Information and Computation 259 (2018) 191–213

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

On the compressibility of finite languages and formal proofs ☆

Sebastian Eberhard, Stefan Hetzl *

Institute of Discrete Mathematics and Geometry, Vienna University of Technology, Wiedner Hauptstraße 8-10, 1040 Wien, Austria

ARTICLE INFO

Article history:
Received 15 December 2015
Available online 5 September 2017

ABSTRACT

We consider the minimal number of productions needed for a grammar to cover a finite language L as the grammatical complexity of L. We study this measure for several types of word and tree grammars and show that it is closely connected to well-known measures for the complexity of formal proofs in first-order predicate logic.
We construct an incompressible sequence of finite word languages and transfer this and several other results about the complexity of word and tree languages to formal proofs.
© 2017 Elsevier Inc. All rights reserved.

1. Introduction

In grammar-based compression, context-free grammars that generate exactly one word are used for representing the input text. The smallest grammar problem asks for the smallest context-free grammar that generates a given word. Its decision version is known to be NP-complete [1], see [2] for the case of a bounded alphabet. However, there is a number of fast algorithms which are practically useful [3–5] or achieve good approximation ratios [6–10]. Grammar-based compression also has the considerable practical advantage that many operations can be performed directly on the compressed representation; see [11].

We are interested in the problem of simultaneously compressing a finite set of words by a single grammar. Traditionally, the grammatical complexity of a finite language L is defined as the minimal number of productions of a grammar G with $L(G) = L$. Each class of grammars thus gives rise to a measure of descriptional complexity. The study of the grammatical complexity of finite languages has been initiated in [12] and continued in [13–16].

Our motivation for investigating this problem is rooted in proof theory and automated deduction: as shown in [17], there is an intimate relationship between a certain class of formal proofs (those with Π_1-cuts) in first-order predicate logic and a certain class of grammars (totally rigid acyclic tree grammars). In particular, the number of production rules in the grammar characterises the number of certain inference rules in the proof. This relationship has been exploited in a method for proof compression whose central combinatorial step is a grammar-based compression of a finite tree language [18–20].

The proof-theoretic application of our work requires a modification of the traditional problem: we are looking for a minimal grammar G s.t. $L(G) \supseteq L$ where L is the finite input language. This is the case because L describes a disjunction which is required to be a tautology (a so-called Herbrand-disjunction, see [21,22]) and if $L' \supseteq L$, then L' also describes a tautology. This condition is similar to (but different from) the one imposed on cover automata [23]: there an automaton A is sought s.t. $L(A) \supseteq L$, but in addition it is required that $L(A) \setminus L$ consists only of words longer than any word in L.

In this paper we consider the minimal number of productions needed for a grammar to cover a finite language L as the grammatical complexity of L. We study this measure for several types of word and tree grammars and show that

☆ Research supported by the Vienna Science Fund (WWTF) project VRG12-004.
* Corresponding author.
E-mail addresses: sebastian.eberhard84@gmail.com (S. Eberhard), stefan.hetzl@tuwien.ac.at (S. Hetzl).

http://dx.doi.org/10.1016/j.ic.2017.09.001
0890-5401/© 2017 Elsevier Inc. All rights reserved.

it is strongly related to well-known measures for the complexity of formal proofs. The central technical results are the construction of an incompressible sequence of finite (word) languages and its use for obtaining a lower bound on the complexity of proofs with $\overline{\Pi}_1$-cuts in terms of the complexity of the shortest cut-free proof. The interest in such a result is motivated by the experience that the length of proofs with cuts is notoriously difficult to control (for propositional logic this is considered the central open problem in proof complexity [24]).

This paper extends [25] in the following respects: we prove the lower bound on an enlarged class of proofs: instead of proofs with Π_1-cuts (i.e., lemmas of the form $\forall x\, A$ for A quantifier-free) we treat proofs with $\overline{\Pi}_1$-cuts here (i.e., lemmas of the form $\forall x_1 \cdots \forall x_n\, A$ for A quantifier-free). This necessitates the introduction of a more general class of tree grammars: vectorial totally rigid acyclic tree grammars. In this paper, we also carry out a thorough investigation of the various types of tree grammars involved, including several results on the relative complexity of them. In contrast to [25], this paper also contains an introduction to the proof-theoretic background and complete proofs of the proof-theoretic results.

In Section 2 we introduce some basic notions concerning the grammatical complexity of finite languages. In Section 3 we construct an incompressible sequence of word languages. In Section 4 we study tree grammars of proof-theoretic relevance. We investigate their relationship to each other and to the word grammars of Section 3. In Section 5 we introduce the basic notions and results of proof theory which are relevant to this paper. In Section 6 we establish the relationship between the complexity of formal proofs and the grammatical complexity of finite languages. Finally, in Section 7, we transfer several results about grammatical complexity, including the incompressibility-result, to proof theory.

2. Grammatical complexity of finite languages

Definition 1. A *context-free grammar (CFG)* is a 4-tuple $G = (N, \Sigma, P, S)$ where N is a finite set of nonterminals, Σ is a finite alphabet, $S \in N$ is the starting symbol and P is a finite set of productions of the form $A \to w$ where $A \in N$ and $w \in (\Sigma \cup N)^*$.

As usual, the one-step derivation relation $\Longrightarrow_G$ of G is defined by $u \Longrightarrow_G v$ iff there is a production $A \to w$ in G s.t. v is obtained from u by replacing an occurrence of A by w. The derivation relation $\Longrightarrow_G^*$ is the reflexive and transitive closure of $\Longrightarrow_G$ and the language of G is $L(G) = \{w \in \Sigma^* \mid S \Longrightarrow_G^* w\}$. We omit the subscript G if the grammar is clear from the context.

Definition 2. A *right-linear grammar* is a context-free grammar (N, Σ, P, S) s.t. all productions in P are of the form $A \to vB$ or $A \to v$ for $A, B \in N$ and $v \in \Sigma^*$.

It is well-known, see e.g., [26], that the languages generated by right-linear grammars are exactly the regular languages.

Definition 3. Let $G = (N, \Sigma, P, S)$ be a context-free grammar. The relation $<_G^1$ on N is defined as follows: $A <_G^1 B$ iff there is a production $A \to w$ in P s.t. B occurs in w. The relation $<_G$ is defined as the transitive closure of $<_G^1$. We say that G is cyclic (respectively acyclic) iff $<_G$ is.

We abbreviate "right-linear acyclic grammar" as "RLAG". Let $A \in N$; then a production whose left hand side is A is called A-production. We write P_A for the set of A-productions in P. For $N' \subseteq N$ we define $P_{N'} = \bigcup_{A \in N'} P_A$. For a language L and a CFG G we say that G covers L if $L(G) \supseteq L$. The size of a CFG $G = (N, \Sigma, P, S)$ is defined as $|G| = |P|$. The length of a right-linear production rule $A \to wB$ or $A \to w$ for $w \in \Sigma^*$ is defined as $|w|$.

Definition 4. The *RLA-complexity* of a finite language L is defined as $\mathrm{RLAc}(L) = \min\{|G| \mid G \text{ RLA s.t. } L(G) \supseteq L\}$. A finite language L is called *RLA-compressible* if $\mathrm{RLAc}(L) < |L|$ and *RLA-incompressible* otherwise.

Note that $\mathrm{RLAc}(L) \leq |L|$ for all finite languages L since L can be generated by a trivial grammar with $|L|$ production rules. All descriptional complexity measures in this paper will be written as $\mathrm{Xc}(\cdot)$ for some formalism X, e.g., $X = \mathrm{RLA}$ as above.

Definition 5. A sequence $(L_n)_{n \geq 1}$ of finite languages is called *RLA-incompressible* if there is an $M \in \mathbb{N}$ s.t. for all $n \geq M$ the language L_n is RLA-incompressible. A sequence $(L_n)_{n \geq 1}$ of finite languages is called *RLA-compressible* if for every $M \in \mathbb{N}$ there is an $n \geq M$ s.t. L_n is RLA-compressible.

We will use the above definition of X-compressibility of a sequence based on the X-compressibility of an element of the sequence also for descriptional complexity measures other than $X = \mathrm{RLA}$.

A variant of our measure of grammatical complexity, the equality formulation, consists in asking for a minimal grammar G with $L(G) = L$. As explained in the introduction, the cover formulation is motivated by our proof-theoretic application; see Section 6. However, the main result on incompressibility also applies to the equality formulation; see Corollary 1. Incompressible finite languages in the sense of the equality formulation have been studied before: [13] considers the sequence

$(L_n)_{n\geq 1}$ induced by an infinite language L by $L_n := \{w \in L \mid |w| \leq n\}$. If L is regular, linear or context-free, then $(L_n)_{n\geq 1}$ is compressible by the respective formalism. This result is applied in [13] to give a proof that $L = \{a^i b^i c^i \mid i \geq 1\}$ is not context-free based on the CF-incompressibility of $(L_{3n})_{n\geq 1}$. In [16], the author considers the language $L = \{a_i a_j \mid 1 \leq i, j \leq n, i \neq j\}$ in the alphabet $\{a_1, \ldots, a_n\}$ and gives a graph-theoretic characterisation of those $K \subseteq L$ which are CF-incompressible.

3. Incompressible word languages

Note that it is trivial to construct an incompressible sequence of languages of constant size, e.g., $L_n = \{a\}$ for a letter a. It is also trivial to construct a sequence of incompressible languages in an infinite alphabet, e.g., $L_n = \{a_1, \ldots, a_n\}$ for letters $a_1, a_2, \ldots$. Consequently, in this section we will construct an incompressible sequence of languages of unbounded size over a finite alphabet.

3.1. Reduced grammars

In this section we will make some preparatory observations on the structure of RLAGs which compress finite languages, leading to the notion of strong compressibility.

Definition 6. Let $G = (N, \Sigma, P, S)$ be an RLAG. Then a rule $A \to w$ is called *trivial* if $A = S$ and $w \in \Sigma^*$. We define $G_t = (N, \Sigma, P_t, S)$ where P_t is the set of trivial rules of G.

We say that a word u is a *subword* of a word w if there are words v_1, v_2 s.t. $w = v_1 u v_2$. We say that a word u is a *prefix* of a word w if there is a word v s.t. $w = uv$.

Definition 7. Let L be a finite language and G be an RLAG that covers L. Then G is called *reduced w.r.t. L* if for every non-trivial production rule $A \to wB$ or $A \to w$ of G there are distinct $u, v \in L \setminus L(G_t)$ s.t. w is a subword of both u and v.

If the language is clear from the context we will say "reduced" instead of "reduced w.r.t. L". Intuitively, in a grammar which is reduced w.r.t. L all production rules are either trivial or useful for the compression of L. The following lemma shows that, for questions of compressibility, it is sufficient to consider reduced RLAGs.

Lemma 1. *Let L be a finite language and G be an RLAG which covers L. Then there is a reduced RLAG G^* which covers L and satisfies $|G^*| \leq |G|$.*

Proof. If G is already reduced we are done. If not, let $G = (N, \Sigma, P, S)$. Then P contains a non-trivial production rule $r\colon A \to wB$ or $r\colon A \to w$ s.t. w is subword of at most one $v \in L \setminus L(G_t)$. Define $P' = P \setminus \{r\}$. If w is subword of no $v \in L$, then r cannot be used in a derivation of a word in L and we define $G' = (N, \Sigma, P', S)$. So suppose that w is a subword of a $v \in L$. If $v \in L(G_t)$, then define $G' = (N, \Sigma, P', S)$. If $v \in L \setminus L(G_t)$, then define $G' = (N, \Sigma, P' \cup \{S \to v\}, S)$. Then G' still covers L and satisfies $|G'| \leq |G|$. Iterating this step will terminate (as the number of non-trivial production rules decreases) and will finish with a reduced RLAG G^* which covers L and satisfies $|G^*| \leq |G|$. □

Definition 8. A language L is called *strongly RLA-compressible* if there is a reduced RLAG G without trivial rules s.t. G covers L and $|G| < |L|$. A sequence $(L_n)_{n\geq 1}$ of finite languages is called *strongly RLA-compressible* if for every $M \in \mathbb{N}$ there is an $n \geq M$ s.t. L_n is strongly RLA-compressible.

Lemma 2. *Let L be an RLA-compressible language, then there is a language $L' \subseteq L$ which is strongly RLA-compressible.*

Proof. Let G be an RLAG which covers L and satisfies $|G| < |L|$. By Lemma 1 there is an RLAG G' which is reduced w.r.t. L and covers L s.t. $|G'| < |L|$. Let G'_{nt} be G' without trivial production rules and let $L' = L(G'_{nt}) \cap L$. Then clearly $L' \subseteq L$ and G'_{nt} does not contain trivial rules. It remains to show that G'_{nt} is reduced w.r.t. L'. To that aim, let $A \to wB$ or $A \to w$ be a non-trivial rule of G_{nt}, then it is also a non-trivial rule of G' hence by reducedness of G' for L we get distinct $u, v \in L \setminus L(G'_t)$ s.t. w is subword of both of them. Now as G' covers L and as $L(G') = L(G'_t) \cup L(G'_{nt})$ (it follows from acyclicity that S cannot appear on the rhs of rules) we have $u, v \in L(G'_{nt})$ hence $u, v \in L'$. □

3.2. Segmented languages

From this section on we will frequently use the alphabet $\Sigma = \{0, 1, s\}$. The letters 0 and 1 will be used for the binary representation of natural numbers, while the letter s will serve as a separator. The RLA-incompressible sequence of languages which we are about to construct is a sequence of segmented languages, a notion which we define now and study in this section.

Definition 9. Let $\Sigma = \{0, 1, \mathrm{s}\}$. A word $w \in \Sigma^*$ s.t. $w = (\mathrm{s}v)^k$ for some $k \geq 1$ and some $v \in \{0,1\}^+$ is called *segmented word*. The word v is called the *building block* of w. Occurrences of v in w are called *segments*. A segmented word $(\mathrm{s}v)^k$ where $|v| = l$ is called a (k,l)-*segmented word*. A language consisting of (k,l)-segmented words is called a (k,l)-*segmented language*.

The following lemma states the key property of segmented languages: long rules are not useful for compression.

Lemma 3. *Let L be a finite (k,l)-segmented language and G be a reduced RLAG that covers L. Then every non-trivial rule of G has length at most l.*

Proof. Suppose that G contains a non-trivial rule $A \to wB$ or $A \to w$ with $|w| > l$ and let w' be the prefix of w of length $l+1$. As G is reduced there are distinct (k,l)-segmented words u, v s.t. w, and hence also w', is subword of both u and v. Since $|w'| = l+1$, there are $w_1, w_2 \in \{0,1\}^*$ s.t. $w' = w_1' \mathrm{s} w_2'$ and $|w_1'| + |w_2'| = l$. But as w' is a subword of u, the building block of u is $w_2'w_1'$ and analogously: as w' is a subword of v, the building block of v is $w_2'w_1'$. Since both, u and v, are (k,l)-segmented words we have $u = v$. Contradiction. □

Lemma 4. *Let L be a finite (k,l)-segmented language that is strongly RLA-compressible. Then $k < |L| - 1$.*

Proof. Note that we have $|w| = k(l+1)$ for all $w \in L$. Let G be a reduced RLAG which compresses L. Then by Lemma 3 every rule in G has length at most l. Hence for deriving a single $w \in L$ the grammar G needs at least $\frac{|w|}{l} > \frac{|w|}{l+1} = k$ rules. Since L is compressible we have $|G| < |L|$ and $L \neq \emptyset$. So, since there is a $w \in L$, we have $k < |G| < |L|$. □

Lemma 5. *Let $(L_n)_{n\geq 1}$ be an RLA-compressible sequence of finite languages s.t. L_n is (k_n, l_n)-segmented and $(k_n)_{n\geq 1}$ is unbounded. Then there is a sequence of finite languages $(L_n')_{n\geq 1}$ s.t.*

1. *$L_n' \subseteq L_n$ for all $n \geq 1$,*
2. *$(L_n')_{n\geq 1}$ is strongly RLA-compressible, and*
3. *$(|L_n'|)_{n\geq 1}$ is unbounded.*

Proof. The pointwise application of Lemma 2 to an infinite subsequence of $(L_n)_{n\geq 1}$ that consists of RLA-compressible languages yields a sequence $(L_n')_{n\geq 1}$ satisfying 1 and 2. By Lemma 4 we have $k_i < |L_i'|$ for infinitely many $i \in \mathbb{N}$, which, together with the assumption that $(k_n)_{n\geq 1}$ is unbounded, entails 3. □

The following Lemma 6 applies the uselessness of long rules for compression to provide an upper bound on the number of segments which a strongly compressing RLAG covers. This upper bound is a key ingredient of the proof of the incompressibility result.

