Strictly Locally Testable and Resources Restricted Control Languages in Tree-Controlled Grammars

Bianca Truthe

Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany

bianca.truthe@informatik.uni-giessen.de

Tree-controlled grammars are context-free grammars where the derivation process is controlled in such a way that every word on a level of the derivation tree must belong to a certain control language. We investigate the generative capacity of such tree-controlled grammars where the control languages are special regular sets, especially strictly locally testable languages or languages restricted by resources of the generation (number of non-terminal symbols or production rules) or acceptance (number of states). Furthermore, the set theoretic inclusion relations of these subregular language families themselves are studied.

1 Introduction

In the monograph [5] by Jürgen Dassow and Gheorghe Păun, *Seven Circumstances Where Context-Free Grammars Are Not Enough* are presented. A possibility to enlarge the generative power of context-free grammars is to introduce some regulation mechanism which controls the derivation in a context-free grammar. In some cases, regular languages are used for such a regulation. They are rather easy to handle and, used as control, they often lead to context-sensitive or even recursively enumerable languages while the core grammar is only context-free.

One such control mechanism was introduced by Karel Čulik II and Hermann A. Maurer in [16] where the structure of derivation trees of context-free grammars is restricted by the requirement that the words of all levels of the derivation tree must belong to a given regular (control) language. This model is called tree-controlled grammar.

Gheorghe Păun proved that the generative capacity of such grammars coincides with that of context-sensitive grammars (if no erasing rules are used) or arbitrary phrase structure grammars (if erasing rules are used). Thus, the question arose to what extend the restrictions can be weakened in order to obtain 'useful' families of languages which are located somewhere between the classes of context-free and context-sensitive languages.

In [6, 7, 8, 9, 27, 29, 30], many subregular families of languages have been investigated as classes for the control languages. In this paper, we continue this research with further subregular language families, especially strictly locally testable languages or languages restricted by resources of the generation (number of non-terminal symbols or production rules) or acceptance (number of states). Furthermore, the set theoretic inclusion relations of these subregular language families themselves are studied.

2 Preliminaries

Throughout the paper, we assume that the reader is familiar with the basic concepts of the theory of automata and formal languages. For details, we refer to [23]. Here we only recall some notation and the definition of contextual grammars with selection which form the central notion of the paper.

Zs. Gazdag, Sz. Iván, G. Kovásznai (Eds.): 16th International Conference on Automata and Formal Languages (AFL 2023)
EPTCS 386, 2023, pp. 253–268, doi:10.4204/EPTCS.386.20

2.1 Languages, grammars, automata

Given an alphabet V, we denote by V^* and V^+ the set of all words and the set of all non-empty words over V, respectively. The empty word is denoted by λ. By V^k, and $V^{\leq k}$ for some natural number k, we denote the set of all words of the alphabet V with exactly k letters and the set of all words over V with at most k letters, respectively. For a word w and a letter a, we denote the length of w by $|w|$ and the number of occurrences of the letter a in the word w by $|w|_a$. For a set A, we denote its cardinality by $|A|$.

A right-linear grammar is a quadruple $G = (N,T,P,S)$ where N is a finite set of non-terminal symbols, T is a finite set of terminal symbols, P is a finite set of production rules of the form $A \to wB$ or $A \to w$ with $A,B \in N$ and $w \in T^*$, and $S \in N$ is the start symbol. Such a grammar is called regular, if all the rules are of the form $A \to xB$ or $A \to x$ with $A,B \in N$ and $x \in T$ or $S \to \lambda$. The language generated by a right-linear or regular grammar is the set of all words over the terminal alphabet which are obtained from the start symbol S by a successive replacement of the non-terminal symbols according to the rules in the set P. A non-terminal symbol A is replaced by the right-hand side w of a rule $A \to w \in P$ in order to derive the next sentential form. The language generated consists of all sentential forms without a non-terminal symbol. Every language generated by a right-linear grammar can also be generated by a regular grammar.

A deterministic finite automaton is a quintuple $A = (V,Z,z_0,F,\delta)$ where V is a finite set of input symbols, Z is a finite set of states, $z_0 \in Z$ is the initial state, $F \subseteq Z$ is a set of accepting states, and δ is a transition function $\delta : Z \times V \to Z$. The language accepted by such an automaton is the set of all input words over the alphabet V which lead letterwise by the transition function from the initial state to an accepting state.

A regular expression over an alphabet V is defined inductively as follows:

1. $\emptyset$ is a regular expression;
2. every element $x \in V$ is a regular expression;
3. if R and S are regular expressions, so are the concatenation $R \cdot S$, the union $R \cup S$, and the Kleene closure R^*;
4. for every regular expression, there is a natural number n such that the regular expression is obtained from the atomic elements $\emptyset$ and $x \in V$ by n operations concatenation, union, or star.

The language $L(R)$ which is described by a regular expression R is also inductively defined: $L(\emptyset) = \emptyset$; $L(x) = \{x\}$ for each $x \in V$; and $L(R \cdot S) = L(R) \cdot L(S)$, $L(R \cup S) = L(R) \cup L(S)$, and $L(R^*) = (L(R))^*$ for regular expressions R and S.

The set of all languages generated by some right-linear grammar coincides with the set of all languages accepted by a deterministic finite automaton and with the set of all languages described by a regular expression. All these languages are called regular and form a family denoted by *REG*. Any subfamily of this set is called a subregular language family.

A context-free grammar is a quadruple $G = (N,T,P,S)$ where N, T, and S are as in a right-linear grammar but the production rules in the set P are of the form $A \to w$ with $A \in N$ and $w \in (N \cup T)^*$.

The language generated by a context-free grammar is the set of all words over the terminal alphabet which are obtained from the start symbol S by replacing sequentially the non-terminal symbols according to the rules in the set P. A language is called context-free if it is generated by some context-free grammar. The family of all context-free languages is denoted by *CF*.

With a derivation of a terminal word by a context-free grammar, we associate a derivation tree which has the start symbol in its root and where every node with a non-terminal $A \in N$ has as children nodes with symbols which form, read from left to right, a word w such that $A \to w$ is a rule of the grammar (if $A \to \lambda$, then the node with A has only one child node and this is labelled with λ). Nodes with terminal symbols or λ have no children. With any derivation tree t of height k and any number $0 \leq j \leq k$, we associate the word of level j and the sentential form of level j which are given by all nodes of depth j read from left to right and all nodes of depth j and all leaves of depth less than j read from left to right, respectively. Obviously, if two words w and v are sentential forms of two successive levels, then $w \Longrightarrow^* v$ holds and this derivation is obtained by a parallel replacement of all non-terminal symbols occurring in the word w.

A context-sensitive grammar is a quadruple $G = (N,T,P,S)$ where N is a finite set of non-terminal symbols, $S \in N$ is the start symbol, T is a finite set of terminal symbols, and P is a finite set of production rules of the form $\alpha \to \beta$ with $\alpha \in (N \cup T)^+ \setminus T^*$, $\beta \in (N \cup T)^*$, and $|\beta| \geq |\alpha|$ with the only exception that $S \to \lambda$ is allowed if the sysmbol S does not occur on any right-hand side of a rule. The language generated by a context-sensitive grammar is the set of all words over the terminal alphabet which are obtained from the start symbol S by replacing sequentially subwords according to the rules in the set P. A language is called context-sensitive if it is generated by some context-sensitive grammar. The family of all context-sensitive languages is denoted by CS. For every context-sensitive language L, there is a context-sensitive grammar $G = (N,T,P,S)$ with $L(G) = L$, where all rules in P are of the form

$$AB \to CD,\ A \to BC,\ A \to B,\ \text{or}\ A \to a$$

with $A,B,C,D \in N$ and $a \in T$, or $S \to \lambda$ if S does not occur on the right-hand side of a rule. Such a grammar is said to be in Kuroda normal form ([17]).

