Autômatos finitos

Rodrigo Gabriel Ferreira Soares

DEINFO - UFRPE

2014

Autômatos

- Neste momento, mostraremos uma representação relativamente modesta de um computador.

Autômatos

- Neste momento, mostraremos uma representação relativamente modesta de um computador.
- Um autômato tem como características comuns com um computador uma unidade de processamento com capacidade fixa, finita.

Autômatos

- Neste momento, mostraremos uma representação relativamente modesta de um computador.
- Um autômato tem como características comuns com um computador uma unidade de processamento com capacidade fixa, finita.
- Ele tem como entrada uma cadeia em uma fita de entrada. Produz como saída uma indicação se a entrada foi considerada aceitável ou não.

Autômatos

- Neste momento, mostraremos uma representação relativamente modesta de um computador.
- Um autômato tem como características comuns com um computador uma unidade de processamento com capacidade fixa, finita.
- Ele tem como entrada uma cadeia em uma fita de entrada. Produz como saída uma indicação se a entrada foi considerada aceitável ou não.
- É um dispositivo de reconhecimento de linguagens.

Autômatos

- Neste momento, mostraremos uma representação relativamente modesta de um computador.
- Um autômato tem como características comuns com um computador uma unidade de processamento com capacidade fixa, finita.
- Ele tem como entrada uma cadeia em uma fita de entrada. Produz como saída uma indicação se a entrada foi considerada aceitável ou não.
- É um dispositivo de reconhecimento de linguagens.
- É muito restrito devido à completa ausência de memória externa ao processador.

Componentes

- Autômatos permitem projetar vários tipos de algoritmos. Exemplo: análise léxica.

Componentes

- Autômatos permitem projetar vários tipos de algoritmos. Exemplo: análise léxica.
  1. Cadeias são recebidas através da leitura de uma **fita de entrada**, que é composta por células, com um símbolo gravado em cada uma delas.

Componentes

- Autômatos permitem projetar vários tipos de algoritmos. Exemplo: análise léxica.
  1. Cadeias são recebidas através da leitura de uma **fita de entrada**, que é composta por células, com um símbolo gravado em cada uma delas.
  2. O **controle finito** é um mecanismo que pode apresentar, em um momento específico, em um de seus **estados internos**. Ele pode ler qualquer símbolo de entrada através do **cabeçote móvel de leitura**.

Componentes

- Autômatos permitem projetar vários tipos de algoritmos. Exemplo: análise léxica.
  1. Cadeias são recebidas através da leitura de uma **fita de entrada**, que é composta por células, com um símbolo gravado em cada uma delas.
  2. O **controle finito** é um mecanismo que pode apresentar, em um momento específico, em um de seus **estados internos**. Ele pode ler qualquer símbolo de entrada através do **cabeçote móvel de leitura**.

Funcionamento

1. Inicialmente, o cabeçote está na posição mais à esquerda da fita e o controle finito está em seu **estado inicial** pré-estabelecido.

Funcionamento

1. Inicialmente, o cabeçote está na posição mais à esquerda da fita e o controle finito está em seu **estado inicial** pré-estabelecido.
2. Em intervalos regulares, o autômato lê um símbolo da fita de entrada e então entra em um novo estado.

Funcionamento

1. Inicialmente, o cabeçote está na posição mais à esquerda da fita e o controle finito está em seu **estado inicial** pré-estabelecido.
2. Em intervalos regulares, o autômato lê um símbolo da fita de entrada e então entra em um novo estado.
3. Esse novo estado depende apenas do estado atual e do símbolo que acabou de ser lido.

Funcionamento

1. Inicialmente, o cabeçote está na posição mais à esquerda da fita e o controle finito está em seu **estado inicial** pré-estabelecido.
2. Em intervalos regulares, o autômato lê um símbolo da fita de entrada e então entra em um novo estado.
3. Esse novo estado depende apenas do estado atual e do símbolo que acabou de ser lido.
   - Autômato finito determinístico.

Funcionamento

1. Inicialmente, o cabeçote está na posição mais à esquerda da fita e o controle finito está em seu **estado inicial** pré-estabelecido.
2. Em intervalos regulares, o autômato lê um símbolo da fita de entrada e então entra em um novo estado.
3. Esse novo estado depende apenas do estado atual e do símbolo que acabou de ser lido.
   - Autômato finito determinístico.
4. Após a leitura de um símbolo de entrada, o cabeçote se movimenta uma célula para a direta na fita de modo que, no próximo movimento, ela leia o símbolo na próxima célula.

Funcionamento

1. Inicialmente, o cabeçote está na posição mais à esquerda da fita e o controle finito está em seu **estado inicial** pré-estabelecido.
2. Em intervalos regulares, o autômato lê um símbolo da fita de entrada e então entra em um novo estado.
3. Esse novo estado depende apenas do estado atual e do símbolo que acabou de ser lido.
   - Autômato finito determinístico.
4. Após a leitura de um símbolo de entrada, o cabeçote se movimenta uma célula para a direta na fita de modo que, no próximo movimento, ela leia o símbolo na próxima célula.
5. Esse processo é repetido continuamente: um símbolo é lido, o cabeçote move-se para a direita e muda o estado interno do controle finito.

Funcionamento

1. Inicialmente, o cabeçote está na posição mais à esquerda da fita e o controle finito está em seu **estado inicial** pré-estabelecido.
2. Em intervalos regulares, o autômato lê um símbolo da fita de entrada e então entra em um novo estado.
3. Esse novo estado depende apenas do estado atual e do símbolo que acabou de ser lido.
   - Autômato finito determinístico.
4. Após a leitura de um símbolo de entrada, o cabeçote se movimenta uma célula para a direta na fita de modo que, no próximo movimento, ela leia o símbolo na próxima célula.
5. Esse processo é repetido continuamente: um símbolo é lido, o cabeçote move-se para a direita e muda o estado interno do controle finito.

Funcionamento

6. Finalmente, o cabeçote alcança o final da cadeia de entrada.

Funcionamento

6. Finalmente, o cabeçote alcança o final da cadeia de entrada.
7. O autômato, então, indica a aprovação ou rejeição do que foi lido de acordo com o estado a que foi conduzido ao final desse processo.

Funcionamento

6. Finalmente, o cabeçote alcança o final da cadeia de entrada.
7. O autômato, então, indica a aprovação ou rejeição do que foi lido de acordo com o estado a que foi conduzido ao final desse processo.
8. Caso o estado resultante pertencer ao conjunto de **estados finais**, a cadeia de entrada é **aceita**.

Funcionamento

6. Finalmente, o cabeçote alcança o final da cadeia de entrada.
7. O autômato, então, indica a aprovação ou rejeição do que foi lido de acordo com o estado a que foi conduzido ao final desse processo.
8. Caso o estado resultante pertencer ao conjunto de **estados finais**, a cadeia de entrada é **aceita**.
9. A linguagem aceita pela máquina é o conjunto de cadeias que ela reconhece.

Definição

Definição

Um **autômato finito determinístico** é uma quíntupla $M = (K, \Sigma, \delta, s, F)$, onde
K é um conjunto finito de **estados**,
Σ é um alfabeto,
$s \in K$ é o **estado inicial**,
$F \subseteq K$ é o conjunto de **estados finais**, e
δ, a **função de transição**, é uma função de $K \times \Sigma$ para K.

Funcionamento

- As regras pelas quais o autômato M escolhe seu próximo estado são codificadas em sua função de transição δ.

Funcionamento

- As regras pelas quais o autômato M escolhe seu próximo estado são codificadas em sua função de transição δ.
- Se M estiver no estado $q \in K$, e o símbolo lido na fita de entrada for $a \in \Sigma$, então $\delta(q, a) \in K$ será o estado, univocamente determinado, para o qual M transitará.

