Theoretical Computer Science 549 (2014) 1–16

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Algorithmic introduction of quantified cuts ☆

Stefan Hetzl [a], Alexander Leitsch [b], Giselle Reis [b], Daniel Weller [a,*]

[a] Institute of Discrete Mathematics and Geometry, Vienna University of Technology, Vienna, Austria
[b] Institute of Computer Languages, Vienna University of Technology, Vienna, Austria

ARTICLE INFO

*Article history:*
Received 23 January 2013
Received in revised form 15 May 2014
Accepted 27 May 2014
Available online 2 June 2014
Communicated by A. Avron

*Keywords:*
Proof compression
Cut-introduction
Classical logic
First-order logic

ABSTRACT

We describe a method for inverting Gentzen's cut-elimination in classical first-order logic. Our algorithm is based on first computing a compressed representation of the terms present in the cut-free proof and then cut-formulas that realize such a compression. Finally, a proof using these cut-formulas is constructed. Concerning asymptotic complexity, this method allows an exponential compression of quantifier complexity (the number of quantifier-inferences) of proofs.



## 1. Introduction

Cut-elimination introduced by Gentzen [15] is the most prominent form of proof transformation in logic and plays an important role in automating the analysis of mathematical proofs. The removal of cuts corresponds to the elimination of intermediate statements (lemmas), resulting in a proof which is analytic in the sense that all statements in the proof are subformulas of the result. Thus a proof of a combinatorial statement is converted into a purely combinatorial proof. Cut-elimination is therefore an essential tool for the analysis of proofs, especially to make implicit parameters explicit.

In this paper we present a method for inverting Gentzen's cut-elimination by computing a proof with cut from a given cut-free proof as input. As cut-elimination is the backbone of proof theory, there is considerable proof-theoretic interest and challenge in understanding this transformation sufficiently well to be able to invert it. But our interest in cut-introduction is not only of a purely theoretical nature. Proofs with cuts have properties that are essential for applications: on the one hand, cuts are indispensable for formalizing proofs in a human-readable way. One the other hand cuts have a very strong compression power in terms of proof length.

Computer-generated proofs are typically analytic, i.e. they only contain logical material that also appears in the theorem shown. This is due to the fact that analytic proof systems have a considerably smaller search space which makes proof-search practically feasible. In the case of the sequent calculus, proof-search procedures typically work on the cut-free fragment. But also resolution is essentially analytic as resolution proofs satisfy the subformula property of first-order logic. An important property of non-analytic proofs is their considerably smaller length. The exact difference depends on the logic (or theory) under consideration, but it is typically enormous. In (classical and intuitionistic) first-order logic there are proofs with cut of length $n$ whose theorems have only cut-free proofs of length $2_n$ (where $2_0 = 1$ and $2_{n+1} = 2^{2_n}$) (see [36]

☆ This work was supported by the projects P22028-N13, I603-N18 and P25160-N25 of the Austrian Science Fund (FWF), by the ERC Advanced Grant ProofCert and the WWTF Vienna Research Group 12-04.

* Corresponding author.
*E-mail addresses:* stefan.hetzl@tuwien.ac.at (S. Hetzl), leitsch@logic.at (A. Leitsch), giselle@logic.at (G. Reis), weller@logic.at (D. Weller).

http://dx.doi.org/10.1016/j.tcs.2014.05.018
0304-3975/

and [31]). The length of a proof plays an important role in many situations such as human readability, space requirements and time requirements for proof checking. For most of these situations general-purpose data compression methods cannot be used as the compressed representation would need to be uncompressed for checking proof theoretical properties. It is therefore of high practical interest to develop proof-search methods which produce non-analytic and hence potentially much shorter proofs. In the method presented in this paper we start with a cut-free proof and abbreviate it by computing useful cuts based on a structural analysis of the cut-free proof.

There is another, more theoretical, motivation for introducing cuts which derives from the foundations of mathematics: most of the central mathematical notions have developed from the observation that many proofs share common structures and steps of reasoning. Encapsulating those leads to a new abstract notion, like that of a group or a vector space. Such a notion then builds the base for a whole new theory whose importance stems from the pervasiveness of its basic notions in mathematics. From a logical point of view this corresponds to the introduction of cuts into an existing proof database. While we cannot claim to contribute much to the understanding of such complex historical processes by the current technical state of the art, this second motivation is still worthwhile to keep in mind, if only to remind us that we are dealing with a difficult problem here.

Gentzen's method of cut-elimination is based on reductions of cut-derivations (subproofs ending in a cut), transforming them into simpler ones; basically the cut is replaced by one or more cuts with lower logical complexity. A naive reversal of this procedure is infeasible as it would lead to a search tree which is exponentially branching on some nodes and infinitely branching on others. Therefore we base our procedure on a deeper proof-theoretic analysis: in the construction of a Herbrand sequent $S'$ corresponding to a cut-free proof $\varphi'$ (see e.g. [3]) obtained by cut-elimination on a proof $\varphi$ of a sequent $S$ with cuts, only the *substitutions* generated by cut-elimination on quantified cuts are relevant. In fact, it is shown in [18] that, for proofs with $\Sigma_1$ and $\Pi_1$-cuts only, $S'$ can be obtained just by computing the substitutions defined by cut-elimination without applying Gentzen's procedure as a whole. Via the cuts in the proof $\varphi$ one can define a tree grammar generating a language consisting exactly of the terms (to be instantiated for quantified variables in $S$) for obtaining the Herbrand sequent $S'$ [18]. Hence, generating a tree grammar $G$ from a set of Herbrand terms $T$ (generating $T$) corresponds to an inversion of the quantifier part of Gentzen's procedure. The computation of such an inversion forms the basis of the method of *cut-introduction* presented in this paper. Such an inversion of the quantifier part of cut-elimination determines which instances of the cut-formulas are used but it does not determine the cut-formulas. In fact, a priori it is not clear that every such grammar can be realized by actual cut-formulas. However, we could show that, for any such tree grammar representing the quantifier part of potential cut-formulas, actual cut-formulas can be constructed. Finally, a proof containing these cut-formulas can be constructed.

Work on cut-introduction can be found at a number of different places in the literature. Closest to our work are other approaches which aim to abbreviate or structure *a given input proof*: [41] is an algorithm for the introduction of atomic cuts that is capable of exponential proof compression. The method [13] for propositional logic is shown to never increase the size of proofs more than polynomially. Another approach to the compression of first-order proofs by introduction of definitions for abbreviating terms is [40].

Viewed from a broader perspective, this paper should be considered part of a large body of work on the generation of non-analytic formulas that has been carried out by numerous researchers in various communities. Methods for lemma generation are of crucial importance in inductive theorem proving which frequently requires generalization [7], see e.g. [25] for a method in the context of rippling [8] which is based on failed proof attempts. In automated theory formation [9,10], an eager approach to lemma generation is adopted. This work has, for example, led to automated classification results of isomorphism classes [34] and isotopy classes [35] in finite algebra. See also [28] for an approach to inductive theory formation. In pure proof theory, an important related topic is Kreisel's conjecture (see footnote 3 on page 400 of [38]) on the generalization of proofs. Based on methods developed in this tradition, [4] describes an approach to cut-introduction by filling a proof skeleton, i.e. an abstract proof structure, obtained by an inversion of Gentzen's procedure with formulas in order to obtain a proof with cuts. The use of cuts for structuring and abbreviating proofs is also of relevance in logic programming: [30] shows how to use focusing in order to avoid proving atomic subgoals twice, resulting in a proof with atomic cuts.

This paper is organized as follows:

In Section 3 we define Herbrand sequents and *extended* Herbrand sequents which represent proofs with cut. The concept of rigid acyclic regular tree grammars is applied to establish a relation between an extended Herbrand sequent $S^*$ and a (corresponding) Herbrand sequent $S'$: the language defined by this grammar is just the set of terms $T$ to be instantiated for quantifiers in the original sequent $S$ to obtain $S'$. Given such a grammar $G$ generating $T$ there exists a so-called *schematic extended Herbrand sequent* $\hat{S}$ in which the (unknown) cut-formulas are represented by monadic second-order variables. It is proved that $\hat{S}$ always has a solution, the *canonical solution*. From this solution, which gives an extended Herbrand sequent and the cut-formulas for a proof, the actual proof with these cuts is constructed.

To make the underlying methods more transparent, Section 3 deals only with end-sequents of the form $\forall x\, F \rightarrow$. In Section 4 the method is generalized to sequents of the form

$$\forall \bar{x}_1 F_1, \ldots, \forall \bar{x}_n F_n \rightarrow \exists \bar{y}_1 G_1, \ldots, \exists \bar{y}_m G_m$$

where the $\bar{x}_i, \bar{y}_j$ are vectors of variables and $\forall z_1 \cdots z_k$ stands for $\forall z_1 \cdots \forall z_k$. This form of sequents is more useful for practical applications and covers all of first-order logic as an arbitrary sequent can be transformed into one of this form by Skolemization and prenexification. We prove that all results obtained in Section 3 carry over to this more general case.

Given a cut-free proof $\varphi$ and a corresponding Herbrand term set $T$, the canonical solution corresponding to a non-trivial minimal grammar generating $T$ yields a proof $\psi$ with lower quantifier-complexity (which is the number of quantifier inferences in a proof) than $\varphi$.

In Section 5 a nondeterministic algorithm CI is defined which is based on the techniques developed in Section 3. We show that CI is, in a suitable sense, an inversion of Gentzen's cut-elimination method. A sequence of cut-free proofs is defined and it is proven that the application of CI to this sequence results in an exponential compression of quantifier complexity.

This paper improves the publication [21] in several crucial directions: (1) the method for introducing a single $\forall$-cut is generalized to a method introducing an arbitrary number of $\forall$-cuts, which requires – among others – a length-preserving transformation of extended Herbrand-sequents to proofs with cuts based on Craig interpolation, (2) we show that our method is, in a suitable sense, an inversion of Gentzen's cut-elimination method, (3) the end-sequent may contain blocks of quantifiers instead of just single ones, (4) the decomposition of terms is represented as a problem of grammars, and (5) it is shown that the proof compression (measured by quantifier complexity) obtained by the new method is exponential while it was only quadratic for the one of [21].

The method CI developed in this paper is a systematic, proof-theoretic method to compress the quantifier complexity of first-order proofs by the introduction of cuts. Still, the generated cut-formulas are all universal. A desirable extension of this method to introduce cuts with alternating quantifiers (which is necessary to obtain super-exponential compressions) is left to future work. Such an extension is highly non-trivial as it first requires the development of an adequate notion of tree grammar for extending the underlying proof-theoretic results to cuts with quantifier alternations.