Definition 10. Let $G = (N, \Sigma, P, S)$ be an RLAG. Let $w \in L(G)$ be a (k,l)-segmented word with building block v and let $i \in \{1, \ldots, k\}$. Let $w_0 = (\mathrm{s}v)^{i-1}$ and $w_1 = (\mathrm{s}v)^{k-i}$, then $w = w_0 \mathrm{s} v w_1$. Let δ be a derivation of w w.r.t. G; then it is of the form

$$S \Longrightarrow^* w_0' A_1 \Longrightarrow w_0 \mathrm{s} v' A_2 \Longrightarrow \cdots \Longrightarrow w_0 \mathrm{s} v'' A_n \Longrightarrow w_0 \mathrm{s} v w_1' A_{n+1} \Longrightarrow^* w$$

for some $A_1, \ldots, A_n, A_{n+1} \in N$ with v', v'' being prefixes of v, w_0' a prefix of w_0 and w_1' a prefix of w_1. We define $\mathrm{nonterms}(w, i, \delta) = \{A_j \mid 1 \leq j \leq n\}$.

So, informally, $\mathrm{nonterms}(w, i, \delta)$ is the set of non-terminals which is involved in the derivation of the i-th segment of w in δ. For a finite (k,l)-segmented language L, we can consider $L \times \{1, \ldots, k\}$ as the *segments of* L. We will now, for an arbitrary set of nonterminals N_0, prove an upper bound on the number of segments (w, i) of L which can be covered using nonterminals from N_0.

Lemma 6. *Let L be a finite (k,l)-segmented language that is strongly compressed by an RLAG $G = (N, \Sigma, P, S)$. For each $w \in L$ fix a derivation δ_w of w w.r.t. G. Let $N_0 \subseteq N$, let $P_0 = P_{N_0}$ and let $S_0 = \{(w, i) \in L \times \{1, \ldots, k\} \mid \mathrm{nonterms}(w, i, \delta_w) \subseteq N_0\}$. Then we have $|S_0| \leq 2^{|P_0|} \cdot |P_0|$.*

Proof. For $w \in L$ define $S_{w,0} = \{i \in \{1, \ldots, k\} \mid \mathrm{nonterms}(w, i, \delta_w) \subseteq N_0\}$. By Lemma 3 every rule of G has length at most l. Due to acyclicity, each $A \in N_0$ can be used at most once in a derivation. Therefore by using all $A \in N_0$ in a derivation one can generate at most $|N_0| \cdot l$ terminal symbols, and hence at most $|N_0|$ segments. We thus obtain $|S_{w,0}| \leq |N_0|$.

Furthermore, define $L_0 = \{w \in L \mid \exists i \text{ s.t. } (w, i) \in S_0\}$. Let $P^* \subseteq P_0$ s.t. P^* contains exactly one production for each nonterminal of N_0 and note that there are at most $2^{|P_0|}$ such P^*. If P^* permits deriving a word that contains a subword $v \in \{0,1\}^l$, then the choice of P^* uniquely determines a word $w \in L$. If P^* does not allow deriving such a word, then P^* may be used

in the derivations δ_w of several $w \in L$; however, it does not contribute to any of its $S_{w,0}$. Therefore we have $|L_0| \leq 2^{|P_0|}$. Putting these two results together, we obtain $|S_0| = \sum_{w \in L_0} |S_{w,0}| \leq |L_0| \cdot |N_0| \leq 2^{|P_0|} \cdot |P_0|$. □

3.3. Ordered grammars

In an RLAG G the ordering $<_G$ is acyclic but, in general, not linear. For technical purposes it will be useful to fix a linearisation of $<_G$ and a corresponding linear order of the productions of G. To that aim we introduce the notion of ordered grammar.

Definition 11. A *right-linear ordered grammar (RLOG)* is a tuple $G = (N, \Sigma, P, A_1)$ where N is a list $A_1, \ldots, A_n$ of nonterminals, P is a list $p_1, \ldots, p_m$ of productions s.t.

1. $G' = (\{A_1, \ldots, A_n\}, \Sigma, \{p_1, \ldots, p_m\}, A_1)$ is an RLAG,
2. if $A_i <_{G'} A_j$ then $i < j$, and
3. $p_1, \ldots, p_m$ are grouped by their left-hand sides, i.e.: $p_1, \ldots, p_m = q_{1,1}, \ldots, q_{1,k_1}, \ldots, q_{n,1}, \ldots, q_{n,k_n}$ where $\{q_{i,1}, \ldots, q_{i,k_i}\} = P_{A_i}$ for all $i \in \{1, \ldots, n\}$.

We say that a RLOG compresses a language L, is reduced w.r.t. L, etc., if the underlying RLAG satisfies the respective property.

Definition 12. Let $G = ((A_1, \ldots, A_n), \Sigma, P, S)$ be a RLOG. Let $w \in L(G)$ be a (k,l)-segmented word, let $i \in \{1, \ldots k\}$ and let δ be a derivation of w w.r.t. G. Let

$$m_1 = \min\{j \in \{1, \ldots, n\} \mid A_j \in \text{nonterms}(w, i, \delta)\},$$
$$m_2 = \max\{j \in \{1, \ldots, n\} \mid A_j \in \text{nonterms}(w, i, \delta)\},$$

and $\text{cost}(w, i, \delta) = \sum_{j=m_1}^{m_2} |P_{A_j}|$.

Note that the cost of the i-th segment of a word w also takes those nonterminals into account which are not used in the derivation δ of w. The following lemma shows that in a strongly compressed segmented language, many segments are cheap.

Lemma 7. *Let L be a finite (k,l)-segmented language and let G be a RLOG that strongly compresses L. Let $w \in L$ and δ be a derivation of w w.r.t. G. Then for at least half of the $i \in \{1, \ldots, k\}$, we have* $\text{cost}(w, i, \delta) < \frac{4|L|}{k}$.

Proof. As G compresses L, it covers L, so by Lemma 3 every rule of G has length at most l. Hence each rule of G can contribute to the costs of at most two segments of w, so we have $2|G| \geq \sum_{i=1}^{k} \text{cost}(w, i, \delta)$. Now suppose that $\lceil \frac{k}{2} \rceil$ segments of w have cost at least $\frac{4|L|}{k}$ each, then $\sum_{i=1}^{k} \text{cost}(w, i, \delta) \geq \lceil \frac{k}{2} \rceil \cdot \frac{4|L|}{k} \geq 2|L|$, which is a contradiction to $|G| < |L|$. □

Definition 13. Let $G = (N, \Sigma, (p_1, \ldots, p_m), A_1)$ be a RLOG and let $s < m$. For $A \in N$ define

$$\text{pmin}(A) = \min\{j \mid p_j \in P_A\} \quad \text{and} \quad \text{pmax}(A) = \max\{j \mid p_j \in P_A\}.$$

Furthermore, for $j \in \{1, \ldots, \lceil \frac{m}{s} \rceil - 1\}$ define

$$N_j = \{A \in N \mid (j-1)s \leq \text{pmin}(A) \text{ and } \text{pmax}(A) < (j+1)s\}.$$

We say that $(N_j)_{j=1}^{\lceil \frac{m}{s} \rceil - 1}$ is the *s-covering of G*.

Note that N_j and N_{j+1} can overlap, but N_j and N_{j+2} can not. Furthermore, note that $|P_{N_j}| \leq 2s$ for all $j \in \{1, \ldots, \lceil \frac{m}{s} \rceil - 1\}$. The following lemma applies Lemma 7 to obtain a lower bound on the number of segments covered by the productions of a single N_j. This lower bound is another key ingredient of the proof of the incompressibility result.

Lemma 8. *Let L be a finite (k,l)-segmented language, let $G = (N, \Sigma, P, S)$ be a RLOG which strongly compresses L and let $|G| > s \geq \frac{4|L|}{k}$. Let $N_1, \ldots, N_{\lceil \frac{|G|}{s} \rceil - 1}$ be the s-covering of G. Let $w \in L$ and δ be a G-derivation of w. Then for at least half of the $i \in \{1, \ldots, k\}$ there is a $j \in \{1, \ldots, \lceil \frac{|G|}{s} \rceil - 1\}$ s.t.* $\text{nonterms}(w, i, \delta) \subseteq N_j$.

Proof. By Lemma 7 at least half of the $i \in \{1, \ldots, k\}$ have $\text{cost}(w, i, \delta) < \frac{4|L|}{k}$. Let i be s.t. $\text{cost}(w, i, \delta) < \frac{4|L|}{k}$, then $\text{cost}(w, i, \delta) < s$. Let $A_0 \in N$ be the nonterminal used for entering the i-th segment of w in δ and let $j_0 = \max\{j \in$

$\{1, \dots, \lceil \frac{|G|}{s} \rceil - 1\} \mid A_0 \in N_j\}$. If $j_0 = \lceil \frac{|G|}{s} \rceil - 1$, then $\text{nonterms}(w, i, \delta) \subseteq N_{j_0}$ because $\bigcup_{j=1}^{\lceil \frac{|G|}{s} \rceil - 1} N_j = N$, N_{j_0} is the last element of this list, and all $A \in \text{nonterms}(w, i, \delta) \setminus \{A_0\}$ occur later than A_0 in the list N. If $j_0 < \lceil \frac{|G|}{s} \rceil - 1$, then $\text{pmin}(A_0) < j_0 s$, for if $\text{pmin}(A_0) \geq j_0 s$, then $A_0 \in N_{j_0+1}$. Therefore $\text{pmin}(A_0) + \text{cost}(w, i, \delta) < \text{pmin}(A_0) + s < (j_0 + 1)s$; hence $\text{nonterms}(w, i, \delta) \subseteq N_{j_0}$. □

3.4. An incompressible sequence of languages

For $n \geq 1$ and $k \in \{0, \dots, 2^n - 1\}$ we write $\text{b}_n(k) \in \{0, 1\}^n$ for the n-bit binary representation of k.

Definition 14 *(Incompressible sequence).* For all $n \geq 1$ define

$$l(n) = \lceil \log(n) \rceil,$$
$$k(n) = \lceil \frac{9n}{l(n) + 1} \rceil, \text{ and}$$
$$L_n = \{(\text{sb}_{l(n)}(i))^{k(n)} \mid 0 \leq i \leq n - 1\}.$$

Note that $l(n)$ is the number of bits required to represent all elements of $\{0, \dots, n - 1\}$ in binary. Note furthermore that for every $n \geq 1$, we have $|L_n| = n$ and all words in L_n have the same length $k(n)(l(n) + 1)$. The number of segments $k(n)$ has been chosen s.t. $k(n)(l(n) + 1)$ is $9n$ padded up to the next multiple of $l(n) + 1$; hence the length of the words in L_n grows linearly in n.

Example 1. *For $n = 5$ we have $l(5) = 3$, $k(5) = 12$ and $L_5 = \{(\text{s000})^{12}, (\text{s001})^{12}, (\text{s010})^{12}, (\text{s011})^{12}, (\text{s100})^{12}\}$.*

Theorem 1. *$(L_n)_{n \geq 1}$ is RLA-incompressible.*

The proof strategy for this theorem is as follows: both Lemmas 6 and 8 assume a strongly compressed segmented language. But while Lemma 6 states an upper bound on the number of segments covered by a certain part of a strongly compressing grammar, Lemma 8 provides a lower bound on the number of segments covered by the productions of a single N_j. The following proof will show these two bounds to be inconsistent, thus deriving the incompressibility of $(L_n)_{n \geq 1}$.

Proof. Suppose that $(L_n)_{n \geq 1}$ is RLA-compressible. Then by Lemma 5 there is a sequence $(L'_n)_{n \geq 1}$ which is strongly compressible by a sequence $(G_n)_{n \geq 1}$ of RLAGs. We consider G_n as RLOG G'_n by fixing an arbitrary linear order satisfying Definition 11. Let us fix for every $n \geq 1$ and every $w \in L'_n$ a derivation δ_w of w w.r.t. G'_n. This is well-defined, since the L'_n are disjoint, and hence δ_w does not depend on n.

First note that for all $n \geq 1$ we have $k(n) = \lceil \frac{9n}{\lceil \log(n) \rceil + 1} \rceil \geq \frac{9n}{\log(n) + 2}$, and since $n \geq |L'_n|$ we have

$$k(n) \geq \frac{9|L'_n|}{\log(|L'_n|) + 2}. \tag{1}$$

Therefore $\frac{4|L'_n|}{k(n)} \leq \frac{4}{9}(\log(|L'_n|) + 2) =: s_n$. Let $N_1, \dots N_{\lceil \frac{|G'_n|}{s_n} \rceil - 1}$ be the s_n-covering of G'_n and define

$$U_n := |\{(w, i) \in L'_n \times \{1, \dots, k(n)\} \mid \exists j \text{ s.t. } \text{nonterms}(w, i, \delta_w) \subseteq N_j\}|.$$

By Lemma 8 we have $U_n \geq \frac{|L'_n| \cdot k(n)}{2}$, which, together with (1), entails

$$U_n \geq \frac{9|L'_n|^2}{2(\log(|L'_n|) + 2)}. \tag{2}$$

On the other hand, applying Lemma 6 to all N_j for $j = 1, \dots, \lceil \frac{|G'_n|}{s_n} \rceil - 1$ and summing up yields

$$U_n \leq \sum_{j=1}^{\lceil \frac{|G'_n|}{s_n} \rceil - 1} 2^{|P_{N_j}|} \cdot |P_{N_j}| \leq (\lceil \frac{|G'_n|}{s_n} \rceil - 1) \cdot 2^{2s_n} \cdot 2s_n.$$

We have

$$2^{2s_n} \cdot 2s_n \leq C|L'_n|^{\frac{8}{9}}(\log(|L'_n|) + 2) \text{ for some } C \in \mathbb{N}$$

and

$$\lceil \frac{|G'_n|}{s_n} \rceil - 1 \leq \frac{|L'_n|}{s_n} = \frac{9|L'_n|}{4(\log(|L'_n|)+2)}$$

and therefore

$$U_n \leq D|L'_n|^{\frac{17}{9}} \text{ for some } D \in \mathbb{N}. \tag{3}$$

Putting (2) and (3) together we obtain

$$|L'_n|^2 \leq E|L'_n|^{\frac{17}{9}}(\log(|L'_n|)+2) \text{ for some } E \in \mathbb{N}. \tag{4}$$

But by Lemma 5 the function $n \mapsto |L'_n|$ is unbounded. Hence there is an $M \in \mathbb{N}$ s.t. for all $n \geq M$ the inequality (4) is not satisfied; contradiction. □

3.5. Remarks

Every sequence of languages which is incompressible in the cover formulation is also incompressible in the (more restricted) equality formulation. Therefore we immediately obtain the following corollary from Theorem 1.

Corollary 1. *There is no sequence $(G_n)_{n\geq 1}$ of RLAGs and $M \in \mathbb{N}$ s.t. $L(G_n) = L_n$ and $|G_n| < |L_n|$ for all $n \geq M$.*

On the other hand, the sequence $(L_n)_{n\geq 1}$ can be compressed by stronger formalisms:

Proposition 1. *There is a sequence $(G_n)_{n\geq 1}$ of acyclic CFGs which compresses $(L_n)_{n\geq 1}$.*

Proof. Let $G_n = (\{S, A_1, \ldots, A_{l(n)}\}, \{0, 1, \mathrm{s}\}, P_n, S)$ where

$$P_n = \{S \to (\mathrm{s}A_1)^{k(n)},\ A_1 \to 0A_2 \mid 1A_2,\ \ldots,\ A_{l(n)} \to 0 \mid 1\}.$$

Then $L(G_n) \supseteq L_n$ for all $n \geq 1$ and $|G_n| = 2\lceil \log(n) \rceil + 1 < n = |L_n|$ for all $n \geq M$ for a certain M. □

The length of the words in L_n grows linearly. Under the condition that $|L_n| = n$ this is the best possible:

Proposition 2. *Let $(L'_n)_{n\geq 1}$ be a sequence of finite languages over a finite alphabet $\Sigma = \{a_1, \ldots a_k\}$ s.t. $|L'_n| = n$ and s.t. there is a sublinear function that bounds the maximal length l_n of a word in L'_n. Then $(L'_n)_{n\geq 1}$ is RLA-compressible.*

Proof. Let $G_n = (\{A_1, \ldots, A_{l_n}\}, \Sigma, P_n, A_1)$ where

$$P_n = \{A_1 \to a_1A_2 \mid \cdots \mid a_kA_2 \mid A_2,\ \ldots,\ A_{l_n} \to a_1 \mid \cdots \mid a_k \mid \varepsilon\}.$$

Then $L(G_n) = \Sigma^{\leq l_n} \supseteq L'_n$ and $|G_n| = (k+1) \cdot l_n$ which, from a certain $M \in \mathbb{N}$ on, is less than $|L'_n| = n$. □

4. Tree languages

4.1. Vectorial grammars

Rigid tree languages, a class of tree languages with equality constraints, have been introduced in [27] with applications in verification in mind. A presentation of this class of languages based on grammars has been given in [17]. Their relevance in the present context stems from the fact that, on the one hand, they describe formal proofs in a sense that will be made precise in Section 6 and, on the other hand, their compression strength equals that of RLAGs in the sense of the below Lemma 12.