We also mention here four classes of languages without a definition since they are mentioned only in the summary of existing results: By MAT, we denote the family of all languages generated by matrix grammars with appearance checking and without erasing rules; by MAT_{fin}, we denote the family of all such languages where the matrix grammar is of finite index ([5], [23]). By $E0L$ ($ET0L$), we denote the family of all languages generated by extended (tabled) interactionless Lindenmayer systems ([22]).

2.2 Complexity measures and resources restricted languages

Let $G = (N,T,P,S)$ be a right-linear grammar, $A = (V,Z,z_0,F,\delta)$ be a deterministic finite automaton, and L be a regular language. Then, we recall the following complexity measures from [4]:

$$\begin{aligned}
State(A) &= |Z|, Var(G) = |N|, Prod(G) = |P|,\\
State(L) &= \min\{\, State(A) \mid A \text{ is a det. finite automaton accepting } L \,\},\\
Var_{RL}(L) &= \min\{\, Var(G) \mid G \text{ is a right-linear grammar generating } L \,\},\\
Prod_{RL}(L) &= \min\{\, Prod(G) \mid G \text{ is a right-linear grammar generating } L \,\}.
\end{aligned}$$

We now define subregular families by restricting the resources needed for generating or accepting their elements:

$$\begin{aligned}
RL_n^V &= \{\, L \mid L \in REG \text{ with } Var_{RL}(L) \leq n \,\},\\
RL_n^P &= \{\, L \mid L \in REG \text{ with } Prod_{RL}(L) \leq n \,\},\\
REG_n^Z &= \{\, L \mid L \in REG \text{ with } State(L) \leq n \,\}.
\end{aligned}$$

2.3 Subregular language families based on the structure

We consider the following restrictions for regular languages. Let L be a language over an alphabet V.

With respect to the alphabet V, the language L is said to be

- *monoidal* if and only if $L = V^*$,
- *nilpotent* if and only if it is finite or its complement $V^* \setminus L$ is finite,
- *combinational* if and only if it has the form $L = V^*X$ for some subset $X \subseteq V$,
- *definite* if and only if it can be represented in the form $L = A \cup V^*B$ where A and B are finite subsets of V^*,
- *suffix-closed* (or *fully initial* or *multiple-entry* language) if and only if, for any two words $x \in V^*$ and $y \in V^*$, the relation $xy \in L$ implies the relation $y \in L$,
- *ordered* if and only if the language is accepted by some deterministic finite automaton

$$A = (V, Z, z_0, F, \delta)$$

  with an input alphabet V, a finite set Z of states, a start state $z_0 \in Z$, a set $F \subseteq Z$ of accepting states and a transition mapping δ where $(Z, \preceq)$ is a totally ordered set and, for any input symbol $a \in V$, the relation $z \preceq z'$ implies $\delta(z,a) \preceq \delta(z',a)$,
- *commutative* if and only if it contains with each word also all permutations of this word,
- *circular* if and only if it contains with each word also all circular shifts of this word,
- *non-counting* (or *star-free*) if and only if there is a natural number $k \geq 1$ such that, for every three words $x \in V^*$, $y \in V^*$, and $z \in V^*$, it holds $xy^kz \in L$ if and only if $xy^{k+1}z \in L$,
- *power-separating* if and only if, there is a natural number $m \geq 1$ such that for every word $x \in V^*$, either $J_x^m \cap L = \emptyset$ or $J_x^m \subseteq L$ where $J_x^m = \{ x^n \mid n \geq m \}$,
- *union-free* if and only if L can be described by a regular expression which is only built by product and star,
- *strictly locally k-testable* if and only if there are three subsets B, I, and E of V^k such that any word $a_1a_2 \ldots a_n$ with $n \geq k$ and $a_i \in V$ for $1 \leq i \leq n$ belongs to the language L if and only if

$$\begin{gathered} a_1a_2\ldots a_k \in B, \\ a_{j+1}a_{j+2}\ldots a_{j+k} \in I \text{ for every } j \text{ with } 1 \leq j \leq n-k-1 \text{ and} \\ a_{n-k+1}a_{n-k+2}\ldots a_n \in E, \end{gathered}$$

- *strictly locally testable* if and only if it is strictly locally k-testable for some natural number k.

We remark that monoidal, nilpotent, combinational, definite, ordered, union-free, and strictly locally (k-)testable languages are regular, whereas non-regular languages of the other types mentioned above exist. Here, we consider among the commutative, circular, suffix-closed, non-counting, and power-separating languages only those which are also regular.

Some properties of the languages of the classes mentioned above can be found in [24] (monoids), [11] (nilpotent languages), [13] (combinational and commutative languages), [19] (definite languages), [12] and [2] (suffix-closed languages), [25] (ordered languages), [3] (circular languages), [18] (non-counting and strictly locally testable languages), [26] (power-separating languages), [1] (union-free languages).

By *FIN*, *MON*, *NIL*, *COMB*, *DEF*, *SUF*, *ORD*, *COMM*, *CIRC*, *NC*, *PS*, *UF*, SLT_k (for any natural number $k \geq 1$), and *SLT*, we denote the families of all finite, monoidal, nilpotent, combinational, definite, regular suffix-closed, ordered, regular commutative, regular circular, regular non-counting, regular power-separating, union-free, strictly locally k-testable, and strictly locally testable languages, respectively.

For any natural number $n \geq 1$, let MON_n be the set of all languages that can be represented in the form $A_1^* \cup A_2^* \cup \cdots \cup A_k^*$ with $1 \leq k \leq n$ where all A_i ($1 \leq i \leq k$) are alphabets. Obviously,

$$MON = MON_1 \subset MON_2 \subset \cdots \subset MON_j \subset \cdots .$$

A strictly locally testable language characterized by three finite sets B, I, and E as above which includes additionally a finite set F of words which are shorter than those of the sets B, I, and E is denoted by $[B,I,E,F]$.

As the set of all families under consideration, we set

$$\begin{aligned}\mathfrak{F} = \{&FIN, NIL, COMB, DEF, SUF, ORD, COMM, CIRC, NC, PS, UF\} \\ &\cup \{\, MON_k \mid k \geq 1 \,\} \cup \{SLT\} \cup \{\, SLT_k \mid k \geq 1 \,\} \\ &\cup \{\, RL_n^V \mid n \geq 1 \,\} \cup \{\, RL_n^P \mid n \geq 1 \,\} \cup \{\, REG_n^Z \mid n \geq 1 \,\}.\end{aligned}$$

2.4 Hierarchy of subregular families of languages

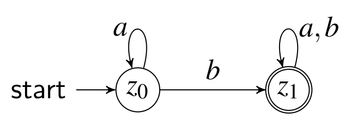
In this section, we present a hierarchy of the families of the aforementioned set $\mathfrak{F}$ with respect to the set theoretic inclusion relation. A summary is depicted in Figure 1.

Before this, we prove some relations of the classes of strictly locally k-testable languages to the subregular language families restricted by resources, which have not been considered in the literature yet.

For this purpose, we first introduce some languages which serve later as witness languages for proper inclusions and incomparabilities.