## 2. A motivating example

Consider the sequents

$$S_n = Pa, \forall x\,(Px \supset Pfx) \to Pf^{2^n}a.$$

The straightforward cut-free proof of $S_n$ in the sequent calculus uses the successor-axiom $2^n$ times. In fact, it is easy to show that every cut-free proof has to contain all of these $2^n$ instances. On the other hand, if we allow the use of cuts, we can give a considerably shorter proof by first showing

$$\forall x\left(Px \supset Pf^2x\right)$$

from the axiom and then using this formula twice to show

$$\forall x\left(Px \supset Pf^4x\right)$$

and so on. In general we cut with the constant-length proofs of

$$\forall x\left(Px \supset Pf^{2^i}x\right) \to \forall x\left(Px \supset P^{2^{i+1}}x\right)$$

and hence obtain a proof of $S_n$ that uses only $O(n)$ inference steps instead of the $\Omega(2^n)$ of the cut-free proof. Note how the structures of these two proofs are reminiscent of the binary and the unary representation of numbers. This proof sequence is an exponential version of the sequences of Statman [36] and Orevkov [31] and has also been considered by Boolos [6].

In this paper we want to leverage this compression power of lemmas by automatically transforming cut-free proofs into proofs using compressing lemmas.

## 3. Proof-theoretic infrastructure

Gentzen's proof of cut-elimination [15] can be understood as the application of a set of local proof rewriting rules with a terminating strategy. A first naive approach to cut-introduction would be to consider the inversion of these local rewriting steps as a search algorithm. While this procedure would in theory allow to reverse every cut-elimination sequence, it becomes clear quickly that it is not feasible in practice: not only would we have to guess an enormous amount of trivialities (e.g. rule permutations) but the inversion of rewriting rules which erase a part of the proof lead to the necessity of correctly guessing an entire subproof. Therefore we need more abstract proof representations.

The proof representations we will be using and their relationships are depicted in Fig. 1. The purpose of this section is to explain these representations and their relationships. As a first orientation let us just mention that the rows of Fig. 1 contain notions on increasingly abstract levels: the level of proofs in the first row, that of formulas in the second row and that of terms in the third row. The transformations in each column are complexity-preserving (in a sense that will be made precise soon). The transformation of an object in the left column to an object in the right column increases its complexity considerably (exponentially in this paper).

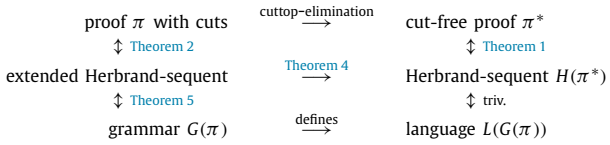


**Fig. 1.** Proof-theoretic setting of this paper.

For this whole section, fix a quantifier-free formula $F$ with one free variable $x$ s.t. $\forall x\, F$ is unsatisfiable. We will, for the sake of simplicity, explain our algorithm first in the setting of proofs of the end-sequent $\forall x\, F \to$. We will show how to abbreviate a given cut-free proof of this sequent by the introduction of cuts, which are of the form $\forall x\, A$ for $A$ quantifier-free, such cuts will be called $\Pi_1$-cuts in the sequel. The algorithm will then be generalized to less restrictive end-sequents in Section 4.

### 3.1. Proofs and Herbrand's theorem

A *sequent* is an ordered pair of sets of formulas, written as $\Gamma \to \Delta$. While the concrete variant of the sequent calculus is of little importance to the algorithms presented in this paper let us, for the sake of precision, fix it to be **G3c** $+ \mathrm{Cut}_{\mathrm{cs}}$[1] from [39].

**Definition 1** *(Herbrand-sequent).* A tautological sequent of the form $H : F[x\backslash t_1], \ldots, F[x\backslash t_n] \to$ is called a *Herbrand-sequent* of $\forall x\, F \to$. We define $|H| = n$ and call it the *complexity of* $H$.

We thus measure the number of instances of $\forall x\, F$ used for showing its unsatisfiability. This complexity-measure is of fundamental importance as the undecidability of first-order logic hinges on it: a bound gives a decision procedure as most general unification can be used for bounding the term size in the number of instances; it then only remains to enumerate all possible instances having at most this bounding size. On the level of proofs we keep track of the number of used instances by counting the number of $\forall_{\mathrm{l}}$- and $\exists_{\mathrm{r}}$-inferences, for the other quantifier rules ($\forall_{\mathrm{r}}$ and $\exists_{\mathrm{l}}$) one application per formula suffices.

**Definition 2.** We define the *quantifier-complexity of a proof* $\pi$, written as $|\pi|_{\mathrm{q}}$ as the number of $\forall_{\mathrm{l}}$- and $\exists_{\mathrm{r}}$-inferences in $\pi$.

Herbrand-sequents then correspond to cut-free proofs in the following sense.

**Theorem 1.** *$\forall x\, F \to$ has a cut-free proof $\pi$ with $|\pi|_{\mathrm{q}} = l$ iff it has a Herbrand-sequent $H$ with $|H| = l$.*

**Proof sketch.** Given $\pi$ we obtain $H$ by reading off the instances of $\forall x\, F$ from the proof $\pi$ and collecting them in a sequent (if $\pi$ contains some duplicate instances we add dummy instances to $H$ for obtaining $|H| = |\pi|_{\mathrm{q}}$).

Given $H$ we first compute any propositional proof of $H$ and obtain a cut-free proof $\pi$ of $\forall x\, F \to$ by introducing the universal quantifier for each of those instances and applying a sufficient number of contractions. □

The above theorem shows that we can think of a Herbrand-sequent as a concise representation of a cut-free proof. A first important step towards our cut-introduction algorithm will be the generalization of this relation to proofs with an arbitrary number of $\Pi_1$-cuts (in a way similar to [17]).

**Definition 3.** Let $u_1, \ldots, u_m$ be terms, let $A_1, \ldots, A_n$ be quantifier-free formulas, let $\alpha_1, \ldots, \alpha_n$ be variables, let $V(t)$ denote the set of variables occurring in the term $t$, and let $s_{i,j}$ for $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant k_j$ be terms s.t.

1. $\mathrm{V}(A_i) \subseteq \{\alpha_i, \ldots, \alpha_n\}$ for all $i$, and
2. $\mathrm{V}(s_{i,j}) \subseteq \{\alpha_{i+1}, \ldots, \alpha_n\}$ for all $i, j$.

Then the sequent

$$H = F[x\backslash u_1], \ldots, F[x\backslash u_m], A_1 \supset \bigwedge_{j=1}^{k_1} A_1[\alpha_1\backslash s_{1,j}], \ldots, A_n \supset \bigwedge_{j=1}^{k_n} A_n[\alpha_n\backslash s_{n,j}] \to$$

is called an *extended Herbrand-sequent* of $\forall x\, F \to$ if $H$ is a tautology.

[1] **G3c** $+ \mathrm{Cut}_{\mathrm{cs}}$ has no structural rules and all its rules are invertible.

What is this cryptic definition supposed to mean? An extended Herbrand-sequent of the above form will represent a proof with $n$ $\Pi_1$-cuts whose cut formulas are $\forall\alpha_1 A_1, \ldots, \forall\alpha_n A_n$ (or sometimes minor variants thereof), the $\alpha_i$ are the eigenvariables of the universal quantifiers in these cut-formulas, the $s_{i,j}$ the terms of the instances of the cut-formulas on the right-hand side of the cut and the $u_i$ the terms of the instances of our end-formula $\forall x F$. The complexity of an extended Herbrand-sequent $H$ of the above form is defined as $|H| = m + \sum_{j=1}^{n} k_j$. One can view an extended Herbrand-sequent together with a propositional proof of it as a particular form of proof in the $\varepsilon$-calculus [24] with the cuts corresponding to the critical formulas. We obtain the following correspondence to the sequent calculus:

**Theorem 2.** *$\forall x F \to$ has a proof $\pi$ with $\Pi_1$-cuts and $|\pi|_q = l$ iff it has an extended Herbrand-sequent $H$ with $|H| = l$.*

While this theorem looks plausible it is not as straightforward to prove as one may expect. Its proof relies on Craig's interpolation theorem [12] which we briefly repeat here for the reader's convenience in the version of [38] and restricted to propositional logic. We split a sequent into two parts by writing it as a *partition* $\Gamma_1 \ ; \ \Gamma_2 \to \Delta_1 \ ; \ \Delta_2$. The purpose of doing so is merely to mark $\Gamma_1, \Delta_1$ as belonging to one and $\Gamma_2, \Delta_2$ as belonging to the other part of the partition. The logical meaning of $\Gamma_1 \ ; \ \Gamma_2 \to \Delta_1 \ ; \ \Delta_2$ is just $\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2$.

**Theorem 3.** *If a quantifier-free sequent $\Gamma_1 \ ; \ \Gamma_2 \to \Delta_1 \ ; \ \Delta_2$ is a tautology, then there is a quantifier-free formula $I$ s.t.*

1. *Both $\Gamma_1 \to \Delta_1, I$ and $I, \Gamma_2 \to \Delta_2$ are tautologies, and*
2. *All atoms that appear in $I$ appear in both $\Gamma_1 \to \Delta_1$ and $\Gamma_2 \to \Delta_2$.*

**Proof.** See [38]. □

**Proof of Theorem 2.** For the left-to-right direction we proceed analogously to the cut-free case: by passing through the proof $\pi$ and reading off the instances of quantified formulas (of both the end-formula and the cuts) we obtain an extended Herbrand-sequent $H$ with $|H| \leqslant |\pi|_q$ (which can be padded with dummy instances if necessary in order to obtain $|H| = |\pi|_q$).

For the right-to-left direction let

$$H = F[x\backslash u_1], \ldots, F[x\backslash u_m], A_1 \supset \bigwedge_{j=1}^{k_1} A_1[\alpha_1\backslash s_{1,j}], \ldots, A_n \supset \bigwedge_{j=1}^{k_n} A_n[\alpha_n\backslash s_{n,j}] \to$$

be an extended Herbrand-sequent and let us begin by introducing some abbreviations. For a set of terms $T$ and a formula $F$, write $F[x\backslash T]$ for the set of formulas $\{F[x\backslash t] \mid t \in T\}$. Abbreviate the "cut-implication" $A_i \supset \bigwedge_{j=1}^{k_i} A_i[\alpha_i\backslash s_{i,j}]$ as $\mathrm{CI}_i$ and let $U = \{u_1, \ldots, u_m\}$. Then $H$ can be written more succinctly as $F[x\backslash U], \mathrm{CI}_1, \ldots, \mathrm{CI}_n \to$.

