Theoretical and Applied Informatics
ISSN 1896–5334
Vol.23 (2011), no. 3-4
pp. 213–228
DOI: 10.2478/v10179-011-0015-7

Parsing Based on n-Path Tree-Controlled Grammars

MARTIN ČERMÁK, JIŘÍ KOUTNÝ, ALEXANDER MEDUNA

Formal Model Research Group
Department of Information Systems
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
icermak, ikoutny, meduna@fit.vutbr.cz

Received 15 November 2011, Revised 1 December 2011, Accepted 4 December 2011

Abstract: This paper discusses recently introduced kind of linguistically motivated restriction placed on tree-controlled grammars—context-free grammars with some root-to-leaf paths in their derivation trees restricted by a control language. We deal with restrictions placed on $n \geq 1$ paths controlled by a deterministic context–free language, and we recall several basic properties of such a rewriting system. Then, we study the possibilities of corresponding parsing methods working in polynomial time and demonstrate that some non-context-free languages can be generated by this regulated rewriting model. Furthermore, we illustrate the syntax analysis of *LL* grammars with controlled paths. Finally, we briefly discuss how to base parsing methods on bottom-up syntax–analysis.

Keywords: regulated rewriting, derivation tree, tree-controlled grammars, path-controlled grammars, parsing, n-path tree-controlled grammars

1. Introduction

The investigation of context-free grammars with restricted derivation trees represents an important trend in today's formal language theory (see [3], [6], [10], [11], [12], [13] and [15]). In essence, these grammars generate their languages just like ordinary context-free grammars do, but their derivation trees have to satisfy some simple prescribed conditions. The present paper continues with the investigation of these grammars.

The idea of restrictions placed on the derivation trees of context-free grammars is introduced in [3], and the resulting grammars restricted in this way are referred to as *tree-controlled grammars*. In essence, the notion of a tree-controlled grammar is defined as follows: take a context-free grammar G and a regular language R. A string w generated by G belongs to the language defined by G and R only if there is derivation tree t for w in G such that all *levels* (except the last one) of t are described by R.

Next, we give an overview of the basic results concerning tree-controlled grammars.

Based on the original definition of a tree-controlled grammar, the modifications, where many well-known types of both controlled grammars and control languages are considered, are studied in [15].

Since tree-controlled grammars generate the class of languages equal to the class of recursively enumerable languages, the question arises whether it is possible to achieve generative power of tree-controlled grammars when the levels in their derivation trees are restricted by sub-regular control languages. This problem is studied in [6], where many types of sub-regular languages are considered.

A new type of restriction in a derivation are studied in [12], where a context-free grammar G and a context-free language R are considered. A string w generated by G belongs to the language defined by G and R only if there is derivation tree t for w in G such that there exists a path of t described by R. Based on this restriction, the authors of [12] introduce *path-controlled grammars* and study several properties of this model. As it is stated in the final remarks, there are many open problems. Some of them are studied in [13] but still many modifications remain unsolved. Based on the non-emptiness problem for the intersection of two languages, in [13] the polynomial recognizability is stated. However, in [13], parsing methods based on the languages generated by path-controlled grammars is not solved at all.

As a generalization of path-controlled grammars, [10] introduces several variants of this rewriting model where not just one, but given number of paths in derivation trees of context-free grammars have to be described by a control language. That is, [10] deals with a generalization of path-controlled grammars (G, R), where a string w generated by G belongs to the language defined by G and R only if there is derivation tree t for w in G such that there exist given n paths of t described by a linear language R, where $n \geq 1$. Such a rewriting system is referred to as n-path tree-controlled grammar. As it is demonstrated in Section 4, this generalized rewriting system can generate some languages not captured in the rewriting system introduced in [12]. Then, based on the common part of all restricted paths of the derivation trees of context-free grammars, [10] introduces several modifications and state pumping properties of some of them. The possibilities of parsing methods working in polynomial time for n-path tree-controlled languages are briefly studied in [9].

The motivation to study the path-based restrictions in more detail is not purely theoretical. Indeed, the motivation is based on the observation that many of the languages commonly used in the practise, including programming and natural languages, are not context-free (see [7], [8], [17], [18], and [20]). Parsing based upon non-context-free grammars represents an alternative approach to capturing context-sensitive dependencies, which are normally handled within the semantic analysis. Simply put, the paper shows that some common tasks performed within semantic analysis can be passed to syntax analysis and, thereby, reduce the work executed by semantic analysis if it is desirable. Indeed, parsers are typically generated automatically based on the grammar's specification and thus, semantic actions that allow to analyse non-context-free languages have to be added manually. For this reason, it is more convenient to capture context-sensitive dependencies already in the grammar's specification. Moreover, strict definition of the restrictions placed on a grammar instead of semantic actions allows stating several fundamental language-family-characterizing properties and their formal proving.

Generating non-context-free languages by more powerful grammars (i.e. context-sensitive grammars where the left hand sides of rewriting rules contain one or more nonterminals) actually leads to several fundamental problems that make their practical usage problematic—namely, it is hard to describe the derivation by graph structure, many problems are undecidable, etc. There are other approaches for extension of generative power of context-free grammars which preserve context-free nature of production rules. However, they achieve higher generative power by constraints on the derivation process (i.e. matrix grammars where in a derivation step a fixed number of context-free rules are required to be applied in a given order—this provides synchronization among different parts of a sentential form and many non-context-free languages can be generated in this way).

Thus, our goal is a formalism that

- generates non-context-free languages (namely those used in linguistics, e.g. $a^k b^k c^k$, $a^k b^l a^k b^l$, for $k, l \geq 1$, etc.) and
- is practically usable (like context-free grammars for which sophisticated parsing methods working in polynomial time has been developed).