Lemma 2.1 *The language $L_1 = \{a\}^*\{b\}\{a,b\}^*$ belongs to $REG_2^Z \setminus SLT$.*

Proof. The language L_1 is accepted by the automaton with two states whose transition function is given in the following diagram (double-circled states are accepting):



Suppose, the language L_1 is strictly locally k-testable for some natural number $k \geq 1$. Then, there exist sets $B \subseteq V^k$, $I \subseteq V^k$, $E \subseteq V^k$, and $F \subseteq V^{\leq k-1}$ such that $L_1 = [B,I,E,F]$. Since the word $a^{2k}ba^{2k}$ belongs to the language L_1, we know that $a^k \in B \cap I \cap E$. But then, also the word a^{2k} belongs to the language which is a contradiction. □

Lemma 2.2 *The language $L_2 = [\{a,b\},\{b,c\},\{a,c\},\emptyset]$ belongs to $SLT_1 \setminus REG_4^Z$.*

Proof. By definition, $L_2 \in SLT_1$.

We now prove that L_2 is not accepted by an deterministic finite automaton with less than five states. Let $L = L_2$ and let R_L be the Myhill-Nerode equivalence relation (see [15]): two words x and y are in this relation if and only if, for all words z, either both words xz and yz belong to the language L or none of them. The words λ, a, b, c, and aa are pairwise not in this relation, as one can check.

Therefore, the index of the language L is at least five. Hence, at least five states are necessary for accepting the language L. □

Lemma 2.3 *For each natural number $n \geq 2$, let $V_n = \{a_1, a_2, \ldots, a_{n-1}\}$ be an alphabet with $n-1$ pairwise different letters and let $L_{3,n} = \{a_1 a_2 \ldots a_{n-1}\}$. Then, every language $L_{3,n}$ for $n \geq 2$ belongs to the set $SLT_2 \setminus REG_n^Z$.*

Proof. The statement $L_{3,n} \in SLT_2$ for $n \geq 2$ can be seen as follows. If $n = 2$, then $L_{3,n} = [\emptyset, \emptyset, \emptyset, \{a_1\}]$, otherwise $L_{3,n} = [\{a_1 a_2\}, \{\, a_p a_{p+1} \mid 2 \leq p \leq n-3 \,\}, \{a_{n-2} a_{n-1}\}, \emptyset]$.

For accepting any language $L_{3,n}$ for $n \geq 2$, at least $n+1$ states are necessary (follows from the fact that the n partial words $a_1 \ldots a_i$ for $0 \leq i \leq n-1$ and $a_1 a_1$ are pairwise not in the Myhill-Nerode relation). □

Lemma 2.4 *For each natural number $n \geq 1$, let $L_{4,n} = \{a^n\}$. Then $L_{4,n}$ belongs to the set $RL_1^P \setminus SLT_n$.*

Proof. The single word a^n can be generated with one rule, hence, $L_{4,n} \in RL_1^P$.

Assume that such a language is strictly locally n-testable. Then, it is $L_{4,n} = [B, I, E, F]$ for suitable sets B, I, E, and F. From $L_{4,n} = \{a^n\}$, it follows that $B = E = \{a^n\}$. But then, also the word a^{n+1} belongs to the language $L_{4,n}$ which is a contradiction. □

Lemma 2.5 *For each natural number $n \geq 1$, let $V_n = \{a_1, a_2, \ldots, a_n\}$ be an alphabet with n pairwise different letters and let $L_{5,n} = V_n^*$. Then, for $n \geq 1$, the language L_n belongs to the set $SLT_1 \setminus RL_n^P$.*

Proof. The language $L_{5,n}$ can be represented as $L_{5,n} = [V, V, V, \{\lambda\}]$. Hence, $L_{5,n} \in SLT_1$ for $n \geq 1$.

For generating a language $L_{5,n}$ for some number $n \geq 1$, at least a non-terminating rule is necessary for every letter a_i ($1 \leq i \leq n$) and additionally a terminating rule. Hence, $L_{5,n} \notin RL_n^P$. □

Lemma 2.6 *The language $L_6 = \{a\}$ belongs to $RL_1^V \setminus SLT_1$.*

Proof. The language L_6 can be generated with a single rule and, hence, with one non-terminal only.

Assume that L_6 is strictly locally 1-testable and can be represented as $[B, I, E, F]$. Then $B = E = \{a\}$. But then, also the word aa belongs to the language which is a contradiction. □

Lemma 2.7 *The language $L_7 = \{a\}\{b\}^*\{a\} \cup \{a\}$ belongs to $SLT_1 \setminus RL_1^V$.*

Proof. The language L_7 is strictly locally 1-testable and can be represented as $[\{a\}, \{b\}, \{a\}, \emptyset]$.

Assume that the language L_7 is generated by a right-linear grammar with one non-terminal symbol only. Let m be the maximal length of the right-hand side of a rule: $m = \max(\{\, w \mid S \to w \in P \,\})$. Then, the word $ab^m a$ cannot be derived in one step. Hence, there is a derivation $S \Longrightarrow ab^p S \Longrightarrow^* ab^m a$ for some number p with $0 \leq p \leq m-2$. But then, also the derivation $S \Longrightarrow ab^p S \Longrightarrow ab^p ab^p S \Longrightarrow^* ab^p ab^m a$ is possible which yields a word which does not belong to the language L_7. Due to this contradiction, we obtain that $L_7 \notin RL_1^V$. □

Lemma 2.8 *The language $L_8 = \{\, a^{3m} \mid m \geq 1 \,\}$ belongs to $RL_1^V \setminus SLT$.*

Proof. The language L_8 is generated by the right-linear grammar $G = (\{S\}, \{a\}, \{S \to a^3 S,\ S \to a^3\}, S)$. Hence, $L_8 \in RL_1^V$.

Assume that the language L_8 is generated by a strictly locally k-testable grammar for some number $k \geq 1$. Then, L_8 has a representation as $[B, I, E, F]$ with $B \cup I \cup\ E \subseteq \{a\}^k$ and $F \subseteq \{a\}^{\leq k-1}$. Since the word a^{3k} belongs to the language L_8, we obtain that B, I, and E contain the word a^k. But then, also the word a^{3k+1} belongs to the language L_8 which is a contradiction. □

Lemma 2.9 *For each natural number $n \geq 1$, let $V_n = \{a_1, a_2, \ldots, a_{n+1}\}$ be an alphabet with $n+1$ pairwise different letters and let $L_{9,n} = \{a_1\}^+\{a_2\}^+ \cdots \{a_{n+1}\}^+$. Then, for $n \geq 1$, the language $L_{9,n}$ belongs to the set $SLT_2 \setminus RL_n^V$.*

Proof. The language $L_{9,n}$ can be represented as

$$L_{9,n} = [\{a_1a_1, a_1a_2\}, \{\, a_pa_p \mid 1 \leq p \leq n+1 \,\} \cup \{\, a_pa_{p+1} \mid 1 \leq p \leq n \,\}, \{a_na_{n+1}, a_{n+1}a_{n+1}\}, \emptyset].$$

Hence, $L_{9,n} \in SLT_2$ for $n \geq 1$.

For generating a language $L_{9,n}$ for some number $n \geq 1$, at least a non-terminal symbol is necessary for every letter a_i $(1 \leq i \leq n+1)$. Hence, $L_{9,n} \notin RL_n^V$. □

We now prove inclusion relations and incomparabilities.

Lemma 2.10 *The class SLT_1 is properly included in the class REG_5^Z.*

Proof. We first prove the inclusion $SLT_1 \subseteq REG_5^Z$.

