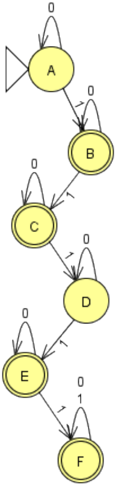


Teoria de Linguagens e Compiladores

Linguagens Regulares

Luiz Eduardo da Silva

Universidade Federal de Alfenas

Agenda

Agenda

O que é um computador?

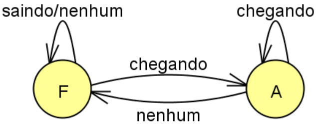
- Para construir uma Teoria Matemática, os computadores reais são complicados (arquitetura, sistema operacional, linguagens de programação, aplicativos).
- Em vez disso, usamos um **modelo computacional**.

Figura: Porta automática

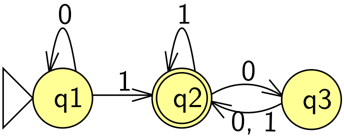
O que é um computador?

- Para construir uma Teoria Matemática, os computadores reais são complicados (arquitetura, sistema operacional, linguagens de programação, aplicativos).
- Em vez disso, usamos um **modelo computacional**.

Figura: Porta automática

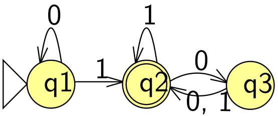


Exemplo



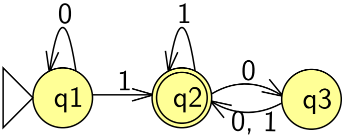
- A figura representa o **diagrama de estados de um AF**
- q_1, q_2 e q_3 são os estados
- O estado q_1 ligado ao triângulo é o **estado inicial**
- O estado q_2 com círculo duplo é o **estado de aceitação**
- As setas representam as **transições** entre estados.

Funcionamento



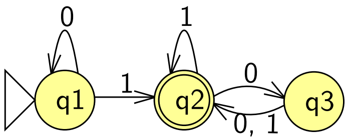
- O AF começa no estado inicial e processa uma entrada, por exemplo 1101.
- Cada símbolo é processado e executada a transição, por exemplo, se a entrada é 1 e o estado atual é q_1, o AF vai para o estado q_2
- O AF aceita ou rejeita a entrada. Se o AF para num estado de aceitação após processar todos os símbolos da entrada, essa é aceita, caso contrário o AF rejeita a cadeia de entrada.

processamento de uma cadeia



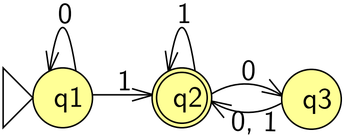
- Seja a cadeia de entrada $w = 1101$.

processamento de uma cadeia



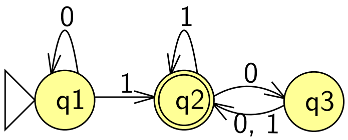
- Seja a cadeia de entrada $w = 1101$.
  1. O AF começa no estado q_1

processamento de uma cadeia



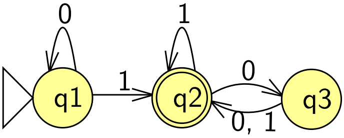
- Seja a cadeia de entrada $w = 1101$.
  1. O AF começa no estado q_1
  2. Lê 1 e vai para o estado q_2

processamento de uma cadeia



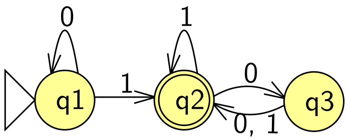
- Seja a cadeia de entrada $w = 1101$.
  1. O AF começa no estado q_1
  2. Lê 1 e vai para o estado q_2
  3. Lê 1 e continua no estado q_2

processamento de uma cadeia



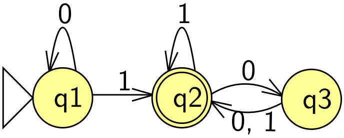
- Seja a cadeia de entrada $w = 1101$.
  1. O AF começa no estado q_1
  2. Lê 1 e vai para o estado q_2
  3. Lê 1 e continua no estado q_2
  4. Lê 0 e vai para o estado q_3

processamento de uma cadeia



- Seja a cadeia de entrada $w = 1101$.
  1. O AF começa no estado q_1
  2. Lê 1 e vai para o estado q_2
  3. Lê 1 e continua no estado q_2
  4. Lê 0 e vai para o estado q_3
  5. Lê 1 e volta para o estado q_2

processamento de uma cadeia



- Seja a cadeia de entrada $w = 1101$.
  1. O AF começa no estado q_1
  2. Lê 1 e vai para o estado q_2
  3. Lê 1 e continua no estado q_2
  4. Lê 0 e vai para o estado q_3
  5. Lê 1 e volta para o estado q_2
  6. O AF **aceita** a cadeia $w = 1101$

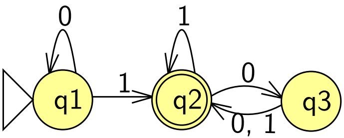
Definição

Um **autômato finito** M é uma 5-upla, $M = (Q, \Sigma, \delta, q_0, F)$

Onde:

1. Q é o conjunto finito de **estados**
2. Σ é o conjunto finito de símbolos, o **alfabeto**
3. $\delta : Q \times \Sigma \rightarrow Q$ é a **função de transição**
4. $q_0 \in Q$ é o **estado inicial**
5. $F \subseteq Q$ é o conjunto de **estados finais**.

Definição formal de um AF



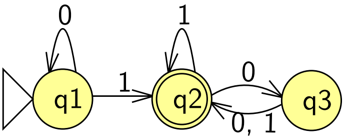
Podemos definir formalmente esse AF como:

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ é descrita como:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

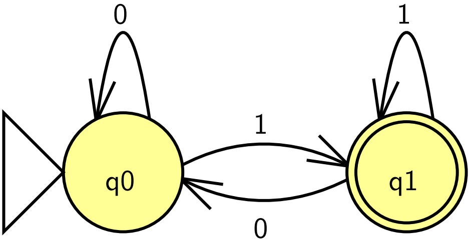
4. q_1 é o **estado inicial**
5. $F = \{q_2\}$

funcionamento

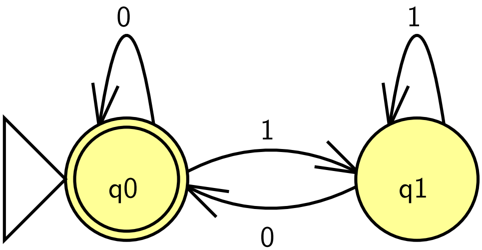


- Seja A o conjunto de todas as cadeias que a máquina M aceita.
- A é chamada de **linguagem da máquina M**, $L(M) = A$.
- M **reconhece/aceita** A.
- Nesse exemplo, $A = \{w | w$ contém pelo menos um símbolo 1 e um número par de 0s segue o último 1$\}$

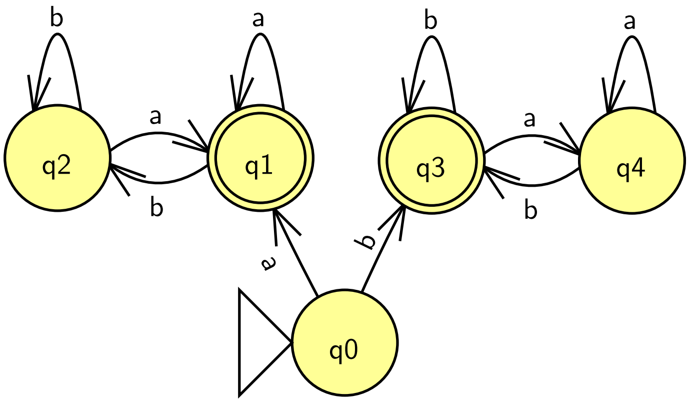
Exemplo1



Exemplo2



Exemplo3



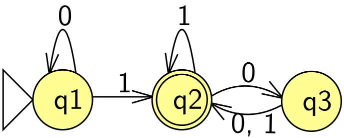
Computação do Autômato

Definição

Para representar a computação do autômato podemos usar duas definições (Vieira,2006):