Thus, in this paper, we study parsing methods for n-path tree-controlled grammars.

In short, the derivations in context-free grammars can be seen in the notions of graph theory–that is, by derivation trees. From this viewpoint, the derivation starts in the root of a tree (which corresponds to the starting symbol of a grammar), continues by sequential application of context-free rules, and ends in leafs of a tree (which corresponds to the terminals of a grammar) without considering any context. From this rough idea, it can be easily seen that during the derivation in a context-free grammar, the paths of the corresponding derivation tree are formed. The paths of the derivation tree have several well-known properties (see [1])–namely, all paths start in the common node (the root) and each of them ends in different node (i.e. leafs). Considering two different paths, there is a node in which both paths split. Taking this into account, we can manage some context information during the derivations in context-free grammars. More precisely, the common part of all restricted paths can form the original part of each individual path. In this way, a context information can be distributed through different branches of the derivation tree. Roughly speaking, we can say that the branches of the derivation tree communicate through their common nodes. However, the derivation tree is constructed as in the context-free case – that is, each branch of a tree can be constructed individually regardless of the other branches.

Section 4 demonstrates context information obtained by the restriction placed on the paths increases the generative power of this regulated rewriting system such that some of non-context-free languages used in the linguistics can be specified. Thus we use context-free grammar with its well-known parsing methods and we restrict some root-to-leaf paths in corresponding derivation trees. As a result, we can generate some of typically non-context-free languages.

The idea of recognition of the strings generated by path-controlled grammars is introduced in [12], where it is demonstrated that path-controlled languages (with just one restricted path) can be recognized in polynomial time. More detailed discussion concerning polynomial recognition of path-controlled languages can be found in [13] where the membership problem is reduced to the non-emptiness problem of the intersection of two languages. In the paper it is demonstrated that membership problem for path-controlled languages is decidable in polynomial time.

However, for the vast majority of practical applications, it is essential to find not only whether or not given string belongs to the language of given grammar, but also how given grammar can generate it – that is, which rules the grammar has to use and in which order. Moreover, the method presented in [13] cannot be straightforwardly modified for the model with $n \geq 2$ controlled paths since it would lead to the question if such intersection contains at least n elements which is much more difficult problem than non-emptiness; however, the problem for $n = 1$ controlled paths corresponds to the method introduced in [13].

Thus, in this paper, after recalling the restrictions placed on $n \geq 1$ paths in the derivation trees of context-free grammars, we introduce LL (see [19]) restriction and we present the ideas how to parse generated language family (not only how to decide membership problem). Essentially, we discuss the following problem: Can we decide whether a string is recognized by n-path tree-controlled grammar in polynomial time, for $n \geq 1$?

In Section 2, we introduce the terminology needed. In Section 3, based on path-controlled grammars, we recall the generalization of path-controlled grammars–n-path tree-controlled grammars. As the main result of this paper, Section 5 discusses whether or not it is possible to parse n-path tree-controlled languages in polynomial time. In the conclusion, we formulate some open problems in the investigation of grammars with restricted paths and the parsing methods of the languages they generate.

2. Preliminaries

This paper assumes that the reader is familiar with the graph theory (see [1]) and the theory of formal languages (see [14]), including the theory of regulated rewriting (see [5]). In this section, we introduce the terminology and the definitions needed in the sequel.

For an alphabet V, V^* denotes the letter monoid (generated by V under the operation concatenation), ε is the unit of V^*, and $V^+ = V^* - \{\varepsilon\}$. For string $x \in V^*$, $|x|$ denotes the length of x and $reversal(x)$ the mirror image of x. Let $x, y \in V^*$; x is *substring* of y if there are two strings $z, z' \in V^*$ with $zxz' = y$. Every subset $L \subseteq V^*$ is a *language* over V.

A *context-free grammar* is a quadruple $G = (V, T, P, S)$ where V is a total alphabet, $T \subseteq V$ is a terminal alphabet, P is a finite set of rules of the form $p : A \rightarrow x$ where p is unique label of a rule, $A \in V - T$, $x \in V^*$, and $S \in V - T$ is the starting symbol. Let $N = V - T$ denote the set of nonterminals in G. A grammar $G = (V, T, P, S)$ is *linear* if and only if all its rules have at most one nonterminal on the right-hand side.

A derivation step in G is defined for $u, v \in V^*$ and $p : A \rightarrow x \in P$ as $uAv \Rightarrow uxv\,[p]$. In the standard manner, we introduce the relations $\Rightarrow^i$, $\Rightarrow^+$, and $\Rightarrow^*$ (see [14]). The language of context-free grammar and linear grammar, G, is called *context-free language* and *linear language*, respectively, and it is defined as $L(G) = \{x \in T^* : S \Rightarrow^* x\}$. The family of linear languages and context-free languages is denoted by **LIN** and **CF**, respectively.

Let $G = (V, T, P, S)$ be a context-free grammar. If there exists $x \in L(G)$ with more than one derivation tree, then G is *ambiguous*; otherwise, G is *unambiguous*. If for all $x \in L(G)$ there is at most m derivation trees, then G is m-ambiguous, for some $m \geq 1$. A context-free language L is *inherently ambiguous* if L is generated by no unambiguous grammar.

Let $G = (V, T, P, S)$ be a context-free grammar and $x \in T^*$. A sequence of nodes obtained by concatenating the labels of all leafs of a derivation tree is called *frontier*. Let ${}_G\triangle(x)$ denote the set of the derivation trees with frontier x in G. Let $t \in {}_G\triangle(x)$. A *path* of t is any sequence of the nodes with the first node equals to the root of t, the last node equals to a leaf of t, and there is an edge in t between each two consecutive nodes of the sequence. Let s be any sequence of the nodes of t. Then $word(s)$ denotes the string obtained by concatenation of all labels of the nodes of s in order from left to right.

