Complexity of Decision Problems on Totally Rigid Acyclic Tree Grammars

Sebastian Eberhard, Gabriel Ebner(✉), and Stefan Hetzl(✉)

TU Wien, Vienna, Austria
gebner@gebner.org, stefan.hetzl@tuwien.ac.at

Abstract. Totally rigid acyclic tree grammars (TRATGs) are an emerging grammatical formalism with numerous applications in proof theory and automated reasoning. We determine the computational complexity of several decision problems on TRATGs: membership, containment, disjointness, equivalence, minimization, and the complexity of minimal cover with a fixed number of nonterminals. We relate non-parametric minimal cover to a problem on regular word grammars of unknown complexity.

1 Introduction

The grammatical formalism of totally rigid acyclic tree grammars (TRATGs) originates in the proof-theoretical study of quantifier inferences in classical first-order logic [8]. These grammars describe the instance terms in proofs with universally quantified prenex cuts. The generated language is finite and isomorphic to a Herbrand disjunction (a tautological set of quantifier-free instances of the proven formula) describing a cut-free proof.

This connection between proof theory and formal language theory has been used for theoretical results on cut-elimination [10], lower bounds on the length of proofs with cut [5], as well as in automated reasoning for lemma generation [6,9] and automated inductive theorem proving [4]. Implementations of these applications are available in the GAPT system [7] for proof theory.

From an abstract point of view, this connection reveals a correspondence between the combinatorial kernel of a proof with cut and the grammar-based compression of a Herbrand disjunction, which is the essential information contained in a cut-free proof. Grammar-based compression is a well-established topic in formal language theory, popular grammar-based text compression algorithms include [12,13,15,17]. Many compression techniques have been adapted and extended to trees [1,14,18]. The smallest grammar problem is known to be NP-complete [2,16] and several approximation algorithms are known. The most important difference between the classical setting and ours is that a TRATG does not compress a single string or tree but a finite set of trees. We compress

Supported by the Vienna Science and Technology Fund (WWTF) project VRG12-004.

M. Hoshi and S. Seki (Eds.): DLT 2018, LNCS 11088, pp. 291–303, 2018.
https://doi.org/10.1007/978-3-319-98654-8_24

with regard to the number of production rules as opposed to, e.g., the size of the grammar as a binary string. Moreover, we cover the input language instead of reproducing it exactly.

In this paper we study and determine the computational complexity of decision problems associated with TRATGs. In Sect. 2 we define TRATGs and their derivations. In the following sections we show the complexity results as summarized in Table 1. We consider the minimal cover problem for both TRATGs and acyclic regular grammars on words in Sect. 8. The version of minimal cover where the number of nonterminals is bounded by a fixed parameter is NP-complete in both cases. Whether the unbounded minimal cover problem is NP-hard remains open in either case.

Table 1. Main complexity results shown in this paper.

TRATG-Membership	NP-complete	Sect. 3, Theorem 1
TRATG-Containment	Π_2^P-complete	Sect. 4, Theorem 2
TRATG-Disjointness	coNP-complete	Sect. 5, Theorem 3
TRATG-Equivalence	Π_2^P-complete	Sect. 6, Theorem 4
TRATG-n-Cover	NP-complete	Sect. 7.1, Theorem 5
Regular-Cover	NP	Sect. 7.2, Lemma 3
TRATG-Minimization	NP-complete	Sect. 8, Theorem 7

Each of these decision problems on TRATGs corresponds to a decision problem on proofs: for instance, TRATG-Membership decides whether a formula is contained in the Herbrand disjunction of a non-erasing cut-normal form.

2 Totally Rigid Acyclic Tree Grammars

We consider terms $t \in \mathcal{T}(\Sigma)$ over a signature Σ of function symbols with arity. Constants and nonterminals are just functions with arity 0. We write $\mathtt{f}/n \in \Sigma$ if the function symbol $\mathtt{f}$ has arity $n \geq 0$. A substitution is defined by a finite map that substitutes constants by other terms. We write $\sigma = [\mathsf{c}_1 \backslash t_1, \ldots, \mathsf{c}_n \backslash t_n]$ for the substitution σ that substitutes c_i by t_i. Substitutions use postfix notation: $t\sigma$ applies the substitution σ to the term t, and $\sigma\tau$ is the substitution such that $t(\sigma\tau) = (t\sigma)\tau$ for all terms t.

Definition 1 (*[8]*). *A TRATG is a tuple* $G = (\mathrm{A}, N, \Sigma, P)$ *such that:*

1. $\mathrm{A} \in N$ *is the start symbol,*
2. N *is a finite set of nonterminals,*
3. Σ *is a finite signature such that* $\Sigma \cap N = \emptyset$,
4. $P \subseteq N \times \mathcal{T}(\Sigma \cup N)$ *is a finite set of productions, writing* $\mathrm{B} \to r$ *for* $(\mathrm{B}, r) \in P$,
5. *there exists a strict linear order* $\prec$ *on the nonterminals such that* $\mathrm{B} \prec \mathrm{C}$ *whenever* $\mathrm{B} \to t \in P$ *for some* t *that contains* C.

Example 1. Let $N = \{\mathrm{A}, \mathrm{B}\}$, $\Sigma = \{\mathtt{f}/2, \mathtt{g}/1, \mathtt{c}/0, \mathtt{d}/0\}$, and productions P such that $\mathrm{A} \to \mathtt{f}(\mathtt{g}(\mathrm{B}), \mathtt{g}(\mathrm{B})) \mid \mathtt{f}(\mathrm{B}, \mathrm{B})$ and $\mathrm{B} \to \mathtt{c} \mid \mathtt{d}$, then $G = (\mathrm{A}, N, \Sigma, P)$ is a TRATG with $\mathrm{A} \prec \mathrm{B}$.

