CFG conversions

Recitation 10/30/15

PDA to CFG

The PDA *P* and CFG *G*

$P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$

First, make sure:

- P has one accept state
- P accepts by empty stack
- Each transition is either push or pop, not both at once

Intuition: G will have variables generating exactly the inputs that cause P to have the net effect of popping a stack symbol X while going from state p to state q.

- For each pair of states p and q in P, G will have a variable A_{pq} that generates all strings x that can take P from p with an empty stack to q with an empty stack
  - P's first move on x has to be push (why?)
  - P's last move on x has to be pop (why?)
- The start variable is $A_{q0qaccept}$

Two cases processing a string x

Last symbol popped is first symbol pushed...

$A_{pq} \rightarrow aA_{rs}b$

a is the first input read

b is the input read at the last move

r is the state after p

s is the state before q

...Or not

→ at some earlier point, the first symbol was popped, so the stack emptied

$A_{pq} \rightarrow A_{pr}A_{rq}$

where r is the state when the stack becomes empty

Rules

A. Add a rule $S \rightarrow [q_0Z_0f]$ for the start state, q_0, and each final state, f.

B. For each (p,ε) in $\delta(qa,A)$ add a rule $[qAp] \rightarrow a$

C. For each transition, in the PDA, that pushes a single character, such as $\delta(q,u,A) = (r,B)$ add rules of the form $[qAp] \rightarrow u[rBp]$ for all states p

D. For each state in the PDA that pushes two (or more) characters, such as $\delta(q,u,A) = (r,BC)$ add rules of the form $[qAp] \rightarrow u[rBt][tCp]$ for all possible combinations of states p and t in the machine

Hopcroft & Ullman exercise 6.3.3

Convert the PDA $P = \{(p,q),(0,1),(X,Z),\delta,q, Z\}$ to a CFG if δ is given by:

1. $\delta(q,1,Z) = \{(q,XZ)\}$
2. $\delta(q,1,X)=\{(q,XX)\}$
3. $\delta(q,0,X)=\{(p,X)\}$
4. $\delta(q,\varepsilon,X)=\{(q,\varepsilon)\}$
5. $\delta(p,1,X)=\{(p,\varepsilon)\}$
6. $\delta(p,0,Z)=\{(q,Z)\}$

Add a rule $S \longrightarrow [q_0 Z_0 f]$ for the start state, q_0, and each final state, f.

S is the start symbol

1. $S \longrightarrow [qZq]$
2. $S \longrightarrow [qZp]$

For each (p,ε) in $\delta(qa,A)$ add a rule $[qAp] \rightarrow a$

The following production comes from rule 4, $\delta(q,\varepsilon,X)=\{(q,\varepsilon)\}$

1. $[qXq] \longrightarrow \varepsilon$

The following production comes from rule 5, $\delta(p,1,X)=\{(p,\varepsilon)\}$

1. $[pXp] \longrightarrow 1$

For each transition, in the PDA, that pushes a single character, such as δ(q,u,A) = (r,B) add rules of the form [qAp] → u[rBp] for all states p

The following productions come from rule 3, δ(q,0,X)={(p,X)}

1. *[qXq]⟶0[pXq]*
2. *[qXp]⟶0[pXp]*

The following two productions come from rule 6, δ(p,0,Z)={(q,Z)}

1. *[pZq]⟶0[qZq]*
2. *[pZp]⟶0[qZp]*

For each state in the PDA that pushes two (or more) characters, such as $\delta(q,u,A) = (r,BC)$ add rules of the form $[qAp] \rightarrow u[rBt][tCp]$ for all possible combinations of states p and t in the machine

The following four productions come from rule 1, $\delta(q,1,Z) = \{(q,XZ)\}$

1. *[qZq] -> 1[qXq][qZq]*
2. *[qZq] -> 1[qXp][pZq]*
3. *[qZp] -> 1[qXq][qZp]*
4. *[qZp] -> 1[qXp][pZp]*

The following four productions come from rule 2, $\delta(q,1,X)=\{(q,XX)\}$

1. *[qXq] -> 1[qXq][qXq]*
2. *[qXq] -> 1[qXp][pXq]*
3. *[qXp] -> 1[qXq][qXp]*
4. *[qXp] -> 1[qXp][pXp]*

CNF to GNF

Review

Chomsky Normal Form

Rules of the forms

$A \rightarrow BC$

$A \rightarrow a$

where $a \in T$ and $A, B, C \in V$

B,C may not be start variable

Greibach Normal Form

Rules of the form

$A \rightarrow a\alpha$

where $\alpha \in V^*$

Construction

1. Modify the rules in R so that if $A_i \rightarrow A_j\gamma \in R$ then $j > i$

2. Starting with A_1 and proceeding to A_m this is done as follows:

(a) Assume that productions have been modified so that for $1 \leq i \leq k$, $A_i \rightarrow A_j\gamma \in R$ only if $j > i$;

(b) If $A_k \rightarrow A_j\gamma$ is a production with $j < k$, generate a new set of productions substituting for the A_j the RHS of each A_j production;

(c) Repeating (b) at most $k-1$ times we obtain rules of the form $A_k \rightarrow A_p\gamma$, $p \geq k$;

(d) Replace rules $A_k \rightarrow A_k\gamma$ by removing left-recursive rules.

Left recursion

A CFG containing rules of the form $A \rightarrow A\alpha|\beta$ is called left-recursive in A.

