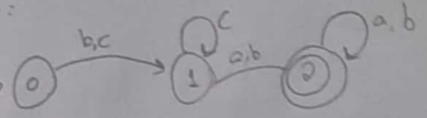
Compiladores - Revisão de LFA - Parte 1

(1) O autômato é um AFD, pois não há transições vazias, cada símbolo leva a um único estado. Ou seja, há somente um caminho e ele é bem definido

Expressão Regular: $ab^*c^*cc^*$
$= a(b|c)^*c^+$

(2) AFD:



Gramática Regular: $G(L) = \{(0,1,2), \{a,b,c\}, 0, (2), \delta\}$

δ: $S \to aA$
$A \to aA$
$A \to bA$
$A \to cB$
$B \to cB$
$B \to \varepsilon$

$\equiv$ $S \to aA$
$A \to aA \mid bA \mid cB$
$B \to cB \mid \varepsilon$

Expressão Regular do complemento
$(b|c)c^*(b|a)^+$ → caso que não começam com a

(3) a. L_1: Linguagem regular (Tipo 3) → Min: AFD
b. L_2: Linguagem livre de contexto (Tipo 2) → Min: AP
c. L_3: Linguagem sensível ao contexto → Min: Mag. turing
d. L_4: Linguagem regular (Tipo 3) → Min: AFD

Gramática L_1
$S \to aA$
$A \to aA \mid bB$
$B \to aC$
$C \to aD$
$D \to aD \mid \varepsilon$

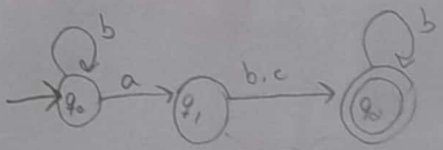
Gramática L_2
$S \to aSa \mid B$
$B \to bB \mid b$

Gramática L_3
$L = \{a^n b^n c^n \mid n \geq 1\}$
$S \to abc$
$S \to aBSc$
$Ba \to aB$
$Bc \to bc$
$Bb \to bb$

(4) $S \to bS \mid aB \mid aC$

δ: $B \to bD$
$D \to bD \mid \varepsilon$
$C \to cE$
$E \to bE \mid \varepsilon$

AFD:



$G = (\{q_0, q_1, q_2\}, \{a,b,c\}, (q_0), (q_2), \delta)$

(5) União:

Dado que L_1 e L_2 são regulares, provar que $L_1 \cup L_2$ é regular

Então, dado os dois AFD's:

$M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$

$M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$

Basta construir um autômato AFD J tal que:

$J = (Q_3, \Sigma, \delta_3, q_0^3, F_3)$ onde:

$Q_3 = Q_1 \times Q_2$

$q_0^3 = (q_0^1, q_0^2)$

$F_3 = \{(p,q) \in Q_1 \times Q_2 \mid p \in F_1 \vee q \in F_2\}$

$\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a))$ para todo $a \in \Sigma$

Conclusão: Como M é um AFD que reconhece $L_1 \cup L_2$, então $L_1 \cup L_2$ é regular.

Intersecção:

Mesmo raciocínio do plano cartesiano anterior; Dado L_1 e L_2, vamos ver se $L_1 \cap L_2$ é regular:

$Q_3 = Q_1 \times Q_2$

$q_0^3 = (q_0^1, q_0^2)$

$F_3 = \{(p,q) \in Q_1 \times Q_2 \mid p \in F_1 \wedge q \in F_2\}$

$\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a))$

↳ Esse AFD aceita uma cadeia somente se ambos os originais aceitarem

Conclusão: É possível construir um AFD que reconhece $L_1 \cap L_2$, portanto é fechada para Intersecção

Complemento

Seja L regular, há um AFD:

$M = (Q, \Sigma, \delta, q_0, F)$

Tal que $L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$

Então Construir um novo AFD:

$M' = (Q, \Sigma, \delta, q_0, Q/F)$

↳ estados finais trocados: todos os estados que não eram finais em M

Intuitivo: $w \in L(M') \longleftrightarrow \delta^*(q_0, w) \in Q/F$

$w \notin L(M) \longleftrightarrow \delta^*(q_0, w) \notin F \longleftrightarrow$

$\delta^*(q_0, w) \in Q/F$

Portanto $L(M') = \neg L(M)$

Compiladores - Revisão de LFA - Parte 2

5) $G=(V, \Sigma, P, S)$ onde:

$V = \{S, A, B\}$

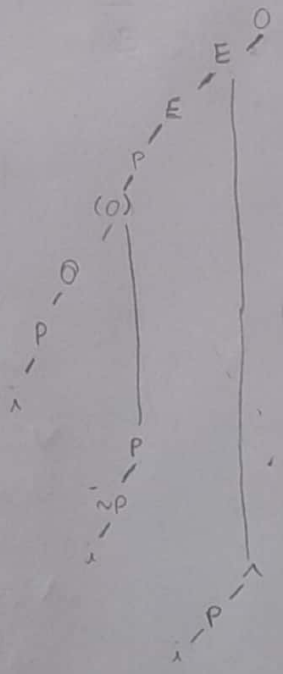
$\Sigma = \{0, 1\}$

$$P = \begin{cases} S \to A \mid B \\ A \to 0A1 \mid 01 \\ B \to 0B11 \mid 011 \end{cases}$$

