paper give a solution to this problem for the case of cuts with one quantifier: the corresponding redundancy is that of the language of a totally rigid acyclic tree grammar.

7.4 Proof Length

From a more theoretical point of view, it is possible to view a proof with cuts as a compressed representation of a cut-free proof or an Herbrand-disjunction (if we fix a particular cut-elimination procedure, for instance the computation of the language of its grammar). The results of this paper show that the size of the proof with cut is intimately related to measures from formal language theory such as automatic complexity [22] or automaticity [21]. By the size condition of Corollary 2, a lower bound on the number of rules of totally rigid acyclic tree grammars translates to a lower bound on the length of proofs with cuts. The perspective of exploiting this connection in order to use techniques from formal language theory for proving lower bounds on proofs seems promising (e.g. for problems 22 and 24 in [3] on the impact of Skolemisation and equational reasoning on the length of proofs).

7.5 Intuitionistic Logic

Another interesting application of these grammars would be the analysis of cut-elimination in intuitionistic logic. Moving from Herbrand's theorem to the existential witness property, the natural expectation would be to find that $|L(\mathrm{G}(\pi))| = 1$.

References

[1] Matthias Baaz and Stefan Hetzl. On the non-confluence of cut-elimination. *Journal of Symbolic Logic*, 76(1):313–340, 2011.

[2] Matthias Baaz and Alexander Leitsch. Cut Normal Forms and Proof Complexity. *Annals of Pure and Applied Logic*, 97:127–177, 1999.

[3] Peter Clote and Jan Krajíček. Open problems. In Peter Clote and Jan Krajíček, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 1–19. Oxford University Press, 1993.

[4] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata: Techniques and Applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

[5] Stephen Cook and Phong Nguyen. *Logical Foundations of Proof Complexity*. Perspectives in Logic. Cambridge University Press, 2010.

[6] Emmanuel Filiot, Jean-Marc Talbot, and Sophie Tison. Satsifiability of a Spatial Logic with Tree Variables. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic (CSL) 2007*, volume 4646 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2007.

[7] Emmanuel Filiot, Jean-Marc Talbot, and Sophie Tison. Tree Automata with Global Constraints. In Masami Ito and Masafumi Toyama, editors, *Developments in Language Theory (DLT) 2008*, volume 5257 of *Lecture Notes in Computer Science*, pages 314–326. Springer, 2008.

[8] Emmanuel Filiot, Jean-Marc Talbot, and Sophie Tison. Tree Automata With Global Constraints. *International Journal of Foundations of Computer Science*, 21(4):571–596, 2010.

[9] Marcelo Finger and Dov Gabbay. Equal Rights for the Cut: Computable Non-analytic Cuts in Cut-based Proofs. *Logic Journal of the IGPL*, 15(5–6):553–575, 2007.

[10] Ferenc Gécseg and Magnus Steinby. Tree Languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages: Volume 3: Beyond Words*, pages 1–68. Springer, 1997.

[11] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1934–1935.

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

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

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

[15] Ulrich Kohlenbach. *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*. Springer, 2008.

[16] Georg Kreisel. Finiteness theorems in arithmetic: An application of Herbrand's theorem for Σ_2-formulas. In J. Stern, editor, *Logic Colloquium 1981*, pages 39–55. North-Holland, 1982.

[17] Horst Luckhardt. Herbrand-Analysen zweier Beweise des Satzes von Roth: Polynomiale Anzahlschranken. *Journal of Symbolic Logic*, 54(1):234–263, 1989.

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

[19] Tsuyoshi Morioka. *Logical Approaches to the Complexity of Search Problems: Proof Complexity, Quantified Propositional Calculus and Bounded Arithmetic*. PhD thesis, University of Toronto, 2005.

[20] Sherif Sakr. XML compression techniques: A survey and comparison. *Journal of Computer and System Sciences*, 75:303–322, 2009.

[21] Jeffrey Shallit and Yuri Breitbart. Automaticity I: Properties of a Measure of Descriptional Complexity. *Journal of Computer and System Sciences*, 53:10–25, 1996.

[22] Jeffrey Shallit and Ming-Wei Wang. Automatic complexity of strings. *Journal of Automata, Languages and Combinatorics*, 6(4):537–554, 2001.

[23] Bruno Woltzenlogel Paleo. Atomic Cut Introduction by Resolution: Proof Structuring and Compression. In E. M. Clark and A. Voronkov, editors, *Logic for Programming, Artifical Intelligence and Reasoning (LPAR-16)*, volume 6355 of *Lecture Notes in Computer Science*, pages 463–480. Springer, 2010.