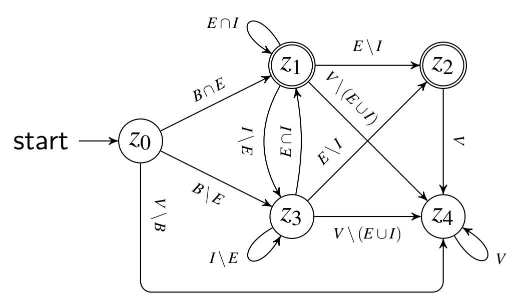
Let L be a strictly locally 1-testable language. Then $L = [B, I, E, F]$ with $B \subseteq V$, $I \subseteq V$, $E \subseteq V$, and $F \subseteq \{\lambda\}$. We construct the following deterministic finite automaton:

$$A = (V, \{z_0, z_1, \ldots, z_4\}, z_0, Z_\mathrm{f}, \delta)$$

where

$$Z_\mathrm{f} = \{z_1, z_2\} \cup \begin{cases} \{z_0\}, & \text{if } \lambda \in F, \\ \emptyset, & \text{otherwise,} \end{cases}$$

and the transition function δ is given by the following diagram (z_0 is an accepting state if and only if $\lambda \in F$):



Due to space reasons, we leave the proof that $L(A) = L$ to the reader. From the construction follows the inclusion $SLT_1 \subseteq REG_5^Z$.

A witness language for the properness of this inclusion is the language $L_1 = \{a\}^*\{b\}\{a,b\}^*$ from Lemma 2.1. □

Lemma 2.11 *The class SLT_1 is incomparable to the classes REG_i^Z for $i \in \{2,3,4\}$.*

Proof. Due to the inclusion relations, it suffices to show that there is a language in the set $REG_2^Z \setminus SLT_1$ and a language in the set $SLT_1 \setminus REG_4^Z$. A language for the first case is $L_1 = \{a\}^*\{b\}\{a,b\}^*$ as shown in Lemma 2.1. A language for the second case is $L_2 = [\{a,b\}, \{b,c\}, \{a,c\}, \emptyset]$ as shown in Lemma 2.2. □

Lemma 2.12 *The classes SLT_k for $k \geq 2$ and SLT are incomparable to the classes REG_n^Z for $n \geq 2$.*

Proof. Due to the inclusion relations, it suffices to show that there is a language in the set $REG_2^Z \setminus SLT$ and a language in each set $SLT_2 \setminus REG_n^Z$ for $n \geq 2$. A language for the first case is $L_1 = \{a\}^*\{b\}\{a,b\}^*$ as shown in Lemma 2.1. Languages for the second case are $L_{3,n} = \{a_1 a_2 \ldots a_{n-1}\}$ as shown in Lemma 2.3. □

Lemma 2.13 *The classes SLT_k for $k \geq 1$ are incomparable to the classes RL_n^P for $n \geq 1$.*

Proof. Due to the inclusion relations, it suffices to show that there is a language in the set $RL_1^P \setminus SLT_k$ for every $k \geq 1$ and a language in each set $SLT_1 \setminus RL_n^P$ for $n \geq 1$. Languages for the first case are $L_{4,k} = \{a^k\}$ for $k \geq 1$ as shown in Lemma 2.4. Languages for the second case are $L_{5,n} = \{a_1, a_2, \ldots, a_n\}^*$ as shown in Lemma 2.5. □

Lemma 2.14 *The class SLT_1 is properly included in the class RL_2^V.*

Proof. Let $L = [B, I, E, F]$ be a strictly locally 1-testable language over an alphabet T. We construct a right-linear grammar $G = (\{S, S'\}, T, P, S)$ with the rules

- $S \to w$ for every word $w \in F \cup (B \cap E)$,
- $S \to wS'$ for every word $w \in B$,
- $S' \to wS'$ for every word $w \in I$, and
- $S' \to w$ for every word $w \in E$.

The language $L(G)$ generated is $F \cup (B \cap E) \cup (BI^*E)$ which is L. Hence, $L \in RL_2^V$ and $SLT_1 \subseteq RL_2^V$. A witness language for the properness of the inclusion is $L_6 = \{a\}$ for which was proved in Lemma 2.6 that it belongs to the set RL_1^V and therefore also to RL_2^V but not to SLT_1. □

Lemma 2.15 *The class SLT_1 is incomparable to the class RL_1^V.*

Proof. There is a language in the set $RL_1^V \setminus SLT_1$, namely $L_6 = \{a\}$ as shown in Lemma 2.6, and a language in the set $SLT_1 \setminus RL_1^V$, namely $L_7 = \{a\}\{b\}^*\{a\} \cup \{a\}$ as shown in Lemma 2.7. □

Lemma 2.16 *The classes SLT_k for $k \geq 2$ and SLT are incomparable to the classes RL_n^V for $n \geq 1$.*

Proof. Due to the inclusion relations, it suffices to show that there is a language in the set $RL_1^V \setminus SLT$ and a language in the set $SLT_2 \setminus RL_n^V$ for every number $n \geq 1$. A language for the first case is $L_8 = \{\, a^{3m} \mid m \geq 1 \,\}$ as shown in Lemma 2.8. A language for the second case is $L_{9,n} = \{a_1\}^+\{a_2\}^+ \cdots \{a_{n+1}\}^+$ as shown in Lemma 2.9. □

A summary of the inclusion relations is given in Figure 1. An edge label in this figure refers to the paper or lemma above where the respective inclusion is proved.

Theorem 2.17 *The inclusion relations presented in Figure 1 hold. An arrow from an entry X to an entry Y depicts the proper inclusion $X \subset Y$; if two families are not connected by a directed path, then they are incomparable.*