- *A* **configuração instantânea**, *representada pelo par* $[e, w]$, *onde* e *é o estado atual e* w *a palavra a ser computada num dado instante;*
- *A relação* $\vdash$ *que mostra a* **transformação** *da configuração instantânea durante a computação da palavra* w.
  - $[q_0, aw] \vdash [q_1, w]$ *se existe uma transição de* q_0 *para* q_1 *para o símbolo a no autômato.*

Exemplo



- Considerando o autômato da Figura e a palavra $w = 1101$, tem-se:

$$[q_1, 1101] \vdash [q_2, 101] \vdash [q_2, 01] \vdash [q_3, 1] \vdash [q_2, \varepsilon]$$

- De forma genérica, a linguagem L reconhecida por um autômato M pode ser formalmente definida como:

$$L(M) = \{w \in \Sigma^* | [q_0, w] \overset{*}{\vdash} [f, \varepsilon], f \in F\}$$

Outra formalização

Definição (Vieira,2006)

- Seja um AFD $M = (Q, \Sigma, \delta, q_0, F)$. A **função de transição estendida** para M, $\hat{\delta} : Q \times \Sigma^* \to Q$, definida recursivamente como:
  - $\hat{\delta}(q, \varepsilon) = q$
  - $\hat{\delta}(q, ay) = \hat{\delta}(\delta(q, a), y)$, para todo $a \in \Sigma$, $y \in \Sigma^*$ e $q \in Q$

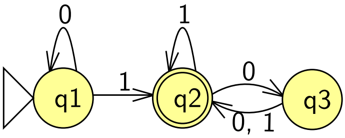
Outra formalização

Definição (Vieira,2006)

- Seja um AFD $M = (Q, \Sigma, \delta, q_0, F)$. A **função de transição estendida** para M, $\hat{\delta} : Q \times \Sigma^* \to Q$, definida recursivamente como:
  - $\hat{\delta}(q, \varepsilon) = q$
  - $\hat{\delta}(q, ay) = \hat{\delta}(\delta(q, a), y)$, para todo $a \in \Sigma$, $y \in \Sigma^*$ e $q \in Q$

Para $w = 001$, temos:

$$\begin{aligned}
\hat{\delta}(q_1, 001) &= \hat{\delta}(\delta(q_1, 0), 01) \\
&= \hat{\delta}(q_1, 01) \\
&= \hat{\delta}(\delta(q_1, 0), 1) \\
&= \hat{\delta}(q_1, 1) \\
&= \hat{\delta}(\delta(q_1, 1), \varepsilon) \\
&= \hat{\delta}(q_2, \varepsilon) \\
&= q_2
\end{aligned}$$



Implementação

- Essa computação pode ser implementada de forma simples. Seja T, a **tabela de transição**, onde as linhas representam estados e as colunas os símbolos do vocabulário.
- A implementação (em pseudocódigo) fica:

```
i = 0;
estado = 1;
while (entrada[i]) {
   estado = T[estado, entrada[i]];
   i++;
}
```

T	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

- Outra definição de linguagem para máquina M, considerando a operação de transição estendida:

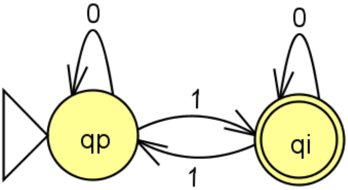
$$L(M) = \{w \in \Sigma^* | \hat{\delta}(q_0, w) \in F\}$$

Projeto 1

Construir um autômato que reconheça cadeia de 0s e 1s com quantidade ímpar de 1s (em qualquer ordem)

Projeto 1

Construir um autômato que reconheça cadeia de 0s e 1s com quantidade ímpar de 1s (em qualquer ordem)

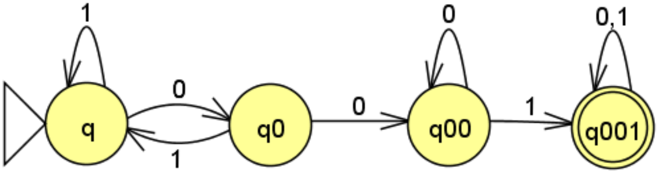


Projeto 2

Construir um autômato para reconhecer uma linguagem regular de todas as cadeias que contém a subcadeia **001**.

Projeto 2

Construir um autômato para reconhecer uma linguagem regular de todas as cadeias que contém a subcadeia **001**.



Na aritmética temos os operadores (soma, subtração, multiplicação, divisão) que são utilizados para expressar valores numéricos. Na teoria da computação, os objetos são **linguagens** e as operações utilizadas são as **operações regulares**: de concatenação (representada pelo símbolo $\circ$), união (representado pelo símbolo $\cup$) e fecho de Kleene (representado pelo símbolo $*$) para definir linguagens regulares.

Definição

Sejam A e B linguagens. As operações regulares de União, Concatenação e Fecho de Kleene (estrela) são definidas da seguinte forma:

- *União:* $A \cup B = \{x | x \in A \text{ ou } x \in B\}$.
- *Concatenação:* $A \circ B = \{xy | x \in A \text{ e } y \in B\}$.
- *Fecho de Kleene:* $A^* = \{x_1 x_2 ... x_k | k \geq 0 \text{ e } x_i \in A\}$.

Exemplo

Seja $\Sigma = \{a, b, c, ..., z\}$ o alfabeto. Sejam $A = \{alan, beto\}$ e $B = \{alto, baixo\}$

- **União**: $A \cup B = \{alan, beto, alto, baixo\}$.
- **Concatenação**:
  $A \circ B = \{alanalto, alanbaixo, betoalto, betobaixo\}$.
- **Fecho de Kleene**:
  $A^* = \{\varepsilon, alan, beto, alanalan, alanbeto, betoalan, betobeto, alanalanalan, alanalanbeto, alanbetoalan, ...\}$.

Definições

Seja $\mathcal{N} = \{0, 1, 2, 3, ...\}$ o conjunto dos números naturais. Dizemos que $\mathcal{N}$ é **fechada sob multiplicação**, pois dados quaisquer dois números naturais x e y, o produto $x \times y$ também está em $\mathcal{N}$.

Teorema

A classe de linguagens regulares é **fechada** *sob a operação de união.*

Ou seja, se A_1 e A_2 são linguagens regulares, então $A = A_1 \cup A_2$ também é uma linguagem regular.

Prova (por construção)

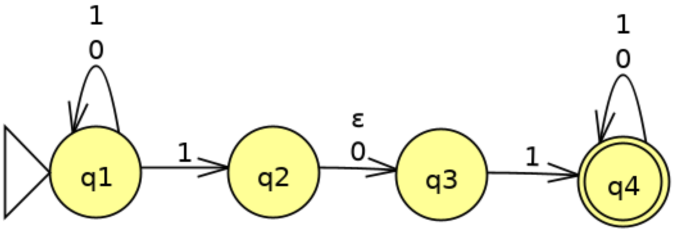
- Sejam $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ e $M2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, autômatos que reconhecem A_1 eA_2, respectivamente.
- Construa $M = (Q, \Sigma, \delta, q_0, F)$ para reconhecer $A_1 \cup A_2$ da seguinte forma:
  1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ e } r_2 \in Q_2\}$.
     Q é o **produto cartesiano**, $Q = Q_1 \times Q_2$.
  2. Σ é o mesmo para M_1 e M_2. Mas poderiam ser diferentes.
  3. Para cada $(r_1, r_2) \in Q$ e cada $a \in \Sigma$, faça:
     $$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$
  4. $q_0 = (q_1, q_2)$
  5. $F = \{(r_1, r_2) | r_1 \in F_1 \text{ ou } r_2 \in F_2\}$.
     Essa expressão é a mesma que $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.
     Não é o mesmo que $F = F_1 \times F_2$ (isso é a **interseção** de duas Máquinas).