For a ranked alphabet (i.e., a term signature) Σ we write $\mathcal{T}(\Sigma)$ for the set of all terms built from function and constant symbols of Σ. In what follows, we will also consider vectors of symbols, written as $\overline{c}$. The union of vectors is the union of their elements, i.e., if $\overline{c} = (c_1, \ldots, c_k)$ and $\overline{d} = (d_1, \ldots, d_l)$, then $\overline{c} \cup \overline{d} = \{c_1, \ldots, c_k, d_1, \ldots, d_l\}$.

Definition 15. Let Σ be a ranked alphabet. A *vectorial totally rigid acyclic tree grammar (VTRATG)* is given by the tuple $(\alpha_{0,1}, N, \Sigma, P)$:

1. $\alpha_{0,1}$ is the start nonterminal.
2. $N = ((\alpha_{0,1}), \overline{\alpha_1}, \ldots, \overline{\alpha_n})$ is a finite sequence of nonterminal vectors $\overline{\alpha_i} = (\alpha_{i,1}, \ldots, \alpha_{i,k_i})$. The nonterminals are pairwise distinct: $\alpha_{i,j} \neq \alpha_{k,l}$ if $(i, j) \neq (k, l)$.
3. Σ is a ranked alphabet, the terminal symbols of the grammar.

4. P is a finite set of vectorial productions. A vectorial production is a pair $\overline{\alpha_i} \to \overline{t}$, where $\overline{t} = (t_1, \ldots, t_{k_i})$ is a vector of terms s.t. $t_j \in \mathcal{T}(\Sigma \cup \overline{\alpha_{i+1}} \cup \cdots \cup \overline{\alpha_n})$. P is required to contain at least one vectorial production for every nonterminal vector $\overline{\alpha_i}$.

Note that G is acyclic in the sense that for each vectorial production $\overline{\alpha_i} \to \overline{t}$, the right hand side $\overline{t}$ only contains nonterminals from the nonterminal vectors $\overline{\alpha_{i+1}}, \ldots, \overline{\alpha_n}$.

Definition 16. Let $G = (\alpha_{0,1}, N, \Sigma, P)$ be a VTRATG with nonterminals $N = ((\alpha_{0,1}), \overline{\alpha_1}, \ldots, \overline{\alpha_n})$. The language of G is defined as $L(G) = \{\alpha_{0,1}[\overline{\alpha_0}\backslash\overline{s_0}] \cdots [\overline{\alpha_n}\backslash\overline{s_n}] \mid \overline{\alpha_i} \to \overline{s_i} \in P\}$.

The property that the language $L(G)$ of a VTRATG does not contain any nonterminals is a consequence of this definition.

Example 2. *Let $G = (\alpha_{0,1}, N, \Sigma, P)$ where $N = ((\alpha_{0,1}), (\alpha_{1,1}, \alpha_{1,2}), (\alpha_{2,1}))$, $\Sigma = \{a/0, b/0, c/0, f/3, g/1\}$ and P consists of the following production rules:*

$$(\alpha_{0,1}) \to (f(\alpha_{1,1}, \alpha_{1,2}, \alpha_{2,1}))$$
$$(\alpha_{1,1}, \alpha_{1,2}) \to (g(\alpha_{2,1}), a) \mid (\alpha_{2,1}, \alpha_{2,1})$$
$$(\alpha_{2,1}) \to (b) \mid (c)$$

Then $L(G) = \{f(g(b), a, b), f(g(c), a, c), f(b, b, b), f(c, c, c)\}$.

The following lemma shows that the language of a VTRATG can be computed by progressively reducing the grammar from the left.

Lemma 9. *Let $G = (\alpha_{0,1}, N, \Sigma, P)$ be a VTRATG with nonterminals $N = ((\alpha_{0,1}), \overline{\alpha_1}, \ldots, \overline{\alpha_n})$ with $n \geq 1$. Define $G' = (\alpha_{0,1}, N', \Sigma, P')$ where*

$$N' = ((\alpha_{0,1}, \overline{\alpha_2}, \ldots, \overline{\alpha_n})), \text{ and}$$
$$P' = \{(\alpha_{0,1}) \to \overline{s_0}[\overline{\alpha_1}\backslash\overline{s_1}] \mid (\alpha_{0,1}) \to \overline{s_0} \in P, \overline{\alpha_1} \to \overline{s_1} \in P\} \cup \{\overline{\alpha_i} \to \overline{s_i} \mid 2 \leq i \leq n\}.$$

Then $L(G') = L(G)$.

Proof. We have

$$\begin{aligned} L(G') &= \{\alpha_{0,1}[\overline{\alpha_0}\backslash\overline{s_0}[\overline{\alpha_1}\backslash\overline{s_1}]][\overline{\alpha_2}\backslash\overline{s_2}] \cdots [\overline{\alpha_n}\backslash\overline{s_n}] \mid \overline{\alpha_i} \to \overline{s_i} \in P\} \\ &= \{\alpha_{0,1}[\overline{\alpha_0}\backslash\overline{s_0}][\overline{\alpha_1}\backslash\overline{s_1}][\overline{\alpha_2}\backslash\overline{s_2}] \cdots [\overline{\alpha_n}\backslash\overline{s_n}] \mid \overline{\alpha_i} \to \overline{s_i} \in P\} \\ &= L(G). \quad \square \end{aligned}$$

The *type* of a VTRATG G as in Definition 15 is the vector $(1, k_1, \ldots, k_n)$ where the leading 1 indicates that the vector $\overline{\alpha_0}$ has length 1 and is included for technical convenience. The *cotype* of G is the vector $(l_0, \ldots, l_n)$ where l_i is the number of vectorial productions in G whose left-hand side is $\overline{\alpha_i}$.

Proposition 3. *Let G be a VTRATG with cotype $(l_0, \ldots, l_n)$. Then $|L(G)| \leq \prod_{i=0}^{n} l_i$.*

Proof. By induction on n. If $n = 0$, then trivially $|L(G)| = l_0$. For the induction step, let $G = (\alpha_{0,1}, ((\alpha_{0,1}), \overline{\alpha_1}, \ldots, \overline{\alpha_n}), \Sigma, P)$ be a VTRATG and let $P_n = \{\overline{\alpha_n} \to \overline{t_1}, \ldots, \overline{\alpha_n} \to \overline{t_{l_n}}\}$ be the productions in P with left-hand side $\overline{\alpha_n}$. For $k \in \{1, \ldots, l_n\}$ define the VTRATG $G_k = (\alpha_{0,1}, ((\alpha_{0,1}), \overline{\alpha}_1, \ldots, \overline{\alpha}_{n-1}), \Sigma, (P \setminus P_n)[\overline{\alpha_n}\backslash\overline{t_k}])$. Then we have $L(G) = \bigcup_{k=1}^{l_n} L(G_k)$. By induction hypothesis $|L(G_k)| \leq \prod_{i=0}^{n-1} l_i$ and therefore $|L(G)| \leq \sum_{k=1}^{l_n} |L(G_k)| \leq \prod_{i=0}^{n} l_i$. $\square$

As a first attempt at a complexity measure of a VTRATG one may be tempted to count the number of vectorial productions, i.e., to define $\sum_{i=0}^{n} l_i$ as the complexity of G. However, this measure trivialises in the following sense.

Proposition 4. *Let L be a finite tree language and let $l_0, \ldots, l_n \in \mathbb{N}$ s.t. $|L| \leq \prod_{i=0}^{n} l_i$. Then there is a VTRAT grammar G with cotype $(l_0, \ldots, l_n)$ and $L(G) = L$.*

Proof. For $i \in \{0, \ldots, n\}$ define $l_i^* = \prod_{0 \leq j < i} l_j$ (hence $l_0^* = 1$) and $\overline{\alpha_i} = (\alpha_{i,1}, \ldots, \alpha_{i,l_i^*})$. Furthermore, for $i \in \{0, \ldots, n-1\}$ let

$$P_i = \{\overline{\alpha_i} \to (\alpha_{i+1,\, j \cdot l_i^* + 1}, \ldots, \alpha_{i+1,\,(j+1)\cdot l_i^*}) \mid 0 \leq j < l_i\}.$$

Define the VTRAT grammar $G_0 = (\alpha_{0,1}, (\overline{\alpha_0}, \ldots, \overline{\alpha_{n-1}}), \Sigma, P_0 \cup \cdots \cup P_{n-1})$. We first claim that for all $k \in \{0, \ldots, n-1\}$ we have

$$\{\alpha_{0,1}[\overline{\alpha_0}\backslash\overline{s_0}]\cdots[\overline{\alpha_k}\backslash\overline{s_k}] \mid \overline{\alpha_i} \to \overline{s_i} \in P_0 \cup \cdots \cup P_k\} = \{\alpha_{k+1,1}, \ldots, \alpha_{k+1,l^*_{k+1}}\}$$

and show this by induction on k: for $k = 0$ we have $\{\alpha_{0,1}[\overline{\alpha_0}\backslash\overline{s_0}] \mid \overline{\alpha_0} \to \overline{s_0} \in P_0\} = \{\alpha_{1,1}, \ldots, \alpha_{1,l^*_1}\}$. For the induction step assume $k > 0$. Then we have

$$\begin{aligned}
&\{\alpha_{0,1}[\overline{\alpha_0}\backslash\overline{s_0}]\cdots[\overline{\alpha_k}\backslash\overline{s_k}] \mid \overline{\alpha_i} \to \overline{s_i} \in P_0 \cup \cdots \cup P_k\} \\
&= \{\alpha_{k,l}[\overline{\alpha_k}\backslash\overline{s_k}] \mid 1 \le l \le l^*_k, \overline{\alpha_k} \to \overline{s_k} \in P_k\} \\
&= \{\alpha_{k+1,j\cdot l^*_k + l} \mid 1 \le l \le l^*_k, 0 \le j < l_k\} \\
&= \{\alpha_{k+1,1}, \ldots, \alpha_{k+1,l^*_{k+1}}\}.
\end{aligned}$$

Finally, let $L = \{t_1, \ldots, t_m\} = \{t_{j\cdot l^*_n+1}, \ldots, t_{(j+1)\cdot l^*_n} \mid 0 \le j < l_n\}$ including duplicates of t_m in case $|L| < l^*_{n+1}$. Define

$$P_n = \{\overline{\alpha_n} \to (t_{j\cdot l^*_n+1}, \ldots, t_{(j+1)\cdot l^*_n}) \mid 0 \le j < l_n\}$$

and $G = (\alpha_{0,1}, (\overline{\alpha}_0, \ldots, \overline{\alpha}_n), \Sigma, P_0 \cup \cdots \cup P_n)$. Then the cotype of G is $(l_0, \ldots, l_n)$ and we have

$$\begin{aligned}
L(G) &= \{\alpha_{n,l}[\overline{\alpha_n}\backslash(t_{j\cdot l^*_n+1}, \ldots, t_{(j+1)\cdot l^*_n})] \mid 1 \le l \le l^*_n, 0 \le j < l_n\} \\
&= \{t_1, \ldots, t_m\}. \quad \square
\end{aligned}$$

Therefore we define the complexity of a VTRATG in a way that also takes the lengths of vectors into account.

Definition 17. Let T be a finite set of vectors of length k, i.e., $T = \{(t_{1,1}, \ldots, t_{1,k}), \ldots, (t_{n,1}, \ldots, t_{n,k})\}$. If $k \ge 2$, we define $T' = \{(t_{1,2}, \ldots, t_{1,k}), \ldots, (t_{n,2}, \ldots, t_{n,k})\}$. Furthermore, we define the *tree complexity* $\|T\|$ of T by induction on k as follows: if $k = 1$, then $\|T\| = |T|$. If $k \ge 2$, let $T = T_1 \uplus \cdots \uplus T_m$ be the partition of T s.t. $\overline{t}, \overline{s} \in T$ are in the same T_i iff $\overline{t}$ and $\overline{s}$ have the same first component and define $\|T\| = m + \sum_{i=1}^m \|T_i\|$.

Note that $\|T\|$ is the number of vertices in the tree representation of the finite set T of vectors.

Definition 18. Let G be a VTRATG with nonterminals $(\overline{\alpha_0}, \ldots, \overline{\alpha_n})$ and vectorial productions P. We define the *complexity of* G as

$$|G| = \sum_{i=0}^{n} \|\{\overline{t} \mid \overline{\alpha_i} \to \overline{t} \in P\}\|.$$

For a finite tree language L we define

$$\mathrm{VTRATc}(L) = \min\{|G| \mid G \text{ VTRATG s.t. } L(G) \supseteq L\}$$

and for a type $\tau \in \{1\} \times \mathbb{N}^n$ we define

$$\mathrm{VTRATc}_\tau(L) = \min\{|G| \mid G \text{ VTRATG of type } \tau \text{ s.t. } L(G) \supseteq L\}.$$

4.2. Flattening vectorial grammars

Definition 19. A *totally rigid acyclic tree grammar (TRATG)* is a VTRATG of type $(1, \ldots, 1)$. For a finite tree language L we define $\mathrm{TRATc}(L) = \min\{|G| \mid G \text{ TRATG s.t. } L(G) \supseteq L\}$.

The type of a VTRATG being $(1, \ldots, 1)$ means that all vectors have length 1; hence the name TRATG. Note that, for a TRATG G, the complexity $|G|$ of G is just the number of production rules of G. On the one hand, every TRATG is a VTRATG and therefore $\mathrm{VTRATc}(L) \le \mathrm{TRATc}(L)$. On the other hand, a VTRATG can be flattened to a TRATG in the sense of the following lemma.

Lemma 10. *Let G be a VTRATG, then there is a TRATG G^{f} s.t. $L(G^{\mathrm{f}}) \supseteq L(G)$ and $|G^{\mathrm{f}}| \le |G|$.*

Proof. Let $(1, k_1, \ldots, k_n)$ be the type of G. We proceed by induction on the number of $i \in \{1, \ldots, n\}$ s.t. $k_i > 1$. If there is no such i, then G is a TRATG. Assume $k_i > 1$ and define

$$G' = (\alpha_{0,1}, ((\alpha_{0,1}), \overline{\alpha_1}, \ldots, \overline{\alpha_{i-1}}, (\alpha_{i,1}), \ldots, (\alpha_{i,k_i}), \overline{\alpha_{i+1}}, \ldots, \overline{\alpha_n}), \Sigma, P')$$

where P' is defined from P by replacing every production of the form $\overline{\alpha_i} \to (t_1, \ldots, t_{k_i})$ by the productions $(\alpha_{i,1}) \to (t_1), \ldots, (\alpha_{i,k_i}) \to (t_{k_i})$. Then we have $L(G') \supseteq L(G)$, $|G'| \le |G|$ and G' has a type with less elements greater than 1. $\square$

Proposition 5. $\mathrm{VTRATc}(L) = \mathrm{TRATc}(L)$ *for every finite tree language* L.

