

String-Partitioning Systems and An Infinite Hierarchy

Zbyněk Křivka & Rudolf Schönecker

Department of Information Systems
Faculty of Information Technologies
Brno University of Technology

Workshop on Formal Models, April 25 - 27, 2006

Outline

- 1 Motivation
- 2 Definitions
 - String partitioning system
 - Finite index of used formal models
 - Programmed grammars
- 3 Our Results
 - Generative power
- 4 Basic proof ideas
 - Basic idea of the first proof's part
 - Basic idea of the second proof's part
- 5 Appendix
 - Modifications of SPS
 - References

Motivation

inspiration and special characteristics

Inspiration:

- another rewriting mechanisms
- models with properties of both automata and grammars
- generative power of such devices?
- different and common properties?

Definition

string partitioning system and it's configuration

SPS is a quadruple $M = (Q, \Sigma, s, R)$

- Q is a finite set of states
- Σ is an alphabet, $\# \in \Sigma$ called *bounder*
- $s \in Q$ is a start state
- R is finite set of rules of the form:
 $p_i\# \rightarrow qx \in R$, where $p, q \in Q$, $i \in I$, $x \in \Sigma^*$.

Configuration of SPS

- is string $c \in Q\Sigma^*$

Definition

derivation step and derived language

Derivation step from $pu\#v$ to $quxv$, where

- $p, q \in Q, u, v, x \in \Sigma^*$
- $\text{occur}(u, \#) = n - 1$
- by using $p_n\# \rightarrow qx \in R$

is $pu\#v \Rightarrow quxv$ $[p_n\# \rightarrow qx]$ in M

Language derived by M , $L(M)$:

- $L(M) = \{w \mid s\# \Rightarrow^* qw, q \in Q, w \in (\Sigma - \{\#\})^*\}$

Simple example of SPS

generation of language $a^n b^n c^n$

$M = (\{s, p, q, f\}, \{a, b, c, \#\}, s, R)$, 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. $f_1\# \rightarrow f\ c$

Example (derivation of string $aaabbbccc$)

$s\# \Rightarrow p\#\#\#[1] \Rightarrow qa\#b\#\#[2] \Rightarrow pa\#\#b\#c[3] \Rightarrow$
 $qaa\#bb\#c[2] \Rightarrow paa\#\#bb\#cc[3] \Rightarrow faaabbb\#\#cc[4] \Rightarrow$
 $faaabbbccc[5].$

$L(M) = \{a^n b^n c^n \mid n \geq 1\}$, with $Ind(M) = 2$

Definition

finite index of used formal models

Finite index of grammar?

- max. number of N 's in sentential form w
- achievable sent. form - $S \Rightarrow^* w$
- leading to string x : $w \in x, x \in \Sigma^*$
- in the most economical derivation

Finite index of SPS?

- max.number of $\#$'s in sentential form

Definition

finite index of used formal models

Index of a language:

- equal to index of grammar/SPS

Family of languages of finite index k

- $\mathcal{L}_k(X)$

Family of all languages of finite index

- $\mathcal{L}_{fin}(X) = \bigcup_{i \geq 1} \mathcal{L}_i(X)$

Definition

programmed grammars

Programmed grammar (PG) – $G = (V, T, P, S)$

- V is a total alphabet
- $T \subseteq V$ is an alphabet of terminals
- $S \in (V - T)$ is the start symbol
- P is a finite set of rules of the form $p: A \rightarrow v, g(p)$
 - $p: A \rightarrow v$ is a context free rule labeled by p
 - $g(p)$ - set of rule labels associated with rule p (following set)
 - after p -application a rule labeled by a label from $g(p)$ has to be applied

Generative power of programmed grammars

Programmed grammars:

- $\mathcal{L}(CF) \subset \mathcal{L}(PG) \subset \mathcal{L}(PG_{ac}) \subset \mathcal{L}(CS) \subset \mathcal{L}(\lambda PG_{ac}) = \mathcal{L}(RE)$
- $\mathcal{L}(CF) \subset \mathcal{L}(PG) \subset \mathcal{L}(\lambda PG) \subset \mathcal{L}(RE)$

Programmed grammars of index k :

