STRING-PARTITIONING SYSTEMS

Ing. Zbyněk KŘIVKA, Doctoral Degree Programme (1)
Dept. of Information Systems, FIT, BUT
E-mail: krivka@fit.vutbr.cz

Supervised by: Dr. Alexander Meduna

ABSTRACT

String-partitioning systems, discussed in this contribution, are based by grammatical rules. Each of these rules is a pure context-free rules whose left hand equals to a special symbol, called a *bounder*. By this bounder, during every derivation or reduction step, this system divides the current string into several parts. These systems define their languages by deriving or, in contrast, reducing strings by using these rules. An infinite hierarchy of language families is obtained.

1 INTRODUCTION

String-partitioning system is absolutely new model in modern language theory based by context-free rules without nonterminals. Motivation for this type of system comes from biology where possibility of choosing mode of derivation is needed.

2 PRELIMINARIES

This paper assumes that the reader is familiar with the theory of formal languages (see [2]).

For a set, Q, $card(Q)$ denotes the cardinality of Q. For an alphabet, V, V^* represents the free monoid generated by V under the operation of concatenation. The identity of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. For $w \in V^*$, $|w|$ denotes the length of w, and for $W \subseteq V$, $occur(w,W)$ denotes the number of occurrences of symbols from W in w and $sym(w,i)$ denotes the i-th symbol of w; for instance, $sym(abcd,3) = c$.

A *context-free grammar* is a quadruple, $G = (V,T,P,S)$, where V is a total alphabet, $T \subseteq V$ is an alphabet of terminals, $S \in (V-T)$ is the start symbol, and P is a finite set of *rules* of the form $q\colon A \to v$, where $A \in (V-T)$, $v \in V^*$ and q is a label of this rule. If $q\colon A \to v \in P$, $x,y \in V^*$, G makes a derivation step from xAy to xvy according to $q\colon A \to v$, symbolically written as $xAy \Rightarrow xvy\ [q\colon A \to v]$ or, simply, $xAy \Rightarrow xvy$. In the standard

manner, we define $\Rightarrow^m$, where $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. For $p \in P$, $rhs(p)$ and $lhs(p)$ denotes right-side and left-side handle of rule p, respectively, $lab(p)$ denotes label of rule p and for set of rules P, $lab(P)$ denotes set of all labels of rules from P. The *language of G*, $L(G)$, is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. A language, L, is *context-free* if and only if $L = L(G)$, where G is a context-free grammar.

A *programmed grammar* (see page 28 in [1]) is a quadruple, $G = (V, T, P, S)$, where V is a total alphabet, $T \subseteq V$ is an alphabet of terminals, $S \in (V - T)$ is the start symbol, and P is a finite set of rules of the form $q\colon A \to v, g(q)$, where $q\colon A \to v$ is a context free rule labeled by q and $g(q)$ is a set of rule labels associated with this rule. After an application of a rule of this form in an ordinary context way, in the next step a rule labeled by a label from $g(q)$ has to be applied; otherwise, G makes a derivation step, symbolically denoted by $\Rightarrow$, by analogy with a context-free grammar. In the standard manner, we define $\Rightarrow^m$, where $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The *language of G*, $L(G)$, is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$.

Let G be a grammar with regulated rewriting, and let V_N, V_T, and S be its nonterminal alphabet, terminal alphabet, and axiom, respectively. For a derivation $D\colon S = w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_r = w \in V_T^*$, according to G, we set $Ind(D,G) = \max\{occur(w_i, V_N) \mid 1 \leq i \leq r\}$, and, for $w \in V_T^*$, we define $Ind(w,G) = \min\{Ind(D,G) \mid D \text{ is a derivation for } w \text{ in } G\}$. The *index of grammar* (see page 151 in [1]) G is defined as $Ind(G) = \sup\{Ind(w,G) \mid w \in L(G)\}$. For a language L in the family $\mathcal{L}(X)$ of languages generated by grammars of some type X, we define $Ind_X(L) = \inf\{Ind(G) \mid L(G) = L, G \text{ is of type } X\}$. For a family $\mathcal{L}(X)$, we set $\mathcal{L}_n(X) = \{L \mid L \in \mathcal{L}(X) \text{ and } Ind_X(L) \leq n\}$, $n \geq 1$ and $\mathcal{L}_{fin}(X) = \bigcup_{n \geq 1} L_n(X)$.

