TWO-SIDED PUSHDOWN AUTOMATA OVER FREE GROUPS

Ing. Radek BIDLO, Doctoral Degree Programme (2)
Dept. of Information Systems, FIT, BUT
E-mail: bidlor@fit.vutbr.cz

Ing. Petr BLATNÝ, Doctoral Degree Programme (2)
Dept. of Information Systems, FIT, BUT
E-mail: blatny@fit.vutbr.cz

Supervised by: Prof. Alexander Meduna

ABSTRACT

In the two-sided pushdown automata discussed in this paper, their two-sided pushdowns are introduced over free groups rather than free monoids. It is demonstrated that these automata with pushdowns introduced in this way characterize the family of recursively enumerable languages.

1 INTRODUCTION

Standardly, in the case of common pushdown automata, their pushdowns are introduced over free monoids generated by the pushdown alphabets under the operation of concatenation. In some studies, however, these automata are defined and investigated with various modifications (see [4], [6]). The present paper represents another study of this kind.

Indeed, in this paper, we introduce the two-sided pushdown automata over free groups with the two-sided pushdowns defined over free groups rather than free monoids. In this way, we significantly increase the power of pushdown automata, because they characterize the family of recursively enumerable languages.

2 DEFINITIONS

This paper assumes that the reader is familiar with the language theory and algebra (see [1], [2], and [3]). Next, this section recalls only the notions used in this paper.

For an alphabet, V, V^* represents the free monoid generated by V under the operation of concatenation. Furthermore, V° represents the free group generated by V under the operation of concatenation. The unit of V° is denoted by ε. For every string, $w \in V^\circ$, there is the *inverse string of* w, denoted by $\overline{w}$, with the property that $w\overline{w} = \overline{w}w = \varepsilon$. For $w \in V^*$ or $w \in V^\circ$, $|w|$ denotes the length of w.

The inverse string of $w = a_1a_2 \dots a_n$, where $a_i \in V$, $i = 1,2,\dots,n$, $n > 0$, is defined as $\overline{w} = \overline{a_n}\overline{a_{n-1}} \dots \overline{a_1}$. And for $w = \varepsilon$ is $\overline{w} = \varepsilon$. Moreover, the reverse of w is defined as $rev(w) = a_na_{n-1} \dots a_1$. The string is said to be *reduced*, if it contains no pairs of the form $x\bar{x}$ or $\bar{x}x$, where $x, \bar{x} \in V^\circ$.

Let $w = uxyv \in V^\circ$ is a string, where $x,y,u,v \in V^\circ$ and $x = \bar{y}$. To express that x and y are mutually inverse and can be erased, we underline xy in $u\underline{xy}v$.

A queue grammar (see [5]) is a sextuple, $Q = (V,T,W,F,s,P)$, where V and W are alphabets satisfying $V \cap W = \emptyset$, $T \subseteq V$, $F \subseteq W$, $s \in (V-T)(W-F)$, and $P \subseteq V \times (W - F) \times V^* \times W$ is a finite relation such that for every $a \in V$, there exists an element $(a,b,x,c) \in P$. If $u,v \in V^*W$, $u = arb$, $v = rxc$, $a \in V$, $r,x \in V^*$, $b,c \in W$, and $(a,b,x,c) \in P$, then $u \Rightarrow v[(a,b,x,c)]$ in Q or, simply, $u \Rightarrow v$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of Q, $L(Q)$, is defined as $L(Q) = \{w : s \Rightarrow^* wf, w \in T^*, f \in F\}$.

A left-extended queue grammar (see [4]) is similar to an ordinary queue grammar except that it records the members of V used when it works. Formally, a left-extended queue grammar is a sextuple, $Q = (V,T,W,F,s,P)$, where V,T,W,F, and s have the same meaning as in a queue grammar. $P \subseteq V \times (W-F) \times V^* \times W$ is a finite relation (as opposed to an ordinary queue grammar, this definition does not require that for every $a \in V$, there exists an element $(a,b,x,c) \in P$). Furthermore, assume that $\# \notin V \cup W$. If $u,v \in V^*\{\#\}V^*W$ so that $u = w\#arb$, $v = wa\#rxc$, $a \in V$, $r,x,w \in V^*$, $b,c \in W$, and $(a,b,x,c) \in P$, then $u \Rightarrow v[(a,b,x,c)]$ in Q or, simply, $u \Rightarrow v$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of Q, $L(Q)$, is defined as $L(Q) = \{v : \#s \Rightarrow^* w\#vf$ for some $w \in V^*$, $v \in T^*$ and $f \in F\}$.