Funcionamento

- As regras pelas quais o autômato M escolhe seu próximo estado são codificadas em sua função de transição δ.
- Se M estiver no estado $q \in K$, e o símbolo lido na fita de entrada for $a \in \Sigma$, então $\delta(q, a) \in K$ será o estado, univocamente determinado, para o qual M transitará.
- A computação de uma cadeia pode ser caracterizada por uma seqüência de configurações de M, que representam o aspecto de M em momentos sucessivos.

Funcionamento

- As regras pelas quais o autômato M escolhe seu próximo estado são codificadas em sua função de transição δ.
- Se M estiver no estado $q \in K$, e o símbolo lido na fita de entrada for $a \in \Sigma$, então $\delta(q, a) \in K$ será o estado, univocamente determinado, para o qual M transitará.
- A computação de uma cadeia pode ser caracterizada por uma seqüência de configurações de M, que representam o aspecto de M em momentos sucessivos.
- Uma **configuração de um autômato finito determinístico** (AFD) é qualquer elemento de $K \times \Sigma^*$.

Funcionamento

- As regras pelas quais o autômato M escolhe seu próximo estado são codificadas em sua função de transição δ.
- Se M estiver no estado $q \in K$, e o símbolo lido na fita de entrada for $a \in \Sigma$, então $\delta(q, a) \in K$ será o estado, univocamente determinado, para o qual M transitará.
- A computação de uma cadeia pode ser caracterizada por uma seqüência de configurações de M, que representam o aspecto de M em momentos sucessivos.
- Uma **configuração de um autômato finito determinístico** (AFD) é qualquer elemento de $K \times \Sigma^*$.
- Um exemplo é $(q_2, ababab)$.

Configurações

- A relação binária $\vdash_M$ se aplica a um par de configurações se a máquina puder passar de uma configuração a outra, como resultado de um simples movimento.

Configurações

- A relação binária $\vdash_M$ se aplica a um par de configurações se a máquina puder passar de uma configuração a outra, como resultado de um simples movimento.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma$, e $\delta(q, a) = q'$.

Configurações

- A relação binária $\vdash_M$ se aplica a um par de configurações se a máquina puder passar de uma configuração a outra, como resultado de um simples movimento.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma$, e $\delta(q, a) = q'$.
- Dizemos assim que (q, w) **leva a** (q', w') **em um passo**.

Configurações

- A relação binária $\vdash_M$ se aplica a um par de configurações se a máquina puder passar de uma configuração a outra, como resultado de um simples movimento.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma$, e $\delta(q, a) = q'$.
- Dizemos assim que (q, w) **leva a** (q', w') **em um passo**.
- $\vdash_M$ é uma função $K \times \Sigma^+$ para $K \times \Sigma^*$.

Configurações

- A relação binária $\vdash_M$ se aplica a um par de configurações se a máquina puder passar de uma configuração a outra, como resultado de um simples movimento.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma$, e $\delta(q, a) = q'$.
- Dizemos assim que (q, w) **leva a** (q', w') **em um passo**.
- $\vdash_M$ é uma função $K \times \Sigma^+$ para $K \times \Sigma^*$.
- Para cada configuração, existe uma configuração seguinte, exceto da forma (q, ϵ), que indica que M consumiu todas as suas entradas, encerrando sua operação.

Aceitação

- Denotamos o fechamento transitivo reflexivo de $\vdash_M$ como $\vdash_M^*$.

Aceitação

- Denotamos o fechamento transitivo reflexivo de $\vdash_M$ como $\vdash_M^*$.
- $(q, w) \vdash_M^* (q', w')$ denota que (q, w) **produz um resultado** (q', w') após um certo número, possivelmente zero, de passos.

Aceitação

- Denotamos o fechamento transitivo reflexivo de $\vdash_M$ como $\vdash_M^*$.
- $(q, w) \vdash_M^* (q', w')$ denota que (q, w) **produz um resultado** (q', w') após um certo número, possivelmente zero, de passos.
- Uma cadeia $w \in \Sigma^*$ é **aceita** por M *sse* existir algum estado $q \in F$, tal que $(s, w) \vdash_M^* (q, \epsilon)$.

Aceitação

- Denotamos o fechamento transitivo reflexivo de $\vdash_M$ como $\vdash_M^*$.
- $(q, w) \vdash_M^* (q', w')$ denota que (q, w) **produz um resultado** (q', w') após um certo número, possivelmente zero, de passos.
- Uma cadeia $w \in \Sigma^*$ é **aceita** por M *sse* existir algum estado $q \in F$, tal que $(s, w) \vdash_M^* (q, \epsilon)$.
- A **linguagem aceita por** M, denotada por $L(M)$, é o conjunto de todas as cadeias aceitas por M.

Exemplo

- Seja M o AFD $(K, \Sigma, \delta, s, F)$ onde $K = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$, e δ é a função na tabela a seguir

Exemplo

- Seja M o AFD $(K, \Sigma, \delta, s, F)$ onde $K = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$, e δ é a função na tabela a seguir

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

Exemplo

- Seja M o AFD $(K, \Sigma, \delta, s, F)$ onde $K = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$, e δ é a função na tabela a seguir

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

- Se M recebe $aabba$ como entrada, sua configuração inicial será $(q_0, aabba)$, daí

Exemplo

- Seja M o AFD $(K, \Sigma, \delta, s, F)$ onde $K = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$, e δ é a função na tabela a seguir

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

- Se M recebe $aabba$ como entrada, sua configuração inicial será $(q_0, aabba)$, daí

$$(q_0, aabba) \vdash_M (q_0, abba) \tag{1}$$

Exemplo

- Seja M o AFD $(K, \Sigma, \delta, s, F)$ onde $K = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$, e δ é a função na tabela a seguir

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

- Se M recebe $aabba$ como entrada, sua configuração inicial será $(q_0, aabba)$, daí

$$(q_0, aabba) \vdash_M (q_0, abba) \quad (1)$$
$$\vdash_M (q_0, bba) \quad (2)$$

Exemplo

- Seja M o AFD $(K, \Sigma, \delta, s, F)$ onde $K = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$, e δ é a função na tabela a seguir

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

- Se M recebe $aabba$ como entrada, sua configuração inicial será $(q_0, aabba)$, daí

$$
\begin{aligned}
(q_0, aabba) &\vdash_M (q_0, abba) && (1)\\
&\vdash_M (q_0, bba) && (2)\\
&\vdash_M (q_1, ba) && (3)
\end{aligned}
$$

Exemplo

- Seja M o AFD $(K, \Sigma, \delta, s, F)$ onde $K = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$, e δ é a função na tabela a seguir

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

- Se M recebe $aabba$ como entrada, sua configuração inicial será $(q_0, aabba)$, daí

$$
\begin{aligned}
(q_0, aabba) &\vdash_M (q_0, abba) && (1)\\
&\vdash_M (q_0, bba) && (2)\\
&\vdash_M (q_1, ba) && (3)\\
&\vdash_M (q_0, a) && (4)
\end{aligned}
$$

Exemplo

- Seja M o AFD $(K, \Sigma, \delta, s, F)$ onde $K = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$, e δ é a função na tabela a seguir

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

- Se M recebe $aabba$ como entrada, sua configuração inicial será $(q_0, aabba)$, daí

$$
\begin{aligned}
(q_0, aabba) &\vdash_M (q_0, abba) && (1)\\
&\vdash_M (q_0, bba) && (2)\\
&\vdash_M (q_1, ba) && (3)\\
&\vdash_M (q_0, a) && (4)\\
&\vdash_M (q_0, \epsilon) && (5)
\end{aligned}
$$

Exemplo

- Assim, $(q_0, aabba) \vdash_M^* (q_0, \epsilon)$, logo $aabba$ é aceita por M.

Exemplo

- Assim, $(q_0, aabba) \vdash_M^* (q_0, \epsilon)$, logo $aabba$ é aceita por M.
- Para representar com mais clareza um AFD, temos o **diagrama de estados**.