3 DEFINITIONS

Let I be a set of positive integers $\{1, 2, \ldots, k\}$. A *string-distributing system* is a quadruple $M = (Q, \Sigma, s, R)$ where Q is a finite set of states, Σ is an alphabet containing a special symbol, #, called a *bounder*, $s \in Q$ is a start state and $R \subseteq Q \times I \times \{\#\} \times Q \times \Sigma^*$ is a finite relation whose members are called *rules*. A rule $(q, n, \#, p, x) \in R$, where $n \in I$, $q, p \in Q$ and $x \in \Sigma^*$, is written as $q_n\# \to px$ hereafter.

A *k-limited configuration* is any string $x \in Q\Sigma^*$ such that $occur(x, \#) \leq k$. Let $pu\#v$, $quxv$ be two k-limited configuration $u, v \in \Sigma^*$, $occur(u, \#) = n - 1$ and $p_n\# \to qx \in R$. Then,

1. M makes a *derivation step* from $pu\#v$ to $quxv$ by using $p_n\# \to qx$, symbolically written $pu\#v \,{}_d\!\Rightarrow quxv\ [p_n\# \to qx]$ in M and
2. M makes a *reduction step* from $quxv$ to $pu\#v$ by using $p_n\# \to qx$, symbolically written $quxv \,{}_r\!\Rightarrow pu\#v[p_n\# \to qx]$ in M.

Let ${}_d\!\Rightarrow^*$ and ${}_r\!\Rightarrow^*$ denote the transition and reflexive closure of ${}_d\!\Rightarrow$ and ${}_r\!\Rightarrow$, respectively.

The *language derived* by M, $L(M, {}_d\!\Rightarrow)$, is defined as $L(M, {}_d\!\Rightarrow) = \{w \mid s\# \,{}_d\!\Rightarrow^* qw,\ q \in Q, w \in (\Sigma - \{\#\})^*\}$.

The *language reduced* by M, $L(M, {}_r\!\Rightarrow)$, is defined as $L(M, {}_r\!\Rightarrow) = \{w \mid qw \,{}_r\!\Rightarrow^* s\#,\ q \in Q, w \in (\Sigma - \{\#\})^*\}$.

Example:

$M = (\{s, p, q, f\}, \{a, b, c, \#\}, s, R)$, where R contains:

1. $s\,{}_1\# \rightarrow p\,\#\#$
2. $p\,{}_1\# \rightarrow q\;a\#b$
3. $q\,{}_2\# \rightarrow p\;\#c$
4. $p\,{}_1\# \rightarrow f\;ab$
5. $f\,{}_1\# \rightarrow f\;c$

$L(M,{}_d\Rightarrow) = \{a^n b^n c^n | n \geq 1\} = L(M,{}_r\Rightarrow)$, holds that $Ind(M) = 2$.
Example of derivation of string *aaabbbccc*: $s\#{}_d\Rightarrow p\#\#[1]\,{}_d\Rightarrow qa\#b\#\;[2]\,{}_d\Rightarrow pa\#b\#c\;[3]\,{}_d\Rightarrow qaa\#bb\#c\;[2]\,{}_d\Rightarrow paa\#bb\#cc\;[3]\,{}_d\Rightarrow faaabbb\#cc\;[4]\,{}_d\Rightarrow faaabbbccc\;[5]$.
Example of reduction of string *aaabbbccc*: $faaabbbccc\,{}_r\Rightarrow faaabbb\#cc\;[5]\,{}_r\Rightarrow paa\#bb\#cc\;[4]\,{}_r\Rightarrow qaa\#bb\#c\;[3]\,{}_r\Rightarrow pa\#b\#c\;[2]\,{}_r\Rightarrow qa\#b\#\;[3]\,{}_r\Rightarrow p\#\#\;[2]\,{}_r\Rightarrow s\#\;[1]$.

Let $\mathcal{L}_{fin}(SPS,{}_d\Rightarrow)$, and $\mathcal{L}_{fin}(P)$ denote the families of string-partitioning system derived languages, and programmed languages of finite index based on context-free grammar, respectively.

4 RESULTS

Lemma 1. $\mathcal{L}_k(P) \subseteq \mathcal{L}_k(SPS,{}_d\Rightarrow)$
For every programmed grammar of index k, G, there is a string-partitioning system of index k, H, such that $L_k(G) = L_k(H,{}_d\Rightarrow)$.

Proof. Let $k \geq 1$ be a positive integer. Let $G = (V,T,P,S)$ is programmed grammar of index k, where $N = V - T$. We construct the string-partitioning system of index k, $H = (Q, T \cup \{\#\}, s, R)$, where $\# \notin T$, $s = \langle\sigma\rangle$, σ is a new symbol, R and Q are constructed by performing the following steps:

1. For each $p\colon S \rightarrow \alpha \in P$, $\alpha \in V^*$, add $\langle\sigma\rangle\,{}_1\# \rightarrow \langle[p\colon S \rightarrow \alpha]\rangle\#$ into R, where $\langle[p\colon S \rightarrow \alpha]\rangle$ is new state in Q.