Proof. $\mathrm{VTRATc}(L) \leq \mathrm{TRATc}(L)$ is obvious since every TRATG is a VTRATG. The other direction follows directly from Lemma 10. □

The flattening-transformation used in Lemma 10 for obtaining a TRATG G^{f} from a VTRATG G without increasing its complexity increases the number of nonterminal vectors. At this point, it is unclear whether this is necessary and one may ask whether there is a flattening-transformation which keeps the number of nonterminal vectors constant but increases the number of productions instead (that is, so to say, to increase the width instead of the height of the grammar). We will now show that this is impossible by exhibiting languages which are $(1,2)$-VTRAT-compressible but $(1,1)$-VTRAT-incompressible. To that aim, we work in the ranked alphabet $\Sigma = \{f/2, s/1, 0/0\}$.

Definition 20. For $n \geq 1$ define $T_n = \{f(s^k(0), s^l(0)) \mid k + l = n\}$.

Lemma 11. *Let* α *be a variable which occurs in a term* u *as strict subterm, let* s_1, s_2 *be terms. If* $u[\alpha \backslash s_1], u[\alpha \backslash s_2] \in T_n$, *then* $s_1 = s_2$.

Proof. If α is a strict subterm of u and $u[\alpha \backslash s_1], u[\alpha \backslash s_2] \in T_n$, then $u = f(u', u'')$ and $s_1 = s^{m_1}(0)$, $s_2 = s^{m_2}(0)$ for some $m_1, m_2 \in \mathbb{N}$.

If u' contains α but u'' does not, then $u' = s^{n'}(\alpha)$ and $u'' = s^{n''}(0)$ for some $n', n'' \in \mathbb{N}$. As $u[\alpha \backslash s_1] \in T_n$ we have $n' + m_1 + n'' = n$ and as $u[\alpha \backslash s_2] \in T_n$ we have $n' + m_2 + n'' = n$ hence $m_1 = m_2$. If u'' contains α but u' does not, proceed symmetrically. If both u' and u'' contain α an analogous argument leads to the equations $n' + n'' + 2m_1 = n$ and $n' + n'' + 2m_2 = n$ and hence to the same conclusion $m_1 = m_2$. □

Theorem 2. $\mathrm{VTRATc}_{(1,2)}(T_n) = O(\sqrt{|T_n|})$ *but* $\mathrm{VTRATc}_{(1,1)}(T_n) \geq |T_n|$.

Proof. We start with the proof of the upper bound. For $m \geq 1$ define the VTRATG G_m of type $(1,2)$ as follows: $G_m = (\alpha_{0,1}, ((\alpha_{0,1}), (\alpha_{1,1}, \alpha_{1,2})), \{f, s, 0\}, P)$ where

$$P = \{(\alpha_{0,1}) \to (f(s^{km}(\alpha_{1,1}), s^{lm}(\alpha_{1,2})) \mid k + l = m - 1\} \cup$$
$$\{(\alpha_{1,1}, \alpha_{1,2}) \to (s^k(0), s^l(0)) \mid k + l = m - 1\}.$$

We claim that $L(G_m) = T_{m^2-1}$. In order to show that, first let $f(s^{r_1}(0), s^{r_2}(0)) \in L(G_m)$. Then $r_1 = km + k'$, $r_2 = lm + l'$ with $k + l = m - 1$ and $k' + l' = m - 1$. Therefore $r_1 + r_2 = (k + l)m + (k' + l') = m^2 - 1$ and hence $f(s^{r_1}(0), s^{r_2}(0)) \in T_{m^2-1}$. For the other direction, let $f(s^{r_1}(0), s^{r_2}(0)) \in T_{m^2-1}$, then $r_1 + r_2 = m^2 - 1$. From the base m representations of r_1 and r_2 we obtain $k, k', l, l' \in \{0, \ldots, m-1\}$ s.t. $r_1 = km + k'$ and $r_2 = lm + l'$. We thus have $m^2 - 1 = (k + l)m + (k' + l')$. Then $k + l < m$ (since otherwise $(k + l)m \geq m^2$ and $(k + l)m + (k' + l') > m^2 - 1$) and $k + l > m - 2$ (since otherwise $(k + l)m \leq (m - 2)m$ and as $k' + l' \leq 2(m - 1)$ we would have $(k + l)m + (k' + l') \leq m^2 - 2$), so $k + l = m - 1$. Then $k' + l' = m^2 - 1 - (m - 1)m = m - 1$ and therefore $f(s^{r_1}(0), s^{r_2}(0)) \in L(G_m)$. We have $|G_m| = 2m$ and $|T_{m^2-1}| = m^2$ which concludes the proof of the upper bound.

For the proof of the lower bound, it will be convenient to write a $(1,1)$-VTRATG as a pair (U, S) of finite sets of terms s.t. the only nonterminal α is only allowed to occur in U. The language of (U, S) is $U[\alpha \backslash S] = \{u[\alpha \backslash s] \mid u \in U, s \in S\}$ and its complexity is $|(U, S)| = |U| + |S|$. We will consider $(U, \emptyset)$ to denote a (1)-VTRATG. The language of $(U, \emptyset)$ is U and its complexity is $|U|$.

Let (U, S) be a VTRATG of type $(1,1)$ or (1) which covers T_n, let $\#U$ be the number of terms in U that contain α. We show that there is a VTRATG (U', S') of type $(1,1)$ or (1) which covers T_n s.t.

$$\#U' < \#U \text{ and } |(U', S')| \leq |(U, S)|$$

To that aim let $u_0 \in U$ s.t. u_0 contains α.

If $u_0 = \alpha$, let $S = S_1 \cup S_2$ with $S_1 \subseteq T_n$ and $S_2 \cap T_n = \emptyset$. As $S_1 \subseteq T_n$, all $s_1 \in S_1$ must be of the form $f(s_1', s_1'')$. Furthermore, all $u \in U \setminus \{\alpha\}$ s.t. there is $s \in S$ with $u[\alpha \backslash s] \in T_n$ must be of the form $f(u', u'')$. So, given $u \in U \setminus \{\alpha\}$ which contains α and $s_1 \in S_1$ the term $u[\alpha \backslash s_1]$ contains at least two occurrences of f hence $u[\alpha \backslash s_1] \notin T_n$. Therefore $((U \setminus \{\alpha\}) \cup S_1, S_2)$ is a VTRATG of type $(1,1)$ or (1) which covers T_n and is smaller than (U, S) both in terms of $|\cdot|$ and $\#\cdot$.

If $u_0 \neq \alpha$ (but α occurs in u_0) write

$$U[\alpha \backslash S] \cap T_n = (u_0[\alpha \backslash S] \cap T_n) \cup \bigcup_{u \in U \setminus \{u_0\}} (u[\alpha \backslash S] \cap T_n).$$

Define $U' = (U \setminus \{u_0\}) \cup (u_0[\alpha \backslash S] \cap T_n)$ and observe that

$$U'[\alpha \backslash S] \cap T_n = (u_0[\alpha \backslash S] \cap T_n) \cup \bigcup_{u \in U \setminus \{u_0\}} (u[\alpha \backslash S] \cap T_n).$$

By Lemma 11, $|u_0[\alpha\backslash S] \cap T_n| \leq 1$ hence (U', S) is a VTRATG of type $(1,1)$ or (1) which covers T_n with $\#U' < \#U$ and $|(U', S')| \leq |(U, S)|$.

Therefore, among all VTRAT grammars of type $(1,1)$ or (1) with minimal complexity there is one with $\#U = 0$, i.e., a (1)-VTRATG. But the only minimal (1)-VTRATG is $(T_n, \emptyset)$ which is of complexity $|T_n|$ and therefore $\mathrm{VTRATc}_{(1,1)} \geq |T_n|$. □

4.3. Relating word and tree languages

Definition 21. For an alphabet Σ define the ranked alphabet $\Sigma^{\mathrm{T}} = \{f_a/1 \mid a \in \Sigma\} \cup \{e\}$. For $w \in \Sigma^*$ define the term w^{T} recursively by $\varepsilon^{\mathrm{T}} = e$, and $(av)^{\mathrm{T}} = f_a(v^{\mathrm{T}})$. For $L \subseteq \Sigma^*$ define $L^{\mathrm{T}} = \{w^{\mathrm{T}} \mid w \in L\}$.

Lemma 12. *For every finite language L:* $\mathrm{TRATc}(L^{\mathrm{T}}) = \mathrm{RLAc}(L)$.

Proof. Let G be a minimal RLAG with $L(G) \supseteq L$. By applying $\cdot^{\mathrm{T}}$ to the production rules of G we obtain a regular tree grammar G' in the terminals Σ^{T} with $L(G') \supseteq L^{\mathrm{T}}$ and $|G'| = |G|$. Since Σ^{T} is unary, the interpretation of G' as a TRATG G'' satisfies $L(G'') = L(G') \supseteq L^{\mathrm{T}}$ and $|G''| = |G|$.

For the other direction let Σ be the alphabet of L and let G be a minimal TRATG with $L(G) \supseteq L^{\mathrm{T}}$, then $G = (N, \Sigma^{\mathrm{T}}, P, S)$. Therefore every production rule in P is of the form $A \to f_{x_1}(\cdots\cdots(f_{x_n}(B))\cdots)$ or $A \to f_{x_1}(\cdots\cdots(f_{x_n}(e))\cdots)$ for some $B \in N$, $n \geq 0$ and $x_1, \ldots, x_n \in \Sigma$. We define an RLAG $G' = (N, \Sigma, P', S)$ by replacing each $A \to f_{x_1}(\cdots\cdots(f_{x_n}(B))\cdots)$ by $A \to x_1 \cdots x_n B$ and each $A \to f_{x_1}(\cdots\cdots(f_{x_n}(e))\cdots)$ by $A \to x_1 \cdots x_n$. Then $|G'| = |G|$. □

Corollary 2. *The sequence of tree languages $(L_n^{\mathrm{T}})_{n\geq 1}$ is TRAT-incompressible.*

Proof. This follows directly from Lemma 12 and Theorem 1. □

5. Proof theory

In this section we introduce the basic notions and results from proof theory which are relevant to this paper. For a thorough introduction to proof theory, the interested reader is referred to [28], for a textbook on the subject to [29].

5.1. The cut-elimination theorem

We will consider formulas and proofs in first-order predicate logic without equality. Formulas are built up inductively from atoms using $\wedge$, $\vee$, $\to$, $\neg$, $\forall$ and $\exists$. The semantics of first-order predicate logic is defined as usual: a *structure* is a pair $\mathcal{M} = (D, I)$ where D, the domain, is an arbitrary set and I, the interpretation, maps constant symbols to elements of D, function symbols to functions over D and predicate symbols to predicates over D. We write $\mathcal{M} \vDash \varphi$ if φ is true in $\mathcal{M}$. This relation is defined as usual by induction on the structure of φ. A formula is called *valid* if it is true in all structures. For a more detailed introduction to first-order predicate logic, the reader is referred to [30].

A *sequent* is an expression of the form $\Gamma \vdash \Delta$ where Γ and Δ are finite multisets of formulas. The intended interpretation of a sequent $\Gamma \vdash \Delta$ is the formula $(\bigwedge_{\varphi\in\Gamma} \varphi) \to (\bigvee_{\psi\in\Delta} \psi)$. A sequent is said to be valid if its interpretation as a formula is. Note that a formula φ can be identified with the sequent $\emptyset \vdash \{\varphi\}$. Usually, multiset-notation is omitted, so the above sequent is just written as $\vdash \varphi$. Thus, a sequent has a syntactic structure which generalises that of a formula.

Example 3. $P(0), \forall x\,(P(x) \to P(s(x))) \vdash P(s(s(0)))$ *is a sequent.*

We define $\Gamma_1 \vdash \Delta_1 \subseteq \Gamma_2 \vdash \Delta_2$ as $\Gamma_1 \subseteq \Gamma_2$ and $\Delta_1 \subseteq \Delta_2$. Analogously, we also use the union $\cup$ and disjoint union $\uplus$ on sequents component-wise. When we speak about a minimal sequent, we mean minimal w.r.t. $\subseteq$. The size of a sequent is $|\Gamma \vdash \Delta| = |\Gamma| + |\Delta|$. The logical complexity $\|\varphi\|$ of a formula φ is the number of logical connectives and atoms it contains. Analogously, the logical complexity $\|\Gamma \vdash \Delta\|$ of a sequent $\Gamma \vdash \Delta$ is the number of logical connectives and atoms it contains. We write $\forall \overline{x}\, A$ as abbreviation for $\forall x_1 \cdots \forall x_n\, A$ and analogously for the existential quantifier $\exists$. A $\overline{\Sigma}_1$*-sequent* is a sequent $\Gamma \vdash \Delta$ where all formulas in Γ are of the form $\forall \overline{x}\, A$ with A quantifier-free and all formulas in Δ are of the form $\exists \overline{x}\, A$ with A quantifier-free. Every sequent can be transformed into a validity-equivalent $\overline{\Sigma}_1$-sequent by Skolemisation and prenexification, see, e.g., [31].

Example 4. *The sequent S defined in Example 3 is a $\overline{\Sigma}_1$-sequent and satisfies $|S| = 3$ and $\|S\| = 6$.*

A *proof* in the sequent calculus is a finite tree whose nodes are labelled with sequents and that is built according to the following logical inference rules. The leaves are of the form $A \vdash A$. The inference rules are:

- The propositional rules:

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

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

$$\frac{\Gamma \vdash \Delta, A \quad B, \Pi \vdash \Lambda}{A \to B, \Gamma, \Pi \vdash \Delta, \Lambda}\ \to_\mathrm{l} \qquad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \to B}\ \to_\mathrm{r}$$

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

- The quantifier rules:

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

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

  The $\forall_\mathrm{r}$- and $\exists_\mathrm{l}$-rules are called *strong quantifier rules* and are subject to the usual condition: α must not appear in the sequent below the rule. The variable α is called *eigenvariable*. The $\forall_\mathrm{l}$- and $\exists_\mathrm{r}$-rules are called *weak quantifier rules*.
- The structural rules:

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ \mathrm{c_l} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ \mathrm{c_r} \qquad \frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ \mathrm{w_l} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ \mathrm{w_r}$$

- The cut:

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

A formula A, B, etc. mentioned explicitly in one of the above definitions of a rule is called *active formula* of its respective rule. In contrast, the multiset Γ, Δ, etc. is called *context* of its respective rule. The structural rules $\mathrm{c_l}$ and $\mathrm{c_r}$ are called *contractions*. The structural rules $\mathrm{w_l}$ and $\mathrm{w_r}$ are called *weakenings*. An active formula of a rule is called *auxiliary formula* if it is in the sequent above the rule and *main formula* if it is in the sequent below the rule. The auxiliary formula of a cut is also called *cut formula* of that cut. The lowest sequent in a proof is called *end-sequent* of that proof.

The soundness and completeness theorems of the sequent calculus state that a sequent is valid iff it is provable.

Example 5. *A proof of the sequent S defined in Example 3 is:*

$$\dfrac{\dfrac{\dfrac{\dfrac{P(0) \vdash P(0) \quad \dfrac{\dfrac{P(s(0)) \vdash P(s(0)) \quad P(s(s(0))) \vdash P(s(s(0)))}{P(s(0)), P(s(0)) \to P(s(s(0))) \vdash P(s(s(0)))}\ \to_\mathrm{l}}{P(s(0)), \forall x\,(P(x) \to P(s(x))) \vdash P(s(s(0)))}\ \forall_\mathrm{l}}{P(0), P(0) \to P(s(0)), \forall x\,(P(x) \to P(s(x))) \vdash P(s(s(0)))}\ \to_\mathrm{l}}{P(0), \forall x\,(P(x) \to P(s(x))), \forall x\,(P(x) \to P(s(x))) \vdash P(s(s(0)))}\ \forall_\mathrm{l}}{P(0), \forall x\,(P(x) \to P(s(x))) \vdash P(s(s(0)))}\ \mathrm{c_l}$$

Thus, the sequent S is valid.

Since a single formula φ may occur several times in a proof, we speak about *formula occurrences*. In the proof of Example 5, there are five occurrences of the formula $\forall x\,(P(x) \to P(s(x)))$. An application of a rule in a proof is called *inference*. The *length* of a proof ψ, written as $|\psi|$ is the number of inferences in ψ. We write $|\psi|_\mathrm{q}$ for the number of weak quantifier inferences in ψ, $|\psi|_\mathrm{q}$ is called the *quantifier complexity of* ψ.