Let $U_i = \{u \in U \mid \mathrm{V}(u) \subseteq \{\alpha_{i+1}, \ldots, \alpha_n\}\}$ for $i = 0, \ldots, n$. First we will show that it suffices to find quantifier-free formulas $A'_1, \ldots, A'_n$ s.t. the sequent

$$H' = F[x\backslash U], A'_1 \supset \bigwedge_{j=1}^{k_1} A'_1[\alpha_1\backslash s_{1,j}], \ldots, A'_n \supset \bigwedge_{j=1}^{k_n} A'_n[\alpha_n\backslash s_{n,j}] \to$$

has a proof of the following *linear form*:

$$\dfrac{\dfrac{\vdots \qquad\qquad \vdots}{F[x\backslash U] \to A'_1, \ldots, A'_n \quad \bigwedge_{j=1}^{k_1} A'_1[\alpha_1\backslash s_{1,j}], F[x\backslash U_1] \to A'_2, \ldots, A'_n}\supset_l}{\begin{array}{c} F[x\backslash U], \mathrm{CI}'_1 \to A'_2, \ldots, A'_n \\ \vdots \\ F[x\backslash U], \mathrm{CI}'_1, \ldots, \mathrm{CI}'_{n-1} \to A'_n \end{array} \qquad \begin{array}{c} \vdots \\ \bigwedge_{j=1}^{k_n} A'_n[\alpha_n\backslash s_{n,j}], F[x\backslash U_n] \to \end{array}}\supset_l$$
$$F[x\backslash U], \mathrm{CI}'_1, \ldots, \mathrm{CI}'_n \to$$

where $\mathrm{CI}'_i$ abbreviates $A'_i \supset \bigwedge_{j=1}^{k_i} A'_i[\alpha_i\backslash s_{i,j}]$. This suffices because in the above proof we can introduce cuts and quantifiers by replacing a segment of the form

$$\dfrac{F[x\backslash U], \mathrm{CI}'_1, \ldots, \mathrm{CI}'_{i-1} \to A'_i, \ldots, A'_n \quad \bigwedge_{j=1}^{k_i} A'_i[\alpha_i\backslash s_{i,j}], F[x\backslash U_i] \to A'_{i+1}, \ldots, A'_n}{F[x\backslash U], \mathrm{CI}'_1, \ldots, \mathrm{CI}'_i \to A'_{i+1}, \ldots, A'_n}\supset_l$$

by

$$\dfrac{\dfrac{\dfrac{F[x\backslash U_{i-1}], \forall x\, F \to A'_i, \ldots, A'_n}{F[x\backslash U_i], \forall x\, F \to A'_i, \ldots, A'_n}\ \forall_{\mathrm{l}}^*}{F[x\backslash U_i], \forall x\, F \to \forall x\, A'_i[\alpha_i\backslash x], A'_{i+1}, \ldots, A'_n}\ \forall_{\mathrm{r}} \quad \dfrac{A'_i[\alpha_i\backslash s_{i,j}]_{j=1}^{k_i}, F[x\backslash U_i] \to A'_{i+1}, \ldots, A'_n}{\forall x\, A'_i[\alpha_i\backslash x], F[x\backslash U_i] \to A'_{i+1}, \ldots, A'_n}\ \forall_{\mathrm{l}}^*}{F[x\backslash U_i], \forall x\, F \to A'_{i+1}, \ldots, A'_n}\ \mathrm{cut}$$

and finishing the proof at its root by

$$\dfrac{F[x\backslash U_n], \forall x\, F \to}{\forall x\, F \to}\ \forall_{\mathrm{l}}^*.$$

This transformation results in a proof whose number of $\forall_{\mathrm{l}}$-inferences is the complexity of the extended Herbrand-sequent as every term of $H$ is introduced exactly once.

Let us now turn to the construction of the $A'_i$. Write

$$L_i \quad \text{for } \mathrm{CI}_1, \ldots, \mathrm{CI}_{i-1}, F[x\backslash U] \to A'_i, \ldots, A'_n, \quad \text{and}$$

$$R_i \quad \text{for } \bigwedge_{j=1}^{k_i} A'_i[\alpha_i\backslash s_{i,j}], F[x\backslash U_i] \to A'_{i+1}, \ldots, A'_n.$$

Note that $L_i$ and $R_i$ depend only on those $A'_j$ with $j \geqslant i$ and note furthermore that $L_{n+1}$ is the extended Herbrand-sequent $H$ which is a tautology by assumption. Fix $i \in \{1, \ldots, n\}$. Assuming $\vDash L_{i+1}$ we will now construct $A'_i$ and show $\vDash L_i$ and $\vDash R_i$.

From $\vDash L_{i+1}$ we obtain

$$\vDash \mathrm{CI}_1, \ldots, \mathrm{CI}_{i-1}, F[x\backslash U] \to A_i, A'_{i+1}, \ldots, A'_n \quad \text{and} \tag{1}$$

$$\vDash \underbrace{\mathrm{CI}_1, \ldots, \mathrm{CI}_{i-1}, F[x\backslash (U \setminus U_i)]}_{\Gamma}, \underbrace{\bigwedge_{j=1}^{k_i} A_i[\alpha_i\backslash s_{i,j}], F[x\backslash U_i]}_{\Pi} \to \underbrace{A'_{i+1}, \ldots, A'_n}_{\Lambda} \tag{2}$$

from an application of $\supset_{\mathrm{l}}$ to $\mathrm{CI}_i$. Applying the propositional interpolation theorem to the partition $\Gamma\ ;\ \Pi \to\ ;\ \Lambda$ of (2) yields $I$ s.t. $\vDash \Gamma \to I$ and $\vDash \Pi, I \to \Lambda$. Furthermore $I$ contains only such atoms which appear in $\Pi \to \Lambda$, hence $\mathrm{V}(I) \subseteq \{\alpha_{i+1}, \ldots, \alpha_n\}$. Define $A'_i$ as $A_i \wedge I$. Observe that $\vDash R_i$ follows from $\vDash \Pi, I \to \Lambda$ and $\vDash L_i$ follows from (1) and $\vDash \Gamma \to I$. Hence $L_i, R_i$ for $i = 1, \ldots, n$ are tautologies. But $L_1, R_1, \ldots, R_n$ are exactly the leaves of the linear proof from above which finishes the proof of the theorem. □

This result does not only generalize Proposition 2 of [21] to the case of an arbitrary number of cuts but also improves it considerably, even for the case of a single cut: the use of interpolants is new in this paper and allows to obtain $|\pi|_{\mathrm{q}} \leqslant |H|$ for an extended Herbrand sequent $H$. In general it is not possible to read back an extended Herbrand-sequent to a proof of linear form without changing the cut formulas as the following example shows. The reason for insisting on this linear form is that it does not contain any duplicate instances which permits to show the property $|\pi|_{\mathrm{q}} = |H|$. The duplication behavior of connectives in this transformation is reminiscent of the complexity results in [2].

**Remark 1.** The complexity of the proof $\pi$ obtained from the extended Herbrand-sequent $H$ can also be bound beyond its pure quantifier complexity $|\pi|_{\mathrm{q}}$. Let $\mathrm{d}(\psi)$ denote the depth of a proof $\psi$, i.e. the maximal number of inferences on a branch and let $\|H\|$ denote the logical complexity of $H$. Then the right-to-left direction of Theorem 2 can be strengthened as follows: there is a constant $c$ s.t. for every extended Herbrand sequent $H$ of $\forall x\, F \to$ with $n$ cuts and $|H| = l$ there is a proof $\pi$ with $n$ $\Pi_1$-cuts, $|\pi|_{\mathrm{q}} = l$ and $\mathrm{d}(\pi) \leqslant c^n \|H\|$. This bound can be obtained from carrying out the proofs of Theorem 3 and Theorem 2 using $\vdash^d$ (derivability in depth $d$) instead of $\vDash$ (validity). It is created by the $n$-fold iteration of transformations of a proof of depth $d$ to a proof of depth $c \cdot d$.

**Example 1.** Let $F = P(x) \wedge (P(c) \supset Q(x)) \wedge (Q(x) \supset P(d)) \wedge \neg P(d)$ and $A_1 = P(\alpha_1)$. Furthermore let $m = 1$, $u_1 = \alpha_1$ and $n = 1$, $k_1 = 1$, $s_{1,1} = c$. Then

$$E = F[x\backslash u_1], \ldots, F[x\backslash u_m], A_1 \supset \bigwedge_{j=1}^{k_1} A_1[\alpha_1\backslash s_{1,j}], \ldots, A_n \supset \bigwedge_{j=1}^{k_n} A_n[\alpha_n\backslash s_{n,j}] \to$$

$$= P(\alpha_1) \wedge \big(P(c) \supset Q(\alpha_1)\big) \wedge \big(Q(\alpha_1) \supset P(d)\big) \wedge \neg P(d), P(\alpha_1) \supset P(c) \to$$

is a tautology and hence an extended Herbrand-sequent of $\forall x\, F \to$.