A string-reading two-sided pushdown automaton over a free group, a **SR2S**$^\circ$ pushdown automaton for short, is an 8-tuple, $M = (Q,\Sigma,\Gamma,R,z,Z_1,Z_2,F)$, where Q is a finite set of states, Σ is an input alphabet, Γ is a pushdown alphabet, $Q \cap (\Sigma \cup \Gamma) = \emptyset$, R is a finite set of rules of the form $u_1|u_2qw \rightarrow v_1|v_2p$ with $u_1,u_2 \in \Gamma$, $v_1,v_2 \in \Gamma^\circ$, $p,q \in Q$, and $w \in \Sigma^*$, $z \in Q$ is the start state, $Z_1 \in \Gamma$ is the start symbol of the front side, $Z_2 \in \Gamma$ is the start symbol of the rear side, and $F \subseteq Q$ is a set of final states. A configuration of M is any string of the form vqy, where $v \in \Gamma^\circ$, $y \in \Sigma^*$, and $q \in Q$. If $u_1|u_2qw \rightarrow v_1|v_2p \in R$, $y = u_1hu_2qwz$, and $x = v_1hv_2pz$, where $u_1,u_2 \in \Gamma$, $h,v_1,v_2 \in \Gamma^\circ$, $q,p \in Q$, and $w,z \in \Sigma^*$, then M makes a move from y to x in M, symbolically written as $y\,{}^\circ\!\Rightarrow x[u_1|u_2qw \rightarrow v_1|v_2q]$ or, simply, $y\,{}^\circ\!\Rightarrow x$. In the standard manner, extend ${}^\circ\!\Rightarrow$ to ${}^\circ\!\Rightarrow^n$, where $n \geq 0$; based on ${}^\circ\!\Rightarrow^n$, define ${}^\circ\!\Rightarrow^+$ and ${}^\circ\!\Rightarrow^*$. We call $Z_1Z_2zw\,{}^\circ\!\Rightarrow^* vqx$ a computation, where $v \in \Gamma^\circ$, $q \in Q$, $w,x \in \Sigma^*$; a computation of the form $Z_1Z_2zw\,{}^\circ\!\Rightarrow^* \varepsilon f$ with $f \in F$ is a successful computation. The language of M, $L(M)^\circ$, is defined as $L(M)^\circ = \{w : Z_1Z_2zw\,{}^\circ\!\Rightarrow^* \varepsilon f$, where $f \in F, w \in \Sigma^*\}$.

A two-sided pushdown automaton over a free group, a **2S**$^\circ$ pushdown automaton for short, is a string-reading two-sided pushdown automaton over a free group, $M = (Q,\Sigma,\Gamma,R,z,Z_1,\ Z_2,F)$, in which every $u_1|u_2qw \rightarrow v_1|v_2p \in R$ satisfies $|w| \leq 1$, where $u_1,u_2 \in \Gamma$, $v_1v_2 \in \Gamma^\circ$, $q,p \in Q$, and $w \in \Sigma^*$.

3 RESULTS

Lemma 3.1 For every recursively enumerable language, L, there exists a left-extended queue grammar, $G = (V,T,W,F,s,P)$, such that $L(G) = L$ and every $(A,q,x,p) \in P$ satisfies $A \in (V-T)$, $q \in (W-F)$, and $x \in ((V-T)^* \cup T^*)$. Formal proof is described in [4].

Corollary 3.1 Let $G = (V,T,W,F,Sq_0,P)$ be a left-extended queue grammar satisfying the properties given in Lemma 1. Grammar G generates every $w \in L(G)$ in this way

$$
\begin{array}{lll}
 & \#Sq_0 & \\
\Rightarrow & x_1\#y_1q_1 & [p_1] \\
\Rightarrow & x_2\#y_2q_2 & [p_2] \\
 & \vdots & \\
\Rightarrow & x_k\#y_kq_k & [p_k] \\
\Rightarrow & x_{k+1}\#y_{k+1}z_1q_{k+1} & [p_{k+1}] \\
 & \vdots & \\
\Rightarrow & x_{k+j-1}\#y_{k+j-1}z_{j-1}q_{k+j-1} & [p_{k+j-1}] \\
\Rightarrow & x_{k+j}\#y_{k+j}z_jq_{k+j} & [p_{k+j}] \\
\Rightarrow & x_{k+j}y_{k+j}\#z_{j+1}q_{k+j+1} & [p_{k+j+1}]
\end{array}
$$

