Conclusive Tree-Controlled Grammars

Dominika Klobučníková Zbyněk Křivka Alexander Meduna

Centre of Excellence IT4Innovations, Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic

iklobucnikova@fit.vut.cz krivka@fit.vut.cz meduna@fit.vut.cz

This paper presents a new approach to regulation of grammars. It divides the derivation trees generated by grammars into two sections—generative and conclusive (the conclusion). The former encompasses generation of symbols up till the moment when the deepest terminal is generated, whereas the latter represents the final steps needed to successfully generate a sentence. A control mechanism based on regulating only the conclusion is presented and subsequently applied to tree-controlled grammars, creating conclusive tree-controlled grammars. As the main result, it is shown that the ratio between depths of generative and conclusive sections does not influence the generative power. In addition, it is demonstrated that any recursively enumerable language is generated by these grammars possessing no more than seven nonterminals while the regulating language is subregular.

1 Introduction

Derivation trees serve as a graph representation of derivations leading to specific sentential forms. Naturally, since this notion corresponds to the rewriting process of a grammar in a deterministic manner, it can be utilized to create a mechanism used to regulate formal grammars.

Such grammar types can be represented by tree-controlled grammars – a tree-controlled grammar is defined as a pair (G,R), where G is an ordinary context-free grammar and R is a regular language (see [1]). The language generated by (G,R) is defined by this equivalence: this language contains a word x if and only if $x \in L(G)$ and there is a derivation tree for x in G such that R contains every word obtained by concatenating the symbols labeling the nodes in the same level for all levels of the tree. Although based upon context-free grammars, tree-controlled grammars are computationally complete—that is, they characterize the family of recursively enumerable language. Considering this advantage, it comes as no surprise that formal language theory intensively investigates these grammars (see [2, 7]).

However, a question presents itself: is it truly necessary to regulate all levels of derivation tree or would it be sufficient to verify only a few key moments of the derivation? A similar notion has been utilized by scattered context grammars with a single context-sensitive rule (see [5]) and by the Geffert normal form (see [3]), both of which are known to be equal to type-0 grammars, which are computationally complete.

Consequently, this paper presents the separation of derivation trees into two distinct parts—the *generative part* and the *conclusive part* (*conclusion* for short, see Figure 1). Intuitively, the former represents the derivation tree starting from the start nonterminal and extending to the level where the lowest (rightmost and furthest from the root) terminal of the potential sentence can be found, while the latter is in charge of verifying—or concluding—the derivation and erasing the remaining auxiliary nonterminals.

The paper proposes a different approach of regulating the derivation tree compared to the classic tree-controlled grammars: instead of regulating the entire tree, which may not prove effective, it focuses specifically on regulation of the subtree forest found in the conclusion of the derivation tree.

This paper proves that these grammars are computationally complete even if their control languages belong to the intersection of the ordered regular language family and the union-free regular language

Submitted to:
NCMA 2022

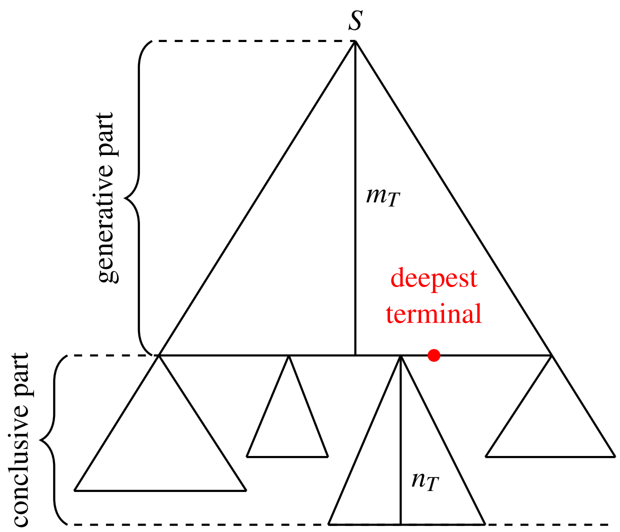


Figure 1: Separation of derivation tree, T, into the generative and conclusive part based on the deepest (lowest) terminal; all symbols located to its right are nonterminal. The generative part starts with the following level; no symbols belong to both generative and conclusive parts.

family, both of which are mutually incomparable and properly contained in the regular language family (see [2]). In fact, we establish a stronger result: any recursively enumerable language is generated by a ten-nonterminal tree-controlled grammar (G,R), where R belongs to the intersection mentioned above, and in addition, R uses no more than seven iterations $(*)$ and no more than ten concatenations.

The paper is organized as follows: first, an introduction to graph theory, derivation trees and their anatomy is given. Then, the conclusive tree-controlled grammars and languages generated by them are introduced, followed by a discussion on their generative power and a comparison with type-0 grammars. At the end of the paper, an overview of the results is given and several open problems are listed.

2 Definitions

This paper assumes that the reader is familiar with language theory (see [4]).

For a set, Q, $\mathrm{card}(Q)$ denotes the cardinality of Q. For an alphabet, V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. For $w \in V^*$, $|w|$ denotes the length of w. Furthermore, $\mathrm{suffix}(w)$ denotes the set of all suffixes of w, and $\mathrm{prefix}(w)$ denotes the set of all prefixes of w. For $w \in V^*$ and $T \subseteq V$, $\mathrm{occur}(w,T)$ denotes the number of occurrences of symbols from T in w. For instance, $\mathrm{occur}(abdabc,\{a,d\}) = 3$. If $T = \{a\}$, where $a \in V$, we simplify $\mathrm{occur}(w,\{a\})$ to $\mathrm{occur}(w,a)$. For a sequence, $x = (a_1,a_2,\ldots,a_n)$, where $a_i \in V$ for $1 \leq i \leq n$, $|x| = n$ denotes the length of x. By $\mathbb{N}$, we denote the set of all positive integers. Let $I \subset \mathbb{N}$ be a finite nonempty set. Then, $\max(I)$ denotes the maximum of I.