Total rigidity is a restriction on the derivations and is named after a corresponding notion for regular tree automata [11]. For TRATGs it means that a derivation may use at most one production per nonterminal. Due to the acyclicity of the productions, the generated language $L(G)$ is always finite.

Definition 2. *Let $G = (\mathrm{A}, N, \Sigma, P)$ be a TRATG. A derivation $\delta \subseteq P$ is a subset of the productions such that for every nonterminal $\mathrm{B} \in N$ there is at most one production $\mathrm{B} \to t \in \delta$ for some t.*

A derivation δ induces a substitution $\sigma_\delta = [\mathrm{B}_1 \backslash t_1] \cdots [\mathrm{B}_n \backslash t_n]$ where $\delta = \{\mathrm{B}_1 \to t_1, \ldots, \mathrm{B}_n \to t_n\}$ and $\mathrm{B}_1 \prec \cdots \prec \mathrm{B}_n$. The term $L(\delta) = \mathrm{A}\sigma_\delta$ is the term derived by δ. The generated language $L(G)$ consists of all terms in $\mathcal{T}(\Sigma)$ that are derivable in G.

In general there is of course more than one choice for the linearized order $\prec$, however Definition 2 is invariant in the concrete choice of the order. We get an equivalent notion of derivation if instead of the substitution σ_δ we consider a sequence $\mathrm{A} \Rightarrow^* t$ where $s \Rightarrow r$ if $r = s[\mathrm{B} \backslash q]$ for some $\mathrm{B} \to q \in \delta$. The substitution σ_δ will be used for the proof of Lemma 1.

Example 2 (continuing Example 1). The derivation $\delta = \{\mathrm{A} \to \mathtt{f}(\mathrm{B}, \mathrm{B}), \mathrm{B} \to \mathtt{d}\}$ derives the term $\mathtt{f}(\mathtt{d}, \mathtt{d})$. In this way, we can compute the generated language $L(G) = \{\mathtt{f}(\mathtt{g}(\mathtt{c}), \mathtt{g}(\mathtt{c})), \mathtt{f}(\mathtt{g}(\mathtt{d}), \mathtt{g}(\mathtt{d})), \mathtt{f}(\mathtt{c}, \mathtt{c}), \mathtt{f}(\mathtt{d}, \mathtt{d})\}$. Note that there are no "mixed" terms such as $\mathtt{f}(\mathtt{c}, \mathtt{d})$.

3 Membership

We first determine the complexity of the membership problem for TRATGs. The term in this problem is represented as a DAG instead of a tree since this is required for the reductions in the following sections. The complexity of the problem is however unchanged if we represent the term as a tree instead.

Problem 1 (TRATG-Membership). Given a TRATG G and a term t represented as a DAG, is $t \in L(G)$?

Clearly TRATG-Membership is in NP, since we can guess a derivation (which is polynomially bounded in the size of G) and check whether $t \in L(G)$. By size, we always refers to the size of a representation in bits. Given sets A and B, we write $A \leq_P B$ if there exists a polynomial-time many-to-one reduction from A to B. For the NP-hardness of TRATG-Membership, we will show that 3SAT $\leq_P$ TRATG-Membership.

Problem 2 (3SAT). Given a propositional formula F in 3CNF, is F satisfiable?

For simplicity, we assume that the formula F is represented in the following syntax: the propositional variables occurring in F are exactly $\mathtt{x}_1, \ldots, \mathtt{x}_m$, and there are exactly n clauses, each containing at most 3 literals. Literals are either $\mathtt{x}_j$ or $\mathtt{neg}(\mathtt{x}_j)$ for some $1 \leq j \leq m$. Clauses are written $\mathtt{or}(l_1, l_2, l_3)$ where l_i is $\mathtt{true}$, $\mathtt{false}$, or a literal for $i \in \{1, 2, 3\}$. The formula F is then $\mathtt{and}(c_1, \ldots, c_n)$ where c_i is a clause for each $1 \leq i \leq n$.

We can now define a TRATG $\mathrm{Sat}_{n,m}$ that will encode 3SAT for n clauses and m variables. In fact, it will generate exactly the satisfiable CNFs. Rigidity provides the main ingredient for this construction: since we can use at most one production per nonterminal, we can "synchronize" across different clauses and make sure that we assign consistent values to the propositional variables. Which value we assign to the $\mathtt{x}_j$ variable is determined by the choice of the production for the Value_j nonterminal.

Definition 3. *Let $n, m > 0$. We define a TRATG $\mathrm{Sat}_{n,m} = (\mathrm{A}, N, \Sigma, P)$. The signature is $\Sigma = \{\mathtt{and}/n, \mathtt{or}/3, \mathtt{neg}/1, \mathtt{false}/0, \mathtt{true}/0, \mathtt{x}_1/0, \ldots, \mathtt{x}_m/0\}$. The productions (and implicitly the nonterminals) are as follows, where $1 \leq i \leq n$, $1 \leq j \leq m$, and $k \in \{1, 2\}$:*