Agenda

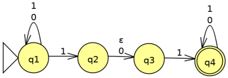
Exemplo

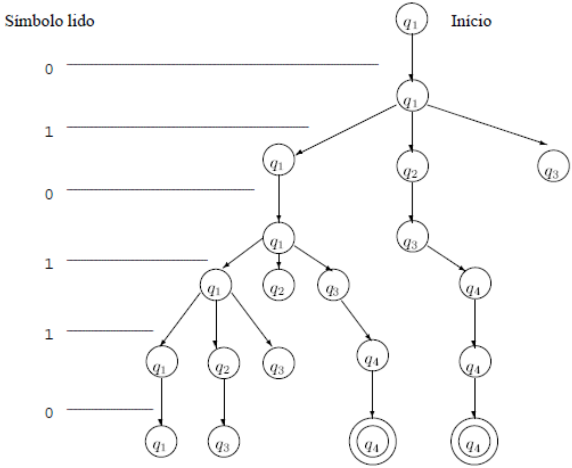
- Quando o passo a ser realizado no autômato é um único, chamamos de **computação determinística**.
- Na **computação não determística**, várias escolhas podem acontecer no próximo passo. Inclusive transições ε ou λ que podem ser realizadas sem consumir nenhum símbolo da cadeia de entrada.

Exemplo de AFN:

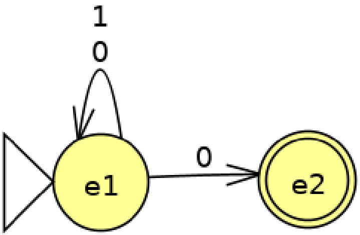


Computação em paralelo





Exemplo 2



$$\begin{array}{llllllllll}
[e_1, 1010] & \vdash & [e_1, 010] & \vdash & [e_1, 10] & \vdash & [e_1, 0] & \vdash & [e_1, \lambda] \\
 & & & \llcorner & [e_2, 10] & & & \llcorner & [e_2, \lambda]
\end{array}$$

Exemplo 3

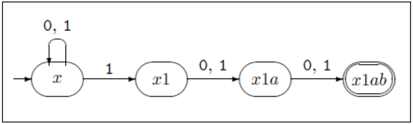


Figura: um afn para cadeias finalizadas por 100, 101, 110 ou 111

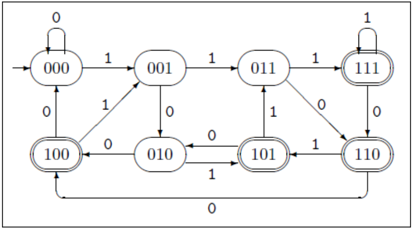


Figura: O afd equivalente

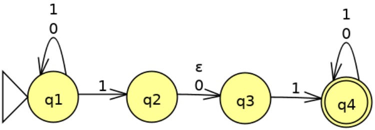
Definição

Um **autômato finito não determinístico** M é uma 5-upla, $M = (Q, \Sigma, \delta, q_0, F)$

Onde:

1. Q é o conjunto finito de **estados**
2. Σ é o conjunto finito de símbolos, o **alfabeto**
3. $\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$ é a **função de transição**
4. $q_0 \in Q$ é o **estado inicial**
5. $F \subseteq Q$ é o conjunto de **estados finais**.

Exemplo



1. $Q = \{q_1, q_2, q_3, q_4\}$
2. $\Sigma = \{0, 1\}$
3. $\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$

δ	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

4. q_1 é o **estado inicial**
5. $F = \{q_4\}$

Equivalência

Para todo **Autômato Finito Não-Determinístico (AFN)** existe um **Autômato Finito Determinístico(AFD)** correspondente.

Equivalência

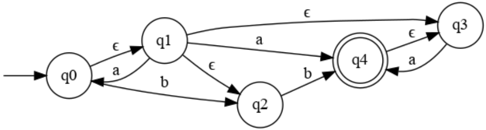
Para todo **Autômato Finito Não-Determinístico (AFN)** existe um **Autômato Finito Determinístico(AFD)** correspondente.

- A demonstração pode ser realizada através da simulação da execução em paralelo da computação do AFN.
- Basicamente, ao invés de mudar para um único estado a cada transição, no AFN poderemos nos deslocar para um conjunto de estados possíveis, dada o não-determinismo de algumas transições.
- Existem $P(Q)$ subconjuntos de estados possíveis para um conjunto Q de estados. Assim se $Q = \{1, 2, 3\}$, $P(Q) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$. Ou seja, para um AFN com k estados, teremos uma AFD <u>com até</u> 2^k estados correspondentes.

Fecho-lâmbda

Para todo **Autômato Finito Não-Determinístico (AFN)** existe um **Autômato Finito Determinístico(AFD)** correspondente.

- Seja $R \in P(Q)$, definimos a função **E** (denominada **fecho-lambda** em Vieira, 2006) da seguinte forma: $E(R) = \{q | q$ pode ser atingido a partir dos estados $r \in R$ através de zero ou mais transições $\varepsilon\}$. Formalmente, $E(R) = \{q | [r, \varepsilon] \vdash^* [q, \varepsilon], r \in R\}$. Exemplo:



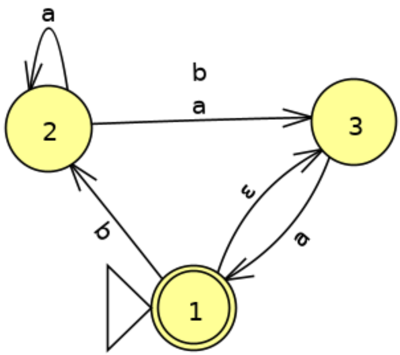
- $E(\{q_0\}) = \{q_0, q_1, q_2, q_3\}$
- $E(\{q_1, q_2\}) = \{q_1, q_2, q_3\}$
- $E(\{q_2\}) = \{q_2\}$

Fecho-lâmbda

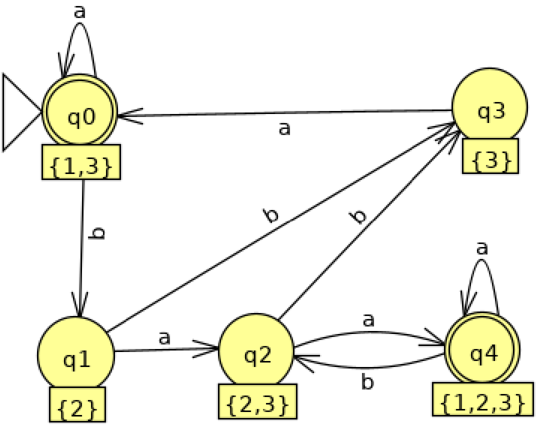
Para todo **Autômato Finito Não-Determinístico (AFN)** existe um **Autômato Finito Determinístico(AFD)** correspondente.

- Através de $P(Q)$ e $E(R)$ podemos construir o AFD $M = \{Q', \Sigma, \delta', q'_0, F'\}$ equivalente ao AFN $N = \{Q, \Sigma, \delta, q_0, F\}$ da seguinte forma:
  - $Q' = P(Q)$.
  - $\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$, para $R \in Q'$, $a \in \Sigma$.
  - $q'_0 = E(\{q_o\})$.
  - $F' = \{R \in Q' | R \text{ contém um estado de aceitação do } AFN\}$

Exemplo - AFN original

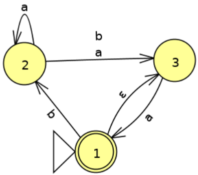


Exemplo - AFD equivalente



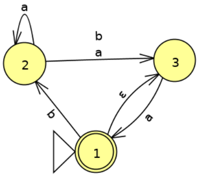
Cálculos

$$q_0' = E(\{q_0\}) = E(\{1\}) = \{1, 3\}$$



Cálculos

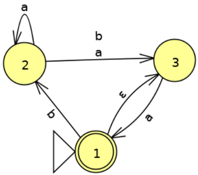
$q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$
$\delta'(R,x) = \bigcup_{r \in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.