Definition 2.1. Union-free regular language *(*UFRL *for short) over an alphabet* Σ *is defined recursively as follows:*

(i) $\{\varepsilon\}$, $\emptyset$ *are* UFRL *over* Σ*;*

(ii) for every $a \in \Sigma$, $\{a\}$ is an UFRL *over Σ;*

(iii) let X, Y be UFRL*, then,*

(a) XY is an UFRL *(concatenation),*

(b) X^* is an UFRL *(iteration).*

The family of *UFRL* is denoted by **UFREG**. TODO: example of UFREG?

Definition 2.2. Extended union-free regular language *(*EUFRL *for short) over an alphabet Σ is defined recursively as follows:*

(i) $\{\varepsilon\}$, $\emptyset$ are EUFRL *over Σ;*

(ii) for every $X \subseteq \Sigma$, X is an EUFRL *over Σ;*

(iii) let X, Y be EUFRL*, then,*

(a) XY is an EUFRL *(concatenation),*

(b) X^* is an EUFRL *(iteration).*

The family of *EUFRL* is denoted by **EUFREG**. TODO: example of EUFREG

A *type-0 grammar* is a quadruple $G = (N, \Sigma, P, S)$, where N and Σ are the finite alphabets of nonterminals and terminals, respectively, such that $N \cap \Sigma = \emptyset$, $S \in N$ is the start nonterminal, and P is the set of productions in the form of $x \rightarrow y$, where $x, y \in (N \cup \Sigma)^*$. Let $V = N \cup \Sigma$. For some $p = x \rightarrow y \in P$ (a production labeled by p), $\text{lhs}(p)$ denotes x as *the left-hand side* of p and $\text{rhs}(p)$ denotes y as *the right-hand side* of p. The direct derivation relation over V^*, symbolically denoted by $\Rightarrow$, is defined as follows: $uxv \Rightarrow uyv$ $[p]$ in G, or simply $uxv \Rightarrow uyv$, if and only if $u, v \in V^*$ and $p : x \rightarrow y \in P$. Let $\Rightarrow^n$ and $\Rightarrow^*$ denote the nth power of $\Rightarrow$, for some $n \geq 0$, and the reflexive-transitive closure of $\Rightarrow$, respectively. The language generated by G is denoted by $L(G)$ and defined as $L(G) = \{x : S \Rightarrow^* x, x \in \Sigma^*\}$. Two grammars are *equivalent* if both generate the same language. The family of languages generated by type-0 grammars (also known as the family of recursively enumerable languages) is denoted by **RE**.

Both, a *right-linear grammar* and a *context-free grammar* is a type-0 grammar, $G = (N, \Sigma, P, S)$, where N, Σ, and S have the same meaning as in the previous definition, and P is the set of productions in the form of $A \rightarrow w$, where $A \in N$, $w \in \Sigma^*(N \cup \{\varepsilon\})$ and $A \in N$, $w \in (N \cup \Sigma)^*$, respectively.

Definition 2.3. *A* type-0 grammar, $G = (\{S, S', A, B\}, \Sigma, P \cup \{ABBBA \rightarrow \varepsilon\}, S)$ *is said to be in the third Geffert normal form if every production, $p \in P$, has one of the following forms:*

(i) $S \rightarrow uSa$,

(ii) $S \rightarrow S'$,

(iii) $S' \rightarrow uS'v$,

(iv) $S' \rightarrow uv$,

where $u \in \{AB, ABB\}^*$, $v \in \{BA, BBA\}^*$, and $a \in \Sigma$.

Recall that type-0 grammars are computationally complete [6].

Lemma 2.4 (Geffert normal form [3])**.** *For every type-0 grammar, G, there exists an equivalent grammar in the third Geffert normal form.*

Let $H = (A, \rho)$ be an oriented graph, where A is the set of *nodes* and ρ is a relation on A consisting of *edges*. A sequence of nodes, $(a_0, a_1, \ldots, a_n)$ for some $n \geq 1$, is a *path of length n* from a_0 to a_n if $(a_{i-1}, a_i) \in \rho$ for all $1 \leq i \leq n$. If $a_0 = a_n$, the path is a *cycle*; H is *acyclic* if it contains no cycles. For a path $(a_0, a_1, \ldots, a_n)$, a_0 is the ancestor of a_n and a_n is the descendant of a_0. A *tree* is an acyclic graph

$T = (A, \rho)$ such that A contains a specific node, called the *root of* T and denoted by $\mathrm{root}(T)$ and every $a \in A - \{\mathrm{root}(T)\}$ is a descendant of $\mathrm{root}(T)$. If a has no descendants, it is a *leaf*. Otherwise, it is an *interior node*. We shall consider T to be an ordered tree, causing every interior node $a \in A$ to have all its direct descendants $b_1 \cdots b_n$ ordered from left to right. The *frontier* of T, denoted as $\mathrm{frontier}(T)$, is the sequence of all leaves of T ordered from left to right. The length of the longest path in T is referred to as the *depth* of T, denoted by $\mathrm{depth}(T)$. For $0 \leq i \leq \mathrm{depth}(T)$, the *i-th level* of T, denoted as $\mathrm{level}(T,i)$, is the sequence of nodes of T in the order defined by the ordered tree with a path of length i from $\mathrm{root}(T)$. A tree $S = (B, \nu)$ is a *subtree* of T if $\emptyset \subset B \subseteq A$, $\nu \subseteq \rho \cap (B \times B)$, and no node in $A - B$ is a descendant of a node in B. S is considered to be an *elementary subtree of* T if $\mathrm{depth}(S) = 1$. All trees in this article are considered to be oriented in top-down manner (so called *out-degree tree*) and ordered from left to right.