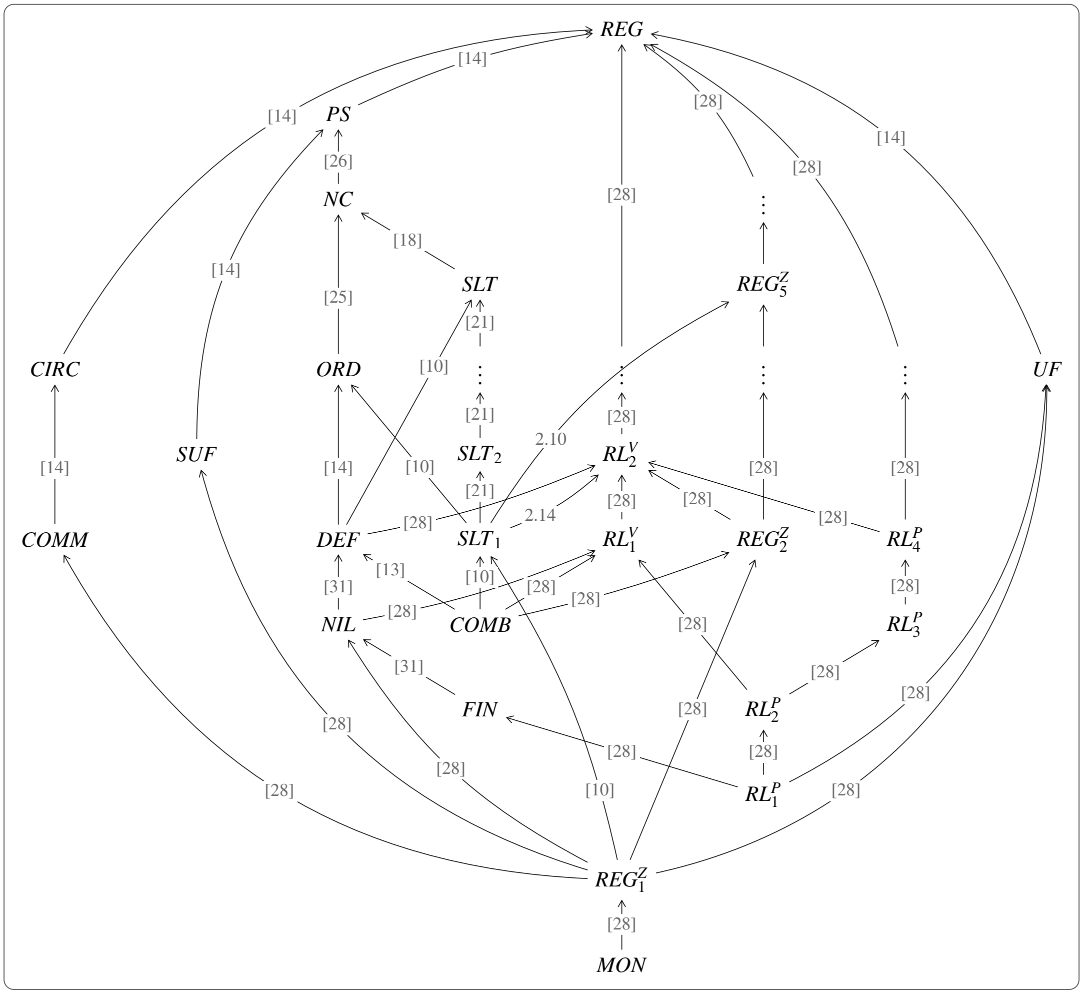


Figure 1: Hierarchy of subregular language families

2.5 Tree-controlled grammars

A tree-controlled grammar is a quintuple $G = (N,T,P,S,R)$ where

- (N,T,P,S) is a context-free grammar with a set N of non-terminal symbols, a set T of terminal symbols, a set P of context-free non-erasing rules (with the only exception that the rule $S \to \lambda$ is allowed if S does not occur on a right-hand side of a rule), and an axiom S,
- R is a regular set over $N \cup T$.

The language $L(G)$ generated by a tree-controlled grammar $G = (N,T,P,S,R)$ consists of all such words $z \in T^*$ which have a derivation tree t where z is the word obtained by reading the leaves from left to right and the words of all levels of t – besides the last one – belong to the regular control language R.

Let $\mathscr{F}$ be a subfamily of *REG*. Then, we denote the family of languages generated by tree-controlled grammars $G = (N,T,P,S,R)$ with $R \in \mathscr{F}$ by $\mathscr{TC}(\mathscr{F})$.

Example 2.18 *As an example, we consider the tree-controlled grammar*

$$G_1 = (\{S\},\{a\},\{S \to SS, S \to a\},S,\{S\}^*).$$

Since the terminal symbol a is not allowed to appear before the last level, on all levels before, any occurrence of S is replaced by SS. Finally, any letter S is replaced by a. Therefore, the levels of an allowed derivation tree consist of the words S, SS, SSSS, …, S^{2^n}, a^{2^n} for some $n \geq 0$. Thus, $L(G_1) = \{\, a^{2^n} \mid n \geq 0 \,\}$. Due to the structure of the control language which is monoidal and can be generated by a grammar with one non-terminal symbol and two rules, we further obtain

$$L(G_1) \in \mathscr{TC}(MON) \cap \mathscr{TC}(RL_1^V) \cap \mathscr{TC}(RL_2^P).$$

Example 2.19 *We now consider the tree-controlled grammar*

$$G_2 = (\{S, A, B, C\}, \{a, b, c\}, P, S, \{S, aAbBcC\})$$

with

$$P = \{S \to aAbBcC, A \to aA, B \to bB, C \to cC, A \to a, B \to b, C \to c\}.$$

By the definition of the control language, any derivation in G_2 has the form

$$S \Longrightarrow aAbBcC \Longrightarrow aaAbbBccC \Longrightarrow \ldots \Longrightarrow a^{n-1}Ab^{n-1}Bc^{n-1}C \Longrightarrow a^n b^n c^n$$

with $n \geq 2$. Thus, the tree-controlled grammar G_2 generates the non-context-free language

$$L(G_2) = \{a^n b^n c^n\} n \geq 2.$$

Due to the structure of the control language which is finite and can be generated by a grammar with one non-terminal symbol and two rules, we further obtain

$$L(G_2) \in \mathscr{TC}(FIN) \cap \mathscr{TC}(RL_1^V) \cap \mathscr{TC}(RL_2^P).$$

In [20] (see also [5]), it has been shown that a language L is generated by a tree-controlled grammar if and only if it is generated by a context-sensitive grammar.

Theorem 2.20 ([20], [5]) *It holds $\mathscr{TC}(REG) = CS$.*

In subsequent papers, tree-controlled grammars have been investigated where the control language belongs to some subfamily of the class *REG* ([6, 7, 8, 9, 27, 29, 30]). In this paper, we continue this research with further subregular language families.

From the definition follows that the subset relation is preserved under the use of tree-controlled grammars: if we allow more, we do not obtain less.

Lemma 2.21 *For any two language classes X and Y with $X \subseteq Y$, we have the inclusion*

$$\mathscr{TC}(X) \subseteq \mathscr{TC}(Y).$$

A summary of the inclusion relations known so far is given in Figure 2. An arrow from an entry X to an entry Y depicts the inclusion $X \subseteq Y$; a solid arrow means proper inclusion; a dashed arrow indicates that it is not known whether the inclusion is proper. If two families are not connected by a directed path, then they are not necessarily incomparable. An edge label in this figure refers to the paper where the respective inclusion is proved.

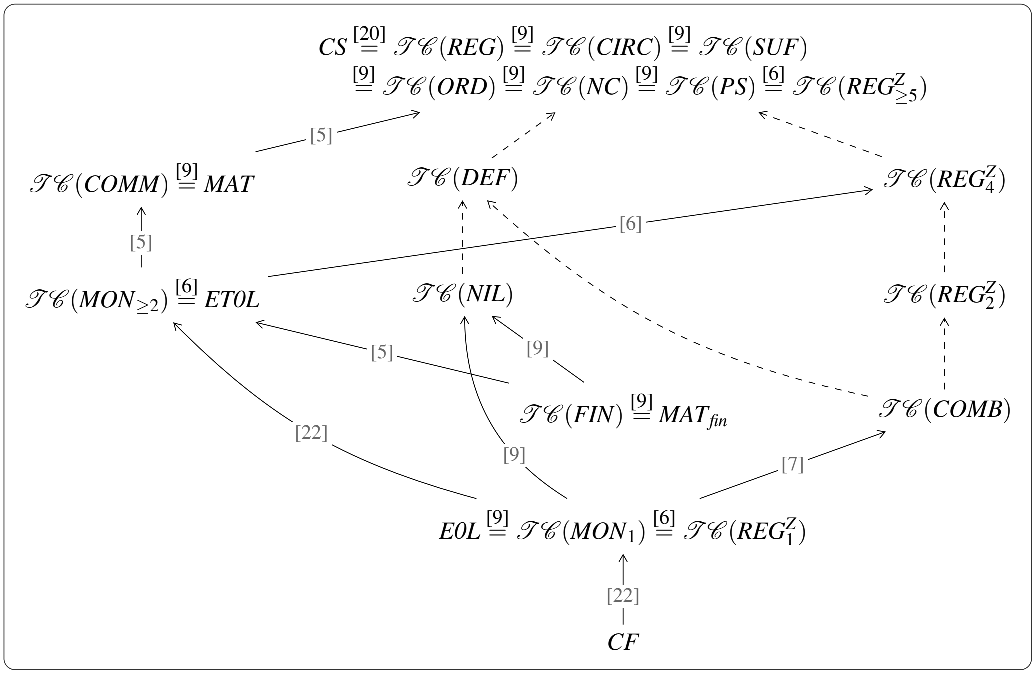


Figure 2: Hierarchy of subregularly tree-controlled language families

3 Results

We insert the classes $\mathscr{TC}(SLT_k)$ for $k \geq 1$, $\mathscr{TC}(SLT)$, $\mathscr{TC}(RL^V_n)$ for $n \geq 1$, and $\mathscr{TC}(RL^P_n)$ for $n \geq 1$ into the existing hierachy (see Figure 2).