where $x_1,\dots,x_{k+j} \in (V-T)^*$, $y_1,\dots,y_{k+j-1} \in (V-T)^*$, $y_{k+j} \in (V-T)$, $z_1,\dots,z_{j+1} \in T^*$, $z_{j+1} = w$, $q_1,\dots,q_{k+j} \in (W-F)$, $q_{k+j+1} \in F$. $p_1,\dots,p_k$ are of the form (A,q,x,p), where $A \in (V-T)$, $p,q \in (W-F)$ and $x \in (V-T)^*$. $p_{k+1},\dots,p_{k+j}$ are of the form (A,q,y,p), where $A \in (V-T)$, $p,q \in (W-F)$ and $y \in T^*$. The last used production, p_{k+j+1}, is of the form (A,p,y,t), where $A \in (V-T)$, $p \in (W-F)$, $y \in T^*$ and $t \in F$.

Theorem 3.1 For every left-extended queue grammar, $G = (V,T,W,F,Sq_0,P)$, satisfying the properties described in Lemma 1, there exists a string-reading two-sided pushdown automaton over a free group, $M = (Q,T,Z,R,z,Z_L,Z_R,F_M)$, such that $L(G) = L(M)$.

Proof 3.1 We construct a string-reading two-sided pushdown automaton over a free group as follows (we will describe only the construction).

$$Q = \{f,z\} \cup \{\langle q,1\rangle, \langle q,2\rangle | q \in W\}$$

$$Z = \{Z_L, Z_R, \overline{Z_L}, \overline{Z_R}\} \cup (V-T) \cup \overline{N}, \text{ where } \overline{N} = \{\overline{x} | x \in (V-T)\}$$

$$F_M = \{f\}$$

The set of rules, R, is constructed in the following way.

1) for the start axiom of G, Sq_0, where $S \in (V-T)$, $q_0 \in (W-F)$,
add $Z_L|Z_Rz \rightarrow Z_L|SZ_R\langle q_0,1\rangle$ to R

2) for every $(A,q,x,p) \in P$, where $A \in (V-T)$, $p,q \in (W-F)$, $x \in (V-T)^*$,
add $Z_L|Z_R\langle q,1\rangle \rightarrow Z_L\overline{A}|xZ_R\langle p,1\rangle$ to R

3) for every $q \in W$
add $Z_L|Z_R\langle q,1\rangle \rightarrow Z_L|Z_R\langle q,2\rangle$ to R

4) for every $(A,q,y,p) \in P$, where $A \in (V-T)$, $p,q \in (W-F)$, $y \in T^*$,
add $Z_L|Z_R\langle q,2\rangle y \rightarrow Z_L\overline{A}|Z_R\langle p,2\rangle$ to R

5) for every $(A,q,y,t) \in P$, where $A \in (V-T)$, $q \in (W-F)$, $t \in F$, $y \in T^*$
add $Z_L|Z_R\langle q,2\rangle y \rightarrow \overline{A}|\varepsilon f$ to R

The construction is completed. If $\langle q,1\rangle$ is the actual state of M, we say that M is in *nonterminal-generating* mode. Similarly, if $\langle q,2\rangle$ is the actual state of M, we say that M is in *terminal-reading* mode, where $q \in W$. The formal proof exceeds the allowed number of pages, so the formal proof is left to the reader.

4 CONCLUSIONS

The paper has presented a new type of pushdown automaton. By modifying the pushdown to the two-sided version and by defining it over free group rather that free monoid, we significantly increase its power. These automata define the family of recursively enumerable languages.

The next investigation of these automata will target the reduction of the pushdown alphabet. Results of it can be a topic of some next paper.

ACKNOWLEDGEMENTS

The authors wish to thank Prof. Alexander Meduna for useful discussions during preparation of this paper.

REFERENCES

[1] Salomaa, A.: Formal Languages, Academic Press, New York, 1973.

[2] Jacobson, N.: Basic Algebra, 2nd ed., W.H. Freeman, New York, 1989.

[3] Meduna, A.: Automata and Languages: Theory and Applications, Springer, London, 2000.

[4] Meduna, A.: Simultaneously One-Turn Two-Pushdown Automata, in International Journal of Computer Mathematics, No. 82 Taylor & Francis Informa plc, pp. 1-9, 2003.

[5] Kleijn, H. C. M., and Rozenberg, G.: On The Generative Power of Regular Pattern Grammars, in Acta Informatica, 20, pp. 391-411, 1983.

[6] Meduna, A., and Kolar, D.: Regulated Pushdown Automata, in Acta Cybernetica, 14, pp. 653-664, 2000.