A *pushdown automaton* is a septuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$, where Q is a finite set of states, Σ is an input alphabet, $q_0 \in Q$ is the initial state, Γ is a pushdown alphabet, δ is a finite set of rules of the form $Zqa \rightarrow \gamma p$, where $p, q \in Q$, $Z \in \Gamma$, $a \in \Sigma \cup \{\varepsilon\}$, $\gamma \in \Gamma^*$ and $Z_0 \in \Gamma$ is the initial pushdown symbol.

A configuration of M is any string from $\Gamma^*Q\Sigma^*$. For any configuration $xAqay$, where $x \in \Gamma^*$, $y \in \Sigma^*$, $q \in Q$ and any $Aqa \rightarrow \gamma p \in \delta$, M makes a move from configuration $xAqay$ to configuration $x\gamma py$ according to $Aqa \rightarrow \gamma p$, written as $xAqay \vdash x\gamma py\,[Aqa \rightarrow \gamma p]$, or, simply, $xAqay \vdash x\gamma py$. Let $\vdash^*$ and $\vdash^+$ denote transitive-reflexive and transitive closure of $\vdash$, respectively. The language of M is defined as $L(M) = \{w \in \Sigma^* : Z_0q_0w \vdash^* f$ and $f \in Q\}$.

3. Definitions

In this section, as the subject of investigation in this paper, we recall the derivation-based generalization of tree-controlled grammars based on n-path restriction.

A *tree-controlled* grammar, TC grammar for short, is a pair (G, R), where $G = (V, T, P, S)$ is a context-free grammar and $R \subseteq V^*$ is a control language over V. The *language that* (G, R) *generates under the* n*-path control by* R, $n \geq 1$, is denoted by ${}_{n-path}L(G, R)$ and it is defined by the following equivalence:

For all $x \in T^*$, $x \in_{n-path} L(G, R)$ if and only if there exists derivation tree $t \in {}_G\triangle(x)$ such that there is a set Q_t of n paths of t such that for each $p \in Q_t$, $word(p) \in R$. Set **n-path-TC** $= \{{}_{n-path}L(G, R) | (G, R)$ is a TC grammar$\}$. Hereafter, TC grammars that generate the language under the n-path control are referred to as n-path TC grammars.

For each context-free grammar G, there is a regular language which describes all paths in a derivation tree of a string w in G (see Prop. 1 in [12]). Since for each context-free grammar G, there is no regular control language that increases the generative power of G with R controling the paths (see Prop 1. and Prop. 2 in [12]), we investigate TC grammar with non-regular control language. On the other hand, as it is demonstrated in Section 4, linear language used to controll the paths in the derivation trees of context-free grammars is strong-enough mechanism to increase the generative power of context-free grammars. Thus, we study n-path TC grammars with linear control languages in the rest of this paper. Therefore, in what it follows, for a TC grammar (G, R), we consider R as a linear language.

4. Examples

In this section, we demonstrate two non-context-free languages that belong to **n-path-TC**. The languages are first introduced for selected $n \geq 1$, and after that, they are presented in a general case. The specific examples for higher values of n tends to be excessively long and they are left to the reader.

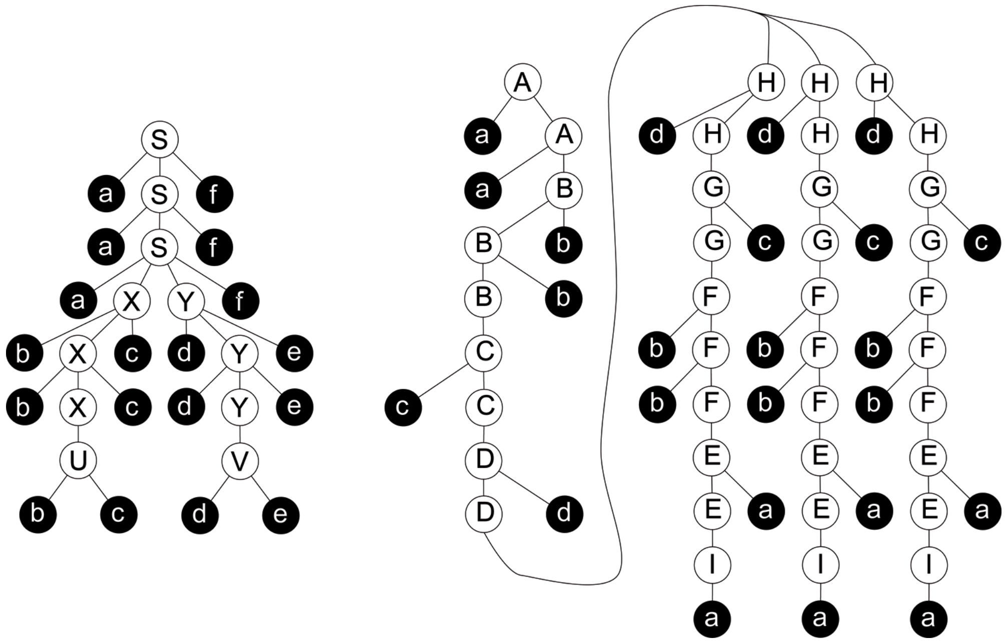

Fig. 1: Illustration of the derivations of $a^3b^3c^3d^3e^3f^3$ with two paths of the form $S^iX^iUb \cup S^iY^iVd$, where $i \geq 1$, in Example 1 (left) and $(a^2cdb^2)^4$ with three paths of the form $A^rB^sC^tD^uH^uG^tF^sE^rIa$, where $r, s, t, u \geq 1$, in Example 3 (right).