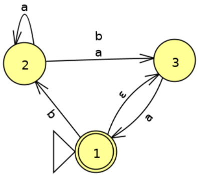
Cálculos

$$q_0' = E(\{q_0\}) = E(\{1\}) = \{1,3\}$$
$\delta'(R,x) = \bigcup_{r\in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$\begin{aligned}\delta'(\{1,3\},a) = \; & \bigcup_{q\in\{1,3\}} E(\delta(q,a)) = \\ & E(\delta(1,a)) \cup E(\delta(3,a)) = \\ & \emptyset \cup \{1,3\} = \{1,3\}\end{aligned}$$



Cálculos

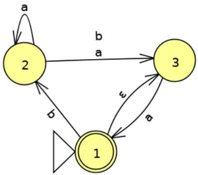


$$q_0' = E(\{q_0\}) = E(\{1\}) = \{1,3\}$$

$\delta'(R,x) = \bigcup_{r \in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$\begin{aligned}
\delta'(\{1,3\},a) = \quad & \bigcup_{q\in\{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\}
\end{aligned}$$

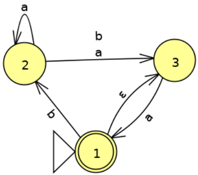
Cálculos



$q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$
$\delta'(R,x) = \bigcup_{r \in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$\begin{aligned}
\delta'(\{1,3\},a) = \quad & \bigcup_{q \in \{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = \quad & E(\delta(2,a)) = \{2,3\}
\end{aligned}$$

Cálculos

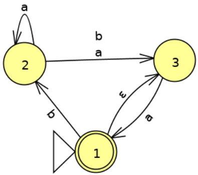


$q_0' = E(\{q_0\}) = E(\{1\}) = \{1,3\}$

$\delta'(R,x) = \bigcup_{r \in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$\begin{aligned}
\delta'(\{1,3\},a) = \quad & \bigcup_{q \in \{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = \quad & E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) = \quad & E(\delta(2,b)) = \{3\}
\end{aligned}$$

Cálculos

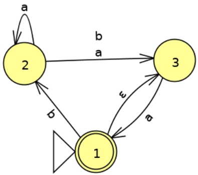


$$q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$$

$$\delta'(R,x) = \bigcup_{r \in R} E(\delta(r,x)), \text{ para } R \in Q',\ x \in \Sigma.$$

$$\begin{aligned}
\delta'(\{1,3\},a) = \quad & \bigcup_{q \in \{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = \quad & E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) = \quad & E(\delta(2,b)) = \{3\} \\
\delta'(\{2,3\},a) = \quad & E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \{2,3\} \cup \{1,3\} = \{1,2,3\}
\end{aligned}$$

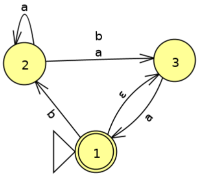
Cálculos



$q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$
$\delta'(R,x) = \bigcup_{r \in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$\begin{aligned}
\delta'(\{1,3\},a) = \quad & \bigcup_{q \in \{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = \quad & E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) = \quad & E(\delta(2,b)) = \{3\} \\
\delta'(\{2,3\},a) = \quad & E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{2,3\},b) = \quad & E(\delta(2,b)) \cup E(\delta(3,b)) = \\
& \{3\} \cup \emptyset = \{3\}
\end{aligned}$$

Cálculos

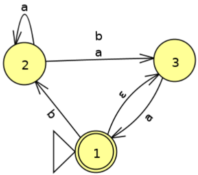


$q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$

$\delta'(R,x) = \bigcup_{r \in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$
\begin{aligned}
\delta'(\{1,3\},a) = &\ \bigcup_{q \in \{1,3\}} E(\delta(q,a)) = \\
&\ E(\delta(1,a)) \cup E(\delta(3,a)) = \\
&\ \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = &\ E(\delta(1,b)) \cup E(\delta(3,b)) = \\
&\ \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = &\ E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) = &\ E(\delta(2,b)) = \{3\} \\
\delta'(\{2,3\},a) = &\ E(\delta(2,a)) \cup E(\delta(3,a)) = \\
&\ \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{2,3\},b) = &\ E(\delta(2,b)) \cup E(\delta(3,b)) = \\
&\ \{3\} \cup \emptyset = \{3\} \\
\delta'(\{3\},a) = &\ E(\delta(3,a)) = \{1,3\}
\end{aligned}
$$

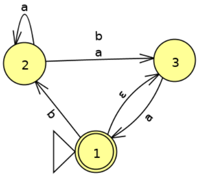
Cálculos



$$
\begin{aligned}
&q_0' = E(\{q_0\}) = E(\{1\}) = \{1,3\}\\
&\delta'(R,x) = \bigcup_{r\in R} E(\delta(r,x)), \text{ para } R \in Q',\ x \in \Sigma.
\end{aligned}
$$

$$
\begin{aligned}
\delta'(\{1,3\},a) = \quad & \bigcup_{q\in\{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = \quad & E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) = \quad & E(\delta(2,b)) = \{3\} \\
\delta'(\{2,3\},a) = \quad & E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{2,3\},b) = \quad & E(\delta(2,b)) \cup E(\delta(3,b)) = \\
& \{3\} \cup \emptyset = \{3\} \\
\delta'(\{3\},a) = \quad & E(\delta(3,a)) = \{1,3\} \\
\delta'(\{3\},b) = \quad & E(\delta(3,b)) = \emptyset
\end{aligned}
$$

Cálculos

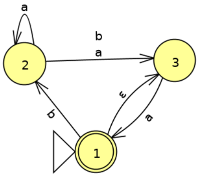


$$q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$$

$\delta'(R,x) = \bigcup_{r \in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$
\begin{aligned}
\delta'(\{1,3\},a) = \quad & \bigcup_{q \in \{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = \quad & E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) = \quad & E(\delta(2,b)) = \{3\} \\
\delta'(\{2,3\},a) = \quad & E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{2,3\},b) = \quad & E(\delta(2,b)) \cup E(\delta(3,b)) = \\
& \{3\} \cup \emptyset = \{3\} \\
\delta'(\{3\},a) = \quad & E(\delta(3,a)) = \{1,3\} \\
\delta'(\{3\},b) = \quad & E(\delta(3,b)) = \emptyset \\
\delta'(\{1,2,3\},a) = \quad & E(\delta(1,a)) \cup E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{2,3\} \cup \{1,3\} = \{1,2,3\}
\end{aligned}
$$

Cálculos

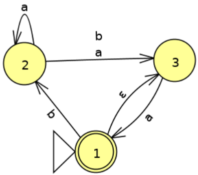


$$q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$$

$\delta'(R,x) = \bigcup_{r\in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$\begin{aligned}
\delta'(\{1,3\},a) = \quad & \textstyle\bigcup_{q\in\{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = \quad & E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) = \quad & E(\delta(2,b)) = \{3\} \\
\delta'(\{2,3\},a) = \quad & E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{2,3\},b) = \quad & E(\delta(2,b)) \cup E(\delta(3,b)) = \\
& \{3\} \cup \emptyset = \{3\} \\
\delta'(\{3\},a) = \quad & E(\delta(3,a)) = \{1,3\} \\
\delta'(\{3\},b) = \quad & E(\delta(3,b)) = \emptyset \\
\delta'(\{1,2,3\},a) = \quad & E(\delta(1,a)) \cup E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{1,2,3\},b) = \quad & E(\delta(1,b)) \cup E(\delta(2,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \{3\} \cup \emptyset = \{2,3\}
\end{aligned}$$