The inclusions follow from the inclusion relations of the respective families of the control languages (see Figure 1 and Lemma 2.21).

In most cases, we obtain that any context-sensitive language can be generated by a tree-controlled grammar where the control language is taken from that family.

Theorem 3.1 *We have $\mathscr{TC}(SLT_k) = CS$ for $k \geq 2$ and $\mathscr{TC}(SLT) = CS$.*

Proof. Let L be a context-sensitive language. Then, there is a context-sensitive grammar $G = (N,T,P,S)$ with $L(G) = L$ which is in Kuroda normal form, where the rule set P can be divided into two sets P_1 and P_2 such that all rules of P_1 are of the form $A \to BC$ or $A \to B$ or $A \to a$ with $A,B,C,D \in N$ and $a \in T$ and all rules of P_2 are of the form $AB \to CD$ with $A,B,C,D \in N$.

We will construct a tree-controlled grammar G_{tc} which simulates the grammar G. Since G_{tc} has only context-free rules, the non-context-free rules of G have to be substituted by context-free rules and some control such that the parts of a non-context-free rule which are independent from the view of the core grammar of G_{tc} remain connected.

We label the non-context-free rules and associate the non-terminal symbols of their left-hand sides with new non-terminal symbols which are marked with the rule label and the position (first or second letter). The context-free rules can be freely applied also in the tree-controlled grammar. A non-context-free rule $p : AB \to CD$ will be simulated by context-free rules

$$A \to A_{p,1},\ B \to B_{p,2},\ A_{p,1} \to C,\ \text{and}\ B_{p,2} \to D.$$

The control language ensures that the rules which belong together (here $A \to A_{p,1}$ and $B \to B_{p,2}$) are applied together (at the same time and next to each other). If a terminal symbol is produced in a sentential form of the grammar G, then it remains there until the whole terminal word is produced. In the tree-controlled grammar G_{tc}, one has to keep track of terminal symbols because they 'disappear' (once produced, they are not present in the next level anymore) and then two non-terminal symbols appear next to each other, although they are not neighbours in the sentential form. So, the tree-controlled grammar should produce placeholders for terminal symbols and replace them by the actual terminal symbols only in the very end. In a tree-controlled grammar, from one level to the next, all non-terminal symbols are replaced. This can be seen as some kind of shortcut where production rules which are independent from each other are applied in parallel.

We construct such a tree-controlled grammar $G_{\mathrm{tc}} = (N_{\mathrm{tc}}, T, P_{\mathrm{tc}}, S, R_{\mathrm{tc}})$. The terminal alphabet and start symbol are the same as in the grammar G. We now give the rules; the non-terminal symbols will be collected later from the rules. At the end, we will give the control language R_{tc}.

In order to simulate the context-free rules directly, we take all non-terminating rules of them from G as they are:

$$P_{\mathrm{cf}} = P \cap (\{\, A \to BC \mid A,B,C \in N \,\} \cup \{\, A \to B \mid A,B \in N \,\}).$$

Instead of the terminating rules, we take rules with a placeholder (for each terminal symbol a, we introduce a unique non-terminal symbol $\hat{a}$), but finally, those placeholders have to be terminated:

$$P_{\mathrm{t}} = \{\, A \to \hat{a} \mid A \in N, a \in T, A \to a \in P \,\} \cup \{\, \hat{a} \to a \mid a \in T \,\}.$$

We give also rules which can delay the derivation such that not everything needs to be replaced in parallel:

$$P_{\mathrm{d}} = \{\, A \to A \mid A \in N \,\} \cup \{\, \hat{a} \to \hat{a} \mid a \in T \,\}.$$

For simulating the non-context-free rules, first rules are applied which mark the position of the intended application such that the control language has the chance to check whether the plan is alright (if it is not, then the derivation will block). In the next step, the markers will be replaced by their actual target non-terminal symbols:

$$P_{\mathrm{cs}} = \bigcup_{p:AB \to CD \in P} \{A \to A_{p,1},\ B \to B_{p,2},\ A_{p,1} \to C,\ B_{p,2} \to D\}.$$

Other rules are not needed, hence,

$$P_{\mathrm{tc}} = P_{\mathrm{cf}} \cup P_{\mathrm{t}} \cup P_{\mathrm{d}} \cup P_{\mathrm{cs}}.$$

The set N_{tc} of non-terminal symbols results as follows:

$$N_{\mathrm{cf}} = N \cup \{\, \hat{a} \mid a \in T \,\},\ N_1 = \{\, A_{p,1} \mid p : AB \to CD \in P \,\},\ N_2 = \{\, B_{p,2} \mid p : AB \to CD \in P \,\},$$
$$N_{12} = \{\, A_{p,1}B_{p,2} \mid p : AB \to CD \in P \,\},\ N_{\mathrm{tc}} = N_{\mathrm{cf}} \cup N_1 \cup N_2.$$

A derivation can go wrong only if the simulation of a non-context-free rule is not properly planned. Hence, as control language, we take

$$R_{\mathrm{tc}} = (N_{\mathrm{cf}} \cup N_{12})^*.$$

Since the context-free rules of the grammar G can be applied independently from each other and do not have to be applied at a certain time (thanks to the rules from the subset P_{d}) and the correct simulation of the non-context-free rules is ensured by the control language R_{tc}, it is not hard to see that the generated languages $L(G)$ and $L(G_{\mathrm{tc}})$ coincide.

The control language R_{tc} is strictly locally 2-testable as can be seen from the following representation: Let

$$B = N_{\mathrm{cf}}^2 \cup N_{\mathrm{cf}}N_1 \cup N_{12}, \qquad I = N_{\mathrm{cf}}^2 \cup N_{\mathrm{cf}}N_1 \cup N_{12} \cup N_2N_{\mathrm{cf}} \cup N_2N_1,$$
$$E = N_{\mathrm{cf}}^2 \cup N_{12} \cup N_2N_{\mathrm{cf}}, \qquad F = N_{\mathrm{cf}} \cup \{\lambda\}.$$

Then $R_{\mathrm{tc}} = [B,I,E,F]$.