---

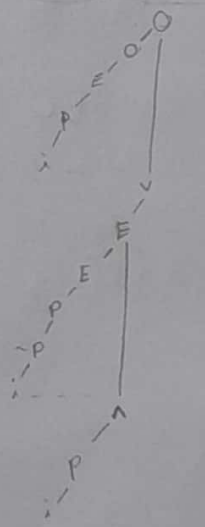
7) a)
- $0 \to E$: E
- $E \to E \wedge P$: $E \wedge P$
- $E \to P$: $P \wedge P$
- $P \to (0)$: $(0) \wedge P$
- $0 \to 0 \vee E$: $(0 \vee E) \wedge P$
- $0 \to P$: $(P \vee E) \wedge P$
- $P \to i$: $(i \vee E) \wedge P$
- $E \to P$: $(i \vee P) \wedge P$
- $P \to \sim P$: $(i \vee \sim P) \wedge P$
- $P \to i$: $(i \vee \sim i) \wedge P$
- $P \to i$: $(i \vee \sim i) \wedge i$

Árvore



Conclusão: Pode ser gerada

7c) $i \vee \sim i \wedge i$
- $0 \to 0 \vee E$: $0 \vee E$
- $0 \to E$: $E \vee E$
- $E \to P$: $P \vee E$
- $P \to i$: $i \vee E$
- $E \to E \wedge P$: $i \vee E \wedge P$
- $E \to P$: $i \vee P \wedge P$
- $P \to \sim P$: $i \vee \sim P \wedge P$
- $P \to i$: $i \vee \sim i \wedge P$
- $P \to i$: $i \vee \sim i \wedge i$



7b) $i \sim i \vee i$

- $0 \to 0 \vee E$: $0 \vee E$

Não há como 0 gerar $i \sim i$

7d) $(i \vee \sim i \wedge)$.

Não há como gerar
) após ^ pois a única transição
aceita é $E \wedge P$ e não há transição à esquerda
de E e/ou P que posicionem ( ) como
desejado

(9) Dado que:

$S \to SS+$

$S \to SS*$

$S \to a$

Cadeia: $aa+a*$

- $S \to SS*$: $SS*$
- $S \to SS+$: $SS+S*$
- $S \to a$: $aS+S*$
- $S \to a$: $aa+S*$
- $S \to a$: $aa+a*$ ↗ Reconhecido

Cadeia: $a+a$

↳ não há S que gera a depois de + sem haver *, então não é aceito na linguagem.

(11) $L = \{a^k b^n c^m \mid n > m+k\}$

$G = (V, \Sigma, P, S)$

$V = \{A, B, C\}$

$\Sigma = \{a, b, c\}$

P =
$S \to ABC$
$A \to aAb \mid \varepsilon$
$C \to bCc \mid \varepsilon$
$B \to bB \mid b$

8) Procedência:

$O \to O \vee E$ (3^o)
$O \to E$
$E \to E \wedge P$ (2^o)
$E \to P$
$P \to (O)$
$P \to \sim P$ (1^o)
$P \to ;$

↓

Sequência de procedência: $\vee$ → menor
$\wedge$ → intermediário
$\sim$ → maior

Níveis de profundidade:
O → mais superficial
E → intermediário
P → mais profundo

Associatividade (com base na recursão)

- $\vee$ é produzido por $O \to O \vee E$
  - ↳ Recursão: à esquerda
  - ↳ Assoc: Esquerda
- $\wedge$ é produzido por $E \to E \wedge P$
  - ↳ Recursão: à esquerda
  - ↳ Assoc: Esquerda
- $\sim$ é produzido por $\sim P$
  - ↳ Recursão: Direita
  - ↳ Assoc: Direita

Compiladores - Revisão de LFA - Parte 3

(11) R: Uma gramática é ambígua quando existe pelo menos uma cadeia da linguagem que ela gera e que pode ser derivada de duas ou mais maneiras distintas, resultando em duas árvores sintáticas distintas.

(12) Dada a gramática

$P \to AB$
$A \to aAb \mid c$
$B \to bBc \mid a$

a) Derivação para acbbbacc

- $P \to AB$: AB
- $A \to aAb$: $aAbB$
- $A \to c$: $acbB$
- $B \to bBc$: $acbbBc$
- $B \to bBc$: $acbbbBcc$
- $B \to a$: $acbbbacc$

(12b) Para $P \to AB$

$(a^m c b^m)$ $(b^x a c^x)$

$\therefore L(G) = \{a^m c b^m b^x a c^x \mid n \geq 0, m \geq 0\}$

(13) Solução: Criar dois não-terminais
<Cmd> para comandos completos (sem else pendente)
<Cmd2> para comandos incompletos (

G: <Cmd> if <Expr> then <Cmd> else <Cmd> | c

<Cmd2> if <Expr> then <Cmd> | if <Expr> then <Cmd2> else <Cmd2>

<Expr> b

Agora a else está sempre associado ao if mais próximo que ainda não tinha um else