Cálculos

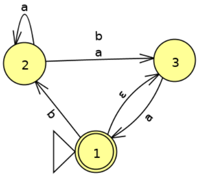


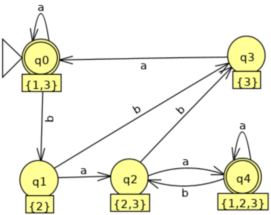
$$
\begin{array}{l}
q_0' = E(\{q_0\}) = E(\{1\}) = \{1,3\} \\
\delta'(R,x) = \bigcup_{r\in R} E(\delta(r,x)), \text{ para } R \in Q',\ x \in \Sigma.
\end{array}
$$

$$
\begin{aligned}
\delta'(\{1,3\},a) = &\ \bigcup_{q\in\{1,3\}} E(\delta(q,a)) = \\
&E(\delta(1,a)) \cup E(\delta(3,a)) = \\
&\emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) = &\ E(\delta(1,b)) \cup E(\delta(3,b)) = \\
&\{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) = &\ E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) = &\ E(\delta(2,b)) = \{3\} \\
\delta'(\{2,3\},a) = &\ E(\delta(2,a)) \cup E(\delta(3,a)) = \\
&\{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{2,3\},b) = &\ E(\delta(2,b)) \cup E(\delta(3,b)) = \\
&\{3\} \cup \emptyset = \{3\} \\
\delta'(\{3\},a) = &\ E(\delta(3,a)) = \{1,3\} \\
\delta'(\{3\},b) = &\ E(\delta(3,b)) = \emptyset \\
\delta'(\{1,2,3\},a) = &\ E(\delta(1,a)) \cup E(\delta(2,a)) \cup E(\delta(3,a)) = \\
&\emptyset \cup \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{1,2,3\},b) = &\ E(\delta(1,b)) \cup E(\delta(2,b)) \cup E(\delta(3,b)) = \\
&\{2\} \cup \{3\} \cup \emptyset = \{2,3\}
\end{aligned}
$$

$F' = \{\{1,3\},\{1,2,3\}\}$

Cálculos





$q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$

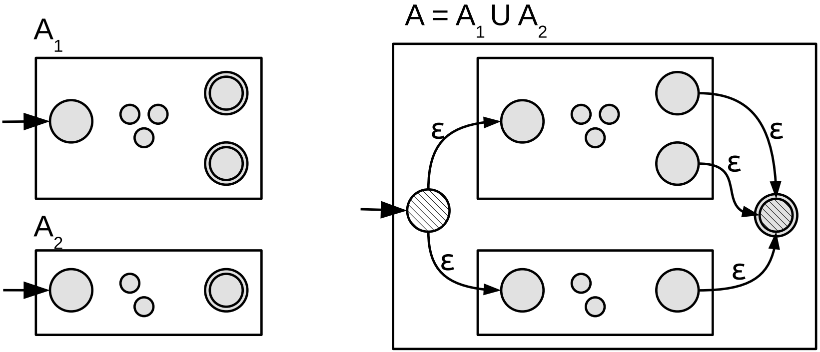
$\delta'(R,x) = \bigcup_{r\in R} E(\delta(r,x))$, para $R \in Q'$, $x \in \Sigma$.

$$\begin{aligned}
\delta'(\{1,3\},a) =\ & \textstyle\bigcup_{q\in\{1,3\}} E(\delta(q,a)) = \\
& E(\delta(1,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{1,3\} = \{1,3\} \\
\delta'(\{1,3\},b) =\ & E(\delta(1,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \emptyset = \{2\} \\
\delta'(\{2\},a) =\ & E(\delta(2,a)) = \{2,3\} \\
\delta'(\{2\},b) =\ & E(\delta(2,b)) = \{3\} \\
\delta'(\{2,3\},a) =\ & E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{2,3\},b) =\ & E(\delta(2,b)) \cup E(\delta(3,b)) = \\
& \{3\} \cup \emptyset = \{3\} \\
\delta'(\{3\},a) =\ & E(\delta(3,a)) = \{1,3\} \\
\delta'(\{3\},b) =\ & E(\delta(3,b)) = \emptyset \\
\delta'(\{1,2,3\},a) =\ & E(\delta(1,a)) \cup E(\delta(2,a)) \cup E(\delta(3,a)) = \\
& \emptyset \cup \{2,3\} \cup \{1,3\} = \{1,2,3\} \\
\delta'(\{1,2,3\},b) =\ & E(\delta(1,b)) \cup E(\delta(2,b)) \cup E(\delta(3,b)) = \\
& \{2\} \cup \{3\} \cup \emptyset = \{2,3\}
\end{aligned}$$

$F' = \{\{1,3\},\{1,2,3\}\}$

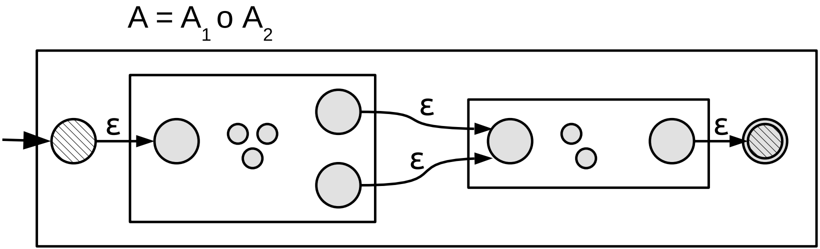
Fecho sobre a União

A classe de linguagens regulares é fechada sob a operação de União.



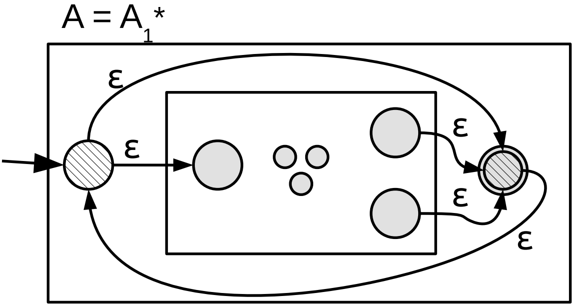
Fecho sobre a Concatenação

A classe de linguagens regulares é fechada sob a operação de Concatenação.

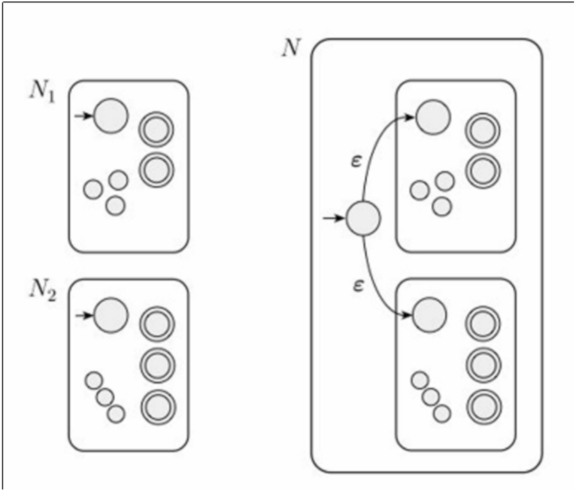


Fecho sobre a Estrela(Kleene)

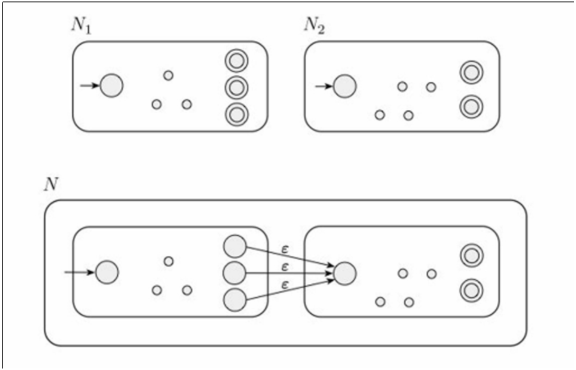
A classe de linguagens regulares é fechada sob a operação de Fecho de Kleene.