Example 1 Consider the TC grammar that generates $n - pathL(G, R)$ and $n = 2$, where

$$\begin{aligned}
&G = (\{S, X, Y, U, V, a, b, c, d, e, f\}, \{a, b, c, d, e, f\}, P, S),\\
&P = \{S \to aSf, \quad S \to aXYf, \quad X \to bXc, \quad Y \to dYe,\\
&\qquad X \to U, \qquad U \to bc, \qquad Y \to V, \qquad V \to de\},\\
&R = \{S^iX^iUb \cup S^iY^iVd \mid i \geq 1\},\\
&{}_{2-path}L(G, R) = \{a^jb^jc^jd^je^jf^j \mid j \geq 1\}.
\end{aligned}$$

Clearly, ${}_{2-path}L(G, R) \notin$ **CF**. The left-hand part of Figure 1 illustrates the derivation tree for the derivation $S \Rightarrow^* a^3b^3c^3c^3d^3e^3f^3$ in (G, R). Clearly, there are two paths described by the strings S^3X^3Ub and S^3Y^3Vd from R.

Example 2 Let (G, R) be a TC grammar that generates ${}_{n-path}L(G, R)$, $n \geq 1$, where

$$\begin{aligned}
G = (&\{S\} \cup \{A_i, B_i \mid 1 \leq i \leq n\} \cup \{a_i \mid 1 \leq i \leq 2n+2\},\\
&\{a_i \mid 1 \leq i \leq 2n+2\}, P, S),\\
P =&\{S \to a_1Sa_{2n+2}, \qquad S \to a_1A_1A_2 \ldots A_na_{2n+2}\} \cup\\
&\{A_{i+1} \to a_{2i+2}A_{i+1}a_{2i+3}, \quad A_{i+1} \to B_{i+1},
\end{aligned}$$

$$\begin{aligned}&\quad B_{i+1} \rightarrow a_{2i+2}a_{2i+3} |\ 0 \leq i \leq n-1\},\\ &R = \bigcup_{i=1}^{n}\{S^k A_i^k B_i a_{2i} |\ k \geq 1\}.\end{aligned}$$

Clearly, $R \in \mathbf{LIN}$. Consider a derivation in (G, R):

$$\begin{aligned}S &\Rightarrow^k a_1^k S a_{2n+2}^k\\ &\Rightarrow a_1^k a_1 A_1 A_2 \ldots A_n a_{2n+2} a_{2n+2}^k\\ &\Rightarrow^{n\cdot k} a_1^{k+1} a_2^k B_1 a_3^k \ldots a_{2n}^k B_n a_{2n+1}^k a_{2n+2}^{k+1}\\ &\Rightarrow^n a_1^{k+1} a_2^{k+1} a_3^{k+1} \ldots a_{2n}^{k+1} a_{2n+1}^{k+1} a_{2n+2}^{k+1}\end{aligned}$$

Clearly, $n \geq 1$ paths are described by R and in this way, (G, R) generates ${}_{n-path}L(G, R) = \{a_1^k \ldots a_{2n+2}^k |\ k \geq 1\} \notin \mathbf{CF}$.

Example 3 Consider the TC grammar (G, R) with ${}_{n-path}L(G, R)$, for $n = 3$, where

$$\begin{aligned}G &= (\{A, B, C, D, E, F, G, H, I, a, b, c, d\}, \{a, b, c, d\}, P, A),\\ P &= \{A \rightarrow aA,\quad A \rightarrow aB,\quad B \rightarrow Bb,\quad B \rightarrow C,\\ &\qquad C \rightarrow cC,\quad C \rightarrow D,\quad D \rightarrow Dd,\quad D \rightarrow HHH,\\ &\qquad E \rightarrow Ea,\quad E \rightarrow I,\quad F \rightarrow bF,\quad F \rightarrow E,\\ &\qquad G \rightarrow Gc,\quad G \rightarrow F,\quad H \rightarrow dH,\quad H \rightarrow G,\quad I \rightarrow a\},\\ R &= \{A^r B^s C^t D^u H^u G^t F^s E^r I a |\ r, s, t, u \geq 0\},\\ {}_{3-path}L(G, R) &= \{(a^r c^t d^u b^s)^4 |\ r > 0, s, t, u \geq 0\}.\end{aligned}$$

Clearly, ${}_{3-path}L(G, R) \notin \mathbf{CF}$. The right-hand part of Figure 1 illustrates the derivation tree for the derivation $S \Rightarrow^* (a^2 c d b^2)^4$ in (G, R). Clearly, there are three paths described by $A^2 B^3 C^2 D^2 H^2 G^2 F^3 E^2 I a$ from R.

Example 4 Let $m \geq 0$ with even m. Let (G, R) be a TC grammar that generates ${}_{n-path}L(G, R)$, $n \geq 1$, where

$$\begin{aligned}G &= (\{A_j, B_j, a_j |\ 1 \leq j \leq m\} \cup \{C\}, \{a_j |\ 1 \leq j \leq m\}, P, A_1),\\ P &= \{A_1 \rightarrow a_1 A_1,\quad A_1 \rightarrow a_1 A_2,\quad B_1 \rightarrow B_1 a_1,\quad B_1 \rightarrow C,\quad C \rightarrow a_1\} \cup\\ &\qquad \{A_m \rightarrow A_m a_m, A_m \rightarrow \{B_m\}^n\} \cup\\ &\qquad \{A_i \rightarrow A_i a_i,\quad A_i \rightarrow A_{i+1} |\ 2 \leq i \leq m-1 \text{ with even } i\} \cup\\ &\qquad \{A_i \rightarrow a_i A_i,\quad A_i \rightarrow A_{i+1} |\ 3 \leq i \leq m-1 \text{ with odd } i\} \cup\\ &\qquad \{B_i \rightarrow a_i B_i,\quad B_i \rightarrow B_{i-1} |\ 2 \leq i \leq m \text{ with even } i\} \cup\\ &\qquad \{B_i \rightarrow B_i a_i,\quad B_i \rightarrow B_{i-1} |\ 3 \leq i \leq m \text{ with odd } i\},\\ R &= \{A_1^{k_1} A_2^{k_2} \ldots A_m^{k_m} B_m^{k_m} B_{m-1}^{k_{m-1}} \ldots B_2^{k_2} B_1^{k_1} C a_1 |\ k_i \geq 0, 1 \leq i \leq m\}.\end{aligned}$$