Let us now try to construct a linear **LK**-proof that corresponds to $E$. Such a proof contains a cut on $\forall x\, P(x)$ as its last inference. The formula $\forall x\, F$ must be instantiated on the left above this cut to obtain $F[x\backslash\alpha_1]$ as $\alpha_1$ is the eigenvariable

of the cut formula. This leaves the right side of the cut as $P(c) \to$ which is not valid, a second instance of $\forall x\, F$ would be needed. The solution used in the proof of Theorem 2 is based on computing a propositional interpolant of $F[x\backslash\alpha_1];\, P(c) \to;$. This can be done e.g. by first computing a proof of the sequent $F[x\backslash\alpha_1],\, P(c) \to$, e.g. the following $\psi =$

$$\dfrac{\dfrac{P(c) \to P(c) \quad \dfrac{\dfrac{Q(\alpha_1) \to Q(\alpha_1) \quad \dfrac{P(\alpha_1), P(d) \to P(d)}{P(\alpha_1), P(d), \neg P(d) \to}\,\neg_l}{P(\alpha_1), Q(\alpha_1), Q(\alpha_1) \supset P(d), \neg P(d) \to}\,\supset_l}{P(\alpha_1), P(c) \supset Q(\alpha_1), Q(\alpha_1) \supset P(d), \neg P(d), P(c) \to}\,\supset_l}{P(\alpha_1) \wedge (P(c) \supset Q(\alpha_1)) \wedge (Q(\alpha_1) \supset P(d)) \wedge \neg P(d), P(c) \to}\,\wedge^*_l$$

The propositional interpolant induced by the partition $F[x\backslash\alpha_1];\, P(c) \to;$ of $\psi$ according to the algorithm of [38] is computed as

$$\dfrac{\dfrac{\neg P(c) \quad \dfrac{\dfrac{\bot}{\bot}\,\neg_l \quad \dfrac{\bot}{\bot}}{\bot \vee \bot}\,\supset_l}{\neg P(c) \vee \bot \vee \bot}\,\supset_l}{\neg P(c) \vee \bot \vee \bot}\,\wedge^*_l$$

which simplifies to $\neg P(c)$. Hence the new cut formula is $\forall x\,(P(x) \wedge \neg P(c))$ which renders the right side of the cut provable as $P(c) \wedge \neg P(c) \to$.

### 3.2. Proofs and grammars

Now that we have established the connection between proofs and (extended) Herbrand-sequents we can move on to the term level of Fig. 1. A first trivial observation is that, assuming the knowledge of $F$, a Herbrand-sequent $H$ for $\forall x\, F \to$ does not carry more information than just the set of terms $T$ s.t. $H = F[x\backslash T] \to$.

A set of terms, in the terminology of formal language theory, is a tree language. The central theoretical result on which this paper is based is an analogous relation between extended Herbrand-sequents (or: proofs with $\Pi_1$-cuts) and *a certain class of tree grammars*. This result has first been proved in [18], see also [22] for a generalization.

Tree languages are a natural generalization of formal (string) languages, see e.g. [14,11]. Many important notions, such as regular and context-free languages carry over from the setting of strings to that of trees. The class of *rigid* tree languages has been introduced in [26] with applications in verification in mind, see [27]. Rigid tree languages augment regular tree languages by the ability to carry out certain equality tests, a property that is very useful for applications.

In the context of proof theory it is more natural to work with grammars than with automata because of the generative nature of cut-elimination. The class of grammars we will use in this paper is a subclass of rigid grammars: the *totally rigid acyclic tree grammars*. We write $\mathcal{T}_\Sigma(V)$ for the set of terms in the first-order signature $\Sigma$ over the set of variables $V$ and $\mathcal{T}_\Sigma$ for $\mathcal{T}_\Sigma(\emptyset)$. For a symbol $f \in \Sigma$ we write $(f/k)$ for denoting the arity $k$ of $f$.

**Definition 4.** A *regular tree grammar* is a tuple $G = \langle N, \Sigma, \tau, P\rangle$, where $N$ is a finite set of non-terminal symbols, $\Sigma$ is a first-order signature, $\tau \in N$ is the *start symbol* and $P$ is a finite set of production rules of the form $\beta \to t$ with $\beta \in N$ and $t \in \mathcal{T}_\Sigma(N)$.

The *one-step derivation relation* $\to^1_G$ of a regular tree grammar $G$ consists of all pairs $u[\beta] \to^1_G u[t]$ where $\beta \to t \in P$. A derivation in $G$ is a finite sequence of terms $t_0 = \tau, t_1, \ldots, t_n$ s.t. $t_i \to^1_G t_{i+1}$. The language of $G$ is defined as $L(G) = \{t \in \mathcal{T}_\Sigma \mid t \text{ has a } G\text{-derivation}\}$.

**Definition 5.** A *rigid tree grammar* is a tuple $G = \langle N, N_R, \Sigma, \tau, P\rangle$, where $\langle N, \Sigma, \tau, P\rangle$, is a regular tree grammar and $N_R \subseteq N$ is the set of *rigid non-terminals*. We speak of a *totally rigid tree grammar* if $N_R = N$. In this case we will just write $\langle N_R, \Sigma, \tau, P\rangle$.

A derivation $t_0 = \tau, t_1, \ldots, t_n = t$ of a term $t \in \mathcal{T}_\Sigma$ in a rigid tree grammar is a derivation in the underlying regular tree grammar that satisfies the additional *rigidity condition*: If there are $i, j < n$, a non-terminal $\beta \in N_R$, and positions $p$ and $q$ such that $t_i|_p = \beta$ and $t_j|_q = \beta$, then $t|_p = t|_q$. The language $L(G)$ of the rigid tree grammar $G$ is the set of all terms $t \in \mathcal{T}_\Sigma$ which can be derived under the rigidity condition. Totally rigid tree grammars are formalisms for specifying sets of substitutions and thus are particularly useful for describing instances generated by cut-elimination.

**Example 2.** 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 generated by the rigid tree grammar $\langle\{\tau, \alpha, \beta\}, \{\alpha\}, \{0/0, s/1, f/2\}, \tau, P\rangle$ where $P = \{\tau \to f(\alpha, \alpha), \alpha \to 0 \mid s(\beta), \beta \to 0 \mid s(\beta)\}$.

**Definition 6.** The *grammar of an extended Herbrand-sequent*

$$H \equiv F[x\backslash u_1], \ldots, F[x\backslash u_m], A_1 \supset \bigwedge_{j=1}^{k_1} A_1[\alpha_1\backslash s_{1,j}], \ldots, A_n \supset \bigwedge_{j=1}^{k_n} A_n[\alpha_n\backslash s_{n,j}] \to$$

is defined as the totally rigid $G(H) = \langle N_R, \Sigma, \tau, P\rangle$ where $N_R = \{\tau, \alpha_1, \ldots, \alpha_n\}$, $\Sigma$ is the signature of $H$ and $P = \{\tau \to u_i \mid 1 \leqslant i \leqslant m\} \cup \{\alpha_i \to s_{i,j} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k_i\}$.

A derivation of the form $\beta \to_G^1 t_1 \to_G^1 \cdots \to_G^1 t_n$ is called cyclic if $\beta \in \mathrm{V}(t_n)$. A grammar is called *acyclic* if it does not have any cyclic derivations. Note that condition 2 of Definition 3 ensures that the grammar of an extended Herbrand-sequent is acyclic. Furthermore, by definition, the grammar of an extended Herbrand-sequent is totally rigid. The language of such a grammar can be written in the following normal form.

**Lemma 1.** *If $G$ is totally rigid and acyclic, then up to renaming of the non-terminals $G = \langle\{\alpha_0, \ldots, \alpha_n\}, \Sigma, \alpha_0, P\rangle$ 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 to rename the non-terminals in such a way that $\alpha_i \to_G^1 t_1 \to_G^1 \cdots \to_G^1 t_n$ and $\alpha_j \in \mathrm{V}(t_n)$ implies $j > i$. The notation based on substitutions is then possible because, due to total rigidity, each $t \in L(G)$ can be derived using at most one production for each non-terminal. See [22] for a detailed proof. □

In particular, the language of a totally rigid acyclic grammar is finite. This lemma also suggests a compact notation for totally rigid acyclic grammars: we write

$$U \circ_{\alpha_1} S_1 \cdots \circ_{\alpha_n} S_n$$

for the grammar $\langle\{\tau, \alpha_1, \ldots, \alpha_n\}, \Sigma, \tau, P\rangle$ where $P = \{\tau \to u \mid u \in U\} \cup \{\alpha_i \to s_i \mid 1 \leqslant i \leqslant n, s_i \in S_i\}$, $\tau$ is some fresh start symbol and $\Sigma$ is the signature of the terms appearing in $P$. Using this notation, we can observe that $L(U) = U$ for a set of terms $U$ and $L(G \circ_\alpha S) = \{u[\alpha\backslash s] \mid u \in L(G), s \in S\}$ for a totally rigid acyclic tree grammar $G$ and a set of terms $S$. If the non-terminals are clear from the context, this notation is further abbreviated as

$$U \circ S_1 \cdots \circ S_n.$$

One can then obtain a cut-elimination theorem based on grammars:

**Theorem 4.** *If $H$ is an extended Herbrand-sequent of $\forall x\, F \to$, then $\{F[x\backslash t] \mid t \in L(G(H))\} \to$ is a Herbrand-sequent of $\forall x\, F \to$.*

**Proof.** This can be shown by following the development of the grammar during a cut-elimination process, see [18,16] and also [22] for a more general result. □

Throughout this whole paper all the grammars we are dealing with will be totally rigid and acyclic. Therefore we will henceforth use *grammar* as synonym for *totally rigid acyclic tree grammar*.

### 3.3. Cut-introduction

We have already observed above that it is not feasible to invert Gentzen's cut-elimination steps literally. The key to our method is that moving from the level of proofs to the level of grammars provides us with a transformation that is much easier to invert. The computation of the language of a grammar can simply be inverted as: *given a finite tree language $L$, find a grammar $G$ s.t. $L(G) = L$.*

The only piece then still missing in Fig. 1 is to obtain an extended Herbrand-sequent from $G$. Note that, for a given $G$, the term-part of the extended Herbrand-sequent is already determined using Lemma 1. What we do not know yet are the cut-formulas. Hence we define:

**Definition 7.** Let $u_1, \ldots, u_m$ be terms, let $X_1, \ldots, X_n$ be monadic second-order variables, let $\alpha_1, \ldots, \alpha_n$ be variables, and let $s_{i,j}$ for $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant k_j$ be terms s.t. $\mathrm{V}(s_{i,j}) \subseteq \{\alpha_{i+1}, \ldots, \alpha_n\}$ for all $i, j$. Then the sequent

$$H = F[x\backslash u_1], \ldots, F[x\backslash u_m], X_1(\alpha_1) \supset \bigwedge_{j=1}^{k_1} X_1(s_{1,j}), \ldots, X_n(\alpha_n) \supset \bigwedge_{j=1}^{k_n} X_n(s_{n,j}) \to$$

is called a *schematic extended Herbrand-sequent* of $\forall x\, F \to$ if $\bigwedge_{t \in L(G(H))} F[x\backslash t] \to$ is a tautology (where $G(H)$ is defined analogously to Definition 6).

A *solution* of a schematic extended Herbrand-sequent $H$ is a substitution $\sigma = [X_i\backslash \lambda\alpha_i.A_i]_{i=1}^n$ s.t. $\mathrm{V}(A_i) \subseteq \{\alpha_i, \ldots, \alpha_n\}$ and $H\sigma$ is a tautology.