Let $G = (N, \Sigma, P, S)$ be a context-free grammar and $A \to x \in P$ be a production. Let T be the elementary tree satisfying $\mathrm{root}(T) = A$ and $\mathrm{frontier}(T) = x$; then, T is considered the *production tree* representing $A \to x$. A *derivation tree* is any tree such that its root belongs to N and each of its elementary subtrees is a production tree representing a rule $p \in P$. The set of all derivation trees, T', such that $\mathrm{frontier}(T') = x$ is denoted by $\Delta_G(x)$; by extension, the set of all derivation trees generating a language, L, is denoted by $\Delta_G(L)$.

The *generative part* of T is the subtree whose root is S and *leaf level* is equivalent to the level of T containing the deepest terminal, whereas the *conclusive part* (or *conclusion*) contains the remaining portion of the tree. Let m_T be the depth of the generative part of T and n_T be the maximum depth of subtrees from the conclusion of T (see Figure 1). If $m_T \geq n_T$, T is said to have a *short conclusion*; otherwise, it is said to have a *long conclusion*. The *depth of conclusion of* T is the maximum depth of a derivation tree located in the conclusive part of T.

Definition 2.5. *Let $G = (N, \Sigma, P, S)$ be a context-free grammar and $x \in \Sigma^*$ be a string. The set of all derivation trees, derivation trees with short conclusion and long conclusion is defined as*

$$\begin{aligned}
{}_{c}\Delta_G(x) &= \Delta_G(x), \\
{}_{sc}\Delta_G(x) &= \{t \in \Delta_G(x) : m_t \geq n_t\}, \text{ and} \\
{}_{lc}\Delta_G(x) &= \{t \in \Delta_G(x) : m_t < n_t\},
\end{aligned}$$

such that m_t, n_t are the depths of the generative and conclusive part of the derivation tree t, respectively.

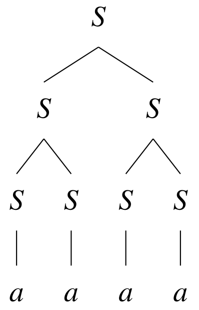
A *tree-controlled grammar* (*TCG* for short) is a pair $H = (G, R)$, where $G = (N, \Sigma, P, S)$ is a context-free grammar and $R \subseteq (N \cup \Sigma)^*$ is a regular control language. The language generated by H is denoted by $L(H)$ and defined by

$$\begin{aligned}
L(H) = \{x : x \in L(G), t \in \Delta_G(x), \\
\text{for all } 0 \leq i < \mathrm{depth}(t), \mathrm{level}(t,i) \in L(R)\}.
\end{aligned}$$

Example 2.6. *Let $H = (G, R)$ with $G = (\{S\}, \{a\}, P, S)$ be a TCG, where P contains*

$$1.\ S \to SS, \quad 2.\ S \to a.$$

with the regular control language $R = \{S\}^*$. In this way, we can generate aaaa in a successful derivation depicted by the following derivation tree where the root is S.



All levels of the derivation tree except for the last one were created by subsequent application of the rule $S \to SS$ until the rule $S \to a$ was applied, forcing the remaining nonterminals to be rewritten in a similar manner; note that the derivation tree does not necessarily reflect the order in which the individual nonterminals were rewritten.

Finally, observe that $L(H) = \{a^{2^n} : n \geq 1\}$ which is a non-context-free language, and that G is a short-conclusive TCG as $n_T = 0$ for all $T \in \Delta_G(L(H))$.

Now, we introduce a modification of tree-controlled grammars that utilizes a level-controlling condition only for the conclusive part of the derivation tree.

Definition 2.7. *Let $H = (G,R)$ be a TCG and $T \in \Delta_G(L(G))$ be a derivation tree generating $L(G)$. Then, the* conclusive condition *($CC(T)$) holds if for all words x, where x is created by concatenating all symbols on a level of the conclusion of T, $x \in R$.*

Definition 2.8. *Let $H = (G,R)$ be a TCG and $L(G) = L_G$. The* conclusive, short-conclusive, *and* long-conclusive language *generated by H is defined by*

$$
\begin{aligned}
{}_cL(H) &= \{x \in L_G : {}_c\Delta_G(x) \cap \Delta_G(L_G) \neq \emptyset \text{ and } CC({}_c\Delta_G(x)) \text{ holds}\},\\
{}_{sc}L(H) &= \{x \in L_G : {}_{sc}\Delta_G(x) \cap \Delta_G(L_G) \neq \emptyset \text{ and } CC({}_{sc}\Delta_G(x)) \text{ holds}\},\\
{}_{lc}L(H) &= \{x \in L_G : {}_{lc}\Delta_G(x) \cap \Delta_G(L_G) \neq \emptyset \text{ and } CC({}_{lc}\Delta_G(x)) \text{ holds}\},
\end{aligned}
$$

respectively.

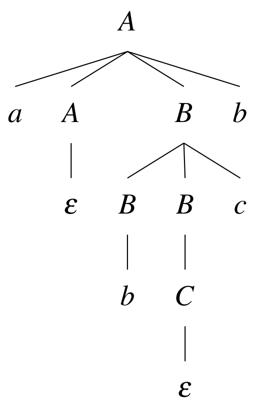
The family of conclusive, short-conclusive, and long-conclusive languages generated by tree-controlled grammars are denoted by **CTC**, **sCTC**, *and* **lCTC**, *respectively.*

For brevity, a TCG generating conclusive, short-conclusive, or long-conclusive language is referred to as a conclusive tree-controlled grammar *(*CTCG *for short, as seen in Example 2.6).*

Example 2.9. *Let $H = (G,R)$ with $G = (N \cup \{A,B,C\}, \Sigma \cup \{a,b\}, P, A)$ be a TCG, where P includes the following rules:*

$$
\begin{array}{lll}
1.\ A \to aABb, & 2.\ A \to \varepsilon, & 3.\ B \to C,\\
4.\ B \to BBc, & 5.\ B \to b, & 6.\ C \to \varepsilon.
\end{array}
$$

According to G, the partial derivation tree, T, depicted below can be constructed.