Clearly, $R \in \mathbf{LIN}$. Observe that $n \geq 1$ paths are described by R and

$$\begin{aligned}{}_{n-path}L(G, R) = \{&(a_1^{k_1+1} a_3^{k_3} \ldots a_{m-1}^{k_{m-1}} a_m^m a_{m-2}^{k_{m-2}} a_{m-4}^{k_{m-4}} \ldots a_2^{k_2})^{n+1}\\ &|\ k_i \geq 0, 1 \leq i \leq m\} \notin \mathbf{CF}.\end{aligned}$$

The details of a derivation in this general case is left to the reader.

5. Syntax analysis of n-path-TC

As it is demonstrated above, some typical non-context-free languages belong to **n-path-TC**. Thus it is natural and indisputably important for practical use to study the parsing methods possibilities for this language family. The first and most important requirement on parsing is polynomial parsability.

Theorem 1 For TC grammar (G, R) where G is unambiguous context-free grammar and R is linear control language, the membership $x \in_{n-path} L(G, R)$, $n \geq 1$, is decidable in $O(|x|^k)$, for some $k \geq 0$.

Proof. For some $n \geq 1$, let ${}_{n-path}L(G, R)$ be the language of a TC grammar (G, R) in which G is unambiguous. We assume R is generated by some unambiguous linear grammar. Since G is unambiguous, it is well-known that we can decide whether $x \in L(G)$, or not, in $O(|x|^2)$–that is, we distinguish two cases: (1) $x \notin L(G)$ and (2) $x \in L(G)$.

- Clearly, if $x \notin L(G)$, then $x \notin_{n-path} L(G, R)$.
- If $x \in L(G)$, then, since G is unambiguous, we can construct unique derivation tree t of $x \in L(G)$ in $O(|x|^2)$. Since each path of t ends in a leaf, t contains $|x|$ paths. Clearly, the height of t is polynomially bounded by some $l \geq 2$ with respect to $|x|$. Thus, for any $x \in L(G)$, the length of each path p of t and therefore also $|word(p)|$ are bounded by l. Because R is unambiguous and $|word(p)| \leq l$ for each $p \in Q_t$, it is well-known that we can decide if $word(p) \in L(R)$ in polynomial time. If for at least n paths, $p_1, p_2, \ldots, p_n$, of derivation tree of x in G, $word(p_i) \in R$ holds for $i \in 1, 2, \ldots, n$, then $x \in_{n-path} L(G, R)$. Hence, the membership problem is decidable in polynomial time.

As straightforwardly follows from above, the proposed way to solve membership problem leads to a parsing method working, in essence, in two phases:

1. construction of derivation tree t of x in G by top-down parsing method,
2. checking that at least $n \geq 1$ paths of t belong to R.

From the practical viewpoint, the situation may occur in which during the phase 1 above we already know that currently constructed derivation tree cannot contain the required number of paths described by the strings from R—informally, we do not have to wait with starting the phase 2 until the phase 1 is completely done (until t is completely constructed).

5.1. Top-Down Parsing of n-path-TC

For some $n \geq 1$, let ${}_{n-path}L(G, R)$ be the language of a TC grammar (G, R) where $G = (V, T, P, S)$ is an unambiguous context-free grammar. We assume that R is generated by unambiguous context–free grammar $G_R = (V_R, V, P_R, S_R)$. Adjust the idea behind Theorem 1 as follows.

We construct labelled derivation tree with the set of labels $\Psi = \{0, 1\}$ and the following semantics. Let p be a path of derivation tree t in G and e be an edge between any two consecutive nodes of p. Then, label $0 \in \Psi$ of any e of p represents p is not described by R—that is, $word(p) \notin R$. Label $1 \in \Psi$ of all e of p represents p can potentially be described by R.

Consider that for the decision if $x \in L(G)$, we use well-known top-down parsing method to construct derivation tree t of x in G—that is, started from S, we construct derivation tree t according to the rules of G such that the frontier of t is equal to x (for more details, see [14]).

Let us suppose that a rule $r : A \rightarrow A_1A_2 \dots A_j \in P$, $j \geq 1$, is used in the derivation step $X \Rightarrow Y$, $X, Y \in V^*$ in G. In addition, we need to determine the value of the labels of the edges between A and each A_j, for $j = 1, 2, \dots, n$, related to the application of r. Let t' be a derivation tree that corresponds to the derivation $S \Rightarrow^* w_1A_1A_2 \dots A_jw_2$ in G, for some $w_1, w_2 \in V^*$. Essentially, t' is a sub-tree of t. Clearly, each path of t' is the beginning part of at least one path in t. Next, we distinguish the following cases:

- If all the edges of t' are labelled, we can proceed to next derivation step in G.
- If some of the edges in t' are not labelled, we need to compute the values of missing labels.

For each unlabelled edge e of path p' in t', we check whether G_R can generate the string of the form $word(p')w$ with $w \in (V_R - V)^*T \cup \{\varepsilon\}$. Since $|word(p')|$ is finite, we can check it in polynomial time. If so, we add label $1 \in \Psi$ to edge e; otherwise, we add label $0 \in \Psi$ to edge e. Note that this phase can be optimized in such a way that we do the test whether G_R can generate $word(p')w$ with $w \in (V_R - V)^*T \cup \{\varepsilon\}$ symbol-by-symbol during the generation of $word(p')$ in G_R. The details of this optimization represents a straightforward task which is left to the reader.