The reason for calling such a substitution $\sigma$ a solution is the close relationship of this problem to unification problems modulo the theory of Boolean algebras, in particular to Boolean unification with constants [29,1]. By comparison with Definition 3 note that if $\sigma$ is a solution for $H$, then $H\sigma$ is an extended Herbrand-sequent. The central property which is of interest right now is that such a sequent always has a solution.

**Definition 8.** Let $H$ be a schematic extended Herbrand-sequent. Define

$$C_1 = \bigwedge_{i=1}^{m} F[x\backslash u_i] \quad \text{and} \quad C_{i+1} = \bigwedge_{j=1}^{k_i} C_i[\alpha_i\backslash s_{i,j}] \quad \text{for } i = 1, \ldots, n.$$

Then

$$\sigma := [X_i\backslash \lambda\alpha_i.C_i]_{i=1}^{n}$$

is called *canonical substitution* of $H$.

We will now show that the canonical substitution is, in fact, a solution.

**Lemma 2.** *Let $H$ and $C_i$ be as in Definition 8. Then $C_{n+1} \to$ is a tautology.*

**Proof.** By definition, $C_{n+1} \to$ is $\bigwedge_{i=1}^{m} \bigwedge_{j_1=1}^{k_1} \cdots \bigwedge_{j_n=1}^{k_n} F[x\backslash u_i][\alpha_1\backslash s_{1,j_1}] \cdots [s_{n,\alpha_n}\backslash j_n] \to$ which by Lemma 1 is $\bigwedge_{t\in L(G(H))} F[x\backslash t] \to$ which is a tautology as $H$ is a schematic extended Herbrand-sequent. □

**Lemma 3.** *Let $H$ be a schematic extended Herbrand-sequent and $\sigma$ be its canonical substitution. Then $\sigma$ is a solution of $H$.*

**Proof.** First note that the variable condition is fulfilled as $\mathrm{V}(C_i) \subseteq \{\alpha_i, \ldots, \alpha_n\}$. Then observe that

$$H\sigma = F[x\backslash u_1], \ldots, F[x\backslash u_m], C_1 \supset \bigwedge_{j=1}^{k_1} C_1[\alpha_1\backslash s_{1,j}], \ldots, C_n \supset \bigwedge_{j=1}^{k_n} C_n[\alpha_n\backslash s_{n,j}] \to$$

is logically equivalent to

$$C_1, C_1 \supset C_2, \ldots, C_n \supset C_{n+1} \to .$$

The unsatisfiability of $C_{n+1}$ follows from Lemma 2, hence $H\sigma$ is a tautology. □

In light of the above result we will henceforth call $\sigma$ the *canonical solution*. Note that the canonical solution permits a sequent calculus proof of a linear form in the sense of the proof of Theorem 2, and hence – for this solution – interpolation is not necessary in the construction of the proof with cuts.

**Theorem 5.** *$\forall x\, F \to$ has an extended Herbrand-sequent $H$ with $|H| = l$ iff there is a totally rigid acyclic tree grammar $G$ with $|G| = l$ s.t. $\bigwedge_{t\in L(G)} F[x\backslash t] \to$ is a tautology.*

**Proof.** The left-to-right direction of this statement follows from Theorem 4 together with the observation that $|G(H)| = |H|$. For the right-to-left direction assume that $G$ is given, let $H$ be the schematic extended Herbrand-sequent of $G$. Then the result follows from Lemma 3. □

Now we have proved all results mentioned in Fig. 1 and can finally describe our approach to cut-introduction. It consists in following this diagram in a clockwise fashion from the cut-free proof to the proof with cut. More specifically, given as input a cut-free proof $\pi$ our algorithm will proceed as follows:

1. Extract the set of terms $T$ of $H(\pi)$ (as in Theorem 1).
2. Find a suitable grammar $G$ s.t. $L(G) = T$.
3. Compute an extended Herbrand-sequent $H$ from $G$ (as in Theorem 5).
4. Construct a proof $\psi$ with cut from $H$ (as in Theorem 2).

**Example 3.** Consider the sequent $\forall x\, F \to$ where

$$F = Pa \wedge (Px \supset Pfx) \wedge \neg Pf^9a.$$

Let $\pi$ be a straightforward cut-free proof of $\forall x\, F \to$, then $|\pi|_{\mathrm{q}} = 9$. Following the above outline of an algorithm we carry out the following steps. Extract the set of terms

$$T = \{a, fa, f^2a, f^3a, f^4a, f^5a, f^6a, f^7a, f^8a\}$$

from $H(\pi)$ following Theorem 1. Compute a grammar $G$ with $L(G) = T$, for example

$$G = \{\alpha, f\alpha, f^2\alpha\} \circ_\alpha \{a, f^3a, f^6a\}.$$

As in the proof of Theorem 5, this grammar induces the schematic extended Herbrand-sequent

$$H = F[x\backslash\alpha], F[x\backslash f\alpha], F[x\backslash f^2\alpha], X(\alpha) \supset \big(X(a) \wedge X\big(f^3a\big) \wedge X\big(f^6a\big)\big) \to$$

whose canonical solution is

$$\sigma = \big[X\backslash\lambda\alpha.\,\big(F[x\backslash\alpha] \wedge F[x\backslash f\alpha] \wedge F[x\backslash f^2\alpha]\big)\big]$$

Hence $H\sigma$ is an extended Herbrand-sequent with $|H\sigma| = |H| = 6$ which in turn induces a proof $\psi$ as in Theorem 2 which has $|\psi|_{\mathrm{q}} = 6$ and contains a single $\Pi_1$-cut whose cut-formula is

$$\forall x\,\big(F \wedge F[x\backslash fx] \wedge F[x\backslash f^2x]\big).$$

Observe that we have decreased the quantifier complexity from $|\pi|_{\mathrm{q}} = 9$ to $|\psi|_{\mathrm{q}} = 6$.

## 4. More general end-sequents

The class of end-sequents considered in the previous section, while leading to a comparatively simple statement of the central results, is clearly too restricted for concrete applications. We will therefore extend our proof-theoretic infrastructure to proofs of end-sequents of the form

$$\forall x_1 \cdots \forall x_{l_1}\, F_1, \ldots, \forall x_1 \cdots \forall x_{l_p} F_p \to \exists x_1 \cdots \exists x_{l_{p+1}}\, F_{p+1}, \ldots, \exists x_1 \cdots \exists x_{l_q} F_q$$

with $l_i \geqslant 0$ and $F_i$ quantifier-free. We say that a sequent in this format is a $\Sigma_1$-sequent. Note that every first-order sequent can be transformed to this form by Skolemization and prenexing. Permitting $l_i$ to be zero allows for quantifier-free formulas such as in the example of Section 2. While the formalism now gets notationally more complicated, the results and proofs remain essentially the same. We write $\bar{x}$ for a vector $(x_1, \ldots, x_n)$ of variables, $\bar{t}$ for a vector $(t_1, \ldots, t_n)$ of terms and $[\bar{x}\backslash\bar{t}]$ for the substitution $[x_1\backslash t_1, \ldots, x_n\backslash t_n]$. For this whole section, we fix a sequent $\Gamma \to \Delta$ of the above form.

**Definition 9.** A tautological sequent of the form

$$\big\{F_i[\bar{x}\backslash\overline{t_{i,j}}] \mid 1 \leqslant i \leqslant p, 1 \leqslant j \leqslant n_i\big\} \to \big\{F_i[\bar{x}\backslash\overline{t_{i,j}}] \mid p < i \leqslant q, 1 \leqslant j \leqslant n_i\big\}$$

is called *Herbrand-sequent* of $\Gamma \to \Delta$.

The size of a Herbrand-sequent is defined as $|H| = \sum_{i=1}^{q} n_i$. Note that we only count formulas obtained by instantiation. Now as we are dealing with blocks of quantifiers it is appropriate to also change the size measure on proofs to consider blocks instead of single quantifiers. To that aim we change the quantifier rules in our sequent calculus to allow the introduction of a block of quantifiers (which is a natural alternative for a number of problems related to proof size, see e.g. [4]):

$$\frac{\forall x_1 \cdots \forall x_n\, A, A[\bar{x}\backslash\bar{t}], \Gamma \to \Delta}{\forall x_1 \cdots \forall x_n\, A, \Gamma \to \Delta}\ \forall^*_{\mathrm{l}} \qquad \frac{\Gamma \to \Delta, A[\bar{x}\backslash\bar{t}], \exists x_1 \cdots \exists x_n\, A}{\Gamma \to \Delta, \exists x_1 \cdots \exists x_n\, A}\ \exists^*_{\mathrm{r}}$$

We write $\|\pi\|_{\mathrm{q}}$ for the number of $\forall^*_{\mathrm{l}}$- and $\exists^*_{\mathrm{r}}$-inferences in the proof $\pi$.

**Theorem 6.** *$\Gamma \to \Delta$ has a cut-free proof $\pi$ with $\|\pi\|_{\mathrm{q}} = l$ iff it has a Herbrand-sequent $H$ with $|H| = l$.*

**Proof.** As for Theorem 1. □

**Definition 10.** Let $\overline{u_{i,1}}, \ldots, \overline{u_{i,m_i}}$ be vectors of terms with $l_i$ elements each. Let $A_1, \ldots, A_n$ be quantifier-free formulas, let $\alpha_1, \ldots, \alpha_n$ be variables, and let $s_{i,j}$ for $1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k_j$ be terms s.t.

1. $\mathrm{V}(A_i) \subseteq \{\alpha_i, \ldots, \alpha_n\}$ for all $i$, and
2. $\mathrm{V}(s_{i,j}) \subseteq \{\alpha_{i+1}, \ldots, \alpha_n\}$ for all $i, j$.

Then the sequent

$$H = \left\{F_i[\bar{x}\backslash\overline{u_{i,j}}] \mid 1 \leqslant i \leqslant p, 1 \leqslant j \leqslant m_i\right\}, A_1 \supset \bigwedge_{j=1}^{k_1} A_1[\alpha_1\backslash s_{1,j}], \ldots, A_n \supset \bigwedge_{j=1}^{k_n} A_n[\alpha_n\backslash s_{n,j}]$$
$$\to \left\{F_i[\bar{x}\backslash\overline{u_{i,j}}] \mid p < i \leqslant q, 1 \leqslant j \leqslant m_i\right\}$$

is called an *extended Herbrand-sequent* of $\Gamma \to \Delta$ if $H$ is a tautology.