Exemplo

- Assim, $(q_0, aabba) \vdash_M^* (q_0, \epsilon)$, logo $aabba$ é aceita por M.
- Para representar com mais clareza um AFD, temos o **diagrama de estados**.
- Ele é um grafo orientado com algumas informações adicionais.

Exemplo

- Assim, $(q_0, aabba) \vdash_M^* (q_0, \epsilon)$, logo $aabba$ é aceita por M.
- Para representar com mais clareza um AFD, temos o **diagrama de estados**.
- Ele é um grafo orientado com algumas informações adicionais.

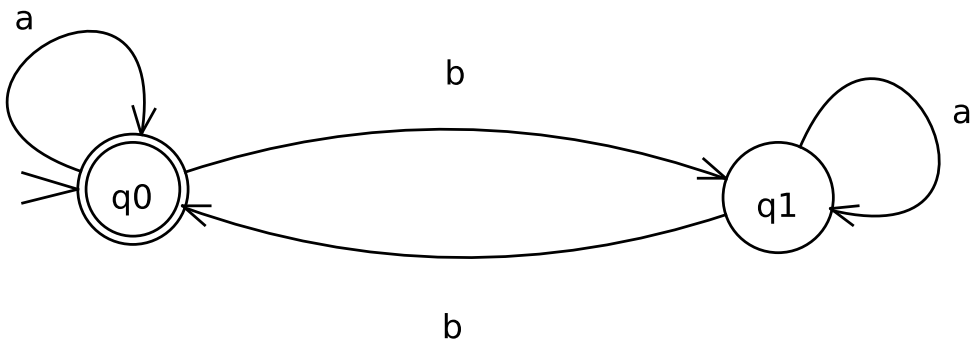


Figura : Diagrama de estados de M.

Exemplo

- Vamos projetar um AFD M que reconhece a linguagem $L(M) = \{w \in \{a, b\}^* :$ w não contém três b's consecutivos$\}$.

Exemplo

- Vamos projetar um AFD M que reconhece a linguagem $L(M) = \{w \in \{a, b\}^* :$ w não contém três b's consecutivos$\}$.
- Seja $M = (K, \Sigma, \delta, s, F)$, onde $K = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0, q_1, q_2\}$, e δ é a função na tabela a seguir

Exemplo

- Vamos projetar um AFD M que reconhece a linguagem $L(M) = \{w \in \{a, b\}^* :$ w não contém três b's consecutivos$\}$.
- Seja $M = (K, \Sigma, \delta, s, F)$, onde $K = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0, q_1, q_2\}$, e δ é a função na tabela a seguir

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_2
q_2	a	q_0
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

Exemplo

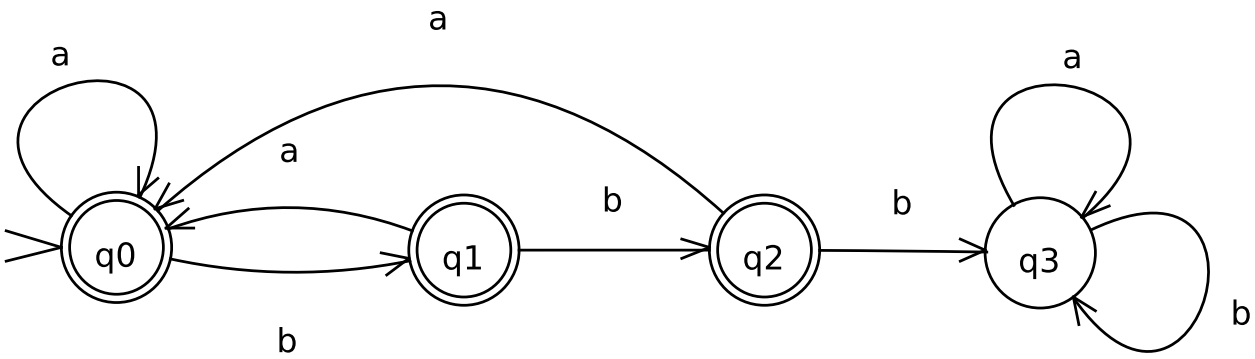


Figura : Diagrama de estados de M.

Exemplo

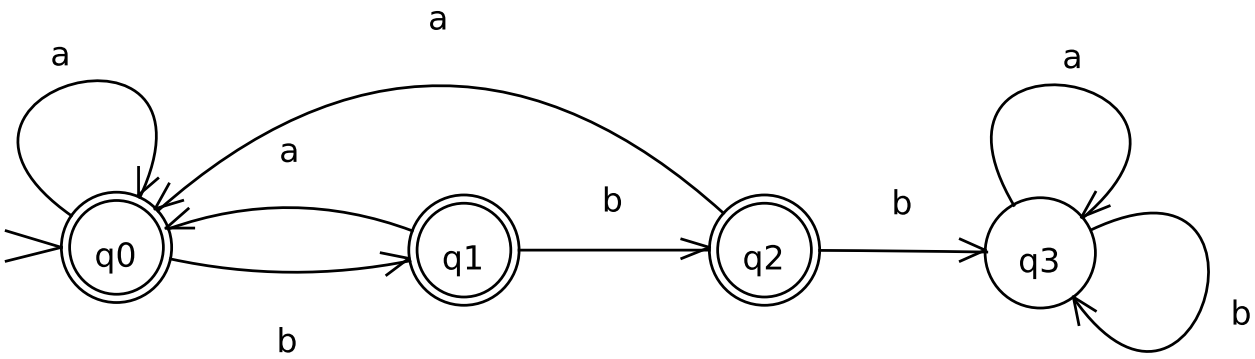


Figura : Diagrama de estados de M.

- M irá permanecer em q_3 independentemente da entrada.

Exemplo

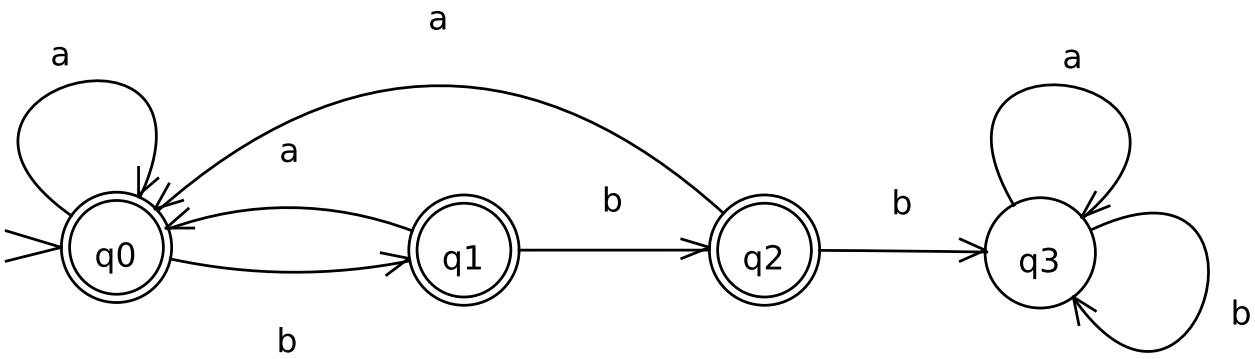


Figura : Diagrama de estados de M.

- M irá permanecer em q_3 independentemente da entrada.
- Diz-se que q_3 é um **estado morto**.

