Grammar Cheatsheet

TLP

2019

Contents

List of Tables

List of Figures

Build CFG 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:

CFG is finite

Given a CFG …

CFG is finite

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

CFG is empty

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

A word belongs to L(G)

CYK

Brute force

Normal Forms

Chomsky

Greibach

$S \to \lambda$ S don't appears in the right member of the same rule.

$A \to a\alpha, A \in N, a \in T, \alpha \in N^*$

1. $G \to G\prime \mid G\prime isinCNF$
2. $G\prime \to G\prime\prime inGNF$

- 2.1 Order the non terminals (ex: S<A<B):

  $A_1 \leftrightarrow S$

  $A_2 \leftrightarrow A$

  $A_2 \leftrightarrow B$

- 2.2 Every rule must be in the form of:

  $A_i \to A_j\alpha, j > i$

- 2.3 In the case that a rule is not in that form:

  ex: $A_2 \rightarrow A_1 A_3$

  Replace A_1 with its right members.

  ex: $A_1 \rightarrow A_2 A_3 \mid A_2 A_2$

  Replacing: $A_2 \rightarrow A_2 A_3 A_3 \mid A_2 A_2 A_3$ (Recursivity to the left)

PDA

Deterministic PDA

$PDA = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic if:

1. $\mid \delta(q, a, A) \mid \leq 1, \forall q \in Q, a \in \Sigma, A \in \Gamma$
2. $\delta(q, \lambda, A) \neq \emptyset, \delta(q, a, A) = \emptyset \forall A \in \Sigma$

LL(k) Grammars

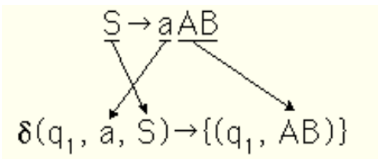
CFG to NPDA

For any context-free grammar in Greibach Normal Form we can build an equivalent nondeterministic pushdown automaton. This establishes that an npda is at least as powerful as a cfg. It will always produce a PDA with **three states**

1. Start state q_0 will serve as initialization.

$$(q_0, \lambda, z) \rightarrow \{(q_1, S_z)\}$$

2. State q_1 will contain the actual grammar computation.

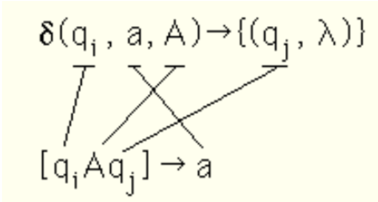


3. Transition q_1 to q_f to accept the string

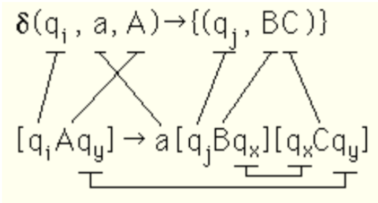
$$delta(q_1, \lambda, z) \rightarrow \{(q_f, z)\}$$

NPDA to CFG

1. Las transiciones del tipo $\delta(q_i, a, A) = (q_j, \lambda)$ se transforman en reglas gramaticas del tipo:



2. Las transiciones del tipo $\delta(q_i, a, A) = (q_j, BC)$ resultan en una multitud de reglas. Una para cada par de estados q_x, q_y en el NPDA, muchas *unreachable* pero las utiles definen la gramatica:



Misc

Eliminate common prefixes

$A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \cdots \mid \alpha\beta_n$

$A \to \gamma_1 \mid \gamma_2 \mid \cdots \mid \gamma_m$

Transform into:

$A \to A\prime$

$A\prime \to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$

Ambiguity

A grammar $G = (N, T, S, P)$ is ambiguous if $\exists$ a word that: - w can be derived with 2 different derivations to the right or left. - w have 2 different derivation trees.