The notion of schematic extended Herbrand-sequent is defined analogously to Definition 7 by replacing the formulas $A_i$ in the above definition by monadic predicate variables $X_i$. The size of a (schematic) extended Herbrand sequent $H$ of the above form is $|H| = \sum_{i=1}^{q} m_i + \sum_{j=1}^{n} k_j$.

**Theorem 7.** *$\Gamma \to \Delta$ has a proof with $\Pi_1$-cuts and $\|\pi\|_q = l$ iff it has an extended Herbrand-sequent $H$ with $|H| = l$.*

**Proof.** Analogous to the proof of Theorem 2, replacing $F[x\backslash U_i]$ by the collection of all instances $F_i[\bar{x}\backslash\overline{u_{i,j}}]$ s.t. all terms in $u_{i,j}$ contain only variables from $\{\alpha_{i+1}, \ldots, \alpha_n\}$. □

The above theorem encapsulates an algorithm for the construction of a proof with $\Pi_1$-cuts from an extended Herbrand-sequent. We will henceforth use the abbreviation PCA for this proof-construction algorithm.

In order to represent term vectors, it is helpful to enrich our signature by new function symbols $f_1, \ldots, f_q$ where $f_i$ has arity $l_i$. The function symbol $f_i$ will serve the purpose of grouping a term-tuple which corresponds to an instantiation of the formula $\forall x_1 \cdots \forall x_{l_i} F_i$ if $i \leqslant p$ (or $\exists x_1 \cdots \exists x_{l_i} F_i$ if $i > p$).

**Definition 11.** The grammar of an extended Herbrand-sequent

$$H = \left\{F_i[\bar{x}\backslash\overline{u_{i,j}}] \mid 1 \leqslant i \leqslant p, 1 \leqslant j \leqslant m_i\right\}, A_1 \supset \bigwedge_{j=1}^{k_1} A_1[\alpha_1\backslash s_{1,j}], \ldots, A_n \supset \bigwedge_{j=1}^{k_n} A_n[\alpha_n\backslash s_{n,j}]$$
$$\to \left\{F_i[\bar{x}\backslash\overline{u_{i,j}}] \mid p < i \leqslant q, 1 \leqslant j \leqslant m_i\right\}$$

is defined as $G(H) = \langle N_R, \Sigma, \tau, P\rangle$ where $N_R = \{\tau, \alpha_1, \ldots, \alpha_n\}$, $\Sigma$ is the signature of $H$ plus $\{f_1, \ldots, f_q\}$ and $P = \{\tau \to f_i(\overline{u_{i,j}}) \mid 1 \leqslant i \leqslant q, 1 \leqslant j \leqslant m_i\} \cup \{\alpha_i \to s_{i,j} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k_i\}$.

Note that this definition also applies to Herbrand-sequents (as in Definition 9): then $n = 0$ and we obtain a trivial grammar $\langle N_R, \Sigma, \tau, P\rangle$ with $N_R = \{\tau\}$. Using this grammar, we define the *Herbrand terms* of a Herbrand-sequent as the set $\{t \mid (\tau \to t) \in P\}$. The *Herbrand terms of a cut-free proof $\pi$* are then the Herbrand terms of the Herbrand-sequent extracted from $\pi$ via Theorem 6.

**Example 4.** Consider the sequent

$$P(0,0), \forall x\forall y\left(P(x,y) \supset P\big(s(x),y\big)\right), \forall x\forall y\left(P(x,y) \supset P\big(x,s(y)\big)\right) \to P\big(s^4(0), s^4(0)\big).$$

Abbreviating $P(x,x) \supset P(s^2(x), s^2(x))$ as $F(x)$ we see that $\forall x\, F(x)$ is a useful cut formula that allows to decrease the number of $\forall_1^*$-inferences. A corresponding extended Herbrand-sequent is

$$E = P(0,0), P(\alpha,\alpha) \supset P\big(s(\alpha),\alpha\big), P\big(s(\alpha),\alpha\big) \supset P\big(s(\alpha),s(\alpha)\big), P\big(s(\alpha),s(\alpha)\big) \supset P\big(s^2(\alpha),s(\alpha)\big),$$
$$P\big(s^2(\alpha),\alpha\big) \supset P\big(s^2(\alpha),s^2(\alpha)\big), F(\alpha) \supset \big(F(0) \wedge F\big(s^2(0)\big)\big) \to P\big(s^4(0),s^4(0)\big)$$

The corresponding grammar is $G(E) = \langle\{\tau,\alpha\}, \{0/0, s/1\}, \tau, P\rangle$, with $|E| = |G(E)| = 6$ and

$$P = \left\{\tau \to f_1(\alpha,\alpha) | f_2\big(s(\alpha),\alpha\big) | f_1\big(s(\alpha),s(\alpha)\big) | f_2\big(s^2(\alpha),\alpha\big), \alpha \to 0 | s^2(0)\right\}.$$

In extension of our compact notation for grammars we write

$$(U_1, \ldots, U_q) \circ_{\alpha_1} S_1 \cdots \circ_{\alpha_n} S_n$$

for the grammar $G(H)$ of Definition 11 where $U_i = \{\overline{u_{i,j}} \mid 1 \leqslant j \leqslant m_i\}$ and $S_i = \{s_{i,j} \mid 1 \leqslant j \leqslant k_i\}$. As before, we leave out the $\alpha_i$ if they are obvious from the context. In this notation the function symbols $f_i$ are implicitly specified by the position of $U_i$ in the vector $(U_1, \ldots, U_q)$. Consequently, each $t \in L((U_1, \ldots, U_q) \circ_{\alpha_1} S_1 \cdots \circ_{\alpha_n} S_n)$ has one of the $f_i$ as top-level symbol and these are the only occurrences of $f_i$. For notational convenience and if $l_i = 1$ for all $i$ we sometimes write $L((U_1, \ldots, U_q) \circ_{\alpha_1} S_1 \cdots \circ_{\alpha_n} S_n)$ as a vector of sets of terms in the form $(T_1, \ldots, T_q)$ where $T_i = \{t \in L((U_1, \ldots, U_q) \circ_{\alpha_1} S_1 \cdots \circ_{\alpha_n} S_n) \mid t = f_i(\bar{s}) \text{ for some } \bar{s}\}$.

**Theorem 8.** *If $H$ is an extended Herbrand-sequent of $\Gamma \to \Delta$, then*

$$\left\{F_i[\bar{x}\backslash\bar{t}] \mid 1 \leqslant i \leqslant p,\ f_i(\bar{t}) \in L\big(G(H)\big)\right\} \to \left\{F_i[\bar{x}\backslash\bar{t}] \mid p < i \leqslant q,\ f_i(\bar{t}) \in L\big(G(H)\big)\right\}$$

*is a Herbrand-sequent of $\Gamma \to \Delta$.*

**Proof.** As for Theorem 4. □

**Definition 12.** Let $H$ be a schematic extended Herbrand sequent. Define

$$C_1 = \bigwedge_{i=1}^{p} \bigwedge_{j=1}^{m_i} F_i[\bar{x}\backslash\overline{u_{i,j}}] \wedge \bigwedge_{i=p+1}^{q} \bigwedge_{j=1}^{m_i} \neg F_i[\bar{x}\backslash\overline{u_{i,j}}] \quad \text{and} \quad C_{i+1} = \bigwedge_{j=1}^{k_i} C_i[\alpha_i\backslash s_{i,j}] \quad \text{for } i = 1, \ldots, n.$$

Then

$$\sigma := [X_i\backslash\lambda\alpha_i.C_i]_{i=1}^{n}$$

is called *canonical substitution* of $H$.

**Lemma 4.** *Let $H$ be a schematic extended Herbrand-sequent and $\sigma$ be its canonical substitution. Then $\sigma$ is a solution of $H$.*

**Proof.** As for Lemma 3. □

As in Section 3, the canonical substitution is hence called *canonical solution*.

**Theorem 9.** *$\Gamma \to \Delta$ has an extended Herbrand-sequent $H$ with $|H| = l$ iff there is a totally rigid acyclic tree grammar $G$ with $|G| = l$ s.t.*

$$\left\{F_i[\bar{x}\backslash\bar{t}] \mid 1 \leqslant i \leqslant p,\ f_i(\bar{t}) \in L\big(G(H)\big)\right\} \to \left\{F_i[\bar{x}\backslash\bar{t}] \mid p < i \leqslant q,\ f_i(\bar{t}) \in L\big(G(H)\big)\right\}$$

*is a tautology.*

**Proof.** Analogous to the proof of Theorem 5, using the canonical solution to obtain an extended Herbrand-sequent from a grammar. □

## 5. The method CI

In Section 3 we have shown that, for any rigid acyclic tree grammar $G$ generating the set of Herbrand terms $T$ of a cut-free proof $\varphi$ of a sequent $S$, there exists a solution of the corresponding schematic extended Herbrand sequent $H$, the canonical solution. This canonical solution yields cut formulas and an extended Herbrand sequent $H^*$ of $S$. By Theorem 2 we can construct a proof $\varphi^*$ with cuts from $H^*$. In combining these transformations in a systematic way we obtain a nondeterministic algorithm $\mathrm{CI}(A)$, where $A$ is an algorithm computing a grammar given a set of terms.[2]

We now define $\mathrm{CI}(A)$:

Input: a cut-free proof $\varphi$ of a $\Sigma_1$-sequent $S$.
(1) Compute the set of Herbrand terms $T$ of $\varphi$.
(2) Compute a rigid acyclic tree grammar $G$ with $L(G) = T$ by $A$.
(3) Construct the canonical solution corresponding to $G$.
(4) Construct the proof with the computed cut-formulas.

### 5.1. Inverting cut-elimination

In this section we show that the method CI is complete by proving that, in a suitable sense, it constitutes an inversion of Gentzen's procedure for cut-elimination. To that aim we will strongly rely on the results of [22,23] which show that the language of a grammar is an invariant of cut-elimination. To be more precise we quickly repeat the most central notions from [22,23] here, for full details the interested reader is referred to these papers.

[2] Note that such algorithms $A$ exist: for example, a trivial algorithm computing a minimal grammar can be implemented by simply searching through all the exponentially-many grammar-candidates. In [20, Sec. 5], we describe a more sophisticated $A$ that is more efficient in practice.