The cut rule formalises the use of a lemma in a mathematical proof. A cut is said to be a $\overline{\Pi}_1$-cut if its cut formula is of the form $\forall \overline{x} A$ for A quantifier-free. There is a procedure for transforming proofs with cuts into cut-free proofs. The cut-elimination theorem stated below, which is based on this procedure, is a cornerstone of proof theory. In order to state this theorem, define the function $(k, n) \mapsto 2^n_k$ for $k, n \in \mathbb{N}$ by induction on k as follows: $2^n_0 = n$ and $2^n_{k+1} = 2^{2^n_k}$.

Theorem 3 *(Cut-elimination theorem). Let ψ be a proof of a sequent $\Gamma \vdash \Delta$ s.t. $\|\varphi\| \leq k$ for every cut formula φ in ψ. Then there is a cut-free proof ψ' of $\Gamma \vdash \Delta$ s.t. $|\psi'| \leq 2^{|\psi|}_{O(k)}$.*

This result was originally shown in [32], see [28] for a contemporary exposition in English. Based on this result, one can consider a proof with cut as a *compressed representation* of a cut-free proof, much like a grammar is a compressed representation of its finite language. Indeed, it is this point of view which is fundamental for the work presented in this paper.

5.2. Complexity of proofs

In this section we discuss complexity measures for proofs and known results about them in order to clarify what kind of bounds we will prove in Section 7.

We have already seen two complexity measures for proofs: the length $|\psi|$ of ψ and the quantifier complexity $|\psi|_{\mathrm{q}}$ of ψ. Another natural measure is the *symbolic complexity of* ψ, i.e., the number of symbols occurring in ψ, written as $\|\psi\|$. Clearly we have $|\psi|_{\mathrm{q}} \leq |\psi| \leq \|\psi\|$ for every proof ψ.

A first crucial observation is that there is no computable upper bound for the complexity of the smallest proof (in any of the three above senses) of a formula in terms of that formula (and hence in any computable complexity measure for formulas). More precisely, writing $\mathcal{F}$ for the set of formulas, we have:

Theorem 4. *There is no computable function* $f : \mathcal{F} \to \mathbb{N}$ *s.t.* $f(\varphi) \geq \min\{|\psi|_{\mathrm{q}} \mid \psi$ *is a proof of* $\vdash \varphi\}$ *for all valid formulas* φ.

This result follows from the undecidability of validity of first-order formulas, the cut-elimination theorem and the techniques used, e.g., in [33] for solving the realisability problem for cut-free proof skeletons. Since this theorem is not required for proving other results in this paper but only for justifying our interest in them, we refrain from giving a proof here.

As a corollary of Theorem 4, every computable function is a lower bound in the following sense: for every computable function $g : \mathcal{F} \to \mathbb{N}$, there is an infinite set of valid formulas $\{\varphi_n \mid n \in \mathbb{N}\}$ s.t. any proof ψ_n of φ_n has $|\psi_n|_{\mathrm{q}} > g(\varphi_n)$ and hence also $|\psi_n| > g(\varphi_n)$ and $\|\psi_n\| > g(\varphi_n)$. Therefore, in this paper we will not be interested in a lower bound on the complexity of a proof *in terms of the formula it proves*. Instead, our results study the complexity of cut-elimination. In one direction, the complexity of cut-elimination is well understood: as stated in Theorem 3, the length of the cut-free proof is at most an iterated exponential in the length of the proof with cut. In addition, corresponding lower bounds have been proved in [34] and [35], see [36]. These lower bounds show that the cut-elimination procedure is optimal in the sense that there is a sequence of formulas $(\varphi_n)_{n\geq 1}$ which possesses a sequence of proofs with cut of length polynomial in n but every sequence of cut-free proofs has a length which grows like the n-fold iteration of the exponential function.

However, the complexity of cut-introduction, i.e., the inversion of cut-elimination, is not at all well understood yet and it is this question to which this paper contributes. More precisely: we analyse the complexity of the shortest proof with cut of some formula φ in terms of the complexity of the shortest cut-free proof of φ.

The existing upper bound on the complexity of cut-elimination has one immediate corollary pertaining to the complexity of cut-introduction: the iterated exponential function is trivially the maximal compression obtainable by cut-introduction. However, to the best of the authors' knowledge, no sequences of formulas are known whose cut-free proofs have a non-trivial maximal compression by cut-introduction. This is not surprising, since lower bounds on the complexity of proofs with cut are notoriously difficult to show (for propositional logic this is considered the central open problem in proof complexity [24]). In this paper we will prove a non-trivial lower bound on the complexity of the shortest proof with $\overline{\Pi}_1$-cuts in terms of the complexity of the shortest cut-free proof (Theorem 9).

5.3. Cut-free proofs and Herbrand-sequents

An *instance* of a formula $Q\overline{x}\,A$ with A quantifier-free and $Q \in \{\forall, \exists\}$ is a formula of the form $A[\overline{x}\backslash\overline{t}]$. We say that $A[\overline{x}\backslash\overline{t}]$ is a *ground instance* if $\overline{t}$ is variable-free.

Definition 22. A sequent $\Gamma' \vdash \Delta'$ is called *Herbrand-sequent* of a $\overline{\Sigma}_1$-sequent $\Gamma \vdash \Delta$ if Γ' consists of ground instances of formulas from Γ and Δ' consists of ground instances of formulas from Δ and $\Gamma' \vdash \Delta'$ is a tautology.

Example 6. *The sequent*

$$P(0),\ P(0) \to P(s(0)),\ P(s(0)) \to P(s(s(0))) \vdash P(s(s(0)))$$

is a Herbrand-sequent of the sequent S defined in Example 3.

Herbrand-sequents are named after J. Herbrand who first proved (a stronger form of) the following theorem [21], see [22] for a contemporary exposition in English.

Theorem 5. *Let S be a* $\overline{\Sigma}_1$*-sequent. Then S is valid iff S has a Herbrand-sequent.*

Thus a Herbrand-sequent of S can be considered a proof of S. Consequently, the size of Herbrand-sequents is of interest to proof complexity.

Definition 23. Let S be a valid $\overline{\Sigma}_1$-sequent. Define $\mathcal{H}(S)$ to be the set of Herbrand-sequents of S and $\mathcal{H}^{\min}(S)$ to be the set of minimal (w.r.t. $\subseteq$) Herbrand-sequents of S. The *Herbrand-complexity of* S is defined as $\mathrm{Hc}(S) = \min\{|H| \mid H \in \mathcal{H}(S)\}$.

Note that $\mathrm{Hc}(S) = \min\{|H| \mid H \in \mathcal{H}^{\min}(S)\}$.

Example 7. *For $n \geq 1$ define the sequent*

$$S_n \;=\; P(0), \forall x\,(P(x) \to P(s(x))) \vdash P(s^{2^n}(0)).$$

Then S_n has the unique minimal Herbrand-sequent

$$H_n \;=\; P(0), P(0) \to P(s(0)), \ldots, P(s^{2^n-1}(0)) \to P(s^{2^n}(0)) \vdash P(s^{2^n}(0)).$$

Therefore $\mathrm{Hc}(S_n) = \Theta(2^n)$.

The Herbrand-complexity plays an important role in proof complexity since it is closely related to the complexity of proof systems used in automated theorem proving: it has been used in [37] for analysing the complexity of the ε-elimination procedure in Hilbert's ε-calculus, in [38] for analysing the complexity of a rule of function introduction into the resolution calculus, in [39] for analysing the effect of Skolemisation on proof length, in [40] for investigating the effect of preprocessing steps on the proof length in a tableau calculus, and in [41] for analysing proof length in the disconnection tableau calculus.

In our setting of the sequent calculus, the Herbrand-complexity is closely related to the number of weak quantifier inferences in a cut-free proof. The following Theorem 6 makes this precise by augmenting Theorem 5 with complexity bounds. In order to prove this result, we first have to define how to read off the term vectors used for instantiating a block of quantifiers in a proof. We will write ε for the empty vector and $(t, \overline{s})$ for the vector obtained from the vector $\overline{s}$ by prepending the term t.

Definition 24. Let ψ be a proof and let μ be an occurrence of a prenex formula. We associate a set $\mathrm{T}(\mu)$ of term vectors to μ by induction on the structure of ψ. If μ is a formula occurrence in an axiom or μ is main formula occurrence of a propositional inference or of a weakening, then $\mathrm{T}(\mu) = \{\varepsilon\}$. If μ is main formula occurrence of an $\exists_{\mathrm{r}}$-inference as in

$$\frac{\Gamma \vdash \Delta, (A[x\backslash t])_\nu}{\Gamma \vdash \Delta, (\exists x\, A)_\mu}\ \exists_{\mathrm{r}}$$

then $\mathrm{T}(\mu) = \{(t, \overline{s}) \mid \overline{s} \in \mathrm{T}(\nu)\}$. The other quantifier rules are handled analogously. If μ is main formula occurrence of a contraction as in

$$\frac{\Gamma \vdash \Delta, A_{\nu_1}, A_{\nu_2}}{\Gamma \vdash \Delta, A_\mu}\ \mathrm{c}_{\mathrm{r}}$$

then $\mathrm{T}(\mu) = \mathrm{T}(\nu_1) \cup \mathrm{T}(\nu_2)$. If μ occurs in the context of an inference and ν is its predecessor in the sequent above the inference, then $\mathrm{T}(\mu) = \mathrm{T}(\nu)$.

Example 8. *Letting μ be the occurrence of $\forall x\,(P(x) \to P(s(x)))$ in the end-sequent of the proof of Example 5, we have $\mathrm{T}(\mu) = \{(0), (s(0))\}$. Note that instantiating $\forall x\,(P(x) \to P(s(x)))$ with $\{(0), (s(0))\}$ and adding the quantifier-free part of the end-sequent of the proof of Example 5 gives the Herbrand-sequent of Example 6.*

Definition 25. Let S be a valid $\overline{\Sigma}_1$-sequent. Define the *cut-free complexity of* S as

$$\mathrm{cfc}(S) = \min\{|\psi|_{\mathrm{q}} \mid \psi \text{ cut-free proof of } S\}.$$

Theorem 6. *Let S be a valid $\overline{\Sigma}_1$-sequent. Then*

$$\mathrm{Hc}(S) - |S| \leq \mathrm{cfc}(S) \leq \|S\|\mathrm{Hc}(S).$$

Proof. For the lower bound on $\mathrm{cfc}(S)$, let ψ be a cut-free proof of S, let $S = \forall \overline{x}\, F_1, \ldots, \forall \overline{x}\, F_r \vdash \exists \overline{x}\, F_{r+1}, \ldots \exists \overline{x}\, F_m$ and, for $i \in \{1, \ldots, m\}$ let μ_i be the occurrence of $Q\overline{x}\, F_i$ in the end-sequent of ψ where $Q = \forall$ if $i \in \{1, \ldots, r\}$ and $Q = \exists$ if $i \in \{r+1, \ldots, m\}$. Let H be the sequent obtained from S by replacing $Q\overline{x}\, F_i$ by $\{F_i[\overline{x}\backslash \overline{t}] \mid \overline{t} \in \mathrm{T}(\mu_i)\}$ for all $i \in \{1, \ldots, m\}$. Then H is a sequent which consists of instances of S. For showing that H is a tautology proceed as follows: construct a proof ψ' by induction on ψ by omitting the quantifier inferences but carrying out all other inferences. Then ψ' is a proof of H, hence H is a tautology. Let $H = H_0 \uplus H_1$ where H_0 are the formulas from S which do not contain a quantifier and H_1 are the instances of formulas from S which do contain quantifiers. Then we have $|H_0| \leq |\psi|_{\mathrm{q}}$ and $|H_1| \leq |S|$ and therefore $|H| \leq |\psi|_{\mathrm{q}} + |S|$ which entails the lower bound on $\mathrm{cfc}(S)$.

For the upper bound, let H be a Herbrand-sequent of S. Since H is tautological, there is a proof χ of H consisting only of propositional and structural inferences. Since H consists only of instances of the formulas in S, there is a proof χ' of S from H consisting only of weak quantifier inferences and structural inferences. Therefore

$$\psi \quad = \quad \begin{array}{l} (\chi) \\ H \\ \vdots \\ S \end{array} (\chi')$$

is a cut-free proof of S. Furthermore, if H contains n instances of a formula φ of S which starts with k quantifiers, then $k \cdot n$ quantifier inferences suffice in χ' for deriving φ from its instances in H. The quantifier-free formulas of S either occur in H or are added in χ' by weakening. In either case, they do not contribute to $|\psi|_{\mathrm{q}}$. Therefore $|\psi|_{\mathrm{q}} \leq \|S\||H|$. □

Corollary 3. *Let $(S_n)_{n\geq 1}$ be a sequence of valid $\overline{\Sigma}_1$-sequents s.t. $\|S_n\| = O(1)$, then*

$$\mathrm{Hc}(S_n) = \Theta(\mathrm{cfc}(S_n)).$$

Proof. This follows directly from Theorem 6 and the observation that $|S| \leq \|S\|$ for any sequent S. □

5.4. Proofs with $\overline{\Pi}_1$-cuts

In this section, we will extend the notion of Herbrand-complexity to cover also proofs with cut. In order to do that, we proceed in the spirit of the above Corollary 3: we count the number of weak quantifier inferences in a proof with cut. Before defining this complexity measure precisely, we make some preparatory observations.

Lemma 13. *Let ψ be a proof with $\overline{\Pi}_1$-cuts of a sequent S. Then there is a proof ψ' with $\overline{\Pi}_1$-cuts of S s.t. $|\psi'|_{\mathrm{q}} \leq |\psi|_{\mathrm{q}}$ and every cut in ψ' is of the form*

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta, A[\overline{x}\backslash\overline{\alpha}]}}{\Gamma \vdash \Delta, \forall \overline{x}\, A}\,\forall_{\mathrm{r}}^{n} \qquad \dfrac{\vdots}{(\forall \overline{x}\, A)_{\mu}, \Pi \vdash \Lambda}}{\Gamma, \Pi \vdash \Delta, \Lambda}\,\mathrm{cut}$$

where $\forall_{\mathrm{r}}^{n}$ denotes n applications of the $\forall_{\mathrm{r}}$-rule, $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$, and $\mathrm{T}(\mu)$ consists only of vectors of length n.

Proof. By common proof-theoretic pruning techniques: shift $\forall_{\mathrm{r}}$ downwards, possibly identifying different eigenvariables of the same x_i until the left hand side of the cut is of the required form. For the right hand side of the cut, shift weakenings downwards until only such ancestor paths of μ remain which instantiate all x_i. □

Therefore, w.l.o.g. we assume that all our proofs are of the shape guaranteed to exist by the above lemma. We then say that $(\alpha_1, \ldots, \alpha_n)$ is the eigenvariable vector of this cut. Symmetrically, we say that $(t_1, \ldots, t_n)$ is a term vector of this cut if $(t_1, \ldots, t_n) \in \mathrm{T}(\mu)$.

Let $C_1, \ldots, C_n$ be the cuts of a proof ψ whose cut formulas contain a quantifier. For $i \in \{1, \ldots, n\}$ let $\overline{\alpha_i}$ be the eigenvariable vector of C_i and $\overline{t_{i,1}}, \ldots, \overline{t_{i,l_i}}$ be the term vectors of C_i. We write $C_i <_{\psi}^{1} C_j$ if one of the $\alpha_{j,p}$ appears in one of the $t_{i,q}$. We write $<_{\psi}$ for the transitive closure of $<_{\psi}^{1}$. The definition of this order is motivated by the following consideration: during cut-elimination the eigenvariable vector $\overline{\alpha}$ of a cut C will be replaced by the term vectors $\overline{t}$ of C. So if another cut C' with eigenvariable vector $\overline{\alpha'}$ has a term vector $\overline{s}$ which contains an α_i, then the final value of $\overline{\alpha'}$ depends on that of $\overline{\alpha}$. This dependence is expressed by $C' <_{\psi} C$. The following result has been shown in [42, Lemma 10].

Lemma 14. *Let ψ be a proof with $\overline{\Pi}_1$-cuts. Then $<_{\psi}$ is acyclic.*

Let $\forall x_1 \cdots \forall x_n\, A$ with A quantifier-free be a $\overline{\Pi}_1$-formula. Then the type of A is $\tau(A) = n$. The type of a $\overline{\Pi}_1$-cut is the type of its cut formula. Let ψ be a proof with $\overline{\Pi}_1$-cuts $C_1, \ldots, C_n$ ordered s.t. $C_i <_{\psi} C_j$ implies $i < j$, then we say that ψ *is of type* $(\tau(C_1), \ldots, \tau(C_n))$. Note that the type of a proof is not unique: different linearisations of $<_{\psi}$ may give rise to different types.