$$
\begin{aligned}
\mathrm{A} &\to \mathtt{and}(\mathrm{Clause}_1, \ldots, \mathrm{Clause}_n) \\
\mathrm{Clause}_i &\to \mathtt{or}(\mathrm{True}_i, \mathrm{Any}_{i,1}, \mathrm{Any}_{i,2}) \\
\mathrm{Clause}_i &\to \mathtt{or}(\mathrm{Any}_{i,1}, \mathrm{True}_i, \mathrm{Any}_{i,2}) \\
\mathrm{Clause}_i &\to \mathtt{or}(\mathrm{Any}_{i,1}, \mathrm{Any}_{i,2}, \mathrm{True}_i) \\
\mathrm{Any}_{i,k} &\to \mathtt{x}_1 \mid \mathtt{neg}(\mathtt{x}_1) \mid \cdots \mid \mathtt{x}_m \mid \mathtt{neg}(\mathtt{x}_m) \mid \mathtt{false} \mid \mathtt{true} \\
\mathrm{True}_i &\to \mathrm{Value}_1 \mid \cdots \mid \mathrm{Value}_m \mid \mathtt{true} \\
\mathrm{Value}_j &\to \mathtt{x}_j \mid \mathtt{neg}(\mathtt{x}_j)
\end{aligned}
$$

It is clear that $\mathrm{Sat}_{n,m}$ can be computed in polynomial time depending on n, since $m \leq 3n$ and we only need to enumerate $|P| = O(n^2)$ productions.

Lemma 1. *Let $n, m > 0$, and F be a propositional formula in 3CNF with exactly n clauses and at most m variables. Then $F \in L(\mathrm{Sat}_{n,m})$ iff F is satisfiable.*

Proof. Let δ be a derivation of F in $\mathrm{Sat}_{n,m}$. We define an interpretation I_δ such that $I_\delta(\mathtt{x}_j) = 1$ iff $\mathrm{Value}_j \to \mathtt{x}_j \in \delta$, for all $j \leq m$. Now we need to show that indeed $I_\delta \models F$. By case analysis on the chosen productions, we have $I_\delta \models \mathrm{Value}_j\sigma_\delta$ for all $1 \leq j \leq m$, then $I_\delta \models \mathrm{True}_i\sigma_\delta$ and $I_\delta \models \mathrm{Clause}_i\sigma_\delta$ for all $1 \leq i \leq n$, and hence $I_\delta \models F$ since $F = \mathrm{A}\sigma_\delta$.

On the other hand, let $F = \mathtt{and}(c_1, \ldots, c_n)$ and $I \models F$ be a satisfying interpretation. We construct a derivation δ with $\mathrm{A} \to \mathtt{and}(\mathrm{Clause}_1, \ldots, \mathrm{Clause}_n) \in \delta$, $\mathrm{Value}_j \to \mathtt{x}_j \in \delta$ if $I \models \mathtt{x}_j$, and $\mathrm{Value}_j \to \mathtt{neg}(\mathtt{x}_j) \in \delta$ if $I \not\models \mathtt{x}_j$.

For $1 \leq i \leq n$, we define the productions for Clause_i, $\mathrm{Any}_{i,1}$, $\mathrm{Any}_{i,2}$, and True_i depending on which literal is true in the i-th clause. Let $\mathtt{or}(l_1, l_2, l_3)$ be the i-th clause, and let j be such that $l_1 = \mathtt{x}_j$ or $l_1 = \mathtt{neg}(\mathtt{x}_j)$. If $I \models l_1$, we let $\mathrm{Clause}_i) \to \mathtt{or}(\mathrm{True}_i, \mathrm{Any}_{i,1}, \mathrm{Any}_{i,2}) \in \delta$, $\mathrm{True}_i \to \mathrm{Value}_j \in \delta$, $\mathrm{Any}_{i,1} \to l_2 \in \delta$

and $\text{Any}_{i,2} \to l_3 \in \delta$. The cases where the second or third literal are true are handled analogously. We can see that this set δ is then indeed a derivation of F in $\text{Sat}_{n,m}$. □

Theorem 1. TRATG-Membership *is* NP*-complete.*

Proof. By reduction from 3SAT using Lemma 1. □

4 Containment

We now consider the problem of determining whether the generated languages of two TRATGs are contained in one another:

Problem 3 (TRATG-Containment). Given TRATGs G_1 and G_2, is $L(G_1) \subseteq L(G_2)$?

The polynomial-time hierarchy is defined as usual, where $\Sigma_0^P = \Pi_0^P = \mathsf{P}$, $\Sigma_{n+1}^P = \mathsf{NP}(\Pi_n^P)$, and $\Pi_{n+1}^P = \mathsf{coNP}(\Sigma_n^P)$. We will show that TRATG-Containment is complete for $\Pi_2^P = \mathsf{coNP}(\mathsf{NP})$. A canonical Π_n^P-complete problem consists of deciding the truth of quantified Boolean formulas (QBF) in Π_n, that is, closed prenex quantified Boolean formulas whose quantifier prefix has n blocks alternating between blocks of universal and existential quantifiers, starting with universal quantifiers. For TRATG-Containment, we hence need Π_2 formulas:

Problem 4 (Π_2-TQBF). Given a closed QBF $\forall \mathtt{y}_1 \cdots \forall \mathtt{y}_k \exists \mathtt{x}_1 \cdots \exists \mathtt{x}_m\, F$ where F is in 3CNF, is F true?