Next, we distinguish the following cases:

- If t' contains no leaf with input edge labelled by 1, then $x \notin {}_{n-path}L(G, R)$.
- If t' contains at least one leaf labelled by symbol of $V - T$, we proceed to next derivation step in G.
- If all the leafs of t' are labelled by the symbols of T and for at least n of the leafs of t, there is an edge labelled by 1, then $x \in_{n-path} L(G, R)$.

	a	b	c	d	e	k	\$
S	1	1			1		
A	2	3			4		
B		5				6	

Table 1. LL table for grammar G from Example 5.

Thus, it is possible to check whether the paths of derivation tree t of LL context-free grammar can potentially be described by given unambiguous context–free language already during the building of t by LL parser.

The following example explains the syntax analysis in more detail.

Example 5 Consider ${}_{n-path}L(G, R)$ where $G = (\{S,\ A,\ B,\ a,\ b,\ c,\ d,\ e,\ k\}, \{a, b, c, d, e, k\},\ P,\ S)$ with $P = \{1 : S \rightarrow AA,\ 2 : A \rightarrow aAd,\ 3 : A \rightarrow bBc, 4 : A \rightarrow e,\ 5 : B \rightarrow bBc,\ 6 : B \rightarrow k\}$ and $R = \{SA^mB^{m-1}k |\ m \geq 1\}$. Obviously, $L(G) = \{a^i(b^jkc^j + e)d^ia^s(b^tkc^t + e)d^s |\ i, j, t, s \geq 0\}$. Clearly, ${}_{1-path}L(G, R) = L(G)$ with $i = j$ or $s = t$, and ${}_{2-path}L(G, R) = L(G)$ with $i = j = s = t$. The LL table for G is constructed by well-known algorithm (see [19]) and it is presented in Table 1.

The syntax analyser uses two pushdown automata and a list of 3-tuples containing state of the second automaton, contents of its pusdown, and a pointer to the first-automaton's pushdown. The first pushdown automaton simulates the construction of derivation tree by the LL table in the well-known way.

- If the top-most symbol on the pushdown is a non-terminal A, the first input symbol is a, and there is a rule $A \rightarrow x$ on position $[A, a]$ in the LL table, then the automaton rewrites A on the pushdown by $reversal(x)$ (expansion step).
- If a on the pushdown's top is a terminal and a is the first input symbol, the automaton reads a from the input and removes a from the pushdown (comparation step).
- Other cases represent a syntax error.

Let us return to the example and consider input string $abkcdaed$. In the beginning, the top-most symbol on the pushdown of the first automaton is S. Since a is the first input symbol and there is rule 1 on the position $[S, a]$ in the table, the automaton rewrites S by AA on its pushdown.

During this computation, the second automaton is parsing the potentially valid paths in the derivation tree. At the beginning, the list contains one item $(q_0, \widehat{S}, 1)$ where 1 represents the first position on the first-automaton's pushdown from the bottom and $\widehat{S}$ is the start pushdown symbol of the second automaton. If the first automaton makes a computation step with symbol a on the pushdown's top and there is a tuple (q, α, p) in the list where p is the pointer to the symbol, the second automaton places α onto its pushdown, automaton moves to q and it makes steps for a as the first input symbol until it does not need next input symbol. For example, after the expansion from S to AA, the second automaton finds list-item $(q_0, \widehat{S}, 1)$ and it moves to state q_0 and places $\widehat{S}$ onto the pushdown.

Then, it simulates step $\widehat{S}q_0S \vdash \beta q$ where $\widehat{S}q_0S \rightarrow \beta q$ is a transition rule of the second automaton. If $a = A$ was a non-terminal and first automaton made expansion by rule $A \rightarrow x$, then the syntax analyser removes the used list-item and if the second automaton did not reject input, then there are l tuples (q, β, i) inserted into the list, for all $i = t+1, \ldots, t+l$ with $l = |x|$, top-most-symbol-position t before the expansion, and β as the current content of the pushdown. Otherwise, for a as a terminal symbol, the comparison is done. If the second automaton accepts, the pointer of the used tuple is rewrited to 0 where 0 denotes accepted path. Assuming that it does not accept with a terminal a, the tuple is removed from the list.

Consider input string $abkcdaed$ again. For some definition of the second automaton, the syntax analysis proceeds as it is given in Table 2.

As it can be seen, only one item with 0 in the last component remains in the list—that is, there is one path belonging to the control language. If we require that $n \geq 2$ paths are described by the control language, the input string is not accepted.

5.2. Bottom-Up Parsing of n-path-TC

Section 5.1 deals in principle with top-down parsing method (LL parser) and its weakness is the assumption that a context-free grammar is unambiguous. Moreover, the method demonstrated in Example 5 assumes that the grammar is LL. Essentially, the same idea is applicable also on bottom-up parsing methods which can handle a larger range of the languages. Therefore, we briefly discuss the ideas of parsing methods for **n-path-TC** in terms of LR parsing.

One of the advantages of bottom-up parsers (e.g. LR parser) is that we do not need to require that in TC grammar (G, R), G is LL grammar. On the other hand, concerning the bottom-up parsing, we have to deal with the ambiguity. However, it is well-known that the question whether a context-free grammar is or is not ambiguous is undecidable, since this problem can be reduced to the Post Correspondence Problem which is undecidable (see [16]).