Definition 26. Let S be a valid $\overline{\Sigma}_1$-sequent and $\tau \in \mathbb{N}^k$. Define

$$\overline{\Pi}_1 c_\tau(S) = \min\{|\psi|_q \mid \psi \text{ is proof of } S \text{ with } \overline{\Pi}_1\text{-cuts of type } \tau\}$$

$$\overline{\Pi}_1 c(S) = \min\{|\psi|_q \mid \psi \text{ is proof of } S \text{ with } \overline{\Pi}_1\text{-cuts}\}$$

$$\Pi_1 c(S) = \min\{|\psi|_q \mid \psi \text{ is proof of } S \text{ with } \overline{\Pi}_1\text{-cuts of a type } (1,\dots,1)\}$$

Note that cfc is an upper bound on $\overline{\Pi}_1 c$, i.e., for every type τ and every valid $\overline{\Sigma}_1$-sequent S we have $\overline{\Pi}_1 c_\tau(S) \leq \mathrm{cfc}(S)$ and hence also $\overline{\Pi}_1 c(S) \leq \mathrm{cfc}(S)$. This follows from the simple observation that a cut-free proof can be transformed to a proof with $\overline{\Pi}_1$-cuts by introducing useless cuts without increasing the number of weak quantifier inferences.

Also note that any $\overline{\Sigma}_1$-cut can be transformed into a $\overline{\Pi}_1$-cut without changing the number of weak quantifier inferences by replacing

$$\dfrac{\dfrac{(\psi_1)}{\Gamma \vdash \Delta, \exists \overline{x}\, A} \quad \dfrac{(\psi_2)}{\exists \overline{x} A, \Pi \vdash \Lambda}}{\Gamma, \Pi \vdash \Delta, \Lambda}\ \text{cut} \qquad \text{by} \qquad \dfrac{\dfrac{(\psi_2')}{\Pi \vdash \Lambda, \forall \overline{x} \neg A} \quad \dfrac{(\psi_1')}{\forall \overline{x} \neg A, \Gamma \vdash \Delta}}{\Gamma, \Pi \vdash \Delta, \Lambda}\ \text{cut}$$

where ψ_1' is obtained from ψ_1 by suitably replacing $\exists_r$-inferences by $\forall_l$-inferences and adding $\neg_l$-inferences as needed and analogously for ψ_2' and ψ_2.

6. Complexity of proofs and complexity of languages

In this section, we come back to formal language theory in order to describe the connection between the complexity of proofs and the complexity of languages. The development in this section follows and extends arguments that can be found in [19]. Intuitively, the following correspondences will be established:

$$\text{cardinality of language} \longleftrightarrow \text{Herbrand complexity}$$

$$\text{complexity of VTRATG} \longleftrightarrow \text{quantifier complexity of proof with } \overline{\Pi}_1\text{-cuts}$$

$$\text{computation of } L(G) \longleftrightarrow \text{cut-elimination}$$

In order to relate proofs to formal languages, we will identify a Herbrand-sequent with a finite tree language as follows: for a $\overline{\Sigma}_1$-sequent $\Gamma \vdash \Delta$ define the ranked alphabet $\Sigma_{\Gamma\vdash\Delta} = \Sigma \cup \{f_{A\vdash} \mid A \in \Gamma\} \cup \{f_{\vdash A} \mid A \in \Delta\}$ where Σ is the set of function and constant symbols appearing in $\Gamma \vdash \Delta$. The Herbrand-sequent $\Gamma' \vdash \Delta'$ of $\Gamma \vdash \Delta$ is then identified with the tree language

$$\{f_{A\vdash}(\bar{t}) \mid A = \forall \overline{x}\, B \in \Gamma,\ B \text{ quantifier-free},\ B[\overline{x}\backslash\bar{t}] \in \Gamma'\} \cup$$

$$\{f_{\vdash A}(\bar{t}) \mid A = \exists \overline{x}\, B \in \Delta,\ B \text{ quantifier-free},\ B[\overline{x}\backslash\bar{t}] \in \Delta'\}.$$

Example 9. *The Herbrand-sequent defined in Example 6 is*

$$\{f_{P(0)\vdash},\ f_{\forall x\,(P(x)\to P(s(x))\vdash}(0),\ f_{\forall x\,(P(x)\to P(s(x))\vdash}(s(0)),\ f_{\vdash P(s(s(0)))}\}$$

as tree language, cf. also Example 8.

Note that the size of a Herbrand-sequent is the size of its tree language. Consequently, $\mathrm{Hc}(S)$ is the size of the smallest tree language representing a Herbrand-sequent of S. Theorem 6 and Corollary 3 show that this size is closely related to the cut-free complexity.

The key result for the connection between the complexity of proofs with cut and that of finite languages is the following Theorem 7, which is an analogue of Theorem 6 for proofs with $\overline{\Pi}_1$-cuts. It permits to analyse the proof complexity of S by analysing the grammatical complexity of the finite tree languages induced by the Herbrand-sequents of S.

Definition 27. Let ψ be a proof with $\overline{\Pi}_1$-cuts of a $\overline{\Sigma}_1$-sequent S. We define a VTRATG $G(\psi)$ as follows: let $C_1, \dots, C_n$ be the cuts of ψ whose cut formulas contain a quantifier, ordered s.t. $C_i <_\psi C_j$ implies $i < j$. Let $\overline{\alpha_i}$ be the eigenvariable vector of C_i and $\overline{t_{i,1}}, \dots, \overline{t_{i,l_i}}$ the term vectors of C_i. Let $S = \forall \overline{x} F_1, \dots, \forall \overline{x} F_r \vdash \exists \overline{x} F_{r+1}, \dots \exists \overline{x} F_m$, let μ_i be the occurrence of $Q\overline{x} F_i$ in the end-sequent of ψ and write f_i for the function symbol $f_{\forall \overline{x} F_i\vdash}$ if $1 \leq i \leq r$ and for $f_{\vdash \exists \overline{x} F_i}$ if $r+1 \leq i \leq m$.

Then $G(\psi) = (\alpha_{0,1}, N, \Sigma, P)$ where $N = ((\alpha_{0,1}), \overline{\alpha_1}, \dots, \overline{\alpha_n})$, Σ is the signature of ψ together with $f_1, \dots, f_m$, and

$$P = \{(\alpha_{0,1}) \to (f_i(\bar{t})) \mid \bar{t} \in \mathrm{T}(\mu_i), 1 \leq i \leq m\} \cup \{\overline{\alpha_i} \to \overline{t_{i,j}} \mid 1 \leq i \leq n, 1 \leq j \leq l_i\}.$$

Example 10. *Let* $\psi_1 =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{P(\alpha_{1,1}) \to P(s(\alpha_{1,1})),\ P(s(\alpha_{1,1})) \to P(s(s(\alpha_{1,1}))) \vdash P(\alpha_{1,1}) \to P(s(s(\alpha_{1,1})))}\ \to_\mathrm{l}}{P(\alpha_{1,1}) \to P(s(\alpha_{1,1})),\ \forall x\,(P(x) \to P(s(x))) \vdash P(\alpha_{1,1}) \to P(s(s(\alpha_{1,1})))}\ \forall_\mathrm{l}}{\forall x\,(P(x) \to P(s(x))),\ \forall x\,(P(x) \to P(s(x))) \vdash P(\alpha_{1,1}) \to P(s(s(\alpha_{1,1})))}\ \mathrm{c}_\mathrm{l}}{\forall x\,(P(x) \to P(s(x))) \vdash P(\alpha_{1,1}) \to P(s(s(\alpha_{1,1})))}\ \forall_\mathrm{r}}{\forall x\,(P(x) \to P(s(x))) \vdash \forall x\,(P(x) \to P(s(s(x))))}$$

and $\psi_2 =$

$$\dfrac{\dfrac{\vdots}{P(0) \to P(s(s(0)))),\ P(0) \vdash P(s(s(0)))}}{\forall x\,(P(x) \to P(s(s(x)))),\ P(0) \vdash P(s(s(0)))}\ \forall_\mathrm{l}$$

and $\psi =$

$$\dfrac{(\psi_1) \qquad\qquad (\psi_2)}{P(0), \forall x\,(P(x) \to P(s(x))) \vdash P(s(s(0)))}\ \mathrm{cut}$$

where the vertical dots abbreviate short (and for the purposes of this example: irrelevant) propositional proofs. Omitting vector notation since all vectors are of length 1*, we have* $G(\psi) = (\alpha_{0,1}, N, \Sigma, P)$ *where* $N = (\alpha_{0,1}, \alpha_{1,1})$*, and* $P =$

$$\alpha_{0,1} \to f_{P(0)\vdash} \mid f_{\forall x\,(P(x)\to P(s(x)))\vdash}(\alpha_{1,1}) \mid f_{\forall x\,(P(x)\to P(s(x)))\vdash}(s(\alpha_{1,1})) \mid f_{\vdash P(s(s(0)))},$$
$$\alpha_{1,1} \to 0$$

The reader is invited to observe that $L(G(\psi))$ *is the Herbrand sequent of Example 9.*

Lemma 15. *Let* S *be a* $\overline{\Sigma}_1$*-sequent and let* ψ *be a proof of* S *with* $\overline{\Pi}_1$*-cuts, then* $L(G(\psi))$ *is a Herbrand-sequent of* S*.*

Proof. We will proceed by induction on the number of cuts in ψ which contain quantifiers. If all cuts in ψ are quantifier-free, then the nonterminals of $G(\psi)$ are of the form $N = ((\alpha_{0,1}))$ and therefore $L(G(\psi)) = \{t \mid (\alpha_{0,1}) \to (t) \in P\}$ where P are the productions of $G(\psi)$. Then an argument as in Theorem 6 shows that $L(G(\psi))$ is a tautology by transforming ψ into a proof ψ' of $L(G(\psi))$ which does not contain quantifier inferences.

Let C_1 be a $<_\psi$-minimal cut which contains quantifiers. Let $\overline{t_1}, \ldots, \overline{t_n}$ be the term vectors of C_1, let $\overline{\alpha_1}$ be the eigenvariable vector of C_1, and let $\overline{\alpha_1}$ be of length k. Then the subproof of ψ which ends with C_1 is of the form

$$\dfrac{\dfrac{(\chi)}{\Gamma \vdash \Delta, A[\overline{x}\backslash\overline{\alpha_1}]}\ \forall_\mathrm{r}^k \qquad \dfrac{\dfrac{\vdots}{A_1[x_k\backslash t_{1,k}], \Pi_1 \vdash \Lambda_1}\ \forall_\mathrm{l} \quad \cdots \quad \dfrac{\vdots}{A_n[x_k\backslash t_{n,k}], \Pi_n \vdash \Lambda_n}\ \forall_\mathrm{l}}{\forall x_k\, A_1, \Pi_1 \vdash \Lambda_1 \quad \cdots \quad \forall x_k\, A_n, \Pi_n \vdash \Lambda_n}}{\Gamma, \Pi \vdash \Delta, \Lambda}\ \mathrm{cut}$$

where $\forall_\mathrm{r}^k$ abbreviates k application of the $\forall_\mathrm{r}$-rule, the $\forall_\mathrm{l}$-inferences on $t_{1,k}, \ldots, t_{1,n}$ may appear in any partial order in the subproof on the right above C_1, and $A_i = A[x_1\backslash t_{i,1}, \ldots, x_{k-1}\backslash t_{i,k-1}]$ for $i \in \{1, \ldots, n\}$. Define ψ' from ψ by replacing this subproof with

$$\dfrac{\dfrac{(\chi[\overline{\alpha_1}\backslash\overline{t_1}])}{\Gamma \vdash \Delta, A_1[x_k\backslash t_{1,k}]} \quad \dfrac{\vdots}{A_1[x_k\backslash t_{1,k}], \Pi_1 \vdash \Lambda_1}}{\Gamma, \Pi_1 \vdash \Delta, \Lambda_1}\ \mathrm{cut} \quad \cdots \quad \dfrac{\dfrac{(\chi[\overline{\alpha_1}\backslash\overline{t_n}])}{\Gamma \vdash \Delta, A_n[x_k\backslash t_{n,k}]} \quad \dfrac{\vdots}{A_n[x_k\backslash t_{n,k}], \Pi_n \vdash \Lambda_n}}{\Gamma, \Pi_n \vdash \Delta, \Lambda_n}\ \mathrm{cut}$$
$$\dfrac{\vdots}{\Gamma, \Pi \vdash \Delta, \Lambda}$$

where the newly introduced cuts replace the $\forall_\mathrm{l}$-inferences on $t_1, \ldots, t_n$ of ψ. Since C_1 is $<_\psi$-minimal, we are in the situation of Lemma 9 and have, in the notation of that Lemma, $G(\psi') = G'(\psi)$ and hence $L(G(\psi')) = L(G(\psi))$. □

Lemma 16. *Let* S *be a* $\overline{\Sigma}_1$*-sequent and let* G *be a VTRATG of type* $(1, \tau)$ *s.t.* $L(G)$ *is a Herbrand sequent of* S*. Then there is a proof* ψ *of* S *with* $\overline{\Pi}_1$*-cuts of type* τ *s.t.* $|\psi|_\mathrm{q} \leq \|S\| \cdot |G|$*.*

Proof. If τ is empty, the result has already been shown in the proof of Theorem 6, so let τ be non-empty. Let $N = ((\alpha_{0,1}), \overline{\alpha_1}, \ldots, \overline{\alpha_n})$ be the nonterminals of G and P the productions of G. Let $S = \Gamma \vdash \Delta$ where $\Gamma = \forall \overline{x} F_1, \ldots, \forall \overline{x} F_r$ and $\Delta = \exists \overline{x} F_{r+1}, \ldots, \exists \overline{x} F_m$. For $k \in \{0, \ldots, n\}$ define $T_k = \{\overline{t} \mid \overline{\alpha_k} \to \overline{t} \in P\}$ and for $i \in \{1, \ldots, m\}$ define $T_{0,i} = \{\overline{t} \mid (\alpha_{0,i}) \to f_i(\overline{t}) \in P\}$ where $f_i = f_{\forall \overline{x} F_i \vdash}$ for $i \in \{1, \ldots, r\}$ and $f_i = f_{\vdash \exists \overline{x} F_i}$ for $i \in \{r+1, \ldots, m\}$. Let l_i be the number of quantifiers in $Q \overline{x} F_i$. Let

$$G_i = \begin{cases} F_i & \text{if } i \in \{1, \ldots, r\} \text{ and } l_i = 0 \\ \bigwedge_{\overline{t} \in T_{0,i}} F_i[\overline{x} \backslash \overline{t}] & \text{if } i \in \{1, \ldots, r\} \text{ and } l_i > 0 \\ \neg F_i & \text{if } i \in \{r+1, \ldots, m\} \text{ and } l_i = 0 \\ \bigwedge_{\overline{t} \in T_{0,i}} \neg F_i[\overline{x} \backslash \overline{t}] & \text{if } i \in \{r+1, \ldots, m\} \text{ and } l_i > 0 \end{cases}$$

Furthermore, let

$$C_1 = \bigwedge_{i=1}^{m} G_i, \text{ and } C_{k+1} = \bigwedge_{\overline{t} \in T_k} C_k[\overline{\alpha_k} \backslash \overline{t}] \text{ for } k \in \{1, \ldots, n-1\}.$$

Define the proof $\psi(G)$ as

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \Delta, \forall \overline{\alpha_1} C_1}(\chi_1) \quad \dfrac{}{\forall \overline{\alpha_1} C_1 \vdash \forall \overline{\alpha_2} C_2}(\chi_2)}{\Gamma \vdash \Delta, \forall \overline{\alpha_2} C_2}\,\text{cut} \quad \dfrac{}{\forall \overline{\alpha_2} C_2 \vdash \forall \overline{\alpha_3} C_3}(\chi_3)}{\Gamma \vdash \Delta, \forall \overline{\alpha_2} C_2}\,\text{cut}$$
$$\vdots$$
$$\dfrac{\Gamma \vdash \Delta, \forall \overline{\alpha_n} C_n \qquad \dfrac{}{\forall \overline{\alpha_n} C_n \vdash}(\chi_{n+1})}{\Gamma \vdash \Delta}\,\text{cut}$$

For $i \in \{1, \ldots, n+1\}$, the proof χ_i proceeds by using strong quantifier inferences with identity substitutions and weak quantifier inferences on the term vectors in T_{i-1}. This leads to quantifier-free tautological sequents H_i for $i \in \{1, \ldots, n\}$ because of the definition of C_i and for $i = n+1$ because H_{n+1} is logically equivalent to $L(G)$ which is tautological by assumption. Since all χ_i are cut-free, ψ is of type τ. Furthermore, we have $|\chi_i|_q = \|T_i\|$ for $i \in \{2, \ldots, n+1\}$. The number of weak quantifier inferences applied to $Q \overline{x} F_i$ in χ_1 is bounded by $\|S\| \cdot |T_{0,i}|$ and therefore $|\chi_1|_q \leq \|S\| \cdot |T_0|$ and we obtain $|\psi(G)|_q \leq \|S\| \cdot |G|$. □

Note that $\psi(G)$ as constructed in the above proof satisfies $G(\psi(G)) = G$.