In the following we will now show that TRATG-Containment is Π_2^P-hard by reducing Π_2-TQBF to it. Such a Π_2-QBF is true if and only if the propositional matrix F is satisfiable for all instances of the universal quantifiers. Theorem 1 shows that we can encode satisfiability as TRATG-Membership. It only remains to encode the instantiation of the universal quantifiers as a TRATG. We will thus define a TRATG that generates exactly the instances of the QBF where we substitute the universal variables by either `true` or `false`:

Definition 4. *Let* $Q = \forall \mathtt{y}_1 \cdots \forall \mathtt{y}_k \exists \mathtt{x}_1 \cdots \exists \mathtt{x}_m\, F$ *be a closed QBF such that* $F = \mathtt{and}(c_1, \ldots, c_n)$ *is in 3CNF. We define a TRATG* $\text{Inst}_Q = (A, N, \Sigma, P)$*. The signature is given by* $\Sigma = \{\mathtt{and}/n, \mathtt{or}/3, \mathtt{neg}/1, \mathtt{true}/0, \mathtt{false}/0, \mathtt{x}_1/0, \ldots, \mathtt{x}_m/0\}$*. The productions and (implicitly) nonterminals are as follows:*

$$\mathrm{A} \to F[\mathtt{y}_1 \backslash \mathrm{Y}_1, \ldots, \mathtt{y}_k \backslash \mathrm{Y}_k]$$
$$\mathrm{Y}_j \to \mathtt{true} \mid \mathtt{false} \quad \textit{for } j \leq k$$

Lemma 2. *Let* $Q = \forall \mathtt{y}_1 \cdots \forall \mathtt{y}_k \exists \mathtt{x}_1 \cdots \exists \mathtt{x}_m\, F$ *be a closed QBF such that* $F = \mathtt{and}(c_1, \ldots, c_n)$ *is in 3CNF. Then* Q *is true iff* $L(\text{Inst}_Q) \subseteq L(\text{Sat}_{n,k+m})$.

Proof. The QBF Q is true if and only if every instance $F[\mathtt{y}_1 \backslash v_1, \ldots, \mathtt{y}_k \backslash v_k]$ is satisfiable where $v_1, \ldots, v_k \in \{\mathtt{true}, \mathtt{false}\}$. The TRATG Inst_Q generates exactly these instances. Hence Q is true iff $L(\mathrm{Inst}_Q) \subseteq L(\mathrm{Sat}_{n,m})$ by Lemma 1. □

Theorem 2. TRATG-Containment *is* Π_2^P*-complete.*

Proof. TRATG-Containment is in Π_2^P because we can guess a derivation δ of a term t in G_1, and then check whether $t \in L(G_2)$. Each derivation can be produced in polynomial time, and checking whether $t \in L(G_2)$ is in NP per Theorem 1. For hardness, we reduce the Π_2^P-complete Π_2-TQBF to TRATG-Containment via Lemma 2. □

5 Disjointness

The complexity of the disjointness problem follows straightforwardly from (the complement of) TRATG-Membership.

Problem 5 (TRATG-Disjointness). Given TRATGs G_1 and G_2, is $L(G_1) \cap L(G_2) = \emptyset$?

Theorem 3. TRATG-Disjointness *is* coNP*-complete.*

Proof. We first show that TRATG-Disjointness is in coNP: for all derivations δ_1 of a term t_1 in G_1 and δ_2 of a term t_2 in G_2, we check that $t_1 \neq t_2$. As usual we can generate a derivation in polynomial time, and checking equality of terms is polynomial as well. Hardness follows via a reduction from the complement of TRATG-Membership: $t \notin L(G)$ if and only if $L(G) \cap L(G_t) = \emptyset$, where G_t is a TRATG such that $L(G_t) = \{t\}$. □

6 Equivalence

Problem 6 (TRATG-Equivalence). Given TRATGs G_1 and G_2, is $L(G_1) = L(G_2)$?

Definition 5. *Let* $G_1 = (\mathrm{A}_1, N_1, \Sigma_1, P_1), G_2 = (\mathrm{A}_2, N_2, \Sigma_2, P_2)$ *be TRATGs such that* $N_1 \cap N_2 = \emptyset$*. Then* $G_1 \cup G_2 = (\mathrm{A}_1, N_1 \cup N_2, \Sigma_1 \cup \Sigma_2, P')$*, where* $P' = P_1 \cup P_2 \cup \{\mathrm{A}_1 \to \mathrm{A}_2\}$*.*

It is easy to see that $L(G_1 \cup G_2) = L(G_1) \cup L(G_2)$. We will also use the notation $G_1 \cup G_2$ for TRATGs that share nonterminals: we then implicitly rename the nonterminals in one TRATG so that they are disjoint.

Theorem 4. TRATG-Equivalence *is* Π_2^P*-complete.*

Proof. We first show that it is in Π_2^P via a reduction to TRATG-Containment: $L(G_1) = L(G_2)$ if and only if $L(G_1) \subseteq L(G_2)$ as well as $L(G_2) \subseteq L(G_1)$—and Π_2^P is closed under intersection. Hardness follows by a reduction from TRATG-Containment: $L(G_1) \subseteq L(G_2)$ iff $L(G_1) \cup L(G_2) = L(G_2)$. This is equivalent to $L(G_1 \cup G_2) = L(G_2)$, an instance of TRATG-Equivalence. □

7 Minimal Cover

7.1 Minimal Cover for Terms

Let $G = (\mathrm{A}, N, \Sigma, P)$ be a TRATG. We say that G covers a set of terms L if $L(G) \supseteq L$, and write $|G| = |P|$ for the number of productions in G. Finding a small TRATG that covers a given finite set forms the combinatorial core of a practical application in automated reasoning: the generation of quantified lemmas [6]. The finite set corresponds to a tautological set of instances, a Herbrand disjunction. Since supersets of tautological disjunctions are tautological as well, we require that the TRATG covers the given finite set of terms, as opposed to $L(G) = L$. The covering TRATGs should have a small number of productions, since they correspond to small proofs that are expected to contain interesting lemmas. We consider a decision version of the problem:

*Problem 7 (*TRATG-Cover*).* Given a finite set of terms L and $k \geq 0$, is there a TRATG G such that $|G| \leq k$ and $L(G) \supseteq L$?