It is also well-known that for some ambiguous context-free grammars, there exists equivalent context-free grammar which is unambiguous. The ambiguity of a context-free grammar can be restricted basically by removing the chain rules (i.e. rules of the form $A \rightarrow B$, $A, B \in V - T$). We assume that a context-free grammar contains only usable rules–that is, only those rules, which can be used during the derivation. Clearly, if

1. PDA	2.PDA	Pointer	Tuples
$Sqabkcdaed$	q_0	$(q_0, \varepsilon, 1)$	$(q_0, \varepsilon, 1)$
$AAqabkcdaed$	q_0	$(q_0, \varepsilon, 1)$	$(q_1, \varepsilon, 1)$, $(q_1, \varepsilon, 2)$
$AdAaqabkcdaed$	q_1	$(q_1, \varepsilon, 2)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, A, 3)$, $(q_1, A, 4)$
$AdAqbkcdaed$	Aq_1	$(q_1, A, 4)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, A, 3)$
$AdcBbqbkcdaed$	Aq_1	$(q_1, A, 3)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, AA, 3)$, $(q_1, AA, 4)$, $(q_1, AA, 5)$
$AdcBqkcdaed$	AAq_1	$(q_1, AA, 5)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, AA, 3)$, $(q_1, AA, 4)$
$Adckqkcdaed$	AAq_1	$(q_1, AA, 4)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, AA, 3)$, $(q_1, A, 4)$
$Adcqcdaed$	Aq_1	$(q_1, A, 4)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, AA, 3)$, $(q_1, A, 0)$
$Adqdaed$	AAq_1	$(q_1, AA, 3)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 2)$, $(q_1, A, 0)$
$Aqaed$	Aq_1	$(q_1, A, 2)$	$(q_1, \varepsilon, 1)$, $(q_1, A, 0)$
$dAaqaed$	q_1	$(q_1, \varepsilon, 1)$	$(q_1, A, 0)$, $(q_1, A, 1)$, $(q_1, A, 2)$, $(q_1, A, 3)$
$dAqed$	Aq_1	$(q_1, A, 3)$	$(q_1, A, 0)$, $(q_1, A, 1)$, $(q_1, A, 2)$
$deqed$	Aq_1	$(q_1, A, 2)$	$(q_1, A, 0)$, $(q_1, A, 1)$, $(q_1, AA, 2)$
dqd	AAq_1	$(q_1, AA, 2)$	$(q_1, A, 0)$, $(q_1, A, 1)$
q	Aq_1	$(q_1, A, 1)$	$(q_1, A, 0)$

Table 2. Parsing of $abkcdaed$ corresponding to TC grammar (G, R) from Example 5.

$G = (V, T, P, S)$ is a context-free grammar with $r : A \rightarrow A \in P$, for some $A \in V - T$, then G is ambiguous since r can be used during the derivation of $x \in L(G)$ arbitrarily many times and thus generate arbitrarily many different derivation trees for x in G.

Obviously, since the chain rules generate nothing, they can be removed from a context-free grammar G without affecting $L(G)$. However, removing the chain rules from G in a TC grammar (G, R) affects the paths in the derivation trees of $x \in L(G)$. Thus the identity ${}_{n-path}L(G, R) ={}_{n-path}L(G', R)$, where G' is obtained by removing the chain rules from G, does not hold; however, the equivalence $L(G) = L(G')$ holds.

Theorem 2 For a TC grammar (G, R), where G is a context-free grammar and $R \in$ **LIN**, there is TC grammar (G', R') such that G' does not contain chain rules and ${}_{n-path}L(G, R) ={}_{n-path}L(G', R')$, $n \geq 1$.

Proof. For some $n \geq 1$, let ${}_{n-path}L(G, R)$ be the language of a TC grammar (G, R) where $G = (V, T, P, S)$. Let G' be a context-free grammar obtained from G by removing the chain rules. Therefore, G' can be constructed by well-known algorithm in polynomial time (see 5.1.3.3 in [14]). We get $G' = (V, T, P', S)$ such that for all $x \in L(G')$, there is no derivation in G' of the form $B \Rightarrow^* A$, for some $A, B \in V - T$.

The paths in the derivation trees of G' are described by the strings of the form $(V - T)^*T$. Basically, we need to read such a strings and remove such symbols $A \in (V - T)$ which corresponds to the application of $B \rightarrow A \in P$ in G. This is done by gsm mapping M (see [5] for the definition) such that M reads the strings s of the form $(V - T)^*T$ and nondeterministically removes or lets unchanged each symbol $A \in (V - T)$ with $B \rightarrow A \in P$ and BA is substring of s. Since **LIN** is closed under gsm mappings (see [4]), also $M(R) \in$ **LIN**. This way, we get $M(R)$ such that $M(R) \neq R$. However, ${}_{n-path}L(G, R) ={}_{n-path}L(G', M(R))$.

Consider TC grammar (G, R) and let (G', R') be constructed as described above. Clearly, G' does not need to be unambiguous since the chain rules are not the only cause of the ambiguity. Consider, however, any $x \in_{n-path} L(G', R')$. Obviously, there is a derivation tree t of x in G'. Since there are no chain rules in G' and for each $x \in L(G')$, $|x|$ is finite, the height of t with respect to $|x|$ is bounded by $log\,|x|/log\,2$. Thus there is at most m, for some $m \geq 1$, derivation trees of x in G' and G is m-ambiguous.

Theorem 3 For TC grammar (G, R) where G is m-ambiguous LR grammar, $m \geq 1$, and an unambiguous language $R \in$ **LIN**, the membership $x \in_{n-path} L(G, R)$, $n \geq 1$, is decidable in $O(|x|^k)$, for some $k \geq 0$.