Altogether, we obtain $CS \subseteq \mathscr{TC}(SLT_2) \subseteq \mathscr{TC}(SLT_k) \subseteq \mathscr{TC}(SLT) \subseteq CS$ for $k \geq 3$. Thus, it holds $\mathscr{TC}(SLT_k) = CS$ for $k \geq 2$ and $\mathscr{TC}(SLT) = CS$. □

Theorem 3.2 *We have $\mathscr{TC}(RL_n^V) = CS$ for $n \geq 1$.*

Proof. The control language $R_{\mathrm{tc}} = (N_{\mathrm{cf}} \cup N_{12})^*$ from the tree-controlled grammar G_{tc} in the proof of Theorem 3.1 can be generated by a right-linear grammar $G' = (\{S'\}, N_{\mathrm{tc}}, P', S')$ where

$$P' = \{\, S' \to xS' \mid x \in N_{\mathrm{cf}} \cup N_{12} \,\} \cup \{\, S' \to x \mid x \in N_{\mathrm{cf}} \cup N_{12} \,\}.$$

Hence, $CS \subseteq \mathscr{TC}(RL_1^V) \subseteq \mathscr{TC}(RL_n^V) \subseteq CS$ for $n \geq 2$. Thus, we conclude $\mathscr{TC}(RL_n^V) = CS$ for $n \geq 1$. □

From the proof of Theorem 3.1, we conclude also the following statement.

Theorem 3.3 *We have $\mathscr{TC}(UF) = CS$.*

Proof. Let $L = \{w_1, w_2, \ldots, w_n\}$ be a finite language. Then $L^* = (\{w_1\}^*\{w_2\}^* \cdots \{w_n\}^*)^*$ and is therefore union-free.

The control language $R_{\mathrm{tc}} = (N_{\mathrm{cf}} \cup N_{12})^*$ from the tree-controlled grammar G_{tc} in the proof of Theorem 3.1 is the Kleene closure of a finite language and, hence, it is union-free. □

Regarding the classes $\mathscr{TC}(RL_n^P)$ for $n \geq 1$, the situation is different since the number of rules depends on the size of the alphabet (which is not necessarily the case for the number of non-terminal symbols or the number of states).

If the control language is generated with one rule only, then either the control language is the empty set (if the right-hand side of the rule contains a non-terminal symbol) or it contains exactly one terminal word. Since the start symbol of the tree-controlled grammar always forms the first level of the derivation tree, it must be contained in the control language (otherwise, the derivation would be blocked right from the beginning). Therefore, we obtain the following result.

Lemma 3.4 *Let $G = (N,T,P,S,R)$ a tree-controlled grammar with $R \in RL_1^P$. Then, the generated language is*

$$L(G) = \begin{cases} \{\, w \mid w \in T^* \text{ and } S \to w \in P \,\}, & \text{if } R = \{S\}, \\ \emptyset, & \text{otherwise}. \end{cases}$$

Proof. If $R = \{S\}$, then every level but the last one of the derivation tree is S and the last level is a terminal word which is produced by S. On the other hand, all terminal words derived from S belong to the generated language.

If $R \neq \{S\}$, then $S \notin R$ since R contains at most one word because $R \in RL_1^P$. Since S is the word of the first level of the derivation tree, there is no derivation possible. Hence, $L(G)$ is empty. □

From this result, the next one immediately follows.

Theorem 3.5 *We have* $\mathcal{TC}(RL_1^P) = FIN$.

Proof. The inclusion $\mathcal{TC}(RL_1^P) \subseteq FIN$ follows from Lemma 3.4. The inclusion $FIN \subseteq \mathcal{TC}(RL_1^P)$ can also be seen from Lemma 3.4: Let L be a finite language over an alphabet T. Then, construct a tree-controlled grammar $G = (\{S\}, T, \{\, S \to w \mid w \in L \,\}, S, \{S\})$. It holds $L(G) = L$ and $L(G) \in \mathcal{TC}(RL_1^P)$. □

If the control language is taken from the family $\mathcal{TC}(RL_2^P)$, then already context-sensitive languages can be generated as the Examples 2.18 and 2.19 show.

Theorem 3.6 *We have* $\mathcal{TC}(RL_1^P) \subset \mathcal{TC}(RL_2^P)$.

Proof. The inclusion follows from Theorem 2.17 and Lemma 2.21. According to Theorem 3.5, the family $\mathcal{TC}(RL_1^P)$ contains finite languages only. As shown in the Examples 2.18 and 2.19, the family $\mathcal{TC}(RL_2^P)$ contains non-context-free languages. □

A summary of all the inclusion relations is given in Figure 3. An arrow from an entry X to an entry Y depicts the inclusion $X \subseteq Y$; a solid arrow means proper inclusion; a dashed arrow indicates that it is not known whether the inclusion is proper. If two families are not connected by a directed path, then they are not necessarily incomparable. An edge label in this figure refers to the paper or theorem above where the respective inclusion is proved.

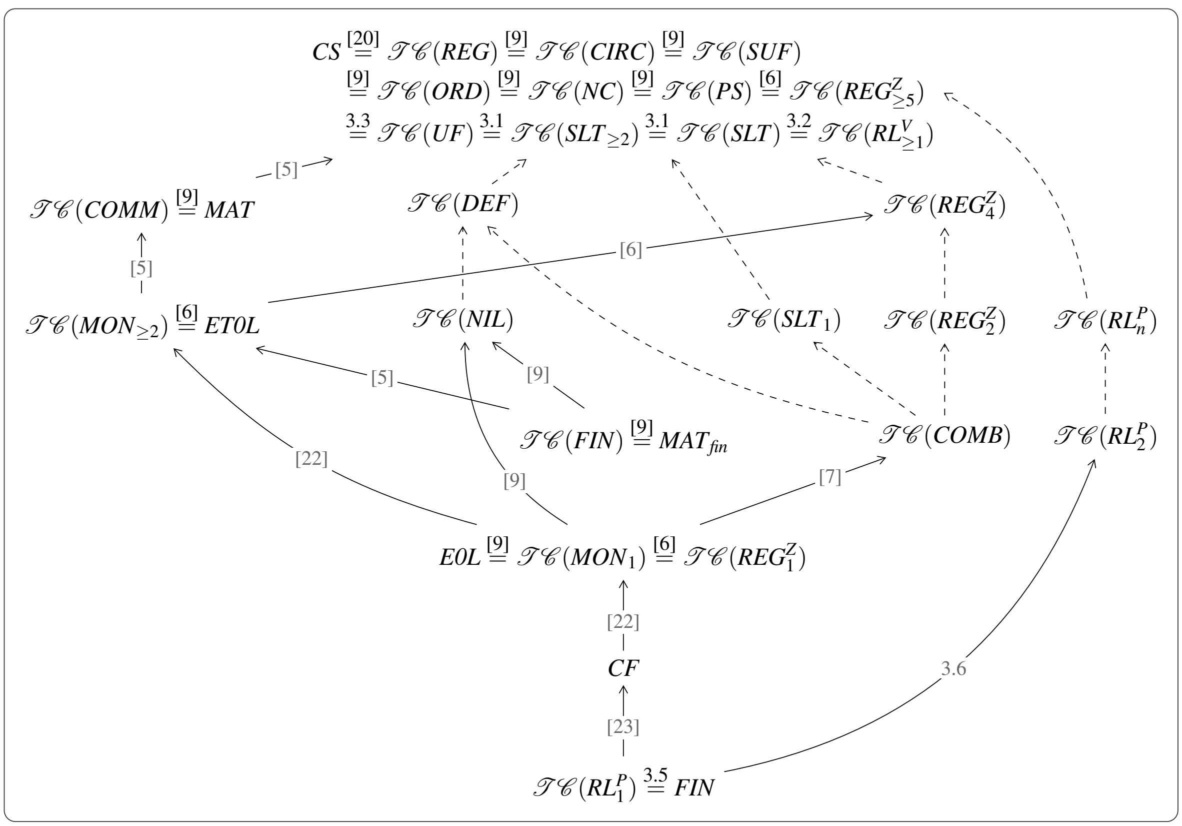


Figure 3: New Hierarchy of subregularly tree-controlled language families

4 Conclusion

There are several families of languages generated by tree-controlled grammars where we do not have a characterization by some other language class. The strictness of some inclusions and the incomparability of some families remain as open problems.