Exemplo

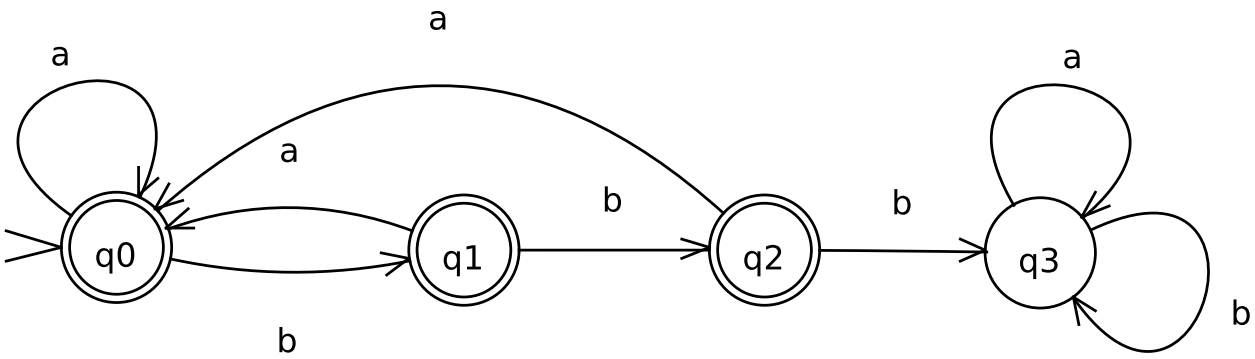


Figura : Diagrama de estados de M.

- M irá permanecer em q_3 independentemente da entrada.
- Diz-se que q_3 é um **estado morto**.
- M fica preso (*trapped*) em q_3, pois não pode escapar desse estado.

Autômatos finitos não-determinísticos

- Capacidade de mudar de estado de forma apenas parcialmente determinada pelo estado atual e pelo símbolo de entrada.

Autômatos finitos não-determinísticos

- Capacidade de mudar de estado de forma apenas parcialmente determinada pelo estado atual e pelo símbolo de entrada.
- Há vários possíveis *estados seguintes* para uma dada combinação de estado e símbolo.

Autômatos finitos não-determinísticos

- Capacidade de mudar de estado de forma apenas parcialmente determinada pelo estado atual e pelo símbolo de entrada.
- Há vários possíveis *estados seguintes* para uma dada combinação de estado e símbolo.
- A escolha do próximo estado não é determinada por nada em nosso modelo – *não-determinismo*.

Autômatos finitos não-determinísticos

- Capacidade de mudar de estado de forma apenas parcialmente determinada pelo estado atual e pelo símbolo de entrada.
- Há vários possíveis *estados seguintes* para uma dada combinação de estado e símbolo.
- A escolha do próximo estado não é determinada por nada em nosso modelo – *não-determinismo*.
- Não são modelos realistas de computadores. São generalizações notacionais úteis para simplificar a descrição de AFDs.

Autômatos finitos não-determinísticos

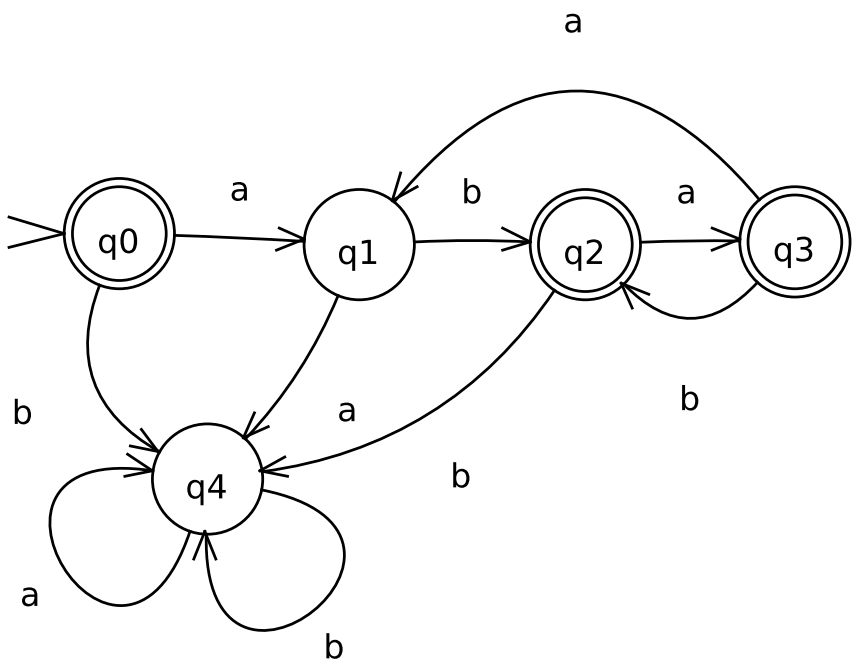
- Capacidade de mudar de estado de forma apenas parcialmente determinada pelo estado atual e pelo símbolo de entrada.
- Há vários possíveis *estados seguintes* para uma dada combinação de estado e símbolo.
- A escolha do próximo estado não é determinada por nada em nosso modelo – *não-determinismo*.
- Não são modelos realistas de computadores. São generalizações notacionais úteis para simplificar a descrição de AFDs.
- Todo autômato finito não-determinístico (AFN) tem um AFD equivalente.

Autômatos finitos não-determinísticos

- Considere o AFD que aceita a linguagem $L = (ab \cup aba)^*$.

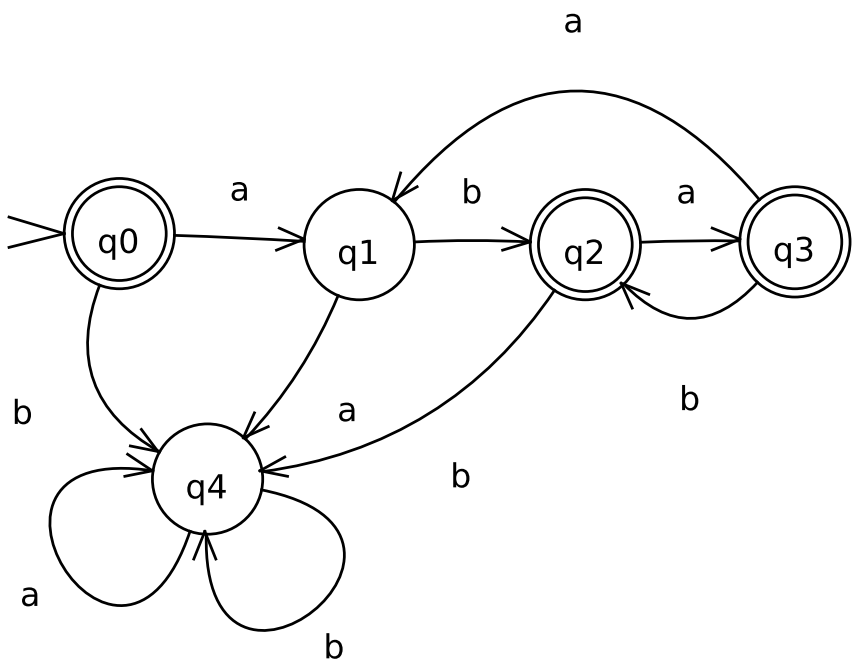
Autômatos finitos não-determinísticos

- Considere o AFD que aceita a linguagem $L = (ab \cup aba)^*$.



Autômatos finitos não-determinísticos

- Considere o AFD que aceita a linguagem $L = (ab \cup aba)^*$.



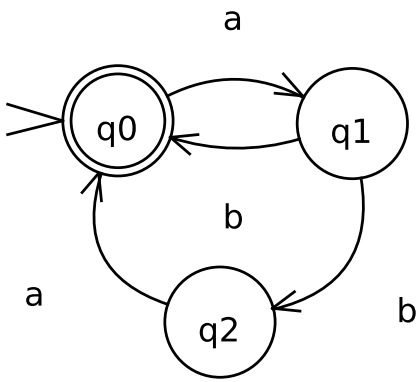
- Há exatamente duas arestas saindo de cada estado.

Autômatos finitos não-determinísticos

- L é aceita pelo seguinte AFN, que é mais simples que o AFD anterior.