Proof. For some $n \geq 1$, let ${}_{n-path}L(G, R)$ be the language of a TC grammar (G, R) in which G is m-ambiguous, for some $m \geq 1$. Thus, if $x \in L(G)$, then we can construct at most m derivation trees of $x \in L(G)$ in $O(m.|x|^2)$ by LR parser. Then, if for at least j paths, $p_1, p_2, \ldots, p_j$, of at least one derivation tree of x in G it holds that $word(p_i) \in R$ for $i \in 1, 2, \ldots, j$, then $x \in_{n-path} L(G, R)$.

Hence, the syntax analysis of ${}_{n-path}TC$ with LR grammar G and unambiguous linear control language can be done in polynomial time also in the case of LR parsing.

6. Conclusion

In this concluding section, we summarize the achieved results and point out some important open questions.

As a generalization of TC grammars that generate the language under path-based control introduced in [12], we have considered TC grammars that generate their languages under n-path control by linear language which were introduced in [10].

We have demonstrated that for $L \in$ **n-path-TC** under assumption that L is generated by TC grammar (G, R) in which G and R are unambiguous and, furthermore, G is restricted to be LL grammar, there is parsing method working in polynomial time.

This method can check whether or not the paths of the derivation tree t of $x \in L(G)$ belongs to control language R in the time of building of t. Moreover, when we consider LR parser for $L \in$ **n-path-TC** under assumption that L is generated by TC grammar (G, R) in which G has bounded ambiguity (i.e. G is unambiguous or m-ambiguous) and unambiguous language $R \in \mathbf{LIN}$, there is also a parsing method working in polynomial time.

However, the open question is whether there is polynomial time parsing method

- if G is not LL,
- if G is ambiguous.

It is also of interest to quantify the worst case of the parsing complexity more precisely.

The open investigation area is represented by the transformation of n-path TC grammars into some normal forms based on Chomsky normal form of underlying context-free grammar which would lead to possibility to use parsing methods based on transformation to Chomsky normal form.

Concerning the parsing methods for TC grammars (and also similar rewriting system), there is great possibility use n-Accepting Restricted Pushdown Automata Systems, which deals with n-tuples of strings. These systems are introduced in [2].

Acknowledgments

This work was supported by the research plan MSM0021630528, FIT-S-11-2 BUT FIT grant, MŠMT grant MEB041003, GAČR grant GD102/09/H042, and EU CZ 1.05/1.1.00/02.0070 grant.

References

1. A. Bondy: *Graph Theory*. Springer, 2010.
2. M. Cermák, A. Meduna: *N-accepting restricted pushdown automata systems*. In 13th International Conference on Automata and Formal Languages, pages 168- 183. Computer and Automation Research Institute, Hungarian Academy of Sci-ences, 2011.
3. K. Culik, H. A. Maurer: *Tree controlled grammars*, Computing, 19:129-139, 1977.
4. J. Dassow, Gh. Păun, and A. Salomaa: *Grammars with controlled derivations*. In G. Rozenberg and A. Salomaa, editors, Handbook of Formal Languages, Volume II, pages 101-154. Berlin: Springer, 1997.
5. J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory*. aun. Springer, Berlin, 1989.

6. J. Dassow, B. Truthe: *Subregularly tree controlled grammars and languages*. In Automata and Fromal Languages – 12th International Conference AFL 2008, Balatonfured, pages 158-169. Computer and Automation Research Institute of the Hungarian Academy of Sciences, 2008.

7. J. Higginbotham: *English is not a context-free language*. Linguistic Inquiry, 15(2):225-234, 1984.

8. A. K. Joshi: *Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions*. Technical Report MS-CIS-85-23, University of Pennsylvania.

9. J. Koutný: *Syntax analysis of tree-controlled languages*. In Proceedings of the 17th Conference and Competition STUDENT EEICT 2011 Volume 3, page 5. Faculty of Information Technology BUT, 2011.

10. J. Koutný, Z. Kˇrivka, A. Meduna: *Pumping properties of path-restricted tree-controlled languages*. In 7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, pages 61-69. Brno University of Technology, 2011.

11. J. Koutný, A. Meduna: *Tree-controlled grammars with restrictions placed upon cuts and paths*. Kybernetika, 2011, in press.

12. S. Marcus, C. Martín-Vide, V. Mitrana, Gh. Păun: *A new-old class of lin aun. guistically motivated regulated grammars*. In Walter Daelemans, Khalil Sima'an, Jorn Veenstra, Jakub Zavrel (Eds.): Computational Linguistics in the Netherlands 2000, Selected Papers from the Eleventh CLIN Meeting, Tilburg, pages 111-125. Language and Computers -Studies in Practical Linguistics 37 Rodopi 2000, 2000.

13. C. Martín-Vide, V. Mitrana: *Further properties of path-controlled grammars*. In Proceedings of FG-MoL 2005: The 10th Conference on Formal Grammar and The 9th Meeting on Mathematics of Language, pages 221-232. University of Edinburgh, Edinburgh, 2005.

14. A. Meduna: *Automata and Languages: Theory and Applications*. Springer, 2005.

15. Gh. Păun: *On the generative capacity of tree controlled grammars*. Computing,aun. 21(3):213-220, 1979.

16. E. L. Post: *A variant of a recursively unsolvable problem*. Bulletion of the American Mathematical Society, 52:264-268, 1946.

17. G. K. Pullum: *On two recent attempts to show that english is not a CFL*. Compu-tational Linguistics, 10(3-4):182-186, 1984.

18. G. K. Pullum, G. Gazdar: *Natural languages and context-free languages*. Linguistics and Philosophy, 4(4):471-504, 1982.

19. Rosenkrantz and Stearns: *Properties of deterministic top down grammars*. In STOC: ACM Symposium on Theory of Computing (STOC), 1969.

20. S. Shieber: *Evidence against the context-freeness of natural language*. Linguistics and Philosophy, 8:333-343, 1985.