Generative power of CD grammar systems with scattered context components

Jakub Martiško
imartisko@fit.vutbr.cz

December 23, 2015

Context Sensitive Grammars

Definition

$G = (N, T, P, S)$, $p = \alpha A \gamma \rightarrow \alpha \beta \gamma$, where
$p \in P, A \in N, \alpha, \beta, \gamma \in (N \cup T)^* \wedge \beta \neq \varepsilon$.

Kuroda normal form

- $AB \rightarrow CD$
- $A \rightarrow BC$
- $A \rightarrow B$
- $A \rightarrow a$

Where $A, B, C, D \in N, a \in T$.

Propagating Scattered Context Grammars

Definition

$G = (N, T, P, S)$, $p = (A_1, A_2, A_3, \ldots, A_n) \rightarrow (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n)$, where $p \in P, A_i \in N, \alpha_i \in (N \cup T)^+$.

Example

Suppose rule $(A, B, C) \rightarrow (a, b, c)$.

AAABBBCCC ↓ Right AAaBbBCCc

CCCBBBAAA ↓ Wrong CCcBbBAAa

$\mathcal{L}(SCG) \subseteq \mathcal{L}(CSG)$

Idea

New symbols are introduced which serve as a counter, which keep track of the current component that should be rewritten. [2]

Example

Suppose rule $i : (A, B, C) \rightarrow (aA, bB, cC)$

aa[i,1]AbbBccC

aaaA[i,2]bbBccC

$\mathcal{L}(SCG) \subseteq \mathcal{L}(CSG)$

Idea

New symbols are introduced which serve as a counter, which keep track of the current component that should be rewritten. [2]

Example

Suppose rule $i : (A, B, C) \rightarrow (aA, bB, cC)$

aaaA[i,2]bbBccC

↓

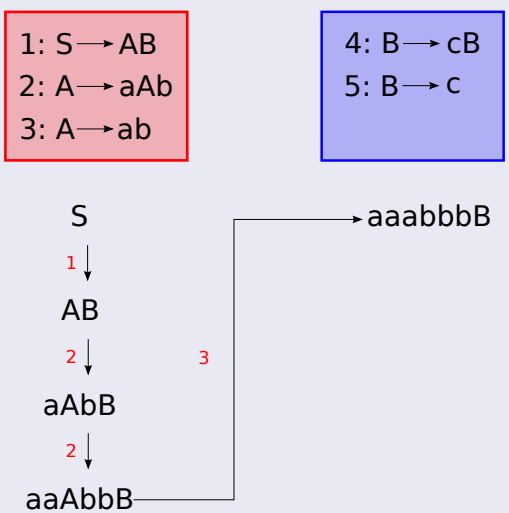
aaaAb[i,2]bBccC

CD Grammar Systems

Definition

Cooperating Distributed grammar system of degree n $M = (N, T, S, P_1, P_2, \ldots, P_n)$. Where P_i are sets of rewriting rules of some form. Working in t mode (component must rewrite everything it can).

Example

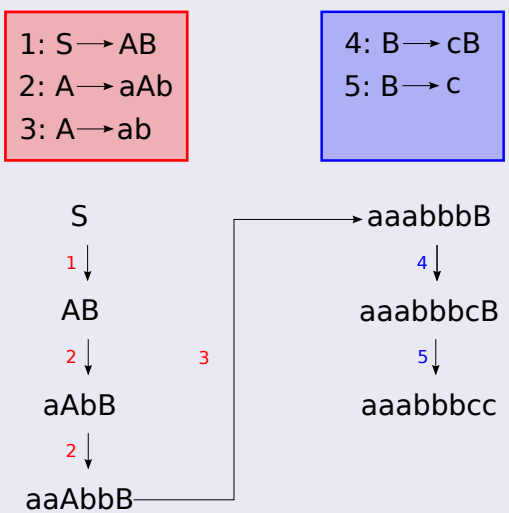


CD Grammar Systems

Definition

Cooperating Distributed grammar system of degree n $M = (N, T, S, P_1, P_2, \ldots, P_n)$. Where P_i are sets of rewriting rules of some form. Working in t mode (component must rewrite everything it can).