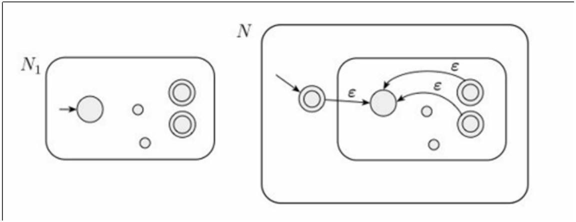
Fecho sobre a União (Sipser)



Fecho sobre a concatenação (Sipser)



Fecho sobre o Fecho de Kleene (Sipser)



Agenda

introdução

Na **aritmética** usamos $+$ e $\times$ para escrever as expressões aritméticas. Ex: $(5+3)\times 8$

Para as **expressões regulares** temos as operações de **união** $\cup$, **concatenação** $\circ$ e a operação estrela (fecho Kleene) $*$. Ex: $(0\cup 1)0^*$

- O resultado de uma expressão aritmética é um **valor**. O resultado de uma expressão regular é uma **linguagem regular**.
- 0 e 1 são abreviações dos conjuntos $\{0\}$ e $\{1\}$. Então $(0\cup 1)$ é o mesmo que $(\{0\}\cup\{1\})$.
- 0^* é o mesmo que $\{0\}^*$.
- E a operação $\circ$ está subentendida na expressão, concatenando $(0\cup 1)$ com 0^*.
- Sem as simplificações, escreveríamos: $(\{0\}\cup\{1\})\circ\{0\}^*$

Definição

Definição indutiva

Sejam R_1 e R_2 expressões regulares. Dizemos que R é uma **expressão regular** se:

1. $R = a$ para $a \in \Sigma$.
2. $R = \varepsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$
5. $R = (R_1 \circ R_2)$
6. $R = (R_1)^*$

- a e ε, representam as linguagens $\{a\}$ e $\{\varepsilon\}$.
- $\emptyset$ representa a linguagem vazia, que não reconhece nenhuma cadeia.
- Parênteses podem ser omitido. Precedência: $* > \circ > \cup$.

Exemplos de ER's

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

1. $0^*10^* = \{w|w$ contém um único $1\}$
2. $\Sigma^*1\Sigma^* = \{w|w$ contém pelo menos um símbolo $1\}$
3. $\Sigma^*001\Sigma^* = \{w|w$ contém a cadeia 001 como subcadeia$\}$
4. $1^*(01^+)^* = \{w|$ todo 0 em w é seguido por pelo menos um $1\}$
   - R^+ é uma simplificação para RR^*.
5. $(\Sigma\Sigma)^* = \{w|w$ é uma cadeia de comprimento par$\}$
6. $01 \cup 10 = \{01, 10\}$
7. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w|w$ começa e termina no mesmo símbolo $\}$
8. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
9. $1^*\emptyset = \emptyset$
10. $\emptyset^* = \{\varepsilon\}$

ER e Compiladores

- **Expressões Regulares** são úteis para construção de analisadores léxicos de linguagens de programação.
- Classes de símbolos como **constantes numéricas**, **constantes literais**, **identificadores** podem ser descritos usando expressões regulares.
- Seja $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, o conjunto dos dígitos decimais. Uma constante numérica pode ser reconhecida usando-se a seguinte expressão regular:

$$(+ \cup - \cup \varepsilon)(D^+ \cup D^+.D^* \cup D^*.D^+)$$

Conversão R $\rightarrow$ AF

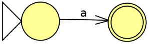
Teorema

Uma linguagem é regular **sse** alguma expressão regular a descreve

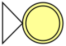
Se uma linguagem é descrita por uma **expressão regular**, então ela é regular.

Para converter R(**expressão regular**) num AFN(**autômato**):

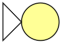
1. $R = a,\ L(R) = \{a\}$



2. $R = \varepsilon$

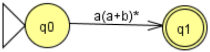


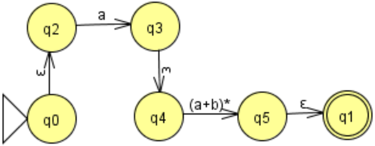
3. $R = \emptyset$

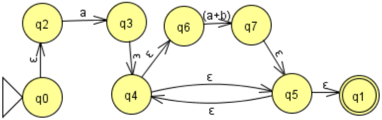


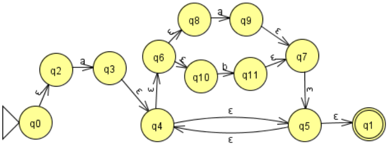
4. $R = (R_1 \cup R_2)$, $R = (R_1 \circ R_2)$ e $R = (R_1)^*$ segue os esquemas apresentados para demonstração das operações fechadas sobre linguagens regulares para AFNs (Slides 37, 38, 39).

Conversão de $R = a(a \cup b)^*$ num AFN

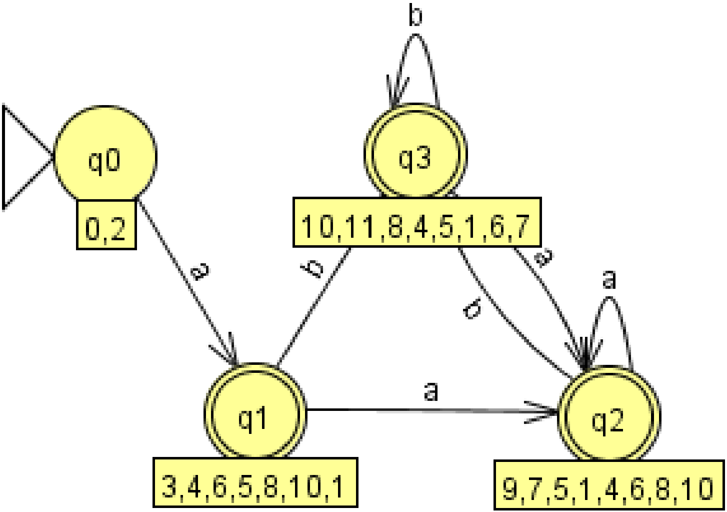








Resultado da conversão AFN → AFD



Conversão AF $\rightarrow$ R

Teorema

Uma linguagem é regular **sse** alguma expressão regular a descreve

Se uma linguagem é regular, então ela é descrita por uma **expressão regular**.

Para converte *AFN*(**autômato**) para *R*(**expressão regular**):

- Usamos um *AFNG* (autômato finito não determinístico generalizado)
- Convertemos *AFDs* para *AFNGs*.
- Convertemos *AFNGs* para *R* (expressões regulares)

Definição

Um **autômato finito não determinístico generalizado** M é uma 5-upla, $M = (Q, \Sigma, \delta, q_{\text{início}}, q_{\text{aceita}})$

Onde:

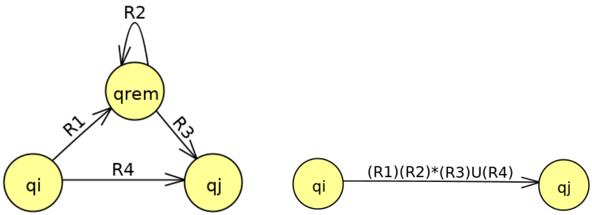
1. Q é o conjunto finito de **estados**
2. Σ é o conjunto finito de símbolos, o **alfabeto**
3. $\delta : (Q - \{q_{\text{aceita}}\}) \times (Q - \{q_{\text{início}}\}) \rightarrow R$ é a **função de transição**
4. $q_{\text{início}}$ é o **estado inicial**
5. q_{aceita} é o **estado de aceitação**.

O símbolo R é a coleção de todas as expressões regulares sobre o alfabeto Σ.

