Context Free Languages 2 of 3

Ex 1:

$\Sigma = \{0,1\}$, $L = \{x \in \Sigma^* : \#_{00}(x) = \#_{11}(x)\}$

Design:

- S generates L
- Add 4 variables $A_{00}, A_{01}, A_{10}, A_{11}$, where A_{ij} generates $\{x \in L : x \text{ starts with } i, \text{ ends with } j\}$

We'll use the L2R Method.

$S \to \varepsilon, A_{00}, A_{01}, A_{10}, A_{11}$
$A_{00} \to 0, 0A_{01}A_{10}, 0A_{10}$
$A_{01} \to 0A_{01}A_{11}, 0A_{11}$
$A_{10} \to 1A_{10}A_{00}, 1A_{00}$
$A_{11} \to 1, 1A_{10}A_{01}, 1A_{01}$

Theorem: (8.1)

For every regex R, there's a CFG G s.t. $\mathcal{L}(G) = \mathcal{L}(R)$

Proof: By structural induction

If $R = \phi$, use no production
If $R = \varepsilon$, use $S \to \varepsilon$
If $R = a$, use $S \to a$
where $a \in \Sigma$

If $R = R_1 + R_2$, then add $S \to S_1, S_2$
If $R = R_1 R_2$, then add $S \to S_1 S_2$
If $R = R_1^*$, then add $S \to \varepsilon, S_1 S$

Definition: (8.4)

A CFG $G = (V, \Sigma, P, S)$ is right linear (r.l.) iff every production has form

$$A \to \varepsilon \text{ or } A \to xB \text{, where } A, B \in V \text{ and } x \in \Sigma^*$$

Also, a CFG $G = (V, \Sigma, P, S)$ is strict right linear (s.r.l.) iff every production has form

$$A \to \varepsilon \text{ or } A \to xB \text{, where } A, B \in V \text{ and } x \in \Sigma^* \text{ and } |x| \le 1$$

Theorem: (8.3)

If $L = \mathcal{L}(G)$ for some s.r.l CFG G, then L is regular.

Theorem: (8.4)

Every regular language can be generated by a s.r.l. CFG.

Proof: (Sketch for 8.3)

Let $G = (V, \Sigma, P, S)$ be a s.r.l CFG.

We construct an NFSA $M = (Q, \Sigma, \delta, s, F)$ s.t. $\mathcal{L}(M) = \mathcal{L}(G)$

Let $Q = V$
$s = S$
$F = \{A \in V : A \to \varepsilon \in P\}$
$\delta = \{B : A \to aB \in P\}$ $\qquad \delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$

$\delta(A, a)$

Then we can prove that M is regular.

Proof: (Sketch for 8.4)

Let $M = (Q, \Sigma, \delta, s, F)$ be a DFSA.

We construct a s.r.l CFG $G = (V, \Sigma, P, S)$ s.t. $\mathcal{L}(G) = \mathcal{L}(M)$.

Let $V = Q$
$S = s$
$P = \{A \to aB : A, B \in V,\ a \in \Sigma \text{ and } \delta(A, a) = B\} \cup \{A \to \varepsilon : A \in F\}$

Corollary: (8.6)

For every r.l. CFG G, there's a s.r.l CFG G' s.t. $\mathcal{L}(G') = \mathcal{L}(G)$

Proof: (Sketch for 8.6)

Look at "Illegal" productions

$A \to a_1 a_2 \cdots a_k B \quad , A, B \in V,\ a_i \in \Sigma,\ k \geq 2$

Replace above with:

$A \to a_1 B_1$
$B_1 \to a_2 B_2$
$B_{k-2} \to a_{k-1} B_{k-1}$
$B_{k-1} \to a_k B$

Theorem: The BIG Result 2.0

Let L be a language. Then the following are equivalent.

(i) $L = \mathcal{L}(R)$ for some regex R
(ii) $L = \mathcal{L}(M)$ for some DFSA M
(iii) $L = \mathcal{L}(M)$ for some NFSA M
(iv) $L = \mathcal{L}(G)$ for some s.r.l CFG G
(v) $L = \mathcal{L}(G)$ for some r.l CFG G

Definition: Pushdown Automata (PDA, section 8.5)

A PDA is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where

- ↳ Q is a finite set of states
- ↳ Σ is the input alphabet
- ↳ Γ is the stack alphabet
- ↳ δ is the transition function
- ↳ q_0 is the initial state
- ↳ $F \subseteq Q$ is the set of accepting state.

$\delta : Q \times (\Sigma \times \{\varepsilon\}) \times (\Gamma \times \{\varepsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$

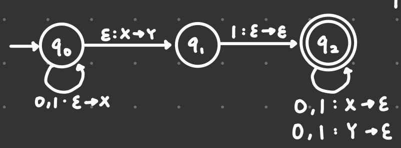
Compared to a NFSA, only the transitions look different. Suppose $(q', Y) \in \delta(q, a, X)$, where $q, q' \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $X, Y \in \Gamma \cup \{\varepsilon\}$

4 possibilities for X and Y (for $\xrightarrow{a : X \to Y}$)

(i) $X = \varepsilon, Y \in \Gamma$ - Push Y onto stack
(ii) $X \in \Gamma, Y = \varepsilon$ - Pop X from stack
(iii) $X, Y \in \Gamma$ - Replace X with Y at top of stack
(iv) $X = Y = \varepsilon$ - Leave stack unchanged

transition can be taken only if X is on top of stack.

Ex 2: (of a PDA)



M

$\mathcal{L}(M) = \{ x \in \Sigma^* : \exists\, v, w \in \Sigma^* \text{ s.t. } |v| = |w| > 0 \text{ and } x = v1w \}$

Definition:

A PDA M accepts a string x iff M can

(i) read all of x,
(ii) end in an accepting state, and
(iii) end with its stack empty

Configuration: Describe the computation of a PDA M after a transition.

(q, x, α),

$q \to$ current state $(q \in Q)$

$x \to$ Portion of input not read yet $(x \in \Sigma^*)$

$\alpha \to$ Stack content $(\alpha \in \Gamma^*)$. Convention: $\alpha = \underset{\uparrow \text{top of stack}}{x_1} x_2 \cdots x_k$

Definition:

Let C, C' be configs

$C \vdash C'$ means M can go from C to C' by taking one transition.

\vdash

$C \vdash^* C'$ means M can go from C to C' by taking zero or more transitions.

M accepts x means $\underbrace{(q_0, x, \varepsilon)}_{\text{initial config}} \vdash^* \underbrace{(q, \varepsilon, \varepsilon)}_{\text{accepting config}}$, where $q \in F$

$\underbrace{C_0 \vdash C_1 \vdash \ldots \vdash C_k}_{\text{accepting computation}}$

Definition:

The language of a PDA M is.

$$\mathcal{L}(M) = \{ x \in \Sigma^* \cdot M \text{ accepts } x \}$$