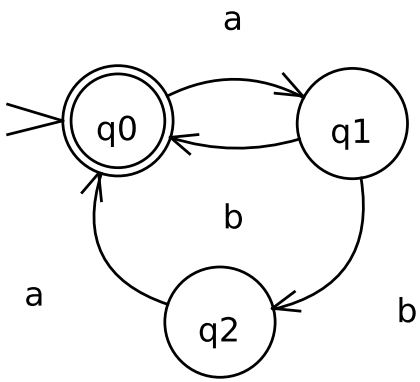
Autômatos finitos não-determinísticos

- L é aceita pelo seguinte AFN, que é mais simples que o AFD anterior.



Autômatos finitos não-determinísticos

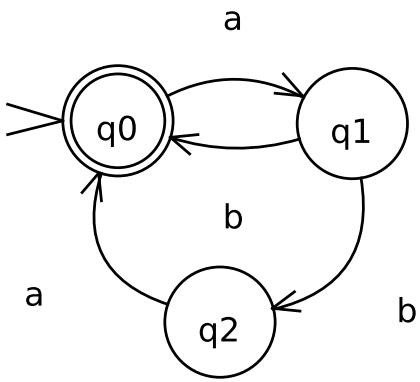
- L é aceita pelo seguinte AFN, que é mais simples que o AFD anterior.



- Ele aceita a cadeia *aba*?

Autômatos finitos não-determinísticos

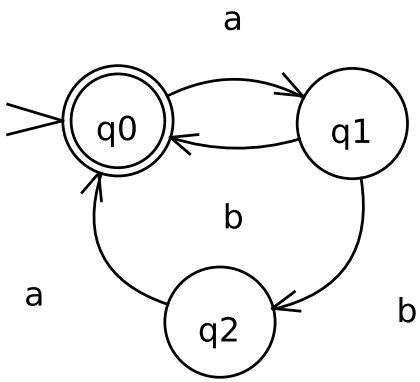
- L é aceita pelo seguinte AFN, que é mais simples que o AFD anterior.



- Ele aceita a cadeia *aba*?
- Aceitar se existir alguma maneira de atingir o estado final a partir do inicial com essa entrada.

Autômatos finitos não-determinísticos

- L é aceita pelo seguinte AFN, que é mais simples que o AFD anterior.



- Ele aceita a cadeia *aba*?
- Aceitar se existir alguma maneira de atingir o estado final a partir do inicial com essa entrada.
- Pode haver vários estados seguintes, bem como nenhum.

Autômatos finitos não-determinísticos

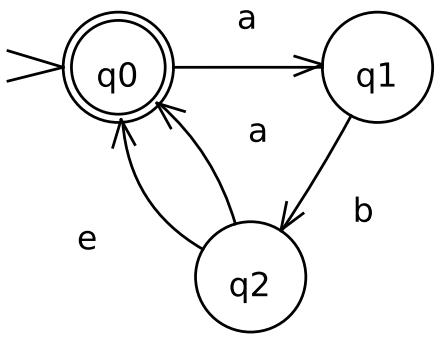
- Um AFN pode conter setas rotuladas com cadeia vazia ϵ.

Autômatos finitos não-determinísticos

- Um AFN pode conter setas rotuladas com cadeia vazia ϵ.
- A máquina seguinte aceita a mesma linguagem L do exemplo anterior.

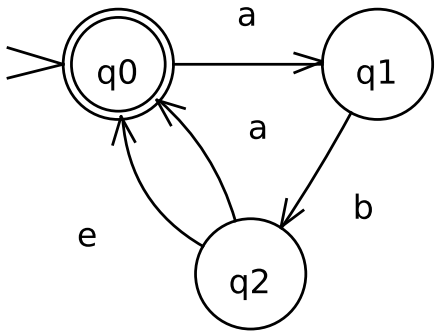
Autômatos finitos não-determinísticos

- Um AFN pode conter setas rotuladas com cadeia vazia ϵ.
- A máquina seguinte aceita a mesma linguagem L do exemplo anterior.



Autômatos finitos não-determinísticos

- Um AFN pode conter setas rotuladas com cadeia vazia ϵ.
- A máquina seguinte aceita a mesma linguagem L do exemplo anterior.



- De q_2, esse dispositivo pode retornar para q_0,
  - após a leitura do símbolo a ou
  - imediatamente, sem consumir qualquer entrada.

Autômatos finitos não-determinísticos

Definição

Um autômato finito não-determinístico é uma quíntupla $M = (K, \Sigma, \Delta, s, F)$, onde
K é um conjunto finito de **estados**,
Σ é um alfabeto,
$s \in K$ é o **estado inicial**,
$F \subseteq K$ é o conjunto de **estados finais**, e
Δ, a **relação de transição**, é um subconjunto de $K \times (\Sigma \cup \{\epsilon\}) \times K$.

Transições, configurações e aceitação

- Cada tripla $(q, u, p) \in \Delta$ é uma transição de M.

Transições, configurações e aceitação

- Cada tripla $(q, u, p) \in \Delta$ é uma transição de M.
- M pode seguir qualquer transição das formas (q, a, p) ou (q, ϵ, p). Se (q, ϵ, p) é executada, então nenhum símbolo de entrada é consumido.

Transições, configurações e aceitação

- Cada tripla $(q, u, p) \in \Delta$ é uma transição de M.
- M pode seguir qualquer transição das formas (q, a, p) ou (q, ϵ, p). Se (q, ϵ, p) é executada, então nenhum símbolo de entrada é consumido.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma \cup \{\epsilon\}$, e $(q, a, q') \in \Delta$.

Transições, configurações e aceitação

- Cada tripla $(q, u, p) \in \Delta$ é uma transição de M.
- M pode seguir qualquer transição das formas (q, a, p) ou (q, ϵ, p). Se (q, ϵ, p) é executada, então nenhum símbolo de entrada é consumido.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma \cup \{\epsilon\}$, e $(q, a, q') \in \Delta$.
- $\vdash_M$ não precisa ser uma função. Pode haver vários pares (q', w'), ou nenhum, tais que $(q, w) \vdash_M (q', w')$.

Transições, configurações e aceitação

- Cada tripla $(q, u, p) \in \Delta$ é uma transição de M.
- M pode seguir qualquer transição das formas (q, a, p) ou (q, ϵ, p). Se (q, ϵ, p) é executada, então nenhum símbolo de entrada é consumido.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma \cup \{\epsilon\}$, e $(q, a, q') \in \Delta$.
- $\vdash_M$ não precisa ser uma função. Pode haver vários pares (q', w'), ou nenhum, tais que $(q, w) \vdash_M (q', w')$.
- $\vdash_M^*$ é o fechamento transitivo reflexivo de $\vdash_M$.

Transições, configurações e aceitação

- Cada tripla $(q, u, p) \in \Delta$ é uma transição de M.
- M pode seguir qualquer transição das formas (q, a, p) ou (q, ϵ, p). Se (q, ϵ, p) é executada, então nenhum símbolo de entrada é consumido.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma \cup \{\epsilon\}$, e $(q, a, q') \in \Delta$.
- $\vdash_M$ não precisa ser uma função. Pode haver vários pares (q', w'), ou nenhum, tais que $(q, w) \vdash_M (q', w')$.
- $\vdash_M^*$ é o fechamento transitivo reflexivo de $\vdash_M$.
- Uma cadeia $w \in \Sigma^*$ é aceita *see* houver um estado $q \in F$ tal que $(s, w) \vdash_M^* (q, \epsilon)$.

Transições, configurações e aceitação

- Cada tripla $(q, u, p) \in \Delta$ é uma transição de M.
- M pode seguir qualquer transição das formas (q, a, p) ou (q, ϵ, p). Se (q, ϵ, p) é executada, então nenhum símbolo de entrada é consumido.
- Se (q, w) e (q', w') são duas configurações de M, então $(q, w) \vdash (q', w')$ *see* $w = aw'$ para algum $a \in \Sigma \cup \{\epsilon\}$, e $(q, a, q') \in \Delta$.
- $\vdash_M$ não precisa ser uma função. Pode haver vários pares (q', w'), ou nenhum, tais que $(q, w) \vdash_M (q', w')$.
- $\vdash_M^*$ é o fechamento transitivo reflexivo de $\vdash_M$.
- Uma cadeia $w \in \Sigma^*$ é aceita *see* houver um estado $q \in F$ tal que $(s, w) \vdash_M^* (q, \epsilon)$.
- $L(M)$ é a linguagem aceita por M.