TRATG-Cover is in NP, since potential TRATGs G have at most $|L|$ productions, hence at most $|L|$ nonterminals, and the size of each production is bounded by the size of L.

Conjecture 1. TRATG-Cover is NP-complete.

So far a proof of Conjecture 1 remains elusive. However we will show that the following version of TRATG-Cover with a fixed parameter is NP-complete:

Problem 8 (TRATG-n-Cover). Given a finite set of terms L and $k \geq 0$, is there a TRATG $G = (\mathrm{A}, N, \Sigma, P)$ such that $|N| \leq n$, $|P| \leq k$, and $L(G) \supseteq L$?

Theorem 5. TRATG-n-Cover *is* NP*-complete for* $n \geq 2$.

The proof of NP-hardness in the following Sect. 7.2 only requires regular grammars for words. A similar approach to prove lower bounds for TRATGs using regular grammars for words was already successfully used in descriptional complexity [5].

7.2 Minimal Cover for Words

It is well known that we can represent words as trees by using unary function symbols instead of letters. We may hence represent the word `hello` as the tree $\mathtt{h}(\mathtt{e}(\mathtt{l}(\mathtt{l}(\mathtt{o}(\epsilon)))))$, for example. Under this correspondence, TRATGs correspond exactly to acyclic regular grammars:

Definition 6. *A regular grammar* $G = (\mathrm{A}, N, \Sigma, P)$ *is a tuple consisting of a start symbol* $\mathrm{A} \in N$*, a finite set of nonterminals* N*, a finite set of letters* Σ *such that* $N \cap \Sigma = \emptyset$*, and a finite set of productions* $P \subseteq N \times \Sigma^*(N \cup \{\epsilon\})$*. We call* G acyclic *if there exists a strict linear order* $\prec$ *on the nonterminals such that* $\mathrm{B} \prec \mathrm{C}$ *whenever* $\mathrm{B} \to w\mathrm{C} \in P$ *for some* $w \in \Sigma^*$.

The one-step derivation relation $\Rightarrow_G$ is defined by $w\mathrm{B} \Rightarrow_G wv$ for $\mathrm{B} \to v \in P$ and $w \in \Sigma^*$. The language $L(G) = \{w \in \Sigma^* \mid \mathrm{A} \Rightarrow_G^* w\}$ then consists of the derivable words. We write $|G| = |P|$ for the number of productions. In the case of acyclic regular grammars, the derivations and the generated language correspond exactly to those for TRATGs. Note that there is no need to require rigidity for derivations here: derivations in acyclic regular grammars can never use more than one production per nonterminal. Rigidity only plays a role for terms, where we can have multiple parallel occurrences of a nonterminal.

Problem 9 (Regular-Cover). Given a finite set of words L and $k \geq 0$, is there an acyclic regular grammar G such that $|G| \leq k$ and $L(G) \supseteq L$?

Lemma 3. Regular-Cover $\leq_P$ TRATG-Cover.

Proof. Treat words as terms with unary function symbols and vice versa. □

Regular-Cover corresponds to TRATG-Cover, and is in NP. Furthermore, if Regular-Cover is NP-hard then so is TRATG-Cover. However the precise complexity of Regular-Cover is open, similar to the case for terms we conjecture it to be NP-complete.

Conjecture 2. Regular-Cover is NP-complete.

Clearly, Conjecture 2 implies Conjecture 1. In the rest of this section, we will show that the restriction of Regular-Cover to a bounded number of nonterminals is NP-hard.

Problem 10 (Regular-n-Cover). Given a finite set of words L and $k \geq 0$, is there an acyclic regular grammar $G = (\mathrm{A}, N, \Sigma, P)$ such that $|N| \leq n$, $|P| \leq k$, and $L(G) \supseteq L$?

We will show the NP-hardness of Regular-n-Cover in several steps: first we reduce 3SAT to Regular-2-Cover using an intermediate problem where we can not only specify words that must be generated by the grammar, but also productions that must be included:

Problem 11 (Regular-2-Cover-Extension). Given a finite set of words L, an acyclic regular grammar $G = (\mathrm{A}, N, \Sigma, P)$ such that $N = \{\mathrm{A}, \mathrm{B}\}$, w contains B whenever $\mathrm{A} \to w \in P$, and $k \geq 0$, is there a superset $P' \supseteq P$ of the productions such that $|P'| \leq |P| + k$, and $L(G') \supseteq L$ where $G' = (\mathrm{A}, N, \Sigma, P')$?

Lemma 4. *3SAT* $\leq_P$ Regular-2-Cover-Extension.

Proof. Consider a propositional formula in CNF with m clauses $C_1, \ldots, C_m$ and n variables (called $x_1, \ldots, x_n$). The number of variables n will be used as the k parameter in Regular-2-Cover-Extension. Assume without loss of generality that $x_j \vee \neg x_j$ is a clause in this formula for all $j \leq n$. We will encode the literals as unary natural numbers. First we define the natural numbers $a_j = 2j$

and $b_j = 2j + 1$ that correspond to x_j and $\neg x_j$, resp. The number $c = 2n + 1$ is their upper bound.