Observe that the generative part of T starts with the nonterminal A at the topmost level of the derivation tree and ends with the level bC. Here, the last terminal symbol b is generated, followed by the conclusion. Consequently, T is a derivation tree with a short conclusion considering $m_T = \max(\{|w| : w \in \{(A,a), (A,A,\varepsilon), (A,B,B,b), (A,B,B,C), (A,B,c), (A,b)\}\}) = 4$*,* $n_T = \max(\{|\{\varepsilon\}|\}) = 1$ *and thus,* $m_T > n_T$*.*

3 Results

It has been established that tree-controlled grammars are computationally complete even if their control set is restricted to a subregular language (see [2, 8]). This section extends the principles to conclusive tree-controlled grammars and establishes their computational completeness using an extended union-free regular control language.

First, the equality of languages generated by subtypes of conclusive tree-controlled grammars is established. Then, using the third Geffert normal form, the section demonstrates that for every type-0 grammar, Q, there exists an equivalent conclusive tree-controlled grammar, $H = (G,R)$, where $G = (N, \Sigma, P, S)$ and R is an extended union-free regular control language.

Theorem 3.1. *Let* $H = (G,R)$*, where* $G = (N,\Sigma,P,S)$*, be a conclusive tree-controlled grammar. Then,* ${}_{lc}L(H) \subseteq {}_{sc}L(H)$*.*

Proof. Introduce a conclusive tree-controlled grammar, $H_\Delta = (G_\Delta, R)$ such that $G_\Delta = (N,\Sigma,P \cup \{S \to S\},S)$. Then, let m and n be the maximum height of generative part and minimum height of conclusion in $\Delta_G(L(G_\Delta))$, respectively. It is apparent that by $(m-n)$ applications of the rule $S \to S$, all sentences $x \in L(G)$ can be generated with a short conclusion. Thus, the theorem holds. □

Next, we present the basic idea describing how to convert a type-0 grammar $Q = (\{S,S',A,B\},T,P \cup \{ABBBA \to \varepsilon\},S)$ in the Third Geffert normal form to an equivalent conclusive tree-controlled grammar $H = (G,R)$. The idea consists in the creation of a derivation in G by context-free productions in an utterly arbitrary way, after which precisely the substring $ABBBA$ located in the middle of the sentential form is erased repeatedly during the conclusion—that is, the controlled part of the derivation tree. In this way, the correctness of the derivation is verified.

More precisely, G generates every $w \in {}_{lc}L(H)$ by performing three consecutive phases: (I), (II), and (III). First, by using context-free productions, the sentential form uSw is derived, where u is a string over $\{AB, ABB\}^*$, and w is a terminal string, $w \in \Sigma^*$. Considering w, phase (I) is not regulated by the control language R.

Phase (II) starts with application of the production $S \rightarrow S'$, which marks the beginning of the conclusion. In this phase, G rewrites the sentential form $uu'S'v'w$, where v' is a string over $\{BA, BBA\}^*$ representing the nonterminal counterparts of $u, u' \in \{AB, ABB\}^*$. Phase (III) is entered upon replacing the nonterminal S' in the sentential form, as its presence is required to generate any additional symbols. Finally, the substring $ABBBA$ found in the middle of the sentential form is repeatedly activated and erased in accordance with the control language using the following rules:

$$A \rightarrow \bar{A}, \quad B \rightarrow \bar{B}, \quad \bar{A} \rightarrow \varepsilon, \quad \bar{B} \rightarrow \varepsilon.$$

To summarize the rewriting process, every sentence $w \in L(G,R)$ is generated by the following sequence of steps as:

$$S \Rightarrow^*_{\text{(I)}} uSw \Rightarrow^*_{\text{(II)}} uu'S'vw \Rightarrow^*_{\text{(III)}} u''\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}v''w \Rightarrow^* w,$$

where $u'' \in \{AB, ABB\}^*$ and $v'' \in \{BA, BBA\}^*$ such that $u'' \in \text{prefix}(u')$ and $v'' \in \text{suffix}(v')$.

Theorem 3.2. *Let L be a recursively enumerable language. Then, there exists an equivalent conclusive tree-controlled grammar, $H = (G,R)$, where R is an extended union-free regular language such that $L = {}_cL(H)$.*

Proof. Let $Q = (N_Q, \Sigma, P_Q, S)$ be a type-0 grammar such that $L(Q) = L$. Without any loss of generality, assume that Q conforms to the third Geffert normal form (see Definition 2.3), and that $N_Q \cap \{\bar{A}, \bar{B}\} = \emptyset$. Let us introduce a conclusive tree-controlled grammar, $G = (N_G, \Sigma, P_G, S)$, and extended union-free regular control language, R.

Construction. Introduce a conclusive tree-controlled grammar $H = (G,R)$ where $G = (N_Q \cup \{\bar{A}, \bar{B}\}$, Σ, P_G, $S)$ with $V_G = N_G \cup \Sigma$ and $V_Q = N_Q \cup \Sigma$. Let $P_G = P_{Gen} \cup P_{Act} \cup P_{Pro} \cup P_{Era}$ be constructed in the following way:

$$\begin{aligned} P_{Gen} &= \{p \in P_Q : \text{occur}(\text{rhs}(p), S') \geq 1\}, \\ P_{Act} &= \{S' \rightarrow u\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}v : S' \rightarrow uv \in P_Q, u \in \{AB, ABB\}^*, v \in \{BA, BBA\}^*\}, \\ P_{Pro} &= \{A \rightarrow A, B \rightarrow B\}, \\ P_{Era} &= \{A \rightarrow \bar{A}, B \rightarrow \bar{B}, \bar{A} \rightarrow \varepsilon, \bar{B} \rightarrow \varepsilon\}. \end{aligned}$$

Set the partial control languages, R_2 and R_3, as extended union-free regular languages (see Definition 2.2) as

$$\begin{aligned} R_2 &= \{S'\}N_Q^*, \\ R_3 &= \{\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}\}N_Q^*, \end{aligned}$$

and the control language, R, an extended union-free regular language, as

$$R = N_Q^* R_2{}^* R_3^*.$$

Observe that $R_2 \subseteq R$, and $R_3 \subseteq R$ without need of union operation, and that R_2 and R_3 correspond to the phases (II) and (III) of conclusion, respectively.