Conversão

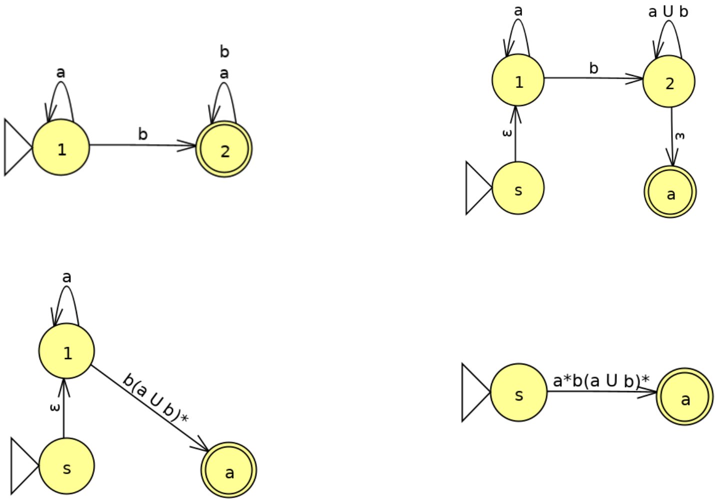
Rotina recursiva para converter o AFNG G na expressão R:
CONVERTE (G)

1. Seja k o número de estados de G
2. Se $k = 2$, então G é $q_{\text{início}} \xrightarrow{R} q_{\text{aceita}}$, retorna R.
3. Se $k > 2$, criamos um G' eliminando um estado q_{rem}, substituindo pela expressão regular que o substitui, conforme esquema:



4. calcule CONVERT(G') e retorne o resultado

Exemplo



Minimização de AFDs

Um AFD M é dito ser um AFD **mínimo** para linguagem $L(M)$ se não existe nenhum outro AFD para $L(M)$ com um número menor de estados.

- Nenhum estado não alcançável do estado inicial deve fazer parte de M.
- Deve-se determinar *grupos de estados equivalentes* e substituir cada grupo por um único estado.

Seja um AFD $M = (Q, \Sigma, \delta, q_0, F)$. Dois estados p e $q \in Q$ são ditos *equivalentes*, $p \approx q$, se e somente se:

$$\text{para todo } w \in \Sigma^*, \hat{\delta}(p, w) \in F \text{ se e somente se, } \hat{\delta}(q, w) \in F.$$

Minimização de AFDs

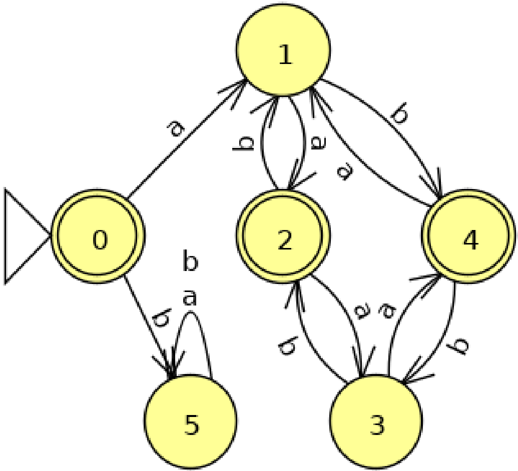
- Seja $[q] = \{q_1, q_2, ...q_n\}$ a classe de equivalência de q na partição induzida por $\approx$.
- Os estados $q_1, q_2, ...q_n$ podem ser substituidos por um único estado no AFD mínimo.

Definição

Seja um AFD $M = (Q, \Sigma, \delta, q_0, F)$. Um autômato reduzido correspondente a M é o AFD $M' = (Q', \Sigma, \delta', q_0', F')$, em que:

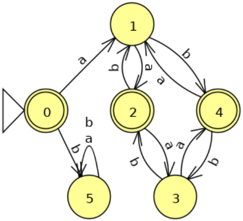
- $Q' = \{[q] | q \in Q\}$;
- $\delta'([q], a) = [\delta(q, a)]$ *para todo* $q \in Q$ *e* $a \in \Sigma$;
- $q_0' = [q_0]$;
- $F' = \{[q] | q \in F\}$.

Algoritmo de Minimização



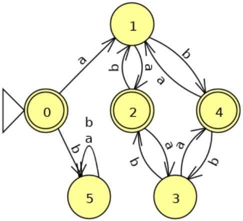
Passo 1

- Dividir o conjunto de estados em dois Grupos, o grupo G_1 do estados **finais** e o grupo G_2 dos estados **não-finais**:
  - $G_1 = F = \{0, 2, 4\}$
  - $G_2 = Q - F = \{1, 3, 5\}$
- Construir a tabela de transição onde, para cada estado é marcado o grupo do estado destino na transição, considerando cada símbolo do vocabulário:



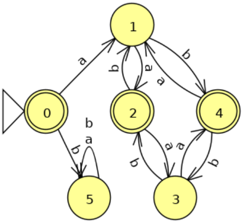
Passo 1

- Dividir o conjunto de estados em dois Grupos, o grupo G_1 do estados **finais** e o grupo G_2 dos estados **não-finais**:
  - $G_1 = F = \{0, 2, 4\}$
  - $G_2 = Q - F = \{1, 3, 5\}$
- Construir a tabela de transição onde, para cada estado é marcado o grupo do estado destino na transição, considerando cada símbolo do vocabulário:



	a	b
0	G_2	G_2
1	G_1	G_1
2	G_2	G_2
3	G_1	G_1
4	G_2	G_2
5	G_2	G_2

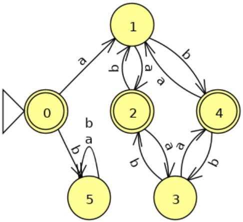
Passo 2



	a	b
0	G_2	G_2
1	G_1	G_1
2	G_2	G_2
3	G_1	G_1
4	G_2	G_2
5	G_2	G_2

- Todos os estados do G_1 fazem as mesmas transições para estado de G_2. No G_2 o estado 5 faz transições para grupos diferentes, logo pertence a outra classe de equivalência $G3 = \{5\}$

Passo 2

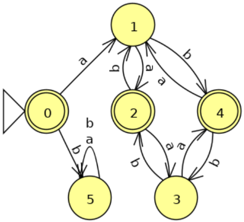


	a	b
0	G_2	G_2
1	G_1	G_1
2	G_2	G_2
3	G_1	G_1
4	G_2	G_2
5	G_2	G_2

- Todos os estados do G_1 fazem as mesmas transições para estado de G_2. No G_2 o estado 5 faz transições para grupos diferentes, logo pertence a outra classe de equivalência $G3 = \{5\}$
- Construir a tabela de transição novamente considerando esses novos grupos:
  - $G_1 = \{0, 2, 4\}$, $G_2 = \{1, 3\}$ e $G_3 = \{5\}$

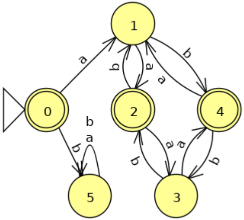
Passo 3

- Grupos:
  - $G_1 = \{0, 2, 4\}$
  - $G_2 = \{1, 3\}$
  - $G_3 = \{5\}$



Passo 3

- Grupos:
  - $G_1 = \{0, 2, 4\}$
  - $G_2 = \{1, 3\}$
  - $G_3 = \{5\}$

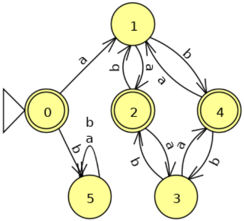


	a	b
0	G_2	G_3
1	G_1	G_1
2	G_2	G_2
3	G_1	G_1
4	G_2	G_2
5	G_3	G_3

- Agora no grupo G_1, o estado 0 faz transições diferentes de 2 e 4. Deve pertencer a outro grupo então.

Passo 4

- Grupos:
  - $G_1 = \{0\}$
  - $G_2 = \{2, 4\}$
  - $G_3 = \{1, 3\}$
  - $G_4 = \{5\}$



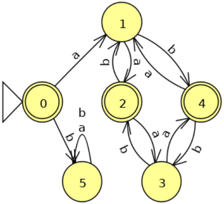
	a	b
0	G_3	G_4
1	G_2	G_2
2	G_3	G_3
3	G_2	G_2
4	G_3	G_3
5	G_4	G_4

- Cada grupo G_n é uma classe de equivalência e vai definir um estado no autômato minimizado.