Let $L = \{\mathtt{s}^c\mathtt{o}_{l,i} \mid i \leq m, l \leq 2n\}$ and $\Sigma = \{\mathtt{o}_{l,i} \mid l \leq 2n, i \leq n\} \cup \{\mathtt{s}\}$. Furthermore, define the acyclic regular grammar $G = (\mathrm{A}, N, \Sigma, P)$ where $N = \{\mathrm{A}, \mathrm{B}\}$, and the productions P are the following:

$$\begin{aligned} \mathrm{B} &\to \mathtt{s}^{c-a_j}\mathtt{o}_{l,i} && \text{for } x_j \in C_i \text{ and } l \leq 2n \\ \mathrm{B} &\to \mathtt{s}^{c-b_j}\mathtt{o}_{l,i} && \text{for } \neg x_j \in C_i \text{ and } l \leq 2n \end{aligned}$$

It remains to show that F is satisfiable iff there exists a set of productions $P' \supseteq P$ such that $|P'| \leq |P| + n$ and $L(G') \supseteq L$ where $G' = (\mathrm{A}, N, \Sigma, P')$.

Left-to-right direction. Let $I \models F$ be a satisfying interpretation, then we construct an acyclic regular grammar G' by adding the following productions to G. We clearly have $|G'| = |G| + n$, and also $L(G') \supseteq L$.

$$\mathrm{A} \to \mathtt{s}^{a_j}\mathrm{B} \quad \text{if } I \models x_j \qquad\qquad \mathrm{A} \to \mathtt{s}^{b_j}\mathrm{B} \quad \text{if } I \not\models x_j$$

Right-to-left direction. Let $G' = (\mathrm{A}, N, \Sigma, P')$ such that $P' \supseteq P$, $L(G') \supseteq L$, and $|P'| \leq |P| + n$. By symmetry, all productions from the nonterminal A are of the form $\mathrm{A} \to \mathtt{s}^r\mathrm{B}$ for some $r \geq 0$. Otherwise there would be an $i \leq m$ such that all constants $\mathtt{o}_{l,i}$ appear in productions from A, and we would have at least $2n$ new productions.

Let δ be a derivation of $\mathtt{s}^c\mathtt{o}_{l,i} \in L$ in G'. Then δ uses the production $\mathrm{A} \to \mathtt{s}^r\mathrm{B}$ for some r, and we have that $r = a_j$ and $x_j \in C_i$ or $r = b_j$ and $\neg x_j \in C_i$. Since $x_j \vee \neg x_j \in F$ for all $j \leq n$, we have $\mathrm{A} \to \mathtt{s}^{a_j}\mathrm{B} \in P'$ or $\mathrm{A} \to \mathtt{s}^{b_j}\mathrm{B} \in P'$ for all $j \leq n$. Because there are at most n such productions, we cannot have both $\mathtt{s}^{a_j}\mathrm{B} \in P'$ and $\mathtt{s}^{b_j}\mathrm{B} \in P'$.

Now define an interpretation I such that $I \models x_j$ iff $\mathrm{A} \to \mathtt{s}^{a_j}\mathrm{B} \in P'$. Note that $I \not\models x_j$ iff $\mathrm{A} \to \mathtt{s}^{b_j}\mathrm{B} \in P'$, and hence I is a model for F by the previous paragraph. □

Lemma 5. Regular-2-Cover-Extension $\leq_P$ Regular-2-Cover.

Proof. Let L, $G = (\mathrm{A}, N, \Sigma, P)$, $N = \{\mathrm{A}, \mathrm{B}\}$, and k be as in the definition of Regular-2-Cover-Extension. We set $m = |L| + |G|$. Without loss of generality, assume that $L \neq \emptyset$. Take $4m$ fresh letters $\mathtt{a}_1, \ldots, \mathtt{a}_{3m}, \mathtt{b}_1, \ldots, \mathtt{b}_m$. We extend the language L to L':

$$\begin{aligned} L' = L &\cup \{\mathtt{a}_i w \mid \mathrm{B} \to w \in P,\ i \leq 3m\} \\ &\cup \{w\mathtt{b}_j \mid \mathrm{A} \to w\mathrm{B} \in P,\ j \leq m\} \\ &\cup \{\mathtt{a}_i\mathtt{b}_j \mid i \leq 3m, j \leq m\} \end{aligned}$$

We need to show that there exists an acyclic regular grammar $G' = (\mathrm{A}, N, \Sigma', P')$ with $\Sigma' = \Sigma \cup \{\mathtt{a}_1, \ldots, \mathtt{a}_{3m}, \mathtt{b}_1, \ldots, \mathtt{b}_m\}$ such that $L(G') \supseteq L'$ and $|G'| \leq |G| + k + 4m$ if and only if there exists an acyclic regular grammar $G'' = (\mathrm{A}, N, \Sigma, P'')$ such that $P \subseteq P''$, $|G''| \leq |G| + k$, and $L(G') \supseteq L$.

Left-to-right direction. Let us first assume that such a grammar G' exists. We can assume that $|G'| \leq |L| + |G| + 4m = 5m$ since there exists a covering grammar of that size with the productions $\{\mathrm{A} \to w \mid w \in L\} \cup P \cup \{\mathrm{A} \to \mathtt{a}_i \mid i \leq 3m\} \cup \{\mathrm{B} \to \mathtt{b}_j \mid j \leq m\}$.

Without loss of generality, assume that the letters $\mathtt{a}_i$ occur only as the first letter of productions of the nonterminal A. We can drop any production that contains $\mathtt{a}_i$ in the middle since $\mathtt{a}_i$ can only occur at the beginning of words in L'. For the same reason we can then replace each production $\mathrm{B} \to \mathtt{a}_i w$ by $\mathrm{A} \to \mathtt{a}_i w$.