**Definition 13.** We denote with $\rightsquigarrow$ the cut-reduction relation defined by allowing the application of the standard reduction rules without any strategy-restriction. The standard reductions include rules such as e.g.

$$\dfrac{\dfrac{\overset{(\pi_1)}{\Gamma \to \Delta, A[x\backslash\alpha]}\,\forall_r \quad \dfrac{\overset{(\pi_2)}{A[x\backslash t], \Pi \to \Lambda}}{\forall x\, A, \Pi \to \Lambda}\,\forall_l}{\Gamma, \Pi \to \Delta, \Lambda}\,\text{cut} \quad \mapsto \quad \dfrac{\overset{(\pi_1[\alpha\backslash t])}{\Gamma \to \Delta, A[x\backslash t]} \quad \overset{(\pi_2)}{A[x\backslash t], \Pi \to \Lambda}}{\Gamma, \Pi \to \Delta, \Lambda}\,\text{cut}\,,$$

see [22, Figure 1] for the complete list. With $\overset{ne}{\rightsquigarrow}$ we denote the *non-erasing part* of $\rightsquigarrow$, i.e. we disallow application of the reduction rule

$$\dfrac{\dfrac{\overset{(\pi_1)}{\Gamma \to \Delta}}{\Gamma \to \Delta, A}\,w_r \quad \overset{(\pi_2)}{A, \Pi \to \Lambda}}{\Gamma, \Pi \to \Delta, \Lambda}\,\text{cut} \quad \mapsto \quad \dfrac{\overset{(\pi_1)}{\Gamma \to \Delta}}{\Gamma, \Pi \to \Delta, \Lambda}\,w^*$$

and its symmetric variant that removes a $w_l$-inference.

**Definition 14.** A proof is called *simple* if every cut is of one of the following forms:

$$\dfrac{\dfrac{\Gamma \to \Delta, A[x\backslash\alpha]}{\Gamma \to \Delta, \forall x\, A}\,\forall_r \quad \forall x\, A, \Pi \to \Lambda}{\Gamma, \Pi \to \Delta, \Lambda}\,\text{cut} \quad \text{or} \quad \dfrac{\Gamma \to \Delta, \exists x\, A \quad \dfrac{A[x\backslash\alpha], \Pi \to \Lambda}{\exists x\, A, \Pi \to \Lambda}\,\exists_l}{\Gamma, \Pi \to \Delta, \Lambda}\,\text{cut} \quad \text{or}$$

$$\dfrac{\Gamma \to \Delta, A \quad A, \Pi \to \Lambda}{\Gamma, \Pi \to \Delta, \Lambda}\,\text{cut}$$

where $A$ is quantifier-free.

Each proof with $\Pi_1 \cup \Sigma_1$-cuts can be pruned to obtain a simple proof by permuting $\forall_r$- and $\exists_l$-inferences down and identifying their eigenvariables when needed. All of the reductions of $\rightsquigarrow$ preserve simplicity with the exception of the following situation:

$$\dfrac{\Gamma \to \Delta, A \quad \dfrac{\dfrac{A, \Pi \to \Lambda, B[x\backslash\alpha]}{A, \Pi \to \Lambda, \forall x\, B}\,\forall_r \quad \forall x\, B, \Sigma \to \Theta}{A, \Pi, \Sigma \to \Lambda, \Theta}\,\text{cut}}{\Gamma, \Pi, \Sigma \to \Delta, \Lambda, \Theta}\,\text{cut}$$

$\rightsquigarrow$

$$\dfrac{\dfrac{\Gamma \to \Delta, A \quad \dfrac{A, \Pi \to \Lambda, B[x\backslash\alpha]}{A, \Pi \to \Lambda, \forall x\, B}\,\forall_r}{\Gamma, \Pi \to \Delta, \Lambda, \forall x\, B}\,\text{cut} \quad \forall x\, B, \Sigma \to \Theta}{\Gamma, \Pi, \Sigma \to \Delta, \Lambda, \Theta}\,\text{cut}$$

where the order of the two cuts is exchanged which motivates the following definition.

**Definition 15.** A *reduction sequence of simple proofs* as one where the above reduction is directly followed by permuting down the $\forall_r$-inference in order to arrive at:

$$\dfrac{\dfrac{\dfrac{\Gamma \to \Delta, A \quad A, \Pi \to \Lambda, B[x\backslash\alpha]}{\Gamma, \Pi \to \Delta, \Lambda, B[x\backslash\alpha]}\,\text{cut}}{\Gamma, \Pi \to \Delta, \Lambda, \forall x\, B}\,\forall_r \quad \forall x\, B, \Sigma \to \Theta}{\Gamma, \Pi, \Sigma \to \Delta, \Lambda, \Theta}\,\text{cut}$$

and symmetrically for the case of $\exists_l$.

We can now state the part of the main result of [23] which is relevant for this paper:

**Theorem 10.** *If $\pi \overset{ne}{\rightsquigarrow} \pi^*$ is a cut-reduction sequence of simple proofs, then $L(G(\pi)) = L(G(\pi^*))$.*

**Proof.** This is the second part of Theorem 7.2 in [23]. □

**Definition 16.** We write $\pi_1 \approx \pi_2$ if $\pi_1$ and $\pi_2$ have the same end-sequent and $G(\pi_1) = G(\pi_2)$.

**Lemma 5.** *$\approx$ is an equivalence relation.*

**Proof.** This follows directly from the definition of $\approx$. □

**Lemma 6.** *For simple $\pi_1, \pi_2$ we have $\pi_1 \approx \pi_2$ iff the schematic extended Herbrand-sequents of $\pi_1$ and $\pi_2$ are identical.*

**Proof.** The left-to-right direction follows from the fact that a schematic extended Herbrand-sequent is uniquely defined by a grammar and the end-sequent. The right-to-left direction follows from being able to read off the grammar from a schematic extended Herbrand-sequent. □

We write $[\pi]$ for the $\approx$-equivalence class of $\pi$, i.e. $[\pi] = \{\psi \text{ simple proof} \mid \psi \approx \pi\}$. For a set $P$ of simple proofs we write ${}^{P}/_{\approx}$ for $\{[\pi] \mid \pi \in P\}$. This notation will, in particular, be used for the set of results of a non-deterministic algorithm such as CI(GG). Here GG refers to the non-deterministic grammar-guessing algorithm which is given a finite tree language $L$ as input and non-deterministically returns any grammar $G$ with $L(G) = L$. Consequently, ${}^{\mathrm{CI(GG)}(\pi^*)}/_{\approx}$ is the set of $\approx$-equivalence classes reachable by applying CI(GG) to $\pi^*$. We can now state the inversion theorem:

**Theorem 11.** *If $\pi \overset{ne}{\rightsquigarrow} \pi^*$ is a cut-reduction sequence of simple proofs and $\pi^*$ is cut-free, then $[\pi] \in {}^{\mathrm{CI(GG)}(\pi^*)}/_{\approx}$.*

**Proof.** By definition of the Herbrand-sequent we have

$$H(\pi^*) = \left\{F_i[\bar{x}\backslash\bar{t}] \mid 1 \leqslant i \leqslant p,\ f_i(\bar{t}) \in L\left(G(\pi^*)\right)\right\} \to \left\{F_i[\bar{x}\backslash\bar{t}] \mid p < i \leqslant q,\ f_i(\bar{t}) \in L\left(G(\pi^*)\right)\right\}.$$

By Theorem 10 we have $L(G(\pi^*)) = L(G(\pi))$ and hence

$$H(\pi^*) = \left\{F_i[\bar{x}\backslash\bar{t}] \mid 1 \leqslant i \leqslant p,\ f_i(\bar{t}) \in L\left(G(\pi)\right)\right\} \to \left\{F_i[\bar{x}\backslash\bar{t}] \mid p < i \leqslant q,\ f_i(\bar{t}) \in L\left(G(\pi)\right)\right\}.$$

Therefore $G(\pi)$ can be obtained from the nondeterministic grammar-guessing algorithm GG applied to $H(\pi^*)$. Let $\psi$ be the proof obtained from the canonical solution of $G(\pi)$, then $\psi \in \mathrm{CI(GG)}(\pi^*)$ and as $G(\psi) = G(\pi)$ we have $\psi \approx \pi$ and therefore $[\pi] \in {}^{\mathrm{CI(GG)}(\pi^*)}/_{\approx}$. □

### 5.2. Proof compression

We will prove in this section that application of CI($A$) to a sequence of cut-free proofs $\varrho_n$ of sequents $S_n$ can result in an exponential compression of $\varrho_n$. But we should take care with respect to which *complexity measure* the proofs are compressed. In fact, the steps (1)–(3) and (5) yield proofs $\chi_n$ of $S_n$ with cuts such that $|\varrho_n|_q$ is exponential in $|\chi_n|_q$, resulting in an exponential compression of quantifier complexity.

As input we take a sequence of shortest cut-free proofs $\varrho_n$ of the sequents

$$s_n\colon Pa, \forall x(Px \supset Pfx) \to Pf^{2^{n+1}}a$$

from Section 2.

We apply step (1) to $\varrho_n$ and obtain a sequence of the corresponding minimal Herbrand sequents

$$s'_n\colon Pa, (Ps \supset Pfs)_{s\in T_n} \to Pf^{2^{n+1}}a,$$

where

$$T_n = \left\{a, fa, \ldots, f^{2^{n+1}-1}a\right\}.$$

Note that the quantifier complexity of any cut-free proof of $S_n$ is $\geqslant 2^{n+1}$ and thus exponential in $n$.

In step (2), $A$ computes the grammars

$$G_n\colon \{\alpha_1, f\alpha_1\} \circ_{\alpha_1} \left\{\alpha_2, f^2\alpha_2\right\} \circ_{\alpha_2} \cdots \circ \left\{\alpha_n, f^{2^{n-1}}\alpha_n\right\} \circ_{\alpha_n} \left\{a, f^{2^n}a\right\},$$

where the $\alpha_i$ are variables and $a$ is a constant symbol.

We now move to step (3) and compute the canonical solution of the schematic extended Herbrand sequent $H_n$ corresponding to $G_n$ where

$$H_n = Pa, (Ps \supset Pfs)_{s\in S_0}, X_1\alpha_1 \supset \bigwedge_{s\in S_1} X_1 s, \ldots, X_n\alpha_n \supset \bigwedge_{s\in S_n} X_n s \to Pf^{2^{n+1}}a.$$

The canonical solution of $H_n$ is

$$\theta_n\colon [X_1 \setminus \lambda\alpha_1.C_1 \mid, \ldots, X_n \setminus \lambda\alpha_n.C_n]$$

where

$$F = Pa \wedge (Px \supset Pfx) \wedge \neg Pf^{2^{n+1}}a,$$
$$C_1 = \bigwedge_{s \in S_0} F\{x \setminus s\},$$
$$C_{i+1} = \bigwedge_{s \in S_i} C_i\{\alpha_i \setminus s\} \quad \text{for } 1 \leqslant i < n.$$

Now let $H_n^*: H_n\theta_n$ be the corresponding extended Herbrand sequent. Then $|H_n^*| = 2(n+1)$.