Passo 5

- $G_1 = \{0\}$
- $G_2 = \{2, 4\}$
- $G_3 = \{1, 3\}$
- $G_4 = \{5\}$

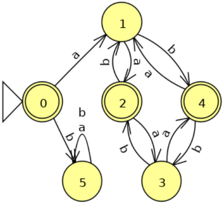
	a	b
0	G_3	G_4
1	G_2	G_2
2	G_3	G_3
3	G_2	G_2
4	G_3	G_3
5	G_4	G_4

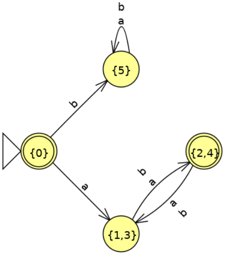


Passo 5

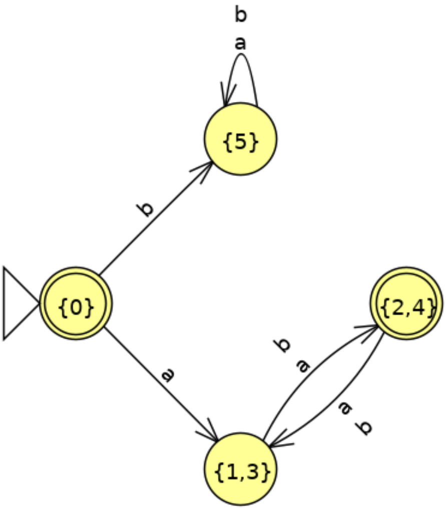
- $G_1 = \{0\}$
- $G_2 = \{2, 4\}$
- $G_3 = \{1, 3\}$
- $G_4 = \{5\}$

	a	b
0	G_3	G_4
1	G_2	G_2
2	G_3	G_3
3	G_2	G_2
4	G_3	G_3
5	G_4	G_4





Resultado da Minimização



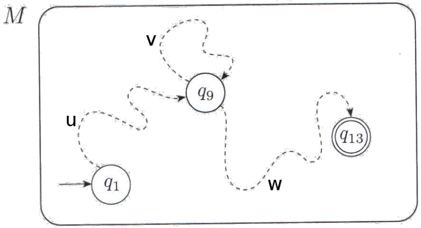
Agenda

Teorema

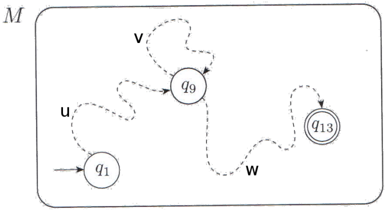
Teorema

Seja um AFD M de k estados, e $z \in L(M)$ tal que $|z| \geq k$. Então existem palavras u, v e w tais que:

- $z = uvw$;
- $v \neq \lambda$; e
- $uv^iw \in L(M)$ para todo $i \geq 0$



Teorema - resumindo



- Se L é regular é aceita por uma AFD M com k estados.
- Se M aceita um cadeia z maior que k símbolos, então alguns estados de M se repetem na verificação de z (imagem ao lado).
- Logo, z pode ser dividida em três partes $z = uvw$, onde $|uv| \leq k$ e $|v| \geq 1$ é a parte que repete da cadeia z.
- Esse ciclo é "bombeado" na cadeia e para qualquer $i \geq 0$, uv^iw é aceita pelo autômato.

Lema do Bombeamento

Lema

Seja L uma linguagem regular. Então existe uma constante $k > 0$, tal que para qualquer palavra $z \in L$ existem u, v e w que satisfazem as seguintes condições:

1. $z = uvw$;
2. $|uv| \leq k$;
3. $v \neq \lambda$; *e*
4. $uv^iw \in L$ *para todo* $i \geq 0$

Uso do Lema do Bombeamento

Para provar que L, infinita, **NÃO** é regular usa-se:

1. supõe-se que L seja linguagem regular;
2. escolhe-se uma palavra z, com tamanho maior que k, a constante do LB.
3. mostra-se que, para toda decomposição de z em u, v e w, existe i tal que $uv^iw \notin L$.

Exemplo de prova

A linguagem $L = \{a^n b^n | n \in N\}$ não é regular.

- Suponha que L é regular. Seja k a constante da LB e seja $z = a^k b^k$.
- Como $|z| > k$, então existem u, v e w, tais que:
  1. $z = uvw$;
  2. $|uv| \leq k$;
  3. $v \neq \lambda$; e
  4. $uv^i w \in L$ para todo $i \geq 0$
- Nesse caso v só tem **a**s, pois $z = uvw = a^k b^k$ e $|uv| \leq k$, e v tem pelo menos um **a**, porque $v \neq \lambda$. Isso implica que $uv^2 w = a^{k+|v|} b^k \notin L$,
- Logo a suposição original de que L é regular não pode ser confirmada.

Outro Exemplo de prova

A linguagem $L = \{xx | x \in \{0,1\}^*\}$ não é regular.

- Suponha que L é regular. Seja k a constante da LB e seja $z = 0^k 1 0^k 1$.
- Como $|z| > k$, então existem u, v e w, tais que:
    1. $z = uvw$;
    2. $|uv| \leq k$;
    3. $v \neq \lambda$; e
    4. $uv^i w \in L$ para todo $i \geq 0$
- Sem a condição $|uv| \leq k$, poderíamos "bombear" z se fizéssemos $u = w = \lambda$. Por essa condição v pode conter somente **0**s e $u = \lambda$; logo $uv^2 w = 0^k 0^k 1 0^k 1 \notin L$.
- Logo a suposição original de que L é regular não pode ser confirmada.

Agenda

Moore e Mealy

- Máquinas de Mealy e de Moore são Autômatos com saída;
- Máquina de Moore associa uma saída a cada estado; e
- Máquina de Mealy associa uma saída a cada transição

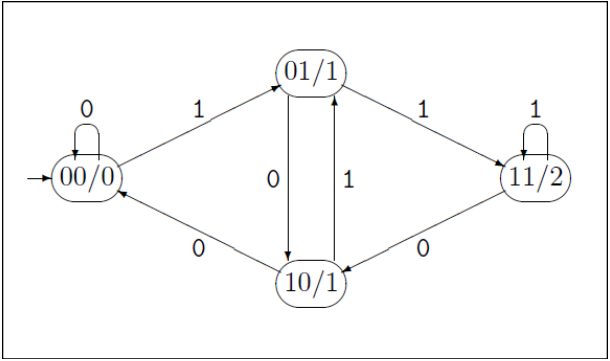
Definição

Uma máquina de Moore é uma sextupla $(E, \Sigma, \Delta, \delta, \sigma, i)$ *em que*

- E *(o conjunto de estados),* Σ *(o alfabeto de entrada),* δ *(a função de transição) e* i *(o estado inicial) são como em AFD's;*
- Δ *é o alfabeto de saida; e*
- $\sigma : E \to \Delta$ *é a função de saída, uma função total.*

Moore

Máquina de Moore que determina o número de 1's presentes nos dois últimos dígitos de uma palavra $\{0,1\}^*$.



Mealy

Definição

Uma máquina de Mealy é uma sextupla $(E, \Sigma, \Delta, \delta, \sigma, i)$ *em que*

- E *(o conjunto de estados),* Σ *(o alfabeto de entrada),* δ *(a função de transição) e* i *(o estado inicial) são como em AFD's;*
- Δ *é o alfabeto de saida; e*
- $\sigma : E \times \Sigma \rightarrow \Delta$ *é a função de saída, uma função total.*

Mealy

Máquina de Mealy que determina o número de 1's presentes nos dois últimos dígitos de uma palavra $\{0,1\}^*$.