2. If $A_1A_2\ldots A_j\ldots A_h \in N^*$, $h \in \{1,2,\ldots,k\}$, $A_i \in N$, $1 \leq i \leq h$, $q \in g(p)$, where $p\colon A_j \rightarrow x_0B_1x_1B_2x_2\ldots x_{n-1}B_nx_n$, $j \in \{1,2,\ldots,h\}$, $q\colon C \rightarrow \alpha$, $C \in N$, $\alpha \in V^*$, for some $n \geq 0$, $x_t \in T^*$, $0 \leq t \leq n$, $B_r \in N$, $1 \leq r \leq n$, $n+h-1 \leq k$ and $D_1D_2\ldots D_{j-1}D_j \ldots D_{j+n-1}D_{j+n}\ldots D_{n+h-1}$ such that $D_1D_2\ldots D_{j-1}D_{j+n}D_{j+n+1}\ldots D_{n+h-1} = A_1A_2 \ldots A_{j-1}A_{j+1}\ldots A_h$ and $D_j\ldots D_{j+n-1} = B_1\ldots B_n$, $lhs(q) = D_d$ for some $d \in \{1,2,\ldots, n+h-1\}$, $D_m \in N$, $1 \leq m \leq n+h-1$
   then into R add
   $\langle A_1A_2\ldots A_{j-1}\;[p\colon\; A_j \rightarrow x_0B_1x_1B_2x_2\ldots x_{n-1}B_nx_n]\;A_{j+1}\ldots A_h\rangle\;{}_j\# \rightarrow \langle D_1D_2\ldots D_d\;[q\colon D_d \rightarrow \alpha]D_{d+1}\ldots D_{n+h-1}\rangle x_0\#x_1\#x_2\ldots x_{n-1}\#x_n$, where $\langle D_1D_2\ldots D_d[q\colon D_d \rightarrow \alpha]\;D_{d+1}\ldots D_{n+h-1}\rangle$ is new state in Q.

□

Basic Idea:

Consider bounder as a special nonterminal, only one in total alphabet. Because of finite index, more information about bounder can be recorded in finite number of states. So, we decode into state signature sequence of all origin nonterminals as was in sentential form in programmed grammar.

Lemma 2. $\mathcal{L}_k(SPS, {}_d\Rightarrow) \subseteq \mathcal{L}_k(P)$
For every string-partitioning system of index k, H, exists equivalent programmed grammar of index k, G, such that $L_k(G) = L_k(H, {}_d\Rightarrow)$.

Proof. Let $k \geq 1$ be a positive integer. Let $H = (Q, T \cup \{\#\}, s, R)$ is string-partitioning system of index k, where $\Sigma = T \cup \{\#\}$. We construct programmed grammar of index k, $G = (V, T, P, S)$, where $N = V - T$, and set of nonterminals N and rules P will be constructed by following steps:

1. $P = \emptyset$,
2. $S = \langle s, 1, 1\rangle$,
3. $N = \{\langle p, i, h\rangle \mid p \in Q, 1 \leq i \leq k, 1 \leq h \leq k\} \cup \{\langle q', i, h\rangle \mid q \in Q, 1 \leq i \leq k, 1 \leq h \leq k\}$ $\cup \{\langle q^{\nabla}, i, h\rangle \mid q \in Q, 1 \leq i \leq k, 1 \leq h \leq k\}$,
4. For every rule $r\colon p_i\# \to qy \in R$, $y = y_0\#y_1\#y_2 \ldots y_{m-1}\#y_m$, $y_0, y_1, y_2 \ldots y_m \in T^*$, if $m = 0$ then $h_{max} = k$ else $h_{max} = k - m + 1$, add following set into P:

   (i) $\{\langle p, j, h\rangle \to \langle q', j, h+m-1\rangle,$
   $\{r' \mid$ if $j+1 = i$ then $r'\colon \langle p, i, h\rangle \to \langle q^{\nabla}, i, h+m-1\rangle$ else $r'\colon \langle p, j+1, h\rangle \to \langle q', j+1, h+m-1\rangle$ $\}$
   $\mid 1 \leq j < i,\ i \leq h \leq h_{max}\}$
   $\cup$