Step (4): By Theorem 2 we can construct proofs $\chi_n$ of $S_n$ with $|\chi_n|_q = |H_n^*| = 2(n+1)$. As $|\varrho_n|_q \geqslant 2^{n+1}$ we obtain

$$|\varrho_n|_q \geqslant 2^{|\chi_n|_q/2},$$

and so $|\varrho_n|_q$ is exponential in $|\chi_n|_q$. Finally we have obtained an exponential compression of quantifier complexity.

Note that an exponential compression of quantifier complexity does not in general imply an exponential compression of *proof length* (taking into account *all* logical inferences). We remark here that such a compression can be obtained in a systematic way by an extended version of CI which searches for a solution of the schematic extended Herbrand sequent that is smaller than the canonical solution, see [20, Sec. 7.2.2].

## 6. Conclusion and future work

We have described a method for the inversion of Gentzen's cut-elimination method by the introduction of quantified cuts into an existing proof. Our method is based on separating the problem into two phases: (1) the computation of a tree grammar for a language and (2) finding a solution of a unification problem.

The work presented in this paper is only a first step and opens up several important directions for future work: a straightforward extension of this algorithm is to use blocks of quantifiers in the cut-formulas. This ability is useful for obtaining additional abbreviations. An equally obvious – but less straightforward – extension of this method is to cover cut-formulas with quantifier alternations. As a prerequisite for this work, the extension of the connection between cut-elimination and tree languages established in [18] to the corresponding class of formulas is necessary.

On the empirical side an important aspect of future work will be to assess the abilities for compression of proofs produced by theorem provers. A first step in this direction has been taken in [19], where we have carried out large-scale experiments with proofs from the TPTP-library [37]. An investigation of proofs from the SMT-LIB [5] is left for future work. Additional features that we consider important for such applications are to include the ability to work modulo simple theories and to systematically compute grammars whose language is a superset of the given set of terms.

On the theoretical side, we could only scratch the surface of many questions in this paper: What is the complexity of grammar computation? What are good exact algorithms? Can we find incompressible tree languages? And more generally: study the complexity of cut-free proofs along the lines of measures such as automatic complexity [33] and automaticity [32]. Also the unification problem poses a number of interesting theoretical challenges: Is the general unification problem of monadic predicate variables modulo propositional logic decidable? If yes, what is its complexity, what are good algorithms? What is the structure of the solution space of unification problems induced by cut-introduction? How can we navigate this structure systematically to find solutions of minimal size? How can we prove lower bounds on the size of such solutions?

## References


[1] Franz Baader, On the complexity of Boolean unification, Inform. Process. Lett. 67 (1998) 215–220.
[2] Matthias Baaz, Stefan Hetzl, Daniel Weller, On the complexity of proof deskolemization, J. Symbolic Logic 77 (2) (2012) 669–686.
[3] Matthias Baaz, Alexander Leitsch, On Skolemization and proof complexity, Fund. Inform. 20 (4) (1994) 353–379.
[4] Matthias Baaz, Richard Zach, Algorithmic structuring of cut-free proofs, in: Computer Science Logic, CSL, 1992, in: Lecture Notes in Computer Science, vol. 702, Springer, 1993, pp. 29–42.
[5] Clark Barrett, Aaron Stump, Cesare Tinelli, The Satisfiability Modulo Theories Library (SMT-LIB), www.SMT-LIB.org, 2010.
[6] George Boolos, Don't eliminate cut, J. Philos. Logic 13 (1984) 373–378.
[7] Alan Bundy, The automation of proof by mathematical induction, in: Andrei Voronkov, John Alan Robinson (Eds.), Handbook of Automated Reasoning, vol. 1, Elsevier, 2001, pp. 845–911.
[8] Alan Bundy, David Basin, Dieter Hutter, Andrew Ireland, Rippling: Meta-Level Guidance for Mathematical Reasoning, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 2005.
[9] Simon Colton, Automated theory formation in pure mathematics, PhD thesis, University of Edinburgh, 2001.
[10] Simon Colton, Automated Theory Formation in Pure Mathematics, Springer, 2002.
[11] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata: techniques and applications, available on http://www.grappa.univ-lille3.fr/tata, 2007, release October, 12th 2007.
[12] William Craig, Three uses of the Herbrand–Gentzen theorem in relating model theory and proof theory, J. Symbolic Logic 22 (3) (1957) 269–285.
[13] Marcelo Finger, Dov Gabbay, Equal rights for the cut: computable non-analytic cuts in cut-based proofs, Log. J. IGPL 15 (5–6) (2007) 553–575.
[14] Ferenc Gécseg, Magnus Steinby, Tree languages, in: G. Rozenberg, A. Salomaa (Eds.), Beyond Words, in: Handbook of Formal Languages, vol. 3, Springer, 1997, pp. 1–68.
[15] Gerhard Gentzen, Untersuchungen über das logische Schließen, Math. Z. 39 (1934–1935) 176–210, 405–431.
[16] Stefan Hetzl, Proofs as tree languages, submitted for publication, preprint available at http://hal.archives-ouvertes.fr/hal-00613713/.

[17] Stefan Hetzl, Describing proofs by short tautologies, Ann. Pure Appl. Logic 159 (1–2) (2009) 129–145.
[18] Stefan Hetzl, Applying tree languages in proof theory, in: Adrian-Horia Dediu, Carlos Martín-Vide (Eds.), Language and Automata Theory and Applications, LATA, 2012, in: Lecture Notes in Computer Science, vol. 7183, Springer, 2012.
[19] Stefan Hetzl, Alexander Leitsch, Giselle Reis, Janos Tapolczai, Daniel Weller, Introducing quantified cuts in logic with equality, in: International Joint Conference on Automated Reasoning, IJCAR-14, 2014, in press.
[20] Stefan Hetzl, Alexander Leitsch, Giselle Reis, Daniel Weller, Algorithmic introduction of quantified cuts, CoRR, arXiv:1401.4330 [abs], 2014.
[21] Stefan Hetzl, Alexander Leitsch, Daniel Weller, Towards algorithmic cut-introduction, in: Logic for Programming, Artificial Intelligence and Reasoning, LPAR-18, in: Lecture Notes in Computer Science, vol. 7180, Springer, 2012, pp. 228–242.
[22] Stefan Hetzl, Lutz Straßburger, Herbrand-confluence for cut-elimination in classical first-order logic, in: Patrick Cégielski, Arnaud Durand (Eds.), Computer Science Logic, CSL, 2012, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 16, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 320–334.
[23] Stefan Hetzl, Lutz Straßburger, Herbrand-confluence, Log. Methods Comput. Sci. 9 (4) (2013).
[24] David Hilbert, Paul Bernays, Grundlagen der Mathematik II, Springer, 1939.
[25] Andrew Ireland, Alan Bundy, Productive use of failure in inductive proof, J. Automat. Reason. 16 (1–2) (1996) 79–111.
[26] Florent Jacquemard, Francis Klay, Camille Vacher, Rigid tree automata, in: Adrian Horia Dediu, Armand-Mihai Ionescu, Carlos Martín-Vide (Eds.), Language and Automata Theory and Applications, LATA, 2009, in: Lecture Notes in Computer Science, vol. 5457, Springer, 2009, pp. 446–457.
[27] Florent Jacquemard, Francis Klay, Camille Vacher, Rigid tree automata and applications, Inform. and Comput. 209 (2011) 486–512.
[28] Moa Johansson, Lucas Dixon, Alan Bundy, Conjecture synthesis for inductive theories, J. Automat. Reason. 47 (3) (2011) 251–289.
[29] Ursula Martin, Tobias Nipkow, Boolean unification – the story so far, J. Symbolic Comput. 7 (3–4) (1989) 275–293.
[30] Dale Miller, Vivek Nigam, Incorporating tables into proofs, in: 16th Conference on Computer Science and Logic, CSL07, in: Lecture Notes in Computer Science, vol. 4646, Springer, 2007, pp. 466–480.
[31] V.P. Orevkov, Lower bounds for increasing complexity of derivations after cut elimination, Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. 88 (1979) 137–161.
[32] Jeffrey Shallit, Yuri Breitbart, Automaticity I: properties of a measure of descriptional complexity, J. Comput. System Sci. 53 (1996) 10–25.
[33] Jeffrey Shallit, Ming-Wei Wang, Automatic complexity of strings, J. Autom. Lang. Comb. 6 (4) (2001) 537–554.
[34] Volker Sorge, Simon Colton, Roy McCasland, Andreas Meier, Classification results in quasigroup and loop theory via a combination of automated reasoning tools, Comment. Math. Univ. Carolin. 49 (2) (2008) 319–339.
[35] Volker Sorge, Andreas Meier, Roy McCasland, Simon Colton, Automatic construction and verification of isotopy invariants, J. Automat. Reason. 40 (2–3) (2008) 221–243.
[36] Richard Statman, Lower bounds on Herbrand's theorem, Proc. Amer. Math. Soc. 75 (1979) 104–107.
[37] Geoff Sutcliffe, The TPTP problem library and associated infrastructure: the FOF and CNF parts, v3.5.0, J. Automat. Reason. 43 (4) (2009) 337–362.
[38] Gaisi Takeuti, Proof Theory, 2nd edition, North-Holland, 1987.
[39] Anne S. Troelstra, Helmut Schwichtenberg, Basic Proof Theory, second edn., Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 2000.
[40] Jiři Vyskočil, David Stanovský, Josef Urban, Automated proof compression by invention of new definitions, in: E.M. Clark, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence and Reasoning, LPAR-16, in: Lecture Notes in Computer Science, vol. 6355, Springer, 2010, pp. 447–462.
[41] Bruno Woltzenlogel Paleo, Atomic cut introduction by resolution: proof structuring and compression, in: E.M. Clark, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence and Reasoning, LPAR-16, in: Lecture Notes in Computer Science, vol. 6355, Springer, 2010, pp. 463–480.