Theorem 7. *Let S be a valid $\overline{\Sigma}_1$-sequent. Then*

$$\min\{\mathrm{VTRATc}_{(1,\tau)}(H) \mid H \in \mathcal{H}(S)\} - |S| \leq \overline{\Pi}_1 \mathrm{c}_\tau(S), \text{ and } \overline{\Pi}_1 \mathrm{c}_\tau(S) \leq \|S\| \min\{\mathrm{VTRATc}_{(1,\tau)}(H) \mid H \in \mathcal{H}(S)\}.$$

Proof. Let ψ be a proof of S with $\overline{\Pi}_1$-cuts of type τ with minimal quantifier complexity, i.e., $|\psi|_q = \overline{\Pi}_1 \mathrm{c}_\tau(S)$. Then $L(G(\psi))$ is a Herbrand-sequent of S by Lemma 15. Furthermore, Definition 27 directly entails that the type of $G(\psi)$ is $(1, \tau)$ and that $|G(\psi)| \leq |\psi|_q + |\{\varphi \in S \mid \varphi \text{ quantifier-free}\}| \leq |\psi|_q + |S|$. Therefore the first in equality holds.

Let G be a VTRATG of type $(1, \tau)$ of minimal complexity s.t. $L(G)$ is a Herbrand-sequent of S. Then by Lemma 16 there is a proof ψ of S with $\overline{\Pi}_1$-cuts of type τ s.t. $|\psi|_q \leq \|S\||G|$ which shows the second inequality. □

Corollary 4. *Let $(S_n)_{n \geq 1}$ be a sequence of valid $\overline{\Sigma}_1$-sequents s.t. $\|S_n\| = O(1)$. Then*

1. $\overline{\Pi}_1 \mathrm{c}_\tau(S_n) = \Theta(\min\{\mathrm{VTRATc}_{(1,\tau)}(H) \mid H \in \mathcal{H}(S_n)\})$.
2. $\overline{\Pi}_1 \mathrm{c}_\tau(S_n) = \Theta(\min\{\mathrm{VTRATc}_{(1,\tau)}(H) \mid H \in \mathcal{H}^{\min}(S_n)\})$.
3. $\overline{\Pi}_1 \mathrm{c}(S_n) = \Theta(\min\{\mathrm{TRATc}(H) \mid H \in \mathcal{H}^{\min}(S_n)\})$.
4. *If, for all $n \geq 1$, S_n has a unique minimal Herbrand-sequent $H_n^{\min}$, then* $\overline{\Pi}_1 \mathrm{c}(S_n) = \Theta(\mathrm{TRATc}(H_n^{\min}))$.

Proof. 1 follows directly from Theorem 7 and the observation that $\|S\| \geq |S|$. 2 follows from 1 and the observation that the smallest grammar which covers a Herbrand-sequent is the smallest grammar which covers a minimal Herbrand-sequent. 3 follows from 2 and Proposition 5. 4 follows directly from 3. □

For 2, 3, and 4 in the above corollary to hold, it is indispensable to use the cover formulation for the grammatical complexity of a finite language. Using the equality formulation would not allow the reduction from all to the minimal Herbrand-sequents.

The strength of this result lies in the fact that it is often possible to construct $(S_n)_{n\geq 1}$ of constant logical complexity s.t. S_n has a unique minimal Herbrand-sequent $H_n^{\min}$. For such a sequence, Corollary 4/4 is applicable, thus reducing, for each n, the problem of determining the minimal number of weak quantifier inferences needed for proving a sequent with $\overline{\Pi}_1$-cuts to the problem of determining the TRAT-complexity of a single tree language.

Example 11. *Letting S_n be the sequents defined in Example 7 note that by using the cut formulas $\forall x\,(P(x) \to P(s^{2^i}(x)))$ for $i \in \{1,\dots,n\}$ one can find a proof π_n with cut having only $O(n)$ inferences. Furthermore, the proof π_n corresponds to the following TRATG G_n. Omitting vector notation since all vectors have length 1 we have $G_n = (\alpha_0, (\alpha_0, \alpha_1, \dots, \alpha_n), \Sigma, P_n)$ where P_n consists of the following rules:*

$$\alpha_0 \to f_{P(0)\vdash} \mid f_{\vdash P(s^{2^n}(0))} \mid f_{\forall x\,(P(x)\to P(s(x)))\vdash}(\alpha_1) \mid f_{\forall x\,(P(x)\to P(s(x)))\vdash}(s(\alpha_1))$$

$$\alpha_1 \to \alpha_2 \mid s^2(\alpha_2),\ \dots,\ \alpha_{n-1} \to \alpha_n \mid s^{2^{n-1}}(\alpha_n), \alpha_n \to 0$$

As the interested reader is invited to verify, the language $L(G_n)$ covers (and is in fact equal to) the minimal Herbrand sequent H_n of S_n (see Example 7). So we have $\mathrm{TRATc}(H_n) = O(n)$ and hence by Corollary 4/4 that $\overline{\Pi}_1\mathrm{c}(S_n) = O(n)$. On the other hand, as shown in Example 7, we have $\mathrm{Hc}(S_n) = \Theta(2^n)$ and thus, by Corollary 3, also $\mathrm{cfc}(S_n) = \Theta(2^n)$.

As a first demonstration of the usefulness of Corollary 4 we show that the complexity of proofs with cut is independent on whether blocks of quantifiers are allowed in the following sense:

Corollary 5. *If $(S_n)_{n\geq 1}$ is a sequence of valid $\overline{\Sigma}_1$-sequents s.t. $\|S_n\| = O(1)$, then $\overline{\Pi}_1\mathrm{c}(S_n) = \Theta(\Pi_1\mathrm{c}(S_n))$.*

Proof. By Corollary 4/1 and Proposition 5. □

Based on a more detailed analysis of the situation in Theorem 7 one can prove a stronger version of this theorem which allows to show that even $\overline{\Pi}_1\mathrm{c}(S) = \Pi_1\mathrm{c}(S)$ for every valid $\overline{\Sigma}_1$-sequent S.

7. Lower bounds on the length of proofs

7.1. Incompressibility by one cut with one quantifier

In this section we will describe a non-trivial application of Corollary 4 transferring Theorem 2 to proofs. To that aim, we work with the ranked alphabet $\Sigma = \{s/1, 0/0\}$. Terms in Σ denote natural numbers. For $n \in \mathbb{N}$ we write $\overline{n}$ for the numeral $s^n(0)$.

Definition 28. For $n \geq 1$ let τ_n be the sequent

$$P(\overline{n}, 0), \forall x \forall y\,(P(s(x), y) \to P(x, s(y))) \vdash P(0, \overline{n})$$

We abbreviate the formula $\forall x \forall y\,(P(s(x), y) \to P(x, s(y)))$ by A. The first step to analysing the descriptional complexity of τ_n is to understand its set of minimal Herbrand-sequents.

Lemma 17. *For every $n \geq 1$ the sequent τ_n has the unique minimal Herbrand sequent $H_n^{\min} =$*

$$P(\overline{n}, 0), P(\overline{n}, 0) \to P(\overline{n-1}, \overline{1}), \dots, P(\overline{1}, \overline{n-1}) \to P(0, \overline{n}) \vdash P(0, \overline{n})$$

Proof. It is easy to see that $H_n^{\min}$ is a Herbrand-sequent of τ_n. For showing that it is the unique minimal Herbrand-sequent, let H be a sequent consisting of instances of τ_n s.t. $H_n^{\min} \not\subseteq H$. We will construct a countermodel for H. If $P(\overline{n}, 0) \vdash \not\subseteq H$, let $\mathcal{M}$ be a structure where $P^{\mathcal{M}}$ is always false, then $\mathcal{M} \not\models H$. Symmetrically, if $\vdash P(0, \overline{n}) \not\subseteq H$, let $\mathcal{M}$ be a structure where $P^{\mathcal{M}}$ is always true, then $\mathcal{M} \not\models H$. So assume $P(\overline{n}, 0) \vdash P(0, \overline{n}) \subseteq H$. Since $\mathcal{H}^{\min}(\tau_n) \not\subseteq H$ we have

$$X = \{(k, l) \in \mathbb{N} \times \mathbb{N} \mid k + l = n - 1,\ P(\overline{k+1}, \overline{l}) \to P(\overline{k}, \overline{l+1}) \vdash \not\subseteq H\} \neq \emptyset.$$

Let $(a, b) \in X$ be s.t. $b \leq l$ for all $(k, l) \in X$. Fix the domain of $\mathcal{M}$ to be $\mathbb{N}$ and the interpretation of 0 and s to the standard interpretation. Define

$$P^{\mathcal{M}}(k, l) = \begin{cases} \text{true} & \text{if } k + l = n - 1 \text{ and } l < b \\ \text{false} & \text{otherwise} \end{cases}$$

We will now define H_1, H_2, H_3 s.t. $H \subseteq H_1 \cup H_2 \cup H_3$. Let

$$H_1 = P(\overline{n}, 0),\ P(\overline{n}, 0) \to P(\overline{n-1}, \overline{1}), \ldots, P(\overline{a+2}, \overline{b-1}) \to P(\overline{a+1}, \overline{b}) \vdash$$
$$H_2 = P(\overline{a}, \overline{b+1}) \to P(\overline{a-1}, \overline{b+2}), \ldots, P(\overline{1}, \overline{n-1}) \to P(0, \overline{n}) \vdash P(0, \overline{n})$$
$$H_3 = H \setminus (H_1 \cup H_2)$$

Now $\mathcal{M} \not\models H_1$, $\mathcal{M} \not\models H_2$, and $\mathcal{M} \not\models H_3$. Therefore $\mathcal{M} \not\models H$. □

The following result shows that cut-free proofs of τ_n cannot be compressed by the introduction of a single cut with a single quantifier.

Theorem 8. *For all $n \geq 1$: $\overline{\Pi}_1 \mathrm{c}_{(1)}(\tau_n) = \Theta(\mathrm{cfc}(\tau_n))$.*

Proof. The Herbrand-sequent $H_n^{\min}$ written as a tree language is

$$\{f_{P(\overline{n},0)\vdash},\ f_{\vdash P(0,\overline{n})}\} \cup \{f_{A\vdash}(\overline{k}, \overline{l}) \mid k + l = n\}$$

which − when identifying $f_{A\vdash}$ with f − contains the language T_n of Definition 20. By Theorem 2 we have $\mathrm{VTRATc}_{(1,1)}(T_n) \geq |T_n|$ and hence $\mathrm{VTRATc}_{(1,1)}(H_n^{\min}) \geq |T_n|$. By Lemma 17 we have $\mathrm{Hc}(\tau_n) = n + 3$ and hence, by Corollary 4/2, that $\overline{\Pi}_1 \mathrm{c}_{(1)}(\tau_n) = \Omega(\mathrm{Hc}(\tau_n))$. But since $\overline{\Pi}_1 \mathrm{c}_\tau(\tau_n) = O(\mathrm{Hc}(\tau_n))$ for any type τ we obtain $\overline{\Pi}_1 \mathrm{c}_{(1)}(\tau_n) = \Theta(\mathrm{Hc}(\tau_n))$ and finally $\overline{\Pi}_1 \mathrm{c}_{(1)}(\tau_n) = \Theta(\mathrm{cfc}(\tau_n))$ by Corollary 3. □

7.2. A lower bound on $\overline{\Pi}_1$-complexity

As main application of Corollary 4 in this paper, we will now use Theorem 1 for obtaining a sequence of sequents which cannot be compressed exponentially by $\overline{\Pi}_1$-cuts (but only at most quadratically). We work with the ranked alphabet $\Sigma = \{f_0/1, f_1/1, f_s/1, e/0, s/1, 0/0\}$ whose first part is the translation of the unranked alphabet $\{0, 1, \mathrm{s}\}$ to trees and whose second part is used for representing natural numbers.

Definition 29. For $n \geq 1$ define the sequent $\sigma_n :=$

$$\forall y \forall v\ P(0, y, v, v), \qquad (P_1)$$
$$\forall x \forall y \forall v\ (P(x, f_s(y), v, v) \to P(s(x), y, e, v)), \qquad (P_2)$$
$$\forall x \forall y \forall u \forall v\ (P(x, f_0(y), u, v) \to P(x, y, f_0(u), v)), \qquad (P_3)$$
$$\forall x \forall y \forall u \forall v\ (P(x, f_1(y), u, v) \to P(x, y, f_1(u), v)), \qquad (P_4)$$
$$\forall v\ (P(\overline{k(2^n)}, e, v, v) \to Q(0, v)), \qquad (Q_1)$$
$$\forall x \forall v\ ((Q(x, f_0(v)) \wedge Q(x, f_1(v))) \to Q(s(x), v)) \qquad (Q_2)$$
$$\vdash$$
$$Q(\overline{n}, e) \qquad (Q_3)$$

These sequents are designed to force the use of the language L_{2^n} of Definition 14 in a Herbrand-sequent. The predicate P is used for checking whether a term represents a segmented word. In $P(x, y, u, v)$ the argument v is fixed to the reversed building block; the term to check is inserted for y. Its current segment is shifted to u by P_3 and P_4 (which reverses it). If u winds up equal to v and y starts with a separator, then x (which counts the number of segments encountered so far) can be incremented by P_2. The predicate Q is used for checking whether a complete binary tree, i.e., all building blocks of L_{2^n}, is present by the axioms Q_1, Q_2, and Q_3.

Lemma 18. *For all $n \geq 1$, the sequent σ_n satisfies $\mathrm{Hc}(\sigma_n) = \Theta((2^n)^2)$ and has a unique minimal Herbrand-sequent which contains*

$$M_n = \{f_{P_1}(w^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}) \mid w \in L_{2^n},\ v \text{ building block of } w\}.$$

Proof. A word $w \in L_{2^n}$ has a building block $v \in \{0,1\}^n$, i.e., $w = (\mathrm{s}v)^{k(2^n)}$. Write $w = a_0 a_1 \cdots a_{|w|-1}$ for $a_i \in \{0, 1, \mathrm{s}\}$. Note that every $i \in \{0, \ldots, |w| - 1\}$ can be written in a unique way as $i = q(i)(n+1) + r(i)$ where $0 \leq r(i) < n + 1$. Also note that $r(i) = 0$ iff $a_i = \mathrm{s}$. For $i \in \{0, \ldots, |w| - 1\}$ define

$$I_{w,i} = \begin{cases} f_{P_2}(q(i), (a_{i+1} \cdots a_{|w|-1})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}) & \text{if } a_i = \mathrm{s} \\ f_{P_3}(q(i), (a_{i+1} \cdots a_{|w|-1})^{\mathrm{T}}, ((a_{q(i)(n+1)+1} \cdots a_{i-1})^{\mathrm{R}})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}) & \text{if } a_i = 0 \\ f_{P_4}(q(i), (a_{i+1} \cdots a_{|w|-1})^{\mathrm{T}}, ((a_{q(i)(n+1)+1} \cdots a_{i-1})^{\mathrm{R}})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}) & \text{if } a_i = 1 \end{cases}$$

Furthermore, define

$$I_w = \{f_{P_1}(w^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}})\} \cup \{I_{w,i} \mid 0 \leq i < |w|\} \cup \{f_{Q_1}((v^{\mathrm{R}})^{\mathrm{T}})\}$$

and

$$J = \{f_{Q_2}(\overline{n - |u| - 1}, u^{\mathrm{T}}) \mid u \in \{0,1\}^{<n}\} \cup \{f_{Q_3}\}.$$

Then we claim that $H_n = \bigcup_{w \in L_{2^n}} I_w \cup J$ is the unique minimal Herbrand-sequent of σ_n. In order to show this, we need to show that 1. H_n is a Herbrand-sequent and 2. that every Herbrand-sequent H_n^* of σ_n satisfies $H_n^* \supseteq H_n$.