The control language, R, assures that both context-free phases of the rewriting process proceed without any restrictions, and that the final phase is only finished successfully if the generated sentence belongs to $L(Q)$.

Basic Idea. Next, we sketch the reason why $L(Q) = {}_cL(H)$. H simulates the derivation steps of Q by using a combination of context-free productions and the subregular control language. Phase (I) is completely contained in the generative part of $\Delta_G(x)$. As phase (II) generates new nonterminals and phase (III) propagates the existing nonterminals at the beginning of a conclusive sentential form, the overall control language contains the prefix from N_Q^*.

All context-free productions found originally in Q are represented by the sets P_{Gen} and P_{Act}. The former set consists of all productions of form (iii) of Q (see Definition 2.3) with the purpose of generating the terminal string and surrounding nonterminals, whereas the latter set represents the productions of form (iv), which may be used to effectively activate the central substring, $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$, and subsequently start the erasing phase.

Similarly, the purpose of sets P_{Pro} and P_{Era} is to assure proper generation of the derivation tree; P_{Pro} serves to propagate the corresponding nonterminals A and B to the lower level of the derivation tree, while P_{Era} is responsible for simulation of the sole context-sensitive rule $ABBBA \rightarrow \varepsilon$ from P_Q used to erase the current center of the sentential form.

Partial control language R_2 simulates the use of context-free productions of form (iii) so it is responsible for the generation of the nonterminal suffix needed to generate a sentence. Finally, once S' has been erased from the sentential form, only the partial control language R_3 may be matched. Thanks to the properties of the Geffert normal form, the activation and subsequent erasing process may occur only at one position in the sentential form at a time. Providing by G, the core substrings, S' and $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$, of any partial control language may only appear at most once in the entire sentential form, and their appearance is mutually exclusive; therefore, R_2 and R_3 may be iterated without disrupting the consistency of the rewriting process. In this way, the equivalence of $L(Q)$ and ${}_cL(H)$ is maintained.

In the following claims (*If*) and (*Only if*), using proof by induction on the number of derivation steps, we prove formally that $L(Q) \subseteq {}_cL(H)$ and ${}_cL(H) \subseteq L(Q)$, respectively.

Define the homomorphism h from V_G to V_Q as $h(X) = X$ for all $X \in V_G - \{\bar{A}, \bar{B}\}$ and $h(X) = \varepsilon$ for $X \in \{\bar{A}, \bar{B}\}$. Furthermore, define homomorphism h' from V_G to V_Q as $h'(X) = X$ for $X \in V_G - \{\bar{A}, \bar{B}\}$ and $h'(\bar{X}) = X$ for $\bar{X} \in \{\bar{A}, \bar{B}\}$.

Claim (If). $S \Rightarrow_Q^n x$ implies that $S \Rightarrow_H^* x'$, where $x = h(x')$ or $x = h'(x')$, $x \in V_Q^*$, $x' \in V_G^*$ for some $n \geq 0$ and if x represents a level in conclusive part of $\Delta_G(x')$, then $x' \in R$.

Proof. Induction Basis: Let $n = 0$. The only possible x is equal to S, as $S \Rightarrow_Q^0 S$. Similarly, $S \Rightarrow_H^0 S$, where $S = h(S)$.
Induction Hypothesis: Suppose that the claim holds for all derivations of length j for some $j \geq 0$.
Induction Step: Consider a derivation of the form

$$S \Rightarrow_Q^{j+1} x.$$

Then, there also exists $y \in V_Q^*$ such that

$$S \Rightarrow_Q^j y \Rightarrow_Q x \; [p].$$

By the induction hypothesis, there exists a derivation

$$S \Rightarrow_H^* y' \text{ where } y = h(y') \text{ or } y = h'(y').$$

Considering Q conforms to the third Geffert normal form, the production $p \in P_Q$ has one of the following forms (see Definition 2.3):

1. a context-free production in accordance with one of forms (i) through (iii),
2. a context-free production in form (iv),
3. the context-sensitive erasing production $ABBBA \rightarrow \varepsilon$.

The possibilities that may occur in H based on the production form are the following.

1. The production p is present in H; $y' \Rightarrow_H x'$ $[p]$, where $x = h(x')$. This means that the level the production is applied on either is not regulated or it corresponds to the partial control language $N_Q^* R_2^*$, depending on whether S or S' is in the middle of x'.
2. Let $p = S' \rightarrow uv \in P_Q$ and $p' = S' \rightarrow u\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}v \in P_G$ for some $u \in \{AB, ABB\}^*$, $v \in \{BA, BBA\}^*$. The production p serves as a transition between generating and erasing phases (II) and (III) in Q, which are regulated by partial control languages $N_Q^* R_2^*$ and $N_Q^* R_3^*$, respectively.
$$\begin{aligned} y\ &= \alpha S'\beta \Rightarrow_Q \alpha uv\beta = x = h(x')\ [p] \\ y' &= \alpha S'\beta \Rightarrow_H \alpha u\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}v\beta = x'\ [p'] \end{aligned}$$
for some $\alpha \in N_Q^*$, and $\beta \in N_Q^* \Sigma^*$. In H, the $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$ substring is erased immediately after being generated together with the generation of another activated substring $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$ in the next level of the derivation tree.
3. The context-sensitive production $ABBBA \rightarrow \varepsilon \in P_Q$ is simulated by consecutive application of context-free productions, $X \rightarrow \bar{X}$, $\bar{X} \rightarrow \varepsilon$ for $X \in \{A, B\}$, to select the $ABBBA$ substring and subsequently erase it. In Q, this erasure is performed in one derivation step and it is sufficient to mark the ensuing $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$ substring in H.
$$\begin{aligned} y\ &= \alpha ABBBA\beta \Rightarrow_Q \alpha\beta = x = h(x') \\ y' &= \alpha\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}\beta \Rightarrow_H^0 x' \end{aligned}$$
for some $\alpha \in N_Q^*$, and $\beta \in N_Q^* \Sigma^*$.

