Chomsky Normal Form

07 February 2021 23:43

Chomsky Normal Form

* Helpful to have a simplified form for G.
* Need to be efficient to check if $w \in L(G)$.
* Should not have loops (in derivation).
* Should not have empty derivations, or useless rules.
* Should have simple rules that are easy to check.

Def 2.8: A context-free grammar is in Chomsky Normal Form if every rule is of the form

$$A \rightarrow BC$$
$$A \rightarrow a$$

where a is a terminal, A, B, C are variables and B and C should not be the start variable.
Also, we allow $S \rightarrow \varepsilon$.

Theorem 2.9: Any CFL is generated by a CFG in

Chomsky Normal Form.

Proof: We convert any CFG into CNF.

Optional Step: Remove useless symbols & productions.

1. Add new start variable S_0.

$$S_0 \rightarrow S$$

This guarantees that S_0 does not appear in the RHS of any rule.

2. Remove ε rules.

Say $A \rightarrow \varepsilon$. Then modify rules with A in RHS. If $R \rightarrow uAv$ was a rule, then

$$R \rightarrow uAv \mid uv.$$

$R \rightarrow A$ becomes $R \rightarrow A \mid \varepsilon$

unless $R \rightarrow \varepsilon$ is already removed.

3. Remove unit rules like $A \rightarrow B$.

When $B \rightarrow u$ appears, add $A \rightarrow u$ unless it was already removed.

4. Restructure long RHS.

4. Restructure long RHS.

Example :

$S \to TST \mid aB$

$T \to B \mid S$

$B \to b \mid \varepsilon$

(1) Add $S_0 \to S$.

$S_0 \to S$

$S \to TST \mid aB$

$T \to B \mid S$

$B \to b \mid \varepsilon$

(2) Remove $B \to \varepsilon$

$S_0 \to S$

$S \to TST \mid aB \mid a$

$T \to B \mid \varepsilon \mid S$

$B \to b$

$T \to \varepsilon$

$S_0 \to S$

$S \to TST \mid TS \mid ST \mid \cancel{S} \mid aB \mid a$

$T \to B \mid S$

$B \to b$

---

(3) Unit Rules

$S \to TST \mid TS \mid ST \mid aB \mid a$

(3) Unit Rules

$S_0 \to S$

$$S_0 \to TST \mid TS \mid ST \mid aB \mid a$$
$$S \to TST \mid TS \mid ST \mid aB \mid a$$
$$T \to B \mid S$$
$$B \to b$$

$T \to B$
$T \to S$

$$S_0 \to TST \mid TS \mid ST \mid aB \mid a$$
$$S \to TST \mid TS \mid ST \mid aB \mid a$$
$$B \to b$$
$$T \to b \mid S$$
$$T \to b \mid TST \mid ST \mid TS \mid aB \mid a$$

---

(4) Restructure rules with more than one symbol in RHS.

let $TS = Z$, $a = A$.

$$S_0 \to ZT \mid TS \mid ST \mid AB \mid a$$
$$S \to ZT \mid TS \mid ST \mid AB \mid a$$
$$T \to ZT \mid TS \mid ST \mid AB \mid a$$
$$Z \to TS$$

$A \longrightarrow a$

$B \longrightarrow b$.

The main advantage of CNF is that there is "predictability" in the derivation of a string. Any string of length n requires exactly $2n-1$ steps for derivation. Why?

Chomsky Normal Form results in an algorithm for checking if w is generated by G.

CYK algorithm

John Cocke 1970

Daniel Younger 1967

Tado Kasami 1965

One naive approach is to try out all derivations with $2n-1$ steps. This is not time efficient. The CYK algorithm is a dynamic programming algorithm.

Use subproblems to solve larger problems.

Use subproblems to solve larger problems.

$\Theta(n^3)$ time.

Let $w = a_1 a_2 \dots a_n$ where each $a_i \in \Sigma$.

$w_{ij} = a_i a_{i+1} \dots a_j$. for all $1 \le i \le j \le n$.

CYK builds T_{ij} for all $1 \le i \le j \le n$, such that $T_{ij} = \{ A \mid A \overset{*}{\Rightarrow} w_{ij} \}$, leading up to T_{1n}. Finally, check if $S \in T_{1n}$. That is, $S \overset{*}{\Longrightarrow} w_{1n} = w$.

T_{11} T_{22} T_{33} .... $T_{n,n}$

T_{12} T_{23} T_{34} .. $T_{n-1,n}$

T_{13} T_{24} ... $T_{n-2,n}$

⋮

$T_{1,n-1}$ $T_{2,n}$

$T_{1,n}$

Order of computing $T_{i,j}$'s.

For $k=0$ to $n-1$, compute all $T_{(i,\,i+k)}$

For $k=0$ to $n-1$, compute all $(i, i+k)$

$k=0$. $T_{i,i} = \{ A \mid A \xRightarrow{*} w_{i,i} = a_i \}$.

All the rules $A \to a_i$

For $k>0$, $A \in T_{i,i+k}$ iff $B \in T_{i,i+j}$ and $C \in T_{i+j+1,i+k}$ and $A \to BC$ is a rule.

If $w = \varepsilon$, accept if $S \to \varepsilon$ is a rule.

For $i=1$ to n

$A \in T_{i,i} \iff A \to a_i$ is a rule.

For $k=1$ to $n-1$.

For $i=1$ to $n-k$

For $j=0$ to $k-1$

Check all rules $R \to AB$.

If $T_{i,i+j}$ contains A, and

$T_{i+j+1,i+k}$ contains B

then $T_{i,i+k} = T_{i,i+k} \cup \{R\}$.

If $S \in T_{1,n}$, say $w \in L(G)$.

If $S \in T_{1,n}$, say $w \in L(G)$.

Else $w \notin L(G)$.

Running time: $O(n^3 r)$ where r is no. of rules.

r is considered to be a constant

Correctness is evident from the algorithm.

Example:

$S \to AB \mid BC$

$A \to BA \mid a$

$B \to CC \mid b$

$C \to AB \mid a$

$w = baaba$

	1	2	3	4	5
5	SAC	SAC	B	SA	AC
4	—	B	SC	B	
3	—	B	AC		
2	SA	AC			
1	B				

Exercise: Verify this matrix.