
Grammar Cheatsheet

TLP

2019

Contents

Build GCL for a given language	3
Reduce a CFG	3
Algoritmo para calcular símbolos co-accesibles	3
Algoritmo para calcular símbolos accesibles	3
Algorithm for:	3
CFG is finite	4
GCL is empty	4
A word belongs to $L(G)$	4
CYK	4
Brute force	4
Normal Forms	4
Chomsky	4
Greibach	4
LL(k) Grammars	4
CFG to NPDA	4
NPDA to CFG	4

List of Tables

List of Figures

Build GCL for a given language

Reduce a CFG

Dada una gramática $G = (N, T, S, P)$:

- Un símbolo útil $\in N \cup T$ es aquel:
 - $X \in N \cup T$ accesible si: $S \Rightarrow^* \alpha X \beta$
 - $X \in N$ co-accesible si: $X \Rightarrow^* \omega, \omega \in T^*$
- El orden importa, primero calcular co-accesibles y luego accesibles.

Algoritmo para calcular símbolos co-accesibles

Símbolos co-accesibles: $S_{co} = \{A \in N \mid A \rightarrow \alpha, \alpha \in T^*\}$

$S_{co_{i+1}} = S_{[co_i]} \{A \in N \mid A \rightarrow \alpha \in P, \alpha \in (S_{[co_i]} \cup T)^*\}$

STOP WHEN: $S_{co_i} = S_{co_{i+1}}$

Algoritmo para calcular símbolos accesibles

Se construye un grafo:

- Los nodos son símbolos(dependencias)
- $X \rightarrow Y$ si $X \rightarrow \alpha Y \beta \in P$

X es accesible si \exists un camino de S hasta X.

Algorithm for:

Given a CFG ...

CFG is finite

1. Reduce the grammar.
2. Transform into CNF.
3. Look for loops in the dependency graph.

GCL is empty

1. Calculate co-accessible symbols.
2. If $S \in S_c \rightarrow L(G) \neq \emptyset$ else $L(G) = \emptyset$

A word belongs to $L(G)$ **CYK****Brute force****Normal Forms****Chomsky****Greibach****LL(k) Grammars****CFG to NPDA****NPDA to CFG**