   To assure that all required nonterminals placed next to each other are affected, the level of the derivation tree is regulated by the partial control language $N_Q^* R_3^*$. Notice that all terminals occurred in the previous levels of $\Delta_G(x')$ so we have only nonterminal strings in the conclusive levels to control by $N_Q^* R_3^*$.

Thus, this claim conforms to the rules of induction. □

Claim (Only if). $S \Rightarrow_H^n x$ implies that $S \Rightarrow_Q^* x'$, where $x \in V_G^*$, $x' \in V_Q^*$, such that $x' = h(x)$ or $x' = h'(x)$ for some $n \geq 0$ and if x represents a level in conclusive part of $\Delta_G(x)$, then $x \in R$.

Proof. Induction Basis: $S \Rightarrow_H^* S = x$ implies $S \Rightarrow_Q^0 S = x'$ where $x = h(x')$.
Induction Hypothesis: Suppose that the claim holds for all derivations of length j for some $j \geq 0$.
Induction Step: Consider a derivation of the form
$$S \Rightarrow_H^{j+1} x.$$
Then, there also exists $y \in V_G^*$, such that
$$S \Rightarrow_H^j y \Rightarrow_H x.$$
By the induction hypothesis, there exists a derivation
$$S \Rightarrow_Q^* y',\ \text{where } h(y) = y' \text{ or } h'(y) = y'.$$
According to the subsets of P_G to which the used production, p, belongs to, four cases in Q follow.

1. Production $p \in P_{Gen}$ containing S' in $\mathrm{rhs}(p)$.

   (a) The form of p depends on the moment of the rewriting when it is applied at; it either serves as the entry point of the conclusion, by using the production $p = S \to S'$, or it is used to generate nonterminals to the right of S' and $p = S' \to uS'v$, $v \in \{BA, BBA\}^*$. The production $S' \to uS'v$ is used on the level described by the partial control language $N_Q^* R_2^*$.

   In this case, the production may be applied in Q in a way analogous to H,

   $$y' \Rightarrow_Q x' \; [p], \text{ where } h(x) = x'.$$

   Considering the structure of the production, it is clear that

   $$\mathrm{occur}(\mathrm{lhs}(p), S') = \mathrm{occur}(\mathrm{rhs}(p), S') = 1$$

   for all $p \in P_{Gen}$; thus, no extra S' symbols are generated. The nonterminals that were not affected by the production, and are already present in the sentential form, are nondeterministically propagated using the productions of P_{Pro}; otherwise, the application of productions of P_{Era} would cause the rewriting process to halt in the future.

   Consequently, the corresponding partial control language, R_2^*, is iterated precisely once while the nonterminal S' is present in the sentential form, as each iteration must contain one of the aforementioned nonterminals.

2. Production $p \in P_{Pro}$ serves to propagate the corresponding nonterminal to the following level of the derivation tree while not affecting the sentential form. Because of this, application of p in Q is equal to

   $$y' \Rightarrow_Q^0 y' = x'.$$

   Considering the production $p \in P_{Pro}$ works with nonterminals $\{A, B\} \subset N_Q$, its application is allowed at any place of the control language, N_Q^*, R_2^*, and R_3^*.

3. Production $p \in P_{Act}$ provides the transition between phases (II) and (III) of the rewriting process, which are described by partial control languages R_2^* and R_3^*, respectively. Necessarily, p has the form of $S' \to u\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}v$, where $u \in \{AB, ABB\}^*$, $v \in \{BA, BBA\}^*$. This process may be described as

   $$y = \alpha S' \beta \Rightarrow_H \alpha u\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}v\beta = x \; [p],$$

   with $h(x) = h(\alpha u\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}v\beta) = h(\alpha uv\beta)$ for some $\alpha \in N_Q^*$, $\beta \in N_Q^*\Sigma^*$, and

   $$y' = \alpha S' \beta \Rightarrow_Q h(\alpha uv\beta) = x' = h(x) \; [S' \to uv].$$

   Subsequently, the $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$ substring is removed in H using the productions of P_{Era}.

4. Productions $p \in P_{Era}$ are used to repeatedly select nonterminals of the $ABBBA$ substring and erase them in an inside-out way. The initial place of erasure is determined by the location of the nonterminal S'.

   (a) Let $p_X = X \to \bar{X}$, $X \in \{A, B\}$. Production p_X nondeterministically marks the nonterminal to be erased. However, this step is only required to simulate the context-sensitive production in H, and therefore application of p_X does not affect the sentential form of Q. Production p_X may be applied in the following way:

   $$y = \alpha uXv\beta \Rightarrow_H \alpha u\bar{X}v\beta = x \; [p_X]$$

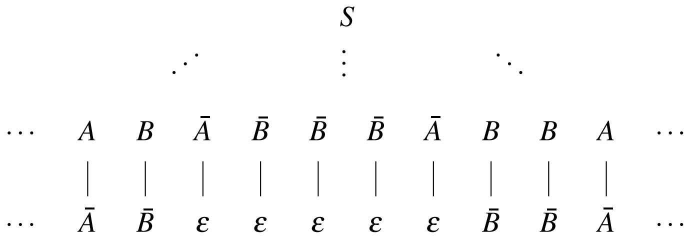

S
··· A B Ā B̄ B̄ B̄ Ā B B A ···
··· Ā B̄ ε ε ε ε ε B̄ B̄ Ā ···

Figure 2: Derivation tree reflecting phase (III) of the rewriting process. Productions of P_{Era} are applied repeatedly on the same level to simulate application of $ABBBA \rightarrow \varepsilon$.