Exemplo

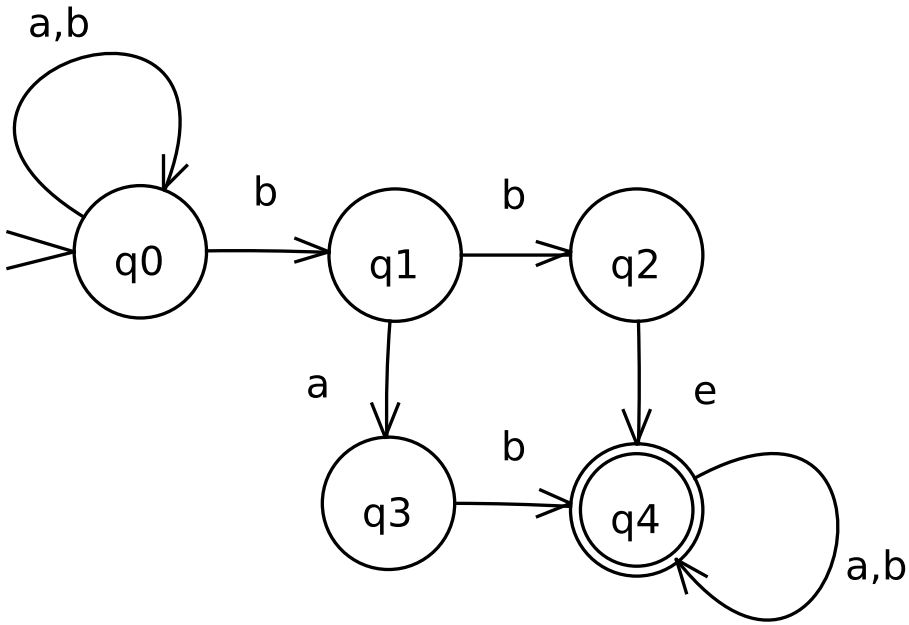
- Seja o AFN que reconhece todas as cadeias que tenham uma ocorrência de bb ou bab.
- Formalmente, $M = (K, \Sigma, \Delta, s, F)$ onde $K = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_4\}$, e

Exemplo

- Seja o AFN que reconhece todas as cadeias que tenham uma ocorrência de *bb* ou *bab*.
- Formalmente, $M = (K, \Sigma, \Delta, s, F)$ onde $K = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_4\}$, e

$$\begin{aligned}\Delta = \quad & \{(q_0, a, q_0), (q_0, b, q_0), (q_0, b, q_1), \\ & (q_1, b, q_2), (q_1, a, q_3), (q_2, \epsilon, q_4), \\ & (q_3, b, q_4), (q_4, a, q_4), (q_4, b, q_4)\}\end{aligned}$$

Exemplo



Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$(q_0, bababab) \quad \vdash_M \quad (q_0, ababab)$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab) \\ &\vdash_M (q_0, babab)\end{aligned}$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\end{aligned}$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$
\begin{aligned}
(q_0, bababab) &\vdash_M (q_0, ababab) \\
&\vdash_M (q_0, babab) \\
&\vdash_M (q_0, abab) \\
&\vdots
\end{aligned}
$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$
\begin{aligned}
(q_0, bababab) &\vdash_M (q_0, ababab) \\
&\vdash_M (q_0, babab) \\
&\vdash_M (q_0, abab) \\
&\vdots \\
&\vdash_M (q_0, \epsilon)
\end{aligned}
$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\\ &\vdots\\ &\vdash_M (q_0, \epsilon)\end{aligned}$$

- Mesma cadeia pode direcionar M para o estado final q_4

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\\ &\vdots\\ &\vdash_M (q_0, \epsilon)\end{aligned}$$

- Mesma cadeia pode direcionar M para o estado final q_4

$$(q_0, bababab) \vdash_M (q_1, ababab)$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\\ &\vdots\\ &\vdash_M (q_0, \epsilon)\end{aligned}$$

- Mesma cadeia pode direcionar M para o estado final q_4

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_1, ababab)\\ &\vdash_M (q_3, babab)\end{aligned}$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\\ &\vdots\\ &\vdash_M (q_0, \epsilon)\end{aligned}$$

- Mesma cadeia pode direcionar M para o estado final q_4

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_1, ababab)\\ &\vdash_M (q_3, babab)\\ &\vdash_M (q_4, abab)\end{aligned}$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\\ &\vdots\\ &\vdash_M (q_0, \epsilon)\end{aligned}$$

- Mesma cadeia pode direcionar M para o estado final q_4

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_1, ababab)\\ &\vdash_M (q_3, babab)\\ &\vdash_M (q_4, abab)\\ &\vdash_M (q_4, bab)\end{aligned}$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\\ &\vdots\\ &\vdash_M (q_0, \epsilon)\end{aligned}$$

- Mesma cadeia pode direcionar M para o estado final q_4

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_1, ababab)\\ &\vdash_M (q_3, babab)\\ &\vdash_M (q_4, abab)\\ &\vdash_M (q_4, bab)\\ &\vdash_M (q_4, ab)\end{aligned}$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\\ &\vdots\\ &\vdash_M (q_0, \epsilon)\end{aligned}$$

- Mesma cadeia pode direcionar M para o estado final q_4

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_1, ababab)\\ &\vdash_M (q_3, babab)\\ &\vdash_M (q_4, abab)\\ &\vdash_M (q_4, bab)\\ &\vdash_M (q_4, ab)\\ &\vdash_M (q_4, b)\end{aligned}$$

Exemplo

- Se forem usadas transições q_0, a, q_0 e (q_0, b, q_0)

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_0, ababab)\\ &\vdash_M (q_0, babab)\\ &\vdash_M (q_0, abab)\\ &\vdots\\ &\vdash_M (q_0, \epsilon)\end{aligned}$$

- Mesma cadeia pode direcionar M para o estado final q_4

$$\begin{aligned}(q_0, bababab) &\vdash_M (q_1, ababab)\\ &\vdash_M (q_3, babab)\\ &\vdash_M (q_4, abab)\\ &\vdash_M (q_4, bab)\\ &\vdash_M (q_4, ab)\\ &\vdash_M (q_4, b)\\ &\vdash_M (q_4, \epsilon)\end{aligned}$$

AFDs e AFNs

- Um AFD é apenas um caso particular de AFN.

AFDs e AFNs

- Um AFD é apenas um caso particular de AFN.
- Em um AFD, a relação de transição Δ é uma função de $K \times \Sigma$ para K.

AFDs e AFNs

- Um AFD é apenas um caso particular de AFN.
- Em um AFD, a relação de transição Δ é uma função de $K \times \Sigma$ para K.
- Ou seja, um AFN será determinístico see não tiver transições da forma $(q, \epsilon, p) \in \Delta$ e para cada $q \in K$ e $a \in \Sigma$, houver um e um só $p \in K$ tal que $(q, a, p) \in \Delta$.

AFDs e AFNs

- Um AFD é apenas um caso particular de AFN.
- Em um AFD, a relação de transição Δ é uma função de $K \times \Sigma$ para K.
- Ou seja, um AFN será determinístico see não tiver transições da forma $(q, \epsilon, p) \in \Delta$ e para cada $q \in K$ e $a \in \Sigma$, houver um e um só $p \in K$ tal que $(q, a, p) \in \Delta$.
- Um AFN pode ser convertido em um AFD equivalente.