- $\mathcal{L}_{fin}(PG) = \mathcal{L}_{fin}(\lambda PG) = \mathcal{L}_{fin}(PG_{ac}) = \mathcal{L}_{fin}(\lambda PG_{ac})$
- $\mathcal{L}(CF) - \mathcal{L}_{fin}(PG) \neq \emptyset$
 $\Rightarrow \mathcal{L}_{fin}(PG)$ is incomparable towards $\mathcal{L}(CF)$

Generative power and infinite hierarchy of string partitioning systems of finite index

infinite hierarchy for SPS

$$\mathcal{L}_k(\text{SPS}) \subset \mathcal{L}_{k+1}(\text{SPS}), \text{ for all } k \geq 1$$

- 1 $\mathcal{L}_k(\text{PG}) \subset \mathcal{L}_{k+1}(\text{PG}), \text{ for all } k \geq 1$ (Gh. Păun, 1980)
- 2 $\mathcal{L}_k(\text{SPS}) = \mathcal{L}_k(\text{PG})$

Theorem 2

$$\mathcal{L}_k(\text{SPS}) = \mathcal{L}_k(\text{PG}), \text{ for every } k \geq 1$$

Proof: 1) $\mathcal{L}_k(\text{PG}) \subseteq \mathcal{L}_k(\text{SPS})$ 2) $\mathcal{L}_k(\text{SPS}) \subseteq \mathcal{L}_k(\text{PG})$

Proof (basic idea)

first part: $PG_k \rightarrow SPS_k$

Conversion: $PG_k \rightarrow SPS_k$

- nonterminals represented by #s and information in state
- each state in SPS_k (2 components) of form:

$$\langle A_1 \dots A_j, q \rangle$$

$$A_1, \dots, A_j \in N_{PG_k}, 0 \leq j \leq k, q \in g(p)$$

- one symbol in $A_1 \dots A_k$ is marked for following rewriting
- q represents next rule to use
- bounders mark positions for former nonterminals

$$x_0 A x_1 B x_2$$

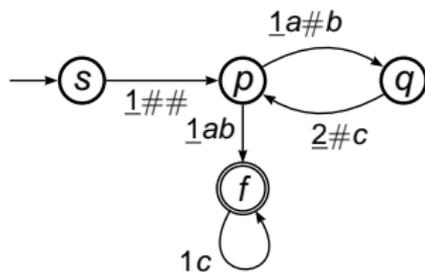


$$\langle AB, q \rangle x_0 \# x_1 \# x_2$$

Proof (demonstration)

second part: $SPS_k \rightarrow PG_k$

$SPS_2 M = (\{s, p, q, f\}, \{a, b, c, \#\}, s, R)$:



- 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 $f_1\# \rightarrow f\ c$

How to construct PG's rule set based on SPS' rules?

basic idea will be presented..

Proof (demonstration)

second part: $SPS_k \rightarrow PG_k$

Conversion: $SPS_k \rightarrow PG_k$

- 1 every $A \in N$ is of form $\langle p, i, h \rangle$, $s := \langle s, 1, 1 \rangle$
- 2 every SPS_k 's rule $p_i \# \rightarrow qy$ simulate by sequence of steps (differs for every number of #s in configuration):

$$p \underline{a} \# b \# c \Rightarrow_{SPS} f \underline{a} \underline{a} b b \# c \quad [p_i \# \rightarrow f ab]$$



a) **renumbering**

$$a \langle p, 1, 2 \rangle b \langle p, 2, 2 \rangle c \Rightarrow_{PG} a \langle f'', 1, 1 \rangle b \langle p, 2, 2 \rangle c \Rightarrow_{PG}$$

b) **rewriting**

$$a \langle \underline{f''}, 1, 1 \rangle b \langle f', 1, 1 \rangle c \Rightarrow_{PG} \underline{a} \underline{a} b b \langle f', 1, 1 \rangle c \Rightarrow_{PG}$$

c) **finalization**

$$a \underline{a} b b \langle f, 1, 1 \rangle c$$

Modifications of SPS

another challenges...

SPS with finite index:

- deterministic variant
- accepting variant
- parallel variant

SPS without index limitation:

- generative power
- properties

References...



J. Dassow, Gh. Păun.

Regulated Rewriting in Formal Language Theory.
Springer, New York, 1989.



A. Meduna.

Automata and Languages, Theory and Applications.
Springer, London, 2000.



T. Kasai

A Hierarchy Between Context-Free and Context-Sensitive
Languages.
Journal of Computer and System Sciences, vol. 4, 1970.