with $uXv \in \{A,\bar{A}\}\{B,\bar{B}\}^3\{A,\bar{A}\}$ and $X \in \{A,B\}$, and whose equivalent in Q would be as follows:

$$y' = \alpha ABBBA\beta \Rightarrow_Q^0 \alpha ABBBA\beta = h'(x).$$

(b) Let $p_{\bar{X}} = \bar{X} \rightarrow \varepsilon$, $X \in \{A,B\}$. Production $p_{\bar{X}}$ erases the previously selected nonterminal to simulate the production $ABBBA \rightarrow \varepsilon \in P$. It can be applied either immediately after the application of a production from P_{Act}, or as the follow-up of productions from P_{Pro} to generate the next level of the derivation tree. Considering the $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$ substring only serves as an intermediary in the erasing process, it may be ignored in Q completely;

$$\begin{aligned} y &= \alpha\bar{u}\bar{X}\bar{v}\beta \Rightarrow_H \alpha\bar{u}\bar{v}\beta = x \\ y' &= \alpha\beta \Rightarrow_Q^0 x' = y' \end{aligned}$$

where $\bar{X} \in \{\bar{A},\bar{B}\}, \bar{u},\bar{v} \in \{\bar{A},\bar{B}\}^*$ such that $\bar{u}\bar{X}\bar{v} \in \{\bar{A}^{i'}\bar{B}^{j'}\bar{A}^{k'} : i' \in \{0,1\}, j' \in \{0,1,2,3\}, k' \in \{0,1\}\}$, $\text{occur}(\alpha,\{\bar{A},\bar{B}\}) = \text{occur}(\beta,\{\bar{A},\bar{B}\}) = 0$ and $h(y) = y'$.

These productions may only be applied during phase (III) of the rewriting process, which is regulated by the partial control language $N_Q^* R_3^*$. Because of this, the productions p_X and $p_{\bar{X}}$ have to be applied on all nonterminals of the $ABBBA$ and $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$ substrings, respectively, as seen in Figure 2. Selection of any other symbols would cause the rewriting process to halt on the following level of the derivation tree.

Because of the properties of the Geffert normal form, the sentential form will always contain at most one occurrence of the substring $ABBBA$. Therefore, the partial control language R_3^* is always iterated precisely once as long as the sentential form contains some $\bar{A}$s or $\bar{B}$s.

It is clear that $p \in P - P_{Pro}$ can only be used during a specific phase of the rewriting process controlled by the corresponding expression. If this were to be violated, a sentential form not described by R would arise, resulting in blocking the derivation in H.

It is important to note that iteration of partial control language R_2 and R_3 depends on the presence of their core substring, S' and $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$, in their respective order. To generate a sentence, x, it is necessary that TCG H goes through all of the following sentential forms.

$$\begin{array}{rl|l} S & \Rightarrow_H^* \alpha_1 S x & \alpha_1 \in N_Q^*, x \in \Sigma^* \\ & \Rightarrow_H \alpha_1 S' x & \\ & \Rightarrow_H^* \alpha_1 \alpha_2 S' \beta_2 x & \alpha_2, \beta_2 \in N_Q^* \\ & \Rightarrow_H^* \alpha_3 \bar{A}\bar{B}\bar{B}\bar{B}\bar{A}\beta_3 x & \alpha_3, \beta_3 \in N_Q^* \\ & \Rightarrow_H^* x & \end{array}$$

This implies that the presence of the individual core substrings is mutually exclusive. As a result, only one of the partial control languages, R_2, R_3, may be positively iterated at a time; in that case, it is iterated precisely once. Thus, ${}_{c}L(H) \subseteq L(Q)$. □

Claim (Long-conclusiveness). ${}_{c}L(H)$ is also long-conclusive.

Proof. Let $x = a_1 a_2 \cdots a_n$ where $a_i \in \Sigma$ for all $1 \leq i \leq n$ generated as

$$S \Rightarrow_H^* \alpha_1 S x \Rightarrow_H \alpha_1 S' x \Rightarrow_H^* \alpha_1 \alpha_2 S' \beta_2 x \Rightarrow_H^* \alpha_3 \bar{A}\bar{B}\bar{B}\bar{B}\bar{A} \beta_3 x \Rightarrow_H^* x$$

where $\alpha_i, \beta_j \in N_Q^*$ for all $1 \leq i \leq 3, 2 \leq j \leq 3$ and $x \in \Sigma^*$. The depth of the generative part for any $T \in \Delta_G(x)$ is $n+1$ because of the properties of the Geffert normal form. Subsequently, the conclusion has the minimum depth of $k+m+1$, where $k \geq 1$ represents the number of levels needed to generate β_2, and $m \geq 1$ is the number of levels needed to verify the derivation or, in other words, erase the substrings $\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}$ located in the middle of the sentential form. Since α_1 contains at least n occurrences of A, $m \geq n$, so $k+m+1 > n+1$ and the theorem holds. □

Claim (Iff). $S \Rightarrow_Q^* x$ if and only if $S \Rightarrow_H^* x$ where $x \in \Sigma^*$.

Proof. Consider $x \in \Sigma^*$ in Claims (*If*) and (*Only if*) of the previous proof. Since $x = x'$ as h and h' for terminal symbols is the identity, thus this claim holds. □

By Claims (*Iff*) and (*Long-conclusiveness*), $L(Q) = {}_{lc}L(H)$, and Theorem 3.2 holds. □

Corollary 3.3. **CTC** = **sCTC** = **lCTC** = **RE**.