   (ii) $\{\langle p, i, h\rangle \to \langle q^{\nabla}, i, h+m-1\rangle,$
   $\{r' \mid$ if $i = h$ then $r'\colon \langle q^{\nabla}, i, h+m-1\rangle \to y_0\langle q', i, h+m-1\rangle y_1\langle q', i+1, h+m-1\rangle y_2 \ldots y_{m-1}\langle q', i+m-1, h+m-1\rangle y_m$ else $r'\colon \langle p, i+1, h\rangle \to \langle q', i+1+m-1, h+m-1\rangle\}$
   $\mid i \leq h \leq h_{max}\}$
   $\cup$

   (iii) $\{\langle p, j, h\rangle \to \langle q', j+m-1, h+m-1\rangle,$
   $\{r' \mid$ if $j = h$ then $r'\colon \langle q^{\nabla}, i, h+m-1\rangle \to y_0\langle q', i, h+m-1\rangle y_1\langle q', i+1, h+m-1\rangle y_2 \ldots y_{m-1}\langle q', i+m-1, h+m-1\rangle y_m$ else $r'\colon \langle p, j+1, h\rangle \to \langle q', j+1+m-1, h+m-1\rangle\}$
   $\mid i < j \leq h,\ i \leq h \leq h_{max}\}$
   $\cup$

   (iv) $\{\langle q^{\nabla}, i, h+m-1\rangle \to y_0\langle q', i, h+m-1\rangle y_1\langle q', i+1, h+m-1\rangle y_2 \ldots y_{m-1}\langle q', i+m-1, h+m-1\rangle y_m,$
   $\{r' \mid r'\colon \langle q', 1, h+m-1\rangle \to \langle q, 1, h+m-1\rangle\}$
   $\mid i \leq h \leq h_{max}\}$
   $\cup$

   (v) $\{\langle q', j, h+m-1\rangle \to \langle q, j, h+m-1\rangle,$
   $\{r' \mid$ if $j < h+m-1$ then $r'\colon \langle q', j+1, h+m-1\rangle \to \langle q, j+1, h+m-1\rangle$ else $r'\colon \langle \tilde{p}, 1, h+m-1\rangle \to \langle \tilde{q}', 1, h+m-1+\tilde{m}-1\rangle$, where $\tilde{p}_{\tilde{i}}\# \to \tilde{q}\tilde{y}_0\#\tilde{y}_1 \ldots \tilde{y}_{\tilde{m}-1}\#\tilde{y}_{\tilde{m}} \in R$, $\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_{\tilde{m}} \in T^*$, if $\tilde{i} = 1$ then $\tilde{q}' := \tilde{q}^{\nabla}\}$
   $\mid 1 \leq j \leq h+m-1,\ i \leq h \leq h_{max}\}$. $\square$

Rigorous proofs (by induction) of both lemmas are left to reader due to insufficient space here.

Basic Idea:

Each new nonterminal has three components:
(1) origin state from system H (marked accordingly to phase of simulation in G),
(2) index of nonterminal in actual sentential form (index is important part of rules in H),
(3) summary number of nonterminals in actual sentential form (in some substeps inconsistent).

Thanks to programming of rules and finite index is the simulation sequence of rule from H always atomic in G or a string of terminals is derived.

Theorem 1. $\mathcal{L}_k(SPS, {}_d\Rightarrow) = \mathcal{L}_k(P)$.
Every language is generated by string-partitioning system of index k if and only if this language is generated even by some programmed grammar of index k.
Proof. This equivalence was proved in the previous Lemma 1 and 2, therefore, this theorem holds. ■

Theorem 2. $\mathcal{L}_{fin}(SPS, {}_d\Rightarrow) = \mathcal{L}_{fin}(P)$.
Proof. Theorem 1 holds for every $k \geq 1$, therefore, this theorem holds too. ■

Corollary 1. Infinite hierarchy $L_k(SPS, {}_d\Rightarrow) \subset L_{k+1}(SPS, {}_d\Rightarrow)$ holds for every $k \geq 1$.
Proof. It follows from Theorem 1 and theorem that $L_k(P) \subset L_{k+1}(P)$ for every $k \geq 1$ in Chapter 3 in [1]. ■

5 CONCLUSION

String-partitioning systems have predominantly theoretical acquisition today. It can help to solve some open problems in regulated grammars of finite index. For example: Is inclusion $L_{fin}(Random\ Context) \subseteq L_{fin}(P)$ proper?

Ideas of future modifications: studying of canonical derivations, parallel versions without no increase of power, removing constraint of finite index.

More details about this new system and other results can be found in [3] (in czech).

REFERENCES

[1] Dassow, J., Păun, G.: Regulated Rewriting in Formal Language Theory, Springer, New York, 1989.

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

[3] Křivka, Z.: String-partitioning systems – essay in course Modern Theoretical Computer Science [in czech], FIT Brno University of Technology, 2004.