AFDs e AFNs

- Um AFD é apenas um caso particular de AFN.
- Em um AFD, a relação de transição Δ é uma função de $K \times \Sigma$ para K.
- Ou seja, um AFN será determinístico see não tiver transições da forma $(q, \epsilon, p) \in \Delta$ e para cada $q \in K$ e $a \in \Sigma$, houver um e um só $p \in K$ tal que $(q, a, p) \in \Delta$.
- Um AFN pode ser convertido em um AFD equivalente.
- Dois autômatos finitos M_1 e M_2 (AFD ou AFN) são **equivalentes** see $L(M_1) = L(M_2)$.

AFDs e AFNs

- Um AFD é apenas um caso particular de AFN.
- Em um AFD, a relação de transição Δ é uma função de $K \times \Sigma$ para K.
- Ou seja, um AFN será determinístico see não tiver transições da forma $(q, \epsilon, p) \in \Delta$ e para cada $q \in K$ e $a \in \Sigma$, houver um e um só $p \in K$ tal que $(q, a, p) \in \Delta$.
- Um AFN pode ser convertido em um AFD equivalente.
- Dois autômatos finitos M_1 e M_2 (AFD ou AFN) são **equivalentes** see $L(M_1) = L(M_2)$.
- **Teorema:** Para cada AFN, há um AFD equivalente.

Convertendo AFNs em AFDs

- Seja $M = (K, \Sigma, \Delta, s, F)$ um AFN. Pretendemos construir um AFD $M' = (K', \Sigma, \delta, s', F')$ equivalente a M.

Convertendo AFNs em AFDs

- Seja $M = (K, \Sigma, \Delta, s, F)$ um AFN. Pretendemos construir um AFD $M' = (K', \Sigma, \delta, s', F')$ equivalente a M.
- A idéia central é visualizar um AFN como se ele ocupasse, em qualquer momento, não um só estado, mas o conjunto de todos os estados que podem ser alcançados a partir do estado inicial por meio da parte da cadeia consumida até então.

Convertendo AFNs em AFDs

- Seja $M = (K, \Sigma, \Delta, s, F)$ um AFN. Pretendemos construir um AFD $M' = (K', \Sigma, \delta, s', F')$ equivalente a M.
- A idéia central é visualizar um AFN como se ele ocupasse, em qualquer momento, não um só estado, mas o conjunto de todos os estados que podem ser alcançados a partir do estado inicial por meio da parte da cadeia consumida até então.
  1. Se M tiver 5 estados $\{q_0, \ldots, q_4\}$, e após a leitura de certa cadeia , atingir o estado q_0, q_2 ou q_3, mas não q_1 ou q_4, seu estado poderia ser $\{q_0, q_2, q_3\}$, em vez de algum membro não-determinado desse conjunto.

Convertendo AFNs em AFDs

- Seja $M = (K, \Sigma, \Delta, s, F)$ um AFN. Pretendemos construir um AFD $M' = (K', \Sigma, \delta, s', F')$ equivalente a M.
- A idéia central é visualizar um AFN como se ele ocupasse, em qualquer momento, não um só estado, mas o conjunto de todos os estados que podem ser alcançados a partir do estado inicial por meio da parte da cadeia consumida até então.
  1. Se M tiver 5 estados $\{q_0, \ldots, q_4\}$, e após a leitura de certa cadeia , atingir o estado q_0, q_2 ou q_3, mas não q_1 ou q_4, seu estado poderia ser $\{q_0, q_2, q_3\}$, em vez de algum membro não-determinado desse conjunto.
  2. Se o próximo símbolo pudesse levar M de q_0 para q_1 ou q_2, de q_2 para q_0 e de q_3 para q_2, então o próximo estado de M poderia ser o conjunto $\{q_0, q_1, q_2\}$.

Convertendo AFNs em AFDs

- Para qualquer estado $q \in K$, seja $E(q)$ o conjunto de todos os estados de M alcançáveis a partir de q, sem a leitura de qualquer símbolo de entrada, isto é,

$$E(q) = \{p \in K : (q, \epsilon) \vdash_M^* (p, \epsilon)\}.$$

Convertendo AFNs em AFDs

- Para qualquer estado $q \in K$, seja $E(q)$ o conjunto de todos os estados de M alcançáveis a partir de q, sem a leitura de qualquer símbolo de entrada, isto é,

$$E(q) = \{p \in K : (q, \epsilon) \vdash_M^* (p, \epsilon)\}.$$

- $E(q)$ é o fechamento do conjunto $\{q\}$ sob a relação

$$\{(p, r) : \text{existe uma transição } (p, \epsilon, r) \in \Delta\}.$$

Convertendo AFNs em AFDs

- Para qualquer estado $q \in K$, seja $E(q)$ o conjunto de todos os estados de M alcançáveis a partir de q, sem a leitura de qualquer símbolo de entrada, isto é,

$$E(q) = \{p \in K : (q, \epsilon) \vdash_M^* (p, \epsilon)\}.$$

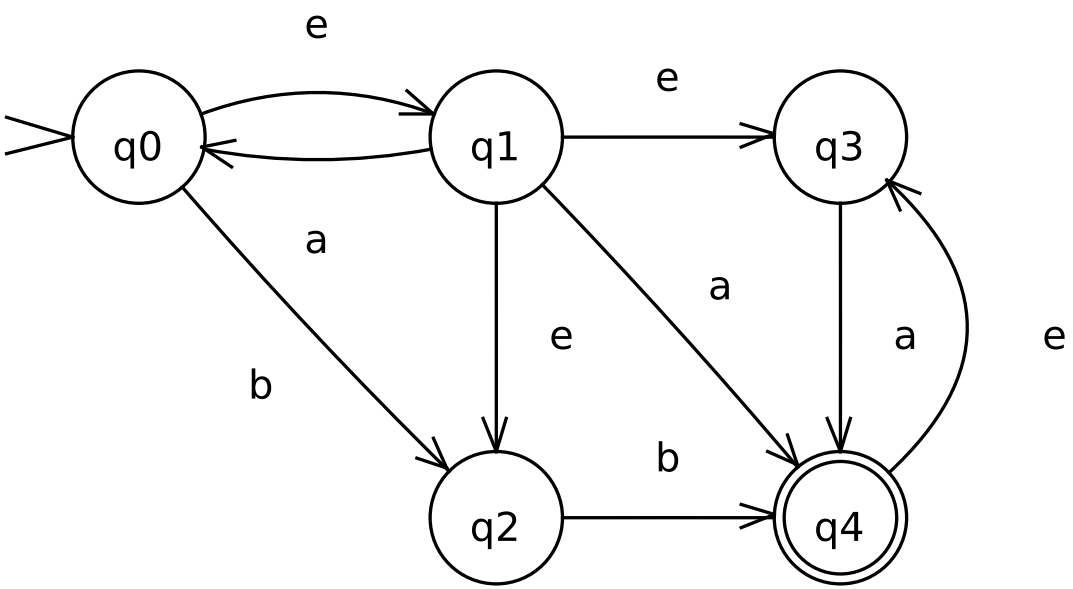
- $E(q)$ é o fechamento do conjunto $\{q\}$ sob a relação

$$\{(p, r) : \text{existe uma transição } (p, \epsilon, r) \in \Delta\}.$$

- $E(q)$ pode ser computado pelo algoritmo
  1. Inicialmente faça $E(q) = \{q\}$
  2. Enquanto existir $(p, \epsilon, r) \in \Delta$ com $p \in E(q)$ e $r \notin E(q)$ faça
     - 2.1 $E(q) = E(q) \cup \{r\}$

Convertendo AFNs em AFDs

- No AFN abaixo, temos $E(q_0) = \{q_0, q_1, q_2, q_3\}$, $E(q_1) = \{q_1, q_2, q_3\}$, $E(q_2) = \{q_2\}$, $E(q_3) = \{q_3\}$ e $E(q_4) = \{q_3, q_4\}$.



Convertendo AFNs em AFDs

- Podemos agora definir formalmente o AFD $M' = (K', \Sigma, \delta, s', F')$ que é equivalente a M.

