4 / 9 / 8

Seja $A = (Q, \Sigma, \delta, s, F)$ um afnd $(\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q)$

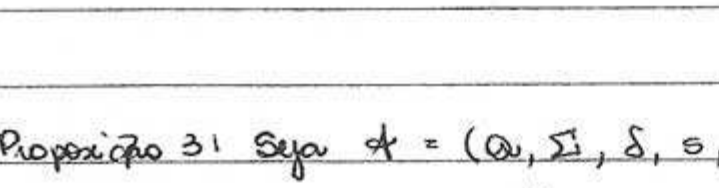
Como estender δ para $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ tal que $\forall q \in Q$, $\forall x \in \Sigma^*$,

$\hat{\delta}(q, x) = \{p \in Q :$ existe em $G(A)$ um passeio $P : q \rightsquigarrow p\}$?

$\forall q \in Q$, λ-fecho$(q) = \{p \in Q :$ existe em $G(A)$ um passeio $P : q \overset{\lambda}{\rightsquigarrow} p\}$

↓ que pode ser o passeio Trivial

Exemplo: $L(a^* b^* c^*)$



λ-fecho$(q_0) = \{q_0, q_1, q_2\}$

λ-fecho$(q_1) = \{q_1, q_2\}$

λ-fecho$(q_2) = \{q_2\}$

$\forall K \subseteq Q$, λ-fecho$(K) = \bigcup_{q \in K} \lambda$-fecho$(q)$.

Vamos definir $\hat{\delta} = Q \times \Sigma^* \to 2^Q$, indutivamente por:

i) $\forall q \in Q$, $\hat{\delta}(q, \lambda) = \lambda$-fecho$(q)$

ii) $\forall q \in Q$, $\forall x \in \Sigma^*$, $\forall \sigma \in \Sigma$, $\hat{\delta}(q, x\sigma) = \lambda$-fecho$(\delta(\hat{\delta}(q, x), \sigma))$

Observações:

① Podemos estudar δ e $\hat{\delta}$ para subconjunto de estados

$\forall K \subseteq Q$

$$\begin{cases} \delta(K, \sigma) = \bigcup_{q \in K} \delta(q, \sigma), \ \forall \sigma \in (\Sigma \cup \{\lambda\}) \\ \hat{\delta}(K, x) = \bigcup_{q \in K} \hat{\delta}(q, x), \ \forall x \in \Sigma^* \end{cases}$$

② $\forall q \in Q$, $\forall \sigma \in (\Sigma \cup \{q\})$

$\hat{\delta}(q, \sigma)$ pode ser diferente de $\delta(q, \sigma)$

Proposição 3: Seja $A = (Q, \Sigma, \delta, s, F)$ um afnd

$\forall p, q \in Q$, $\forall x \in \Sigma^*$, existe em $G(A)$ um passeio $P : p \overset{x}{\rightsquigarrow} q$

se e somente se $q \in \hat{\delta}(p, x)$

Corolário 4: Seja $A = (Q, \Sigma, \delta, s, F)$ um afnd

$L(A) = \{ w \in \Sigma^* : \hat{\delta}(s, w) \cap F \neq \emptyset \}$

Lema 5: Para cada afnd A com λ-transições, existe um afnd B em λ-transições tal que $L(A) = L(B)$

Algoritmo para determinar λ-fecho (K)

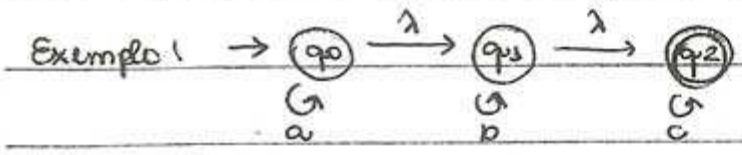
- Recebe { um afnd $A = (Q, \Sigma, \delta, s, F)$, onde $\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$.
  um subconjunto de estados $K \subseteq Q$
- Determina : Fecho = λ-fecho (K)

```
Fecho ← ∅
Inicializa Fila (Fila);
para (cada estado q ∈ K) faça                    } O(|Q|)
    Fecho ← Fecho ∪ {q};
    Insere Fila (Fila, q);
enquanto (não Fila Vazia (Fila)) faça            } O(|Q|²)
    Remove Fila (Fila, q);
    para (cada estado p ∈ δ(q, λ)) faça          } O(|Q|)
        se (p ∉ Fecho)
            então
                Fecho ← Fecho ∪ {p}
                Insere Fila (Fila, p);
devolve Fecho
```

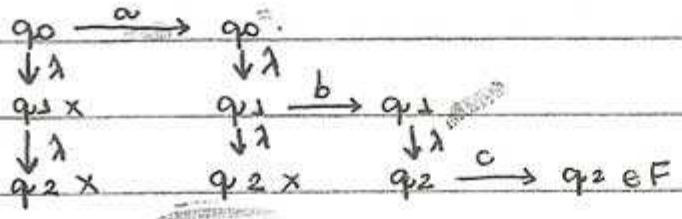
Observação: O consumo de tempo do algoritmo no pior caso é $O(|Q|^3)$

/ /

Exemplo 1:



$x = abc$



Algoritmo para verificar se x é aceita por A

- Recebe { uma palavra $x \in \Sigma^*$
  uma afnd $A = (Q, \Sigma, \delta, s, F)$
- Verifica se x é aceita por A

$K \leftarrow \lambda\text{-fecho}(s)$; $\quad O(|Q|^2)$

entrada $\leftarrow x$,

enquanto (não leu toda entrada e $K \neq \emptyset$) faça

$\sigma \leftarrow$ leia o próximo símbolo de entrada

$K \leftarrow \lambda\text{-fecho}(\delta(K, \sigma))$; $\quad \rightarrow O(|Q|^2)$

$\delta(K,\sigma) = \bigcup_{q \in K} (q, \sigma)$

se $(K \cap F \neq \emptyset)$ $O(|Q|^2)$

então x é aceita por A

senão x não é aceita por A

Observação! No pior caso, o consumo de tempo desse algoritmo é $O(|x| \cdot |Q|^2)$

/ /

NRec(Σ) = Rec(Σ)

A : $(Q_A, \Sigma, \delta_A, s_A, F_A)$ afd $\delta_A : Q_A \times \Sigma \to Q_A$

$B = (Q_A, \Sigma, \delta_B, s_A, F_A)$ afnd

$\forall q \in Q_A$, $\delta_B(q, \sigma) = \{\delta_A(q, \sigma)\}$ $\forall q \in Q_A$, $\delta_B(q, \lambda) = \emptyset$

$\forall \sigma \in \Sigma$