String-Partitioning Systems

Rudolf Schönecker

April 4, 2006

String partitioning system - basics

Definition (SPS)

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*, for some set of positive integers I.

Definition (Rules)

A rule $(q, n, \#, p, x) \in R$, where $n \in I$, $q, p \in Q$ and $x \in \Sigma^*$, is written as $q\ {}_n\# \rightarrow px$ hereafter.

String partitioning system - basics

$occur(w, W)$ - the nr. of occurrences of symbols from W in w

Definition (k-limited configuration)

is any string $x \in Q\Sigma^*$ such that $occur(x, \#) \leq k$

Definition (derivation step)

Let $pu\#v$, $quxv$ be two k-limited configuration $u, v \in \Sigma^*$, $occur(u, \#) = n - 1$ and $p\ _n\# \rightarrow qx \in R$.

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

String partitioning system - basics

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

Definition (SPS language)

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

String partitioning system - example

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. $f\ {}_1\# \rightarrow f\ c$

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

String partitioning system - example

Example (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 (Example of reduction of string *aaabbbccc*)

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

PG definition

Definition (Programmed grammar)

is a quadruple, $G = (V, T, P, S)$, where

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

Finite index definition

Definition ($G = (V, T, P, S)$, $N = V - T$)

For $D: S = w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_r = w \in T^*$, for $w \in T^*$ in G:

- $Ind(D, G) = \max\{occur(w_i, N) \mid 1 \leq i \leq r\}$
- $Ind(w, G) = \min\{Ind(D, G) \mid D$ is a derivation for w in $G\}$
- $Ind(G) = \sup\{Ind(w, G) \mid w \in L(G)\}$

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 is of type $X\}$

For a family $\mathcal{L}(X)$, we set

- $\mathcal{L}_k(X) = \{L \mid L \in \mathcal{L}(X)$ and $Ind_X(L) \leq k\}$, $k \geq 1$
- $\mathcal{L}_{fin}(X) = \bigcup_{n \geq 1} L_n(X)$

PG generative power

Summary (Generative power)

For programmed grammars stands:

- $\mathcal{L}(2) \subset \mathcal{L}(P, CF) \subset \mathcal{L}(1)$

For programmed grammars of index k stands:

- $\mathcal{L}_k(P, CF) \subset \mathcal{L}_{k+1}(P, CF)$, *for all* $k \geq 1$
- $\mathcal{L}(CF) - \mathcal{L}_{fin}(P, CF) \neq \emptyset$
- $\mathcal{L}_{fin}(P, CF)$ *is incomparable towards* $\mathcal{L}(CF)$

Results

Lemma ($\mathcal{L}_k(P, CF) \subseteq \mathcal{L}_k(SPS, {}_d\Rightarrow)$)

For every programmed grammar of index k, G, there is a string-partitioning system of index k, H, such that $L_k(G) = L_k(H, {}_d\Rightarrow)$.

Lemma ($\mathcal{L}_k(SPS, {}_d\Rightarrow) \subseteq \mathcal{L}_k(P, CF)$)

For every string-partitioning system of index k, H, exists equivalent programmed grammar of index k, G, such that $L_k(G) = L_k(H, {}_d\Rightarrow)$.

Main result

$\mathcal{L}_k(SPS, {}_d\Rightarrow) = \mathcal{L}_k(P, CF)$, for every $k \geq 1$.

Results

Infinite hierarchy of languages

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

Proof:

Because of $\mathcal{L}_k(P, CF) \subset \mathcal{L}_{k+1}(P, CF)$, for all $k \geq 1$, and $\mathcal{L}_k(SPS, {}_d\Rightarrow) = \mathcal{L}_k(P, CF)$, for every $k \geq 1$.