Furthermore, we can assume that G' is symmetric in the new symbols, that is, $\mathrm{A} \to \mathtt{a}_i w \in P'$ if and only if $\mathrm{A} \to \mathtt{a}_j w \in P'$ for all $i, j \leq 3m$ and words w. Otherwise pick an $i \leq 3m$ such that $W = \{w \mid \mathrm{A} \to \mathtt{a}_i w \in P'\}$ is of minimal size. We then remove all productions containing an $\mathtt{a}_j$ for some $j \leq 3m$, and replace them by the productions $\mathrm{A} \to \mathtt{a}_j w$ for $w \in W$. The resulting grammar still covers L' since L' is symmetric under permutation of the letters $\mathtt{a}_i$.

Now for every $i \leq 3m$ there is at most one production of the form $\mathrm{A} \to \mathtt{a}_i w$ for some w—if there were two, then by $\mathtt{a}_i$-symmetry we would have $2 \cdot 3m = 6m$ productions, exceeding the previously obtained upper bound of $5m$ productions. This also implies that $\mathrm{B} \to \mathtt{b}_j \in P'$ for all $j \leq m$. By a symmetry argument for $\mathtt{b}_j$ there are no other productions that contain $\mathtt{b}_j$. We also have that $\mathrm{B} \to w \in P'$ whenever $\mathrm{B} \to w \in P$.

We can now construct the acyclic regular grammar G'' by removing all productions from G' that contain one of the new symbols $\mathtt{a}_i$ or $\mathtt{b}_j$. We have $L(G'') \supseteq L$ since $L(G') \supseteq L$ and because productions that contain the new symbols cannot be used in derivations of words in L. Furthermore, $|G''| \leq |G'| - 4m = |G| + k$. It remains to show that $\mathrm{A} \to w\mathrm{B} \in P'$ whenever $\mathrm{A} \to w\mathrm{B} \in P$ for some $w \in \Sigma^*$. Recall that L' contains $w\mathtt{b}_j$ for all $j \leq m$. However the only occurrence of $\mathtt{b}_j$ in P' is in the production $\mathrm{B} \to \mathtt{b}_j \in P'$. Hence $\mathrm{A} \to w\mathrm{B} \in P'$.

Right-to-left direction. In the other case, we assume that such a grammar G'' exists, and need to construct G'. We obtain this grammar G' by adding the productions $\mathrm{A} \to \mathtt{a}_i\mathrm{B}$ and $\mathrm{B} \to \mathtt{b}_j$ for all $i \leq 3m$ and $j \leq m$. Then G' has $4m$ more productions than G'' and covers L'. □

Lemma 6. Regular-n-Cover $\leq_P$ Regular-$(n+1)$-Cover.

Proof. Let L be a finite set of words, $k \geq 0$, and define $m = 2k$. Assume that $|L| \geq 1$, $k \geq 2$, and $n \geq 1$—otherwise we can directly compute the answer. Take $m+1$ fresh letters $\mathtt{a}_1, \ldots, \mathtt{a}_m, \mathtt{b}$. We define $L' = \{\mathtt{a}_1, \ldots, \mathtt{a}_m\} \cdot (L \cup \{\mathtt{b}\})$. It remains to show that there exists an acyclic regular grammar G with n nonterminals such that $|G| \leq k$ and $L(G) \supseteq L$ if and only if there exists an acyclic regular grammar G' with $n+1$ nonterminals such that $|G'| \leq k + m + 1$ and $L(G') \supseteq L'$.

Left-to-right direction. Let $G = (\mathrm{B}, N, \Sigma, P)$ be an acyclic regular grammar such that $|N| = n$, $|G| \leq k$, and $L(G) \supseteq L$. We set $G' = (\mathrm{A}, N \cup \{\mathrm{A}\}, \Sigma \cup \Sigma', P \cup P')$ where A is a fresh nonterminal, $\Sigma' = \{\mathtt{a}_1, \ldots, \mathtt{a}_m, \mathtt{b}\}$, and $P' = \{\mathrm{A} \to \mathtt{a}_i\mathrm{B} \mid i \leq m\} \cup \{\mathrm{B} \to \mathtt{b}\}$. Clearly $|N \cup \{\mathrm{A}\}| = n + 1$, $|G'| = |G| + m + 1 \leq k + m + 1$, and $L(G') \supseteq L'$.

Right-to-left direction. Let $G' = (\mathrm{A}, N, \Sigma, P')$ be an acyclic regular grammar such that $|N| = n+1$, $|G| \leq k+m+1$, and $L(G') \supseteq L'$. Via the same argument as used in the proof of Lemma 5 and noting that $2m > k + m$, we can assume that there exists a nonterminal $\mathrm{B} \neq \mathrm{A}$ such that all productions from A are of the form $\mathrm{A} \to \mathtt{a}_i\mathrm{B}$ for some $i \leq m$. Let $P \subseteq P'$ be the set of productions whose left side is not A and whose right side is not $\mathtt{b}$. The acyclic regular grammar $G = (\mathrm{B}, N \setminus \{\mathrm{A}\}, \Sigma, P)$ then has the desired properties. □

Theorem 6. Regular-n-Cover *is* NP*-complete for* $n \geq 2$.

Proof. By reduction from 3SAT using Lemma 4 to 6. □

Theorem 5, the NP-completeness of TRATG-n-Cover, now directly follows from Theorem 6.

8 Minimization

Problem 12 (TRATG-Minimization). Given a TRATG $G = (\mathrm{A}, N, \Sigma, P)$, a set of terms L such that $L(G) \supseteq L$, and $k \geq 0$, is there a subset $P' \subseteq P$ of the productions such that $|P'| \leq k$ and $L(G') \supseteq L$ where $G' = (\mathrm{A}, N, \Sigma, P')$?