In the present paper, we have only considered tree-controlled grammars without erasing rules. For tree-controlled grammars where erasing rules are allowed, several results have been published already (see, e. g., [7, 29, 30]). Also in this situation, there are some open problems.

Another direction for future research is to consider other subregular language families or to relate the families of languages generated by tree-controlled grammars to language families obtained by other grammars/systems with regulated rewriting.

References

[1] Janusz A. Brzozowski (1962): *Regular Expression Techniques for Sequential Circuits*. Ph.D. thesis, Princeton University, Princeton, NJ, USA.

[2] Janusz A. Brzozowski, Galina Jirásková & C. Zou (2014): *Quotient complexity of closed languages*. *Theory of Computing Systems* 54, pp. 277–292, doi:10.1007/s00224-013-9515-7.

[3] Jürgen Dassow (1979): *On the circular closure of languages*. *Elektronische Informationsverarbeitung und Kybernetik/Journal of Information Processing and Cybernetics* 15(1–2), pp. 87–94.

[4] Jürgen Dassow, Florin Manea & Bianca Truthe (2011): *On contextual grammars with subregular selection languages*. In Markus Holzer, Martin Kutrib & Giovanni Pighizzini, editors: *Descriptional Complexity of Formal Systems – 13th International Workshop, DCFS 2011, Gießen/Limburg, Germany, July 25 – 27, 2011. Proceedings*, *LNCS* 6808, Springer-Verlag, pp. 135–146, doi:10.1007/978-3-642-22600-7_11.

[5] Jürgen Dassow & Gheorghe Păun (1989): *Regulated Rewriting in Formal Language Theory*. *EATCS Monographs in Theoretical Computer Science* 18, Springer-Verlag, doi:10.1007/978-3-642-74932-2.

[6] Jürgen Dassow, Ralf Stiebe & Bianca Truthe (2009): *Two collapsing hierarchies of subregularly tree controlled languages*. *Theoretical Computer Science* 410(35), pp. 3261–3271, doi:10.1016/j.tcs.2009.03.005.

[7] Jürgen Dassow, Ralf Stiebe & Bianca Truthe (2010): *Generative capacity of subregularly tree controlled grammars*. *International Journal of Foundations of Computer Science* 21(5), pp. 723–740, doi:10.1142/S0129054110007520.

[8] Jürgen Dassow & Bianca Truthe (2008): *On two hierarchies of subregularly tree controlled languages*. In Cezar Câmpeanu & Giovanni Pighizzini, editors: *Descriptional Complexity of Formal Systems, 10th International Workshop, Charlottetown, Prince Edward Island, Canada, July 16–18, 2008, Proceedings*, University of Prince Edward Island, pp. 145–156.

[9] 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*, Computer and Automation Research Institute, Hungarian Academy of Sciences, pp. 158–169.

[10] Jürgen Dassow & Bianca Truthe (2022): *On the generative capacity of contextual grammars with strictly locally testable selection languages*. In Henning Bordihn, Géza Horváth & György Vaszil, editors: *12th International Workshop on Non-Classical Models of Automata and Applications (NCMA 2022), Debrecen, Hungary, August 26 – 27, 2022. Proceedings*, *EPTCS* 367, Open Publishing Association, pp. 65–80, doi:10.4204/EPTCS.367.5.

[11] Ferenc Gécseg & István Peák (1972): *Algebraic Theory of Automata*. Academiai Kiado, Budapest.

[12] Arthur Gill & Lawrence T. Kou (1974): *Multiple-entry finite automata*. Journal of Computer and System Sciences 9(1), pp. 1–19, doi:10.1016/S0022-0000(74)80034-6.

[13] Ivan M. Havel (1969): *The theory of regular events II*. Kybernetika 5(6), pp. 520–544.

[14] Markus Holzer & Bianca Truthe (2015): *On relations between some subregular language families*. In Rudolf Freund, Markus Holzer, Nelma Moreira & Rogério Reis, editors: Seventh Workshop on Non-Classical Models of Automata and Applications (NCMA), Porto, Portugal, August 31 – September 1, 2015, Proceedings, books@ocg.at 318, Österreichische Computer Gesellschaft, pp. 109–124.

[15] John E. Hopcroft & Jeffrey D. Ullman (1979): *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading.

[16] Karel Čulik II & Hermann A. Maurer (1977): *Tree controlled grammars*. Computing 19(2), pp. 129–139, doi:10.1007/BF02252350.

[17] Sige Yuki Kuroda (1964): *Classes of languages and linear bounded automata*. Information and Control 7(2), pp. 207–223, doi:10.1016/S0019-9958(64)90120-2.

[18] Robert McNaughton & Seymour Papert (1971): *Counter-Free Automata*. MIT Press, Cambridge, USA.

[19] Micha A. Perles, Michael O. Rabin & Eliahu Shamir (1963): *The theory of definite automata*. IEEE Trans. Electronic Computers 12, pp. 233–243, doi:10.1109/PGEC.1963.263534.

[20] Gheorghe Păun (1979): *On the generative capacity of tree controlled grammars*. Computing 21, pp. 213–220, doi:10.1007/BF02253054.

[21] Stefano Crespi Reghizzi & Pierluigi San Pietro (2011): *From regular to strictly locally testable languages*. In Petr Ambrož, Štěpán Holub & Zuzana Masáková, editors: 8th International Conference WORDS 2011, EPTCS 63, pp. 103–111, doi:10.4204/EPTCS.63.14.

[22] Grzegorz Rozenberg & Arto Salomaa (1980): *The Mathematical Theory of L Systems*. Academic Press.

[23] Grzegorz Rozenberg & Arto Salomaa, editors (1997): *Handbook of Formal Languages*. Springer-Verlag, Berlin.

[24] Huei-Jan Shyr (1991): *Free Monoids and Languages*. Hon Min Book Co., Taichung, Taiwan.

[25] Huei-Jan Shyr & Gabriel Thierrin (1974): *Ordered automata and associated languages*. Tankang Journal of Mathematics 5(1), pp. 9–20.

[26] Huei-Jan Shyr & Gabriel Thierrin (1974): *Power-separating regular languages*. Mathematical Systems Theory 8(1), pp. 90–95, doi:10.1007/BF01761710.

[27] Ralf Stiebe (2008): *On the complexity of the control language in tree controlled grammars*. In Jürgen Dassow & Bianca Truthe, editors: Colloquium on the Occasion of the 50th Birthday of Victor Mitrana, Otto-von-Guericke-Universitäat Magdeburg, Germany, June 27, 2008, Proceedings, Otto-von-Guericke-Universitäat Magdeburg, Germany, pp. 29–36.

[28] Bianca Truthe (2018): *Hierarchy of Subregular Language Families*. Technical Report, Justus-Liebig-Universität Giessen, Institut für Informatik, IFIG Research Report 1801.

[29] 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.

[30] György Vaszil (2012): *On the nonterminal complexity of tree controlled grammars*. In Henning Bordihn, Martin Kutrib & Bianca Truthe, editors: Languages Alive – Essays Dedicated to Jürgen Dassow on the Occasion of His 65th Birthday, LNCS 7300, Springer, pp. 265–272, doi:10.1007/978-3-642-31644-9_18.

[31] Barbara Wiedemann (1978): *Vergleich der Leistungsfähigkeit endlicher determinierter Automaten*. Diplomarbeit, Universität Rostock.