Example



Context Sensitive Grammar Systems

Generative power

CD grammar systems with CS components working in t mode are as powerful as standard CSGs.

Proof (idea)

Main idea – component activation can be simulated by CS programmed grammars where success field contains rules of the current component and failure field switches to new component [1].

Main result: $\mathcal{L}(SCGS, t) = \mathcal{L}(CSG)$

$\mathcal{L}(SCGS, t) \subseteq \mathcal{L}(CSG)$

Proof is a combination of previous proofs. For each SCG component a CS component, that simulates it, is created. All the CS components can be reduced to CDGS with single component.

$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$

Main problem (suppose rule $AB \to CD$ and rule $(A, B) \to (C, D)$).

AAABBB

↓

AACDBB

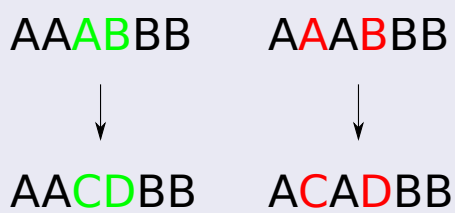
Main result: $\mathcal{L}(SCGS, t) = \mathcal{L}(CSG)$

$\mathcal{L}(SCGS, t) \subseteq \mathcal{L}(CSG)$

Proof is a combination of previous proofs. For each SCG component a CS component, that simulates it, is created. All the CS components can be reduced to CDGS with single component.

$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$

Main problem (suppose rule $AB \to CD$ and rule $(A, B) \to (C, D)$).



$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 1

Idea

- Using two SCG components working in t mode.
- No ε-rules.
- All terminals in rules are replaced by nonterminal variant of the symbol.
- First component – applies the rule.
- Second component – checks, whether the rule was applied in context sensitive way.
- New symbols are introduced – $|A|, |B[,]C| \ldots$

$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 2

First component

Works in two phases, current phase is determined by the first symbol (+ or -). During the first phase, one rule of the input grammar is simulated and then the component switches to second phase.

Example

-A...AAABBB...

+A...AA|C[]D|BB...

$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 2

First component

Works in two phases, current phase is determined by the first symbol (+ or -). During the first phase, one rule of the input grammar is simulated and then the component switches to second phase.
During the second phase, all remaining symbols are rewriten to their context free variant.

Example

+A...AA|A[]B|BB...

$\downarrow *$

|+A|...|A||A||C[]D||B||B|...

$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 3

Second component

Works in two phases, current phase is again determined by the first symbol (all rules check the first symbol using the first part of the SC rule). The first phase is initialization.

Example

|+A||A||A|...|C[]D||B||B|...

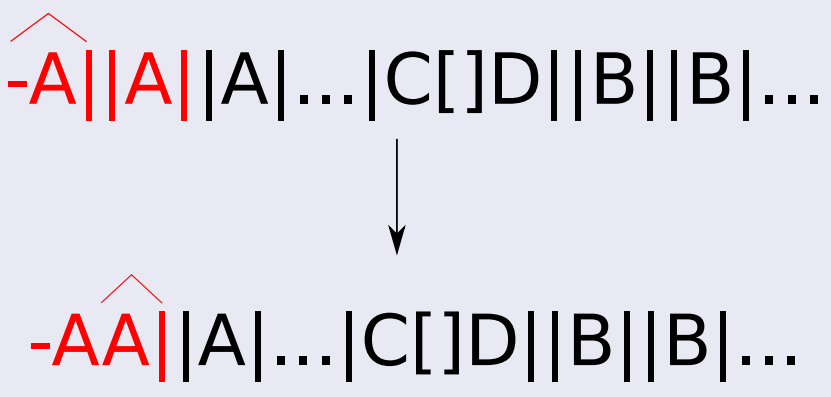
-A||A||A|...|C[]D||B||B|...