For 1. note that the instances of $P_1, \ldots, P_4$ in I_w form the chain of implications:

$$\begin{aligned}
f_{P_1}(w^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}): &\quad P(0, w^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}) \\
I_{w,0}: &\quad P(0, w^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}) \to P(\overline{1}, (v(sv)^{k(2^n)-1})^{\mathrm{T}}, e, (v^{\mathrm{R}})^{\mathrm{T}}) \\
&\vdots \\
I_{w,|w|-1}: &\quad P(\overline{k(2^n)}, f_{a_{|w|-1}}(e), ((b_0 \cdots b_{n-2})^{\mathrm{R}})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}) \to P(\overline{k(2^n)}, e, (v^{\mathrm{R}})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}})
\end{aligned}$$

where $v = b_0 \cdots b_{|v|-1}$ for $b_j \in \{0,1\}$. This chain allows to infer $P(\overline{k(2^n)}, e, (v^{\mathrm{R}})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}})$ in propositional logic. Hence, from the instances of Q_1 in H_n we obtain $Q(0, (v^{\mathrm{R}})^{\mathrm{T}})$ for all $v \in \{0,1\}^n$, i.e., $Q(0, u^{\mathrm{T}})$ for all $u \in \{0,1\}^n$. Now, by using the instances of Q_2 we can infer

$$\begin{aligned}
&Q(0, u^{\mathrm{T}}) \quad \text{for all } u \in \{0,1\}^n, \\
&Q(\overline{1}, u^{\mathrm{T}}) \quad \text{for all } u \in \{0,1\}^{n-1}, \\
&\vdots \\
&Q(\overline{n}, e)
\end{aligned}$$

in propositional logic which — together with Q_3 — shows that H_n is a tautology.

For 2. we proceed as in the proof of Lemma 17. Suppose H is a sequent of instances of S_n s.t. $H_n \not\subseteq H$. We construct a countermodel $\mathcal{M}$ of H. Take as domain all variable-free terms in the signature $\Sigma = \{f_0, f_1, f_s, e, s, 0\}$ and define $t^{\mathcal{M}} = t$. We can now freely choose the truth values of all variable-free atoms built from Σ and P and Q.

In order to choose these truth values, we can identify H_n with the unary/binary tree whose root is $Q(\overline{n}, e)$, whose leaves are $\{P(0, w^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}, (v^{\mathrm{R}})^{\mathrm{T}}) \mid w \in L_{2^n}\}$ and whose edges are defined as follows: a variable-free atom A has the two variable-free atoms A_1 and A_2 as parents if $(A_1 \wedge A_2) \to A$ is instance of Q_2 and a variable-free atom A has the single parent A_0 if $A_0 \to A$ is instance of P_2, P_3, P_4, or Q_1. It is straightforward to check that this is well-defined, i.e., that every node thus reachable from the root $Q(\overline{n}, e)$ has either 1 or 2 parents.

We define the truth values of atoms as follows: all leaves of H_n which appear in H are set to true. A node of H_n which appears in H is set to true iff all its parents are set to true. Since $H_n \not\subseteq H$, at least one edge from H_n is missing, hence the root $Q(\overline{n}, e)$ is set to false. By setting all atoms of H which do not appear in H_n to true, the sequent H evaluates to $\top, \ldots, \top \vdash \bot$, i.e., false in $\mathcal{M}$.

Now we have

$$\mathrm{Hc}(\sigma_n) = |H_n| = (\sum_{w \in L_{2^n}} |I_w|) + |J|$$

because $I_w \cap J = \emptyset$ for all w and $w_1 \neq w_2$ implies $I_{w_1} \cap I_{w_2} = \emptyset$. Furthermore,

$$= 2^n(|w| + 2) + |\{0,1\}^{<n}| + 1 = 2^n \Theta(2^n) + 2^n = \Theta((2^n)^2). \quad \square$$

Before we prove the lower bound on the $\overline{\Pi}_1$-complexity of σ_n we need to make a simple observation about tree languages:

Lemma 19. *Let $L \subseteq \mathcal{T}(\Sigma)$ be finite, let $w \mapsto v_w$ be a mapping from L to $\mathcal{T}(\Sigma)$, let $f \notin \Sigma$ be a binary function symbol and define $L' = \{f(w, v_w) \mid w \in L\}$. Then* $\mathrm{TRATc}(L') \geq \mathrm{TRATc}(L)$.

Proof. Let G' be a minimal TRATG which covers L'. Replace every occurrence of a term $f(t_1, t_2)$ in a production rule of G' by t_1. The grammar G thus obtained covers L and satisfies $|G| \leq |G'|$. $\square$

Theorem 9. *For all $n \geq 1$: $\overline{\Pi}_1\mathrm{c}(\sigma_n) = \Omega(\sqrt{\mathrm{cfc}(\sigma_n)})$.*

Proof. Applying Lemma 19 to Corollary 2 shows that $(M_n)_{n\geq 1}$ is TRAT-incompressible, i.e., there is an $M \in \mathbb{N}$ s.t. for all $n \geq M$ and every TRATG G_n which covers M_n we have $|G_n| \geq |M_n| = 2^n$. By Lemma 18, M_n is contained in the unique minimal Herbrand-sequent of S_n and hence Corollary 4/4 shows that $\overline{\Pi}_1\mathrm{c}(S_n) = \Omega(\mathrm{TRATc}(M_n)) = \Omega(2^n)$. On the other hand, by Lemma 18, we have $\mathrm{Hc}(S_n) = \Theta((2^n)^2)$ and hence, by Corollary 3, also $\mathrm{cfc}(S_n) = \Theta((2^n)^2)$. □

This theorem should be considered in the light of Example 11 which shows that there can be an exponential difference between the $\overline{\Pi}_1$-complexity and the cut-free complexity. What Theorem 9 thus provides is a non-trivial lower bound on $\overline{\Pi}_1\mathrm{c}(\cdot)$ in terms of $\mathrm{cfc}(\cdot)$.

Theorem 9 can be strengthened to take the total number of inferences into account: i.e., letting ψ_n be the shortest (w.r.t. $|\psi_n|$) proof of σ_n one can show that $\overline{\Pi}_1\mathrm{c}(\sigma_n) = \Omega(\sqrt{|\psi_n|})$ by constructing a cut-free proof that follows the informal proof that H_n is a tautology in Lemma 18.

In contrast to Section 7.1 and Theorem 8, in this section a square gets lost in the translation from tree languages to proofs. In fact, we do not know whether $\overline{\Pi}_1\mathrm{c}(S_n) = \Theta(\sqrt{\mathrm{Hc}(S_n)})$. But we conjecture that a $\overline{\Pi}_1$-incompressible sequence exists. More precisely:

Conjecture. *There is a sequence $(S_n)_{n\geq 1}$ of $\overline{\Sigma}_1$-sequents in a finite signature and of constant logical complexity s.t. $\overline{\Pi}_1\mathrm{c}(S_n) = \Theta(\mathrm{Hc}(S_n))$.*

8. Conclusion

We have investigated several measures of the grammatical complexity of finite word and tree languages. We have related these measures to each other and to well-known measures for the complexity of formal proofs. These results, in particular Corollary 4, show that there is a tight connection between the complexity of proofs with cut in first-order predicate logic and the grammatical complexity of Herbrand-sequents. We have demonstrated the strength of this connection by transferring several complexity results from the setting of formal languages to that of proofs. In particular, we have constructed a sequence of incompressible word languages and showed that it yields a lower bound on the length of proofs with $\overline{\Pi}_1$-cuts.

The problem of compressing a finite language by a grammar has received only little attention in the literature so far. Consequently there is a number of interesting open questions, for example: what is the complexity of the smallest grammar problem in this setting? How difficult is the approximation of the smallest grammar? Can approximation algorithms and techniques be carried over from the case of one word to this setting? How does the situation change when we do not minimise the number of production rules but the symbol complexity of the grammar?

Fast approximation algorithms for computing a minimal VTRATG that covers a given finite input language are also of high practical value for the cut-introduction method [18,19] and its implementation [20].

Furthermore, it would be interesting to extend this connection between the grammatical complexity of Herbrand sequents and the length of proofs with cut to larger classes of proofs, e.g., to $\overline{\Pi}_2$-cuts based on [43,44].

Acknowledgments

The trick underlying the proof of Proposition 4 is due to Giselle Reis. The authors are grateful to the anonymous referees for many helpful comments and suggestions which led to a considerable improvement of the paper.

References

[1] J.A. Storer, T.G. Szymanski, The macro model for data compression (extended abstract), in: ACM Symposium on Theory of Computing, STOC, 1978, ACM, New York, NY, USA, 1978, pp. 30–39.
[2] K. Casel, H. Fernau, S. Gaspers, B. Gras, M.L. Schmid, On the complexity of grammar-based compression over fixed alphabets, in: I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, D. Sangiorgi (Eds.), 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 55, 2016, pp. 122:1–122:14.
[3] C.G. Nevill-Manning, I.H. Witten, Identifying hierarchical structure in sequences: a linear-time algorithm, J. Artif. Intell. Res. 7 (1997) 67–82.
[4] N.J. Larsson, A. Moffat, Offline dictionary-based compression, in: Data Compression Conference, DCC, 1999, IEEE Computer Society, 1999, pp. 296–305.
[5] J.C. Kieffer, E.-H. Yang, Grammar-based codes: a new class of universal lossless source codes, IEEE Trans. Inf. Theory 46 (3) (2000) 737–754.
[6] W. Rytter, Application of Lempel–Ziv factorization to the approximation of grammar-based compression, Theor. Comput. Sci. 302 (1–3) (2003) 211–222.
[7] M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, A. Shelat, The smallest grammar problem, IEEE Trans. Inf. Theory 51 (7) (2005) 2554–2576.
[8] H. Sakamoto, A fully linear-time approximation algorithm for grammar-based compression, J. Discret. Algorithms 3 (2–4) (2005) 416–430.
[9] A. Jez, Approximation of grammar-based compression via recompression, Theor. Comput. Sci. 592 (2015) 115–134.
[10] A. Jez, A really simple approximation of smallest grammar, Theor. Comput. Sci. 616 (2016) 141–150.
[11] M. Lohrey, Algorithmics on SLP-compressed strings: a survey, Groups Complex. Cryptol. 4 (2) (2012) 241–299.
[12] W. Bucher, H.A. Maurer, K. Culik, D. Wotschke, Concise description of finite languages, Theoretical Computer Science 14 (1981) 227–246.
[13] W. Bucher, A note on a problem in the theory of grammatical complexity, Theor. Comput. Sci. 14 (1981) 337–344.
[14] B. Alspach, P. Eades, G. Rose, A lower-bound for the number of productions required for a certain class of languages, Discrete Appl. Math. 6 (2) (1983) 109–115.
[15] W. Bucher, H.A. Maurer, K. Culik, Context-free complexity of finite languages, Theor. Comput. Sci. 28 (1984) 277–285.
[16] Z. Tuza, On the context-free production complexity of finite languages, Discrete Appl. Math. 18 (3) (1987) 293–304.

[17] S. Hetzl, Applying tree languages in proof theory, in: A.-H. Dediu, C. Martín-Vide (Eds.), Language and Automata Theory and Applications, LATA, 2012, in: Lecture Notes in Computer Science, vol. 7183, Springer, 2012, pp. 301–312.
[18] S. Hetzl, A. Leitsch, D. 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.
[19] S. Hetzl, A. Leitsch, G. Reis, D. Weller, Algorithmic introduction of quantified cuts, Theor. Comput. Sci. 549 (2014) 1–16.
[20] S. Hetzl, A. Leitsch, G. Reis, J. Tapolczai, D. Weller, Introducing quantified cuts in logic with equality, in: S. Demri, D. Kapur, C. Weidenbach (Eds.), International Joint Conference on Automated Reasoning, IJCAR, 2014, in: Lecture Notes in Computer Science, vol. 8562, Springer, 2014, pp. 240–254.
[21] J. Herbrand, Recherches sur la théorie de la démonstration, Ph.D. thesis, Université de Paris, 1930.
[22] S.R. Buss, On Herbrand's Theorem, in: Logic and Computational Complexity, in: Lecture Notes in Computer Science, vol. 960, Springer, 1995, pp. 195–209.
[23] C. Câmpeanu, N. Santean, S. Yu, Minimal cover-automata for finite languages, Theor. Comput. Sci. 267 (1–2) (2001) 3–16.
[24] P. Pudlák, Twelve problems in proof complexity, in: E.A. Hirsch, A.A. Razborov, A.L. Semenov, A. Slissenko (Eds.), International Computer Science Symposium in Russia, CSR, in: Lecture Notes in Computer Science, vol. 5010, Springer, 2008, pp. 13–27.
[25] S. Eberhard, S. Hetzl, Compressibility of finite languages by grammars, in: J. Shallit, A. Okhotin (Eds.), Descriptional Complexity of Formal Systems, DCFS, 2015, in: Lecture Notes in Computer Science, vol. 9118, Springer, 2015, pp. 93–104.
[26] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.
[27] F. Jacquemard, F. Klay, C. Vacher, Rigid tree automata and applications, Inf. Comput. 209 (2011) 486–512.
[28] S. Buss, An introduction to proof theory, in: The Handbook of Proof Theory, North-Holland, 1999, pp. 2–78.
[29] G. Takeuti, Proof Theory, 2nd edition, North-Holland, Amsterdam, 1987.
[30] D. van Dalen, Logic and Structure, Springer, 2008.
[31] J. Harrison, Handbook of Practical Logic and Automated Reasoning, Cambridge University Press, 2009.
[32] G. Gentzen, Untersuchungen über das logische Schließen II, Math. Z. 39 (3) (1935) 405–431.
[33] J. Krajíček, P. Pudlák, The number of proof lines and the size of proofs in first order logic, Arch. Math. Log. 27 (1988) 69–84.
[34] R. Statman, Lower bounds on Herbrand's theorem, Proc. Am. Math. Soc. 75 (1979) 104–107.
[35] V. Orevkov, Lower bounds for increasing complexity of derivations after cut elimination, Zap. Nauč. Semin. LOMI 88 (1979) 137–161.
[36] P. Pudlák, The lengths of proofs, in: S. Buss (Ed.), Handbook of Proof Theory, Elsevier, 1998, pp. 547–637.
[37] G. Moser, R. Zach, The epsilon calculus and Herbrand complexity, Stud. Log. 82 (1) (2006) 133–155.
[38] M. Baaz, A. Leitsch, Complexity of resolution proofs and function introduction, Ann. Pure Appl. Log. 57 (1992) 181–215.
[39] M. Baaz, A. Leitsch, On skolemization and proof complexity, Fundam. Inform. 20 (4) (1994) 353–379.
[40] U. Egly, Quantifiers and the system KE: some surprising results, in: G. Gottlob, E. Grandjean, K. Seyr (Eds.), Computer Science Logic, CSL, 1998, in: Lecture Notes in Computer Science, vol. 1584, Springer, 1998, pp. 90–104.
[41] R. Letz, G. Stenz, Generalised handling of variables in disconnection tableaux, in: D.A. Basin, M. Rusinowitch (Eds.), International Joint Conference on Automated Reasoning, IJCAR, 2004, in: Lecture Notes in Computer Science, vol. 3097, Springer, 2004, pp. 289–306.
[42] S. Hetzl, On the form of witness terms, Arch. Math. Log. 49 (5) (2010) 529–554.
[43] B. Afshari, S. Hetzl, G.E. Leigh, Herbrand disjunctions, cut elimination and context-free tree grammars, in: T. Altenkirch (Ed.), Typed Lambda Calculi and Applications, TLCA, 2015, in: LIPIcs, vol. 38, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 1–16.
[44] B. Afshari, S. Hetzl, G.E. Leigh, Herbrand confluence for first-order proofs with Π_2-cuts, in: D. Probst, P. Schuster (Eds.), Concepts of Proof in Mathematics, Philosophy, and Computer Science, de Gruyter, 2016, pp. 5–40.