The language generated by such rules is of the form $A \overset{*}{\Rightarrow} \beta\alpha^n$. If we replace the rules $A \rightarrow A\alpha|\beta$ with

$A \rightarrow \beta B|\beta,\ B \rightarrow \alpha B|\alpha$

where B is a new variable, then the language generated by A is the same while no left-recursive A-rules are used in the derivation

Example

Example

Convert the CFG $G = (\{A_1, A_2, A_3\}, \{a, b\}, R, A_1)$ where $R=$

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 | b$

$A_3 \rightarrow A_1A_2 | a$

into Greibach normal form.

1. Modify the rules in R so that if $A_i \rightarrow A_j \gamma \in R$ then $j > i$

Only A_3 rules violate the condition→only A_3 rules need to be changed→

$A_3 \rightarrow \mathbf{A_1} A_2 | a$

$A_3 \rightarrow \mathbf{A_2 A_3} A_2 | a$

$A_3 \rightarrow \mathbf{A_2} A_3 A_2 | a$

A_2 has two possibilities

$A_3 \rightarrow \mathbf{A_3 A_1} A_3 A_2 | \mathbf{b} A_3 A_2 | a$

Original rules:

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1$

$A_2 \rightarrow b$

$A_3 \rightarrow A_1 A_2$

$A_3 \rightarrow a$

(d) Replace rules $A_k \rightarrow A_k\gamma$ by removing L-recursive rules.

$A_3 \rightarrow A_3A_1A_3A_2 | bA_3A_2 | a$

replace with:

$A_3 \rightarrow bA_3A_2B_3 | bA_3A_2$

$A_3 \rightarrow aB_3 | a$

$B_3 \rightarrow A_1A_3A_2B_3 | A_1A_3A_2$

All A_3 rules are done!

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 | b$

$A_3 \rightarrow A_3A_1A_3A_2 | bA_3A_2 | a$

replace the rules $A \rightarrow A\alpha | \beta$ with

$A \rightarrow \beta B | \beta$, $B \rightarrow \alpha B | \alpha$

Make A_2 rules start with terminal

$A_2 \longrightarrow \mathbf{A_3} A_1 | b$

$A_2 \longrightarrow bA_3A_2B_3A_1 | bA_3A_2A_1 | aB_3A_1 | aA_1 | b$

$A_1 \longrightarrow A_2A_3$

$A_2 \longrightarrow A_3A_1 | b$

$A_3 \longrightarrow bA_3A_2B_3 | bA_3A_2 | aB_3 | a$

$B_3 \longrightarrow A_1A_3A_2B_3 | A_1A_3A_2$

Make A_1 rules start with terminal

$A_1 \longrightarrow A_2A_3$

$A_2 \longrightarrow bA_3A_2B_3A_1|bA_3A_2A_1|aB_3A_1|aA_1|b$

$A_3 \longrightarrow bA_3A_2B_3|bA_3A_2|aB_3|a$

$B_3 \longrightarrow A_1A_3A_2B_3|A_1A_3A_2$

$A_1 \longrightarrow \mathbf{A_2}A_3$

$A_1 \longrightarrow bA_3A_2B_3A_1A_3|bA_3A_2A_1A_3|aB_3A_1A_3|aA_1A_3|bA_3$

Make B_3 start with terminal

$A_1 \rightarrow bA_3A_2B_3A_1A_3 | bA_3A_2A_1A_3 | aB_3A_1A_3 | aA_1A_3 | bA_3$

$A_2 \rightarrow bA_3A_2B_3A_1 | bA_3A_2A_1 | aB_3A_1 | aA_1 | b$

$A_3 \rightarrow bA_3A_2B_3 | bA_3A_2 | aB_3 | a$

$B_3 \rightarrow A_1A_3A_2B_3 | A_1A_3A_2$

$B_3 \rightarrow \mathbf{A_1}A_3A_2B_3$

$B_3 \rightarrow bA_3A_2B_3A_1A_3A_3A_2B_3 | bA_3A_2A_1A_3A_3A_2B_3 | aB_3A_1A_3A_3A_2B_3 | aA_1A_3A_3A_2B_3 | bA_3A_3A_2B_3$

$B_3 \rightarrow \mathbf{A_1}A_3A_2$

$B_3 \rightarrow bA_3A_2B_3A_1A_3A_3A_2 | bA_3A_2A_1A_3A_3A_2 | aB_3A_1A_3A_3A_2 | aA_1A_3A_3A_2 | bA_3A_3A_2$

Done!

$A_3 \rightarrow bA_3A_2B_3|bA_3A_2|aB_3|a$

$A_2 \rightarrow bA_3A_2B_3A_1|bA_3A_2A_1|aB_3A_1|aA_1|b$

$A_1 \rightarrow bA_3A_2B_3A_1A_3|bA_3A_2A_1A_3|aB_3A_1A_3|aA_1A_3|bA_3$

$B_3 \rightarrow bA_3A_2B_3A_1A_3A_3A_2B_3|bA_3A_2A_1A_3A_3A_2B_3|aB_3A_1A_3A_3A_2B_3|aA_1A_3A_3A_2B_3|bA_3A_3A_2B_3|bA_3A_2B_3A_1$
$A_3A_3A_2|bA_3A_2A_1A_3A_3A_2|aB_3A_1A_3A_3A_2|aA_1A_3A_3A_2|bA_3A_3A_2$