$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 3

Second component

Works in two phases, current phase is again determined by the first symbol (all rules check the first symbol using the first part of the SC rule). The first phase is initialization. During the second phase, there is always exactly one symbol marked with $\wedge$ symbol. This symbol serves as a current symbol. Each rule checks the current symbol and a symbol right of it.

Example

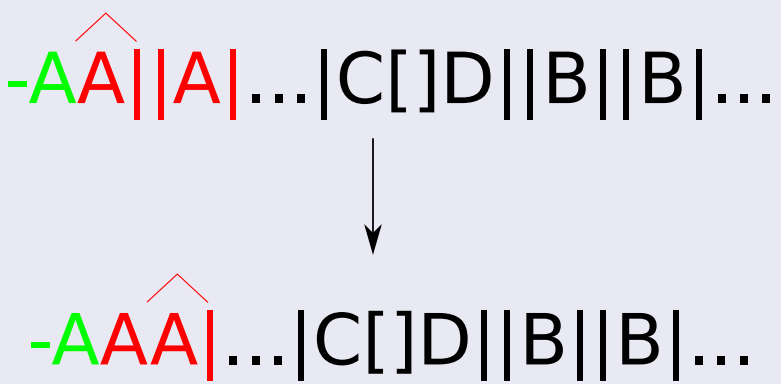


$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 3

Second component

Works in two phases, current phase is again determined by the first symbol (all rules check the first symbol using the first part of the SC rule). The first phase is initialization. During the second phase, there is always exactly one symbol marked with $\wedge$ symbol. This symbol serves as a current symbol. Each rule checks the current symbol and a symbol right of it.

Example

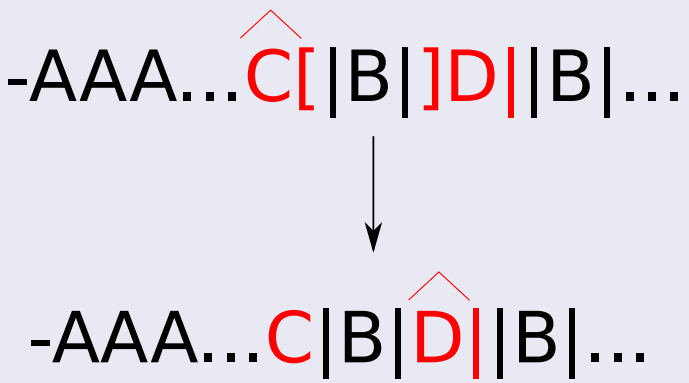


$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 4

Second component – Errors

Scattered context grammars do not guarantee, that adjacent symbols will be always rewritten.

Example



$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 4

Second component – Errors

Scattered context grammars do not guarantee, that adjacent symbols will be always rewritten. Current symbol ($\wedge$) marks the leftmost symbol, that can be rewritten. Second component can rewrite only symbol right of this symbols but it can not "return" to any symbol left of it.

Example

-AAA...C|B|$\hat{D}$||B|...

Finished ↓ Rewritable

-AAA...C|B|D$\hat{B}$|...

$\mathcal{L}(CSG) \subseteq \mathcal{L}(SCGS, t)$ Proof, part 4

Second component – Errors

Scattered context grammars do not guarantee, that adjacent symbols will be always rewritten. Current symbol ($\wedge$) marks the leftmost symbol, that can be rewritten. Second component can rewrite only symbol right of this symbols but it can not "return" to any symbol left of it. Rules of the form $|A| \rightarrow |A|$ will force the component to loop endlesly.

Example

-AAA...C|B|DB...

-AAA...C|B|DB...

References


E. Csuhaj-Varjú.
Grammar Systems: A Grammatical Approach to Distribution and Cooperation.
Topics in computer mathematics. Gordon and Breach, 1994.


J. Dassow and G. Păun.
Regulated rewriting in formal language theory.
EATCS monographs on theoretical computer science. Springer, 1989.