Contents

1. Build CFG for a given language **2**

2. Reduce a CFG **2**

3. Algorithm for: **2**

3.1 CFG is finite 2

3.2 CFG is empty **3**

4. A word belongs to L(G) **3**

4.1 CYK 3

4.2 Brute force 3

5. Chomsky Normal Form **3**

6. Greibach Normal Form **4**

7. PDA **5**

8. LL(k) Grammars **5**

9. CFG to NPDA **5**

10. NPDA to CFG **5**

11. Misc **6**

11.1 Eliminate common prefixes 6

11.2 Ambiguity 6

1. Build CFG for a given language

2. 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.

2.1 Algoritmo para calcular símbolos co-accesibles

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

$S_{co_i+1} = S_{co_i}\{A \in N \mid A \to \alpha \in P, \alpha \in (S_{co_i} \cup T)^*\}$

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

2.2 Algoritmo para calcular símbolos accesibles

Se construye un grafo:

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

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

3. Algorithm for:

Given a CFG …

3.1 CFG is finite

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

3.2 CFG is empty

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

4. A word belongs to L(G)

4.1 CYK

4.2 Brute force

5. Chomsky Normal Form

La CNF es una gramatica del tipo:

1. $A \rightarrow BC$, donde $A, B,$ y C, son no-terminales o
2. $A \rightarrow a$ donde A es un no-terminal y a es una terminal
3. Cabe notar que un CNF no tiene simbolos inutiles (se debe reducir antes) ni tampoco tiene producciones ϵ

Los pasos para transformar una CFG a una CNF son:

a. Conseguir que todos los cuerpos de tamano 2 o mas consistan solo de no-terminales.
b. Romper los cuerpos de tamano 3 o superior en cuerpos pequenos para cumplir la condicion anterior.

ejemplo:

$$
\begin{array}{lcl}
E & \rightarrow & EPT \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
T & \rightarrow & TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
F & \rightarrow & LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
I & \rightarrow & a \mid b \mid IA \mid IB \mid IZ \mid IO \\
A & \rightarrow & a \\
B & \rightarrow & b \\
Z & \rightarrow & 0 \\
O & \rightarrow & 1 \\
P & \rightarrow & + \\
M & \rightarrow & * \\
L & \rightarrow & (\\
R & \rightarrow &)
\end{array}
$$

$$
\begin{array}{lcl}
E & \rightarrow & EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
T & \rightarrow & TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
F & \rightarrow & LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
I & \rightarrow & a \mid b \mid IA \mid IB \mid IZ \mid IO \\
A & \rightarrow & a \\
B & \rightarrow & b \\
Z & \rightarrow & 0 \\
O & \rightarrow & 1 \\
P & \rightarrow & + \\
M & \rightarrow & * \\
L & \rightarrow & (\\
R & \rightarrow &) \\
C_1 & \rightarrow & PT \\
C_2 & \rightarrow & MF \\
C_3 & \rightarrow & ER
\end{array}
$$

6. Greibach Normal Form

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

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

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

- 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:

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

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

  ex: $A_2 \rightarrow A_1A_3$

  Replace A_1 with its right members.

  ex: $A_1 \rightarrow A_2A_3 \mid A_2A_2$

  Replacing: $A_2 \rightarrow A_2A_3A_3 \mid A_2A_2A_3$ (Recursivity to the left)

7. 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. $\delta(q, \lambda, A) \neq \emptyset, \delta(q, a, A) = \emptyset \forall A \in \Sigma$

8. LL(k) Grammars

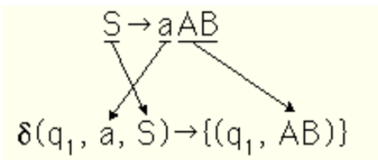
9. CFG to NPDA

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.

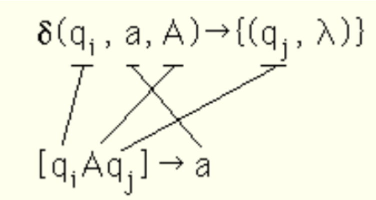


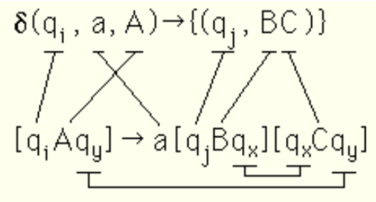
3. Transition q_1 to q_f to accept the string

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

10. 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:





11. Misc

11.1 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:

$A \to A\prime$

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

11.2 Ambiguity

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