This optimization problem plays a central role in a practical algorithm [3] to find minimal TRATGs which cover a given set of terms, as in TRATG-n-Cover. We will now show that TRATG-Minimization is NP-complete via reduction from Set Cover.

Problem 13 (Set Cover). Given a finite set X, a finite collection $C \subseteq \mathcal{P}(X)$ of subsets such that $\bigcup C = X$, and $k \geq 0$, is there a sub-collection $C' \subseteq C$ such that $\bigcup C = X$ and $|C'| \leq k$?

Theorem 7. TRATG-Minimization *is* NP*-complete.*

Proof. The problem is in NP because we can check $|P'| \leq k$ in polynomial time, and reduce $L(G') \supseteq L$ to TRATG-Membership. Hardness follows by reduction from Set Cover: we pick a fresh nonterminal A, set $N = \{\mathrm{A}\} \cup C$ (treating the subsets as nonterminals), $L = X$, and $P = \{\mathrm{A} \to U \mid U \in C\} \cup \{U \to x \mid x \in U \in C\}$. A subset $P' \subseteq P$ of the productions of with $k + |X|$ elements then directly corresponds to a sub-collection C' of such that $|C'| \leq k$: for every $x \in X$, there is at least one production $U \to x \in P'$ for some $U \in C$. These are $|X|$ productions, there are hence at most k productions of the form $\mathrm{A} \to U$ for some U, which yield the sub-collection $C' = \{U \mid \mathrm{A} \to U \in P'\}$. □

9 Conclusion

We have determined the computational complexity of the membership, containment, disjointness, equivalence, and minimization problems on TRATGs. For minimal cover we could only determine the complexity of a variant where the

number of nonterminals is bounded. The complexity of minimal cover without a fixed parameter remains unknown, even for regular grammars on words.

The reductions in this paper generally use an unbounded number of symbols. In particular, the reductions for minimal cover require this unboundedness in an essential way to constrain the covering grammar. It remains as future work to study reductions which do not increase the number of symbols.

Each of these decision problems directly corresponds to a decision problem in proof theory concerning the result of non-erasing cut-elimination. Showing that minimal cover is polynomial-time decidable could lead to fast compression algorithms using TRATGs, which can immediately be used for applications in automated deduction [6].

Acknowledgments. The authors would like to thank the reviewers for many helpful suggestions which led to a considerable improvement of this paper.

References

1. Busatto, G., Lohrey, M., Maneth, S.: Efficient memory representation of XML document trees. Inf. Syst. **33**(4–5), 456–474 (2008)
2. Casel, K., Fernau, H., Gaspers, S., Gras, B., Schmid, M.L.: On the complexity of grammar-based compression over fixed alphabets. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) Leibniz International Proceedings in Informatics (LIPIcs) 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016), vol. 55, pp. 122:1–122:14 (2016)
3. Eberhard, S., Ebner, G., Hetzl, S.: Algorithmic compression of finite tree languages by rigid acyclic grammars. ACM Trans. Comput. Log. **18**(4), 26:1–26:20 (2017)
4. Eberhard, S., Hetzl, S.: Inductive theorem proving based on tree grammars. Ann. Pure Appl. Log. **166**(6), 665–700 (2015)
5. Eberhard, S., Hetzl, S.: On the compressibility of finite languages and formal proofs. Inf. Comput. **259**, 191–213 (2018)
6. Ebner, G., Hetzl, S., Leitsch, A., Reis, G., Weller, D.: On the generation of quantified lemmas. J. Autom. Reason. pp. 1–32 (2018)
7. Ebner, G., Hetzl, S., Reis, G., Riener, M., Wolfsteiner, S., Zivota, S.: System description: GAPT 2.0. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS (LNAI), vol. 9706, pp. 293–301. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40229-1_20
8. Hetzl, S.: Applying tree languages in proof theory. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 301–312. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28332-1_26
9. Hetzl, S., Leitsch, A., Reis, G., Weller, D.: Algorithmic introduction of quantified cuts. Theor. Comput. Sci. **549**, 1–16 (2014)
10. Hetzl, S., Straßburger, L.: Herbrand-confluence. Log. Methods Comput. Sci. **9**(4) (2013)
11. Jacquemard, F., Klay, F., Vacher, C.: Rigid tree automata and applications. Inf. Comput. **209**(3), 486–512 (2011)
12. Kieffer, J.C., Yang, E.H.: Grammar-based codes: a new class of universal lossless source codes. IEEE Trans. Inf. Theory **46**(3), 737–754 (2000)

13. Larsson, N.J., Moffat, A.: Offline dictionary-based compression. In: Data Compression Conference (DCC 1999), pp. 296–305. IEEE Computer Society (1999)
14. Lohrey, M., Maneth, S., Mennicke, R.: XML tree structure compression using repair. Inf. Syst. **38**(8), 1150–1167 (2013)
15. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: a linear-time algorithm. J. Artif. Intell. Res. **7**, 67–82 (1997)
16. Storer, J.A., Szymanski, T.G.: The macro model for data compression (extended abstract). In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC 1978), pp. 30–39. ACM, New York (1978)
17. Storer, J.A., Szymanski, T.G.: Data compression via textural substitution. J. ACM **29**(4), 928–951 (1982)
18. Yamagata, K., Uchida, T., Shoudai, T., Nakamura, Y.: An effective grammar-based compression algorithm for tree structured data. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS (LNAI), vol. 2835, pp. 383–400. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39917-9_25