Proof. By Theorem 3.2, **lCTC** = **RE**. By Church's thesis (e.g., see page 630 in [6]), **sCTC** ⊆ **RE**, and by Theorem 3.1, **lCTC** ⊆ **sCTC**, so **sCTC** = **RE**. □

Observe that the control language R can be replaced by a union-free language $\hat{R} = (A^*B^*S'^*)^* (\{\bar{A}\bar{B}\bar{B}\bar{B}\bar{A}\}^*(A^*B^*S'^*)^*)^*$ while the previous proof technique still works.

Lemma 3.4. *The union-free regular language, $\hat{R}$, can be generated by a right-linear grammar, $G_{\hat{R}}$, with the nonterminal complexity of* 1.

Proof. Recall that $N_G = \{S, S', A, B, \bar{A}, \bar{B}\}$. Let $G_{\hat{R}} = (\{\hat{S}\}, N_G, P_{\hat{R}}, \hat{S})$ be a right-linear grammar, where $P_{\hat{R}}$ is defined as follows:

$$P_{\hat{R}} = \{\hat{S} \rightarrow x\hat{S} : x \in \{A, B, S'\}\} \cup \{\hat{S} \rightarrow \bar{A}\bar{B}\bar{B}\bar{B}\bar{A}\hat{S}, \hat{S} \rightarrow \varepsilon\}.$$

It can easily be verified that $L(G_{\hat{R}}) = \hat{R}$, and therefore Lemma 3.4 holds. □

It has already been shown that any recursively enumerable language can be generated by a tree-controlled grammar with a regular control language whose nonterminal complexity is equal to 7 (see [7]); however, the result was heavily dependent on the use of union operation. We improve this result by the introduction of conclusiveness of the tree-controlled grammar and by omitting the use of union operation altogether.

Theorem 3.5. *Any recursively enumerable language, L, can be generated by a conclusive tree-controlled grammar controlled by a union-free regular language using seven nonterminals.*

Proof. Let Q' be a type-0 grammar such that $L = L(Q')$ and construct $H' = (G', \hat{R})$ as a conclusive tree-controlled grammar such that $L(Q') = L(H')$, where G' is constructed in accordance with the proof of Theorem 3.2. $G' = (\{S, S', A, B, \bar{A}, \bar{B}\}, \Sigma_{G'}, P_{G'}, S)$; similarly, let $G_{\hat{R}}$ be a right-linear grammar constructed in accordance with the proof of Lemma 3.4, such that $L(G_{\hat{R}}) = \hat{R}$ and $G_{\hat{R}} = (\{\hat{S}\}, \{S, S', A, B, \bar{A}, \bar{B}\}, P_{\hat{R}}, \hat{S})$. Clearly, the nonterminal complexity of grammars G' and $G_{\hat{R}}$ is 6 and 1, respectively, bringing the overall nonterminal complexity of H' to 7. □

4 Conclusion and Open Problems

We conclude this paper by remarking on some of the properties of conclusive tree-controlled grammars. Although this modification is based upon the same principle as the original tree-controlled grammars, its main advantage lies in the fact that until all terminals have been generated, the derivation tree is not regulated. Thus, the modification offers significantly lower descriptional complexity, making it better suited for practical use—this is the case especially for the short-conclusive variant of the grammar (see Definition 2.8). Observe that all families of languages generated by conclusive tree-controlled grammars are equivalent, meaning the length of the conclusion should not affect the generative power in any way.

Finally, we propose four open problems regarding the conclusive modification of tree-controlled grammars:

1. Consider conclusive tree-controlled grammars with a short conclusion. What is the minimum depth of conclusion needed to maintain the computational completeness of the grammars?
2. What is the minimum possible nonterminal complexity of conclusive tree-controlled grammars? Can it be further restricted beyond seven nonterminals?
3. Introduce a modification of conclusive tree-controlled grammars whose core grammar is at most linear. Does this modification affect the generative power of conclusive tree-controlled grammars?
4. Study other formal grammars working in a conclusive way, where generative and conclusive parts of the derivation tree can be distinguished.

Acknowledgment

This work was supported by the Ministry of Education, Youth and Sports of Czech Republic project ERC.CZ no. LL1908 and the BUT grant FIT-S-20-6293.

References

[1] K Culik II & H A Maurer (1977): *Tree controlled grammars*. *Computing* 19(2), pp. 129–139, doi:10.1007/BF02252350.

[2] Jürgen Dassow & Bianca Truthe (2008): *Subregularly Tree Controlled Grammars and Languages*. In Erzsébet Csuhaj-Varjú & Zoltán Ésik, editors: *Automata and Formal Languages, 12th International Conference, AFL 2008, Balatonfüred, Hungary, May 27-30, 2008, Proceedings*, pp. 158–169.

[3] Viliam Geffert (1991): *Normal forms for phrase-structure grammars*. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications* 25(5), pp. 473–496.

[4] John E. Hopcroft, Rajeev Motwani & Jeffrey D. Ullman (2006): *Introduction to Automata Theory, Languages, and Computation*. Pearson.

[5] Zbyněk Křivka & Alexander Meduna (2021): *Scattered Context Grammars with One Non-Context-Free Production are Computationally Complete*. Fundamenta Informaticae 179(4), pp. 361–384, doi:10.3233/FI-2021-2028.

[6] Alexander Meduna (2000): *Automata and Languages: Theory and Applications*. Springer, London.

[7] Sherzod Tuarev, Jürgen Dassow & Mohd Selamat (2011): *Nonterminal complexity of tree controlled grammars*. Theoretical Computer Science 412, pp. 5789–5795, doi:10.1016/j.tcs.2011.06.033.

[8] Sherzod Turaev, Jürgen Dassow, Florin Manea & Mohd Hasan Selamat (2012): *Language classes generated by tree controlled grammars with bounded nonterminal complexity*. Theoretical Computer Science 449, pp. 134–144, doi:10.1016/j.tcs.2012.04.013.