Convertendo AFNs em AFDs

- Podemos agora definir formalmente o AFD $M' = (K', \Sigma, \delta, s', F')$ que é equivalente a M.
  - $K' = 2^K$
  - $s' = E(s)$
  - $F' = \{Q \subseteq K : Q \cap F \neq \emptyset\}$
  - Para cada $Q \subseteq K$ e cada símbolo $a \in \Sigma$, defina

$$\delta(Q, a) = \bigcup \{E(p) : p \in K \text{ e } (q, a, p) \in \Delta \text{ para algum } q \in Q\}$$

Convertendo AFNs em AFDs

- Podemos agora definir formalmente o AFD $M' = (K', \Sigma, \delta, s', F')$ que é equivalente a M.
  - $K' = 2^K$
  - $s' = E(s)$
  - $F' = \{Q \subseteq K : Q \cap F \neq \emptyset\}$
  - Para cada $Q \subseteq K$ e cada símbolo $a \in \Sigma$, defina

$$\delta(Q, a) = \bigcup \{E(p) : p \in K \text{ e } (q, a, p) \in \Delta \text{ para algum } q \in Q\}$$

- $\delta(Q, a)$ é o conjunto de todos os estados de M para os quais M pode ir através da leitura da entrada a, possivelmente executando várias transições em vazio.

Convertendo AFNs em AFDs

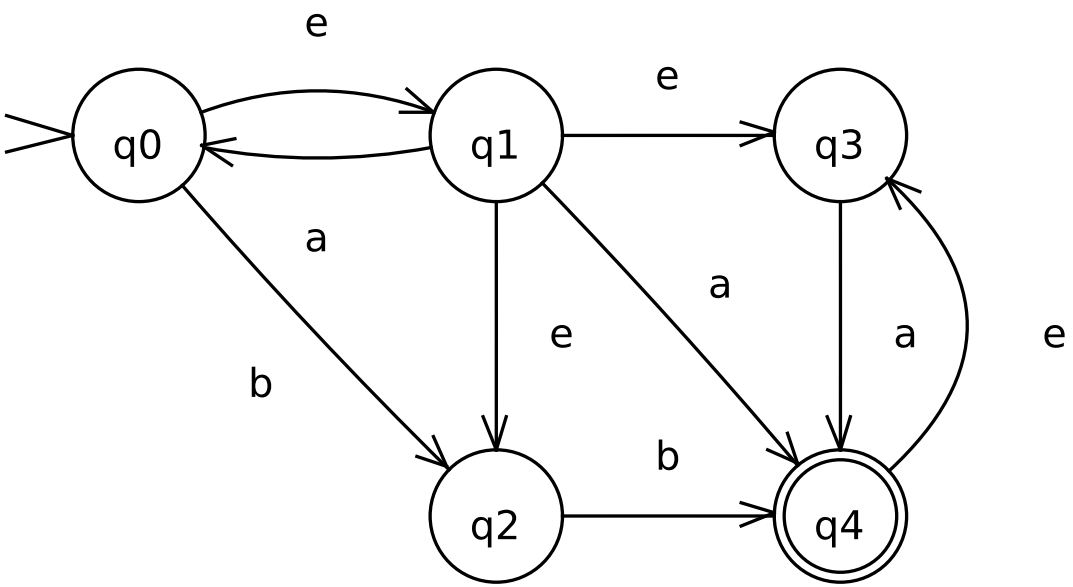
- Podemos agora definir formalmente o AFD $M' = (K', \Sigma, \delta, s', F')$ que é equivalente a M.
  - $K' = 2^K$
  - $s' = E(s)$
  - $F' = \{Q \subseteq K : Q \cap F \neq \emptyset\}$
  - Para cada $Q \subseteq K$ e cada símbolo $a \in \Sigma$, defina

$$\delta(Q, a) = \bigcup \{E(p) : p \in K \text{ e } (q, a, p) \in \Delta \text{ para algum } q \in Q\}$$

- $\delta(Q, a)$ é o conjunto de todos os estados de M para os quais M pode ir através da leitura da entrada a, possivelmente executando várias transições em vazio.
- Como as únicas transições de q_1, para a, são (q_1, a, q_0) e (q_1, a, q_4), então $\delta(\{q_1\}, a) = E(q_0) \cup E(q_4) = \{q_0, q_1, q_2, q_3, q_4\}$

Exemplo

- Seja o AFN anterior.



Exemplo

- Definimos $E(q)$ para cada estado q de M. Como $s' = \{q_0, q_1, q_2, q_3\}$ e (q_1, a, q_0), (q_1, a, q_4) e (q_3, a, q_4), são todas as transições (q, a, p) para algum $q \in s'$, então

$$\delta(s', a) = E(q_0) \cup E(q_4) = \{q_0, q_1, q_2, q_3, q_4\}.$$

Exemplo

- Definimos $E(q)$ para cada estado q de M. Como $s' = \{q_0, q_1, q_2, q_3\}$ e (q_1, a, q_0), (q_1, a, q_4) e (q_3, a, q_4), são todas as transições (q, a, p) para algum $q \in s'$, então

$$\delta(s', a) = E(q_0) \cup E(q_4) = \{q_0, q_1, q_2, q_3, q_4\}.$$

- Analogamente, (q_0, b, q_2) e (q_2, b, q_4), são todas as transições (q, b, p) para algum $q \in s'$, então

$$\delta(s', b) = E(q_2) \cup E(q_4) = \{q_2, q_3, q_4\}.$$

Exemplo

- Repetindo esse cálculo para os estados recém criados, temos

$$\begin{aligned}
\delta(\{q_0, q_1, q_2, q_3, q_4\}, a) &= \{q_0, q_1, q_2, q_3, q_4\} \\
\delta(\{q_0, q_1, q_2, q_3, q_4\}, b) &= \{q_2, q_3, q_4\} \\
\delta(\{q_2, q_3, q_4\}, a) &= E(q_4) = \{q_3, q_4\} \\
\delta(\{q_2, q_3, q_4\}, b) &= E(q_4) = \{q_3, q_4\}.
\end{aligned}$$

Exemplo

- Repetindo esse cálculo para os estados recém criados, temos

$$\begin{aligned}
\delta(\{q_0, q_1, q_2, q_3, q_4\}, a) &= \{q_0, q_1, q_2, q_3, q_4\} \\
\delta(\{q_0, q_1, q_2, q_3, q_4\}, b) &= \{q_2, q_3, q_4\} \\
\delta(\{q_2, q_3, q_4\}, a) &= E(q_4) = \{q_3, q_4\} \\
\delta(\{q_2, q_3, q_4\}, b) &= E(q_4) = \{q_3, q_4\}.
\end{aligned}$$

- Em seguida,

$$\begin{aligned}
\delta(\{q_3, q_4\}, a) &= E(q_4) = \{q_3, q_4\} \\
\delta(\{q_3, q_4\}, b) &= \emptyset.
\end{aligned}$$

Exemplo

- Repetindo esse cálculo para os estados recém criados, temos

$$\begin{aligned}
\delta(\{q_0, q_1, q_2, q_3, q_4\}, a) &= \{q_0, q_1, q_2, q_3, q_4\} \\
\delta(\{q_0, q_1, q_2, q_3, q_4\}, b) &= \{q_2, q_3, q_4\} \\
\delta(\{q_2, q_3, q_4\}, a) &= E(q_4) = \{q_3, q_4\} \\
\delta(\{q_2, q_3, q_4\}, b) &= E(q_4) = \{q_3, q_4\}.
\end{aligned}$$

- Em seguida,

$$\begin{aligned}
\delta(\{q_3, q_4\}, a) &= E(q_4) = \{q_3, q_4\} \\
\delta(\{q_3, q_4\}, b) &= \emptyset.
\end{aligned}$$

- E, finalmente,

$$\delta(\emptyset, a) = \delta(\emptyset, b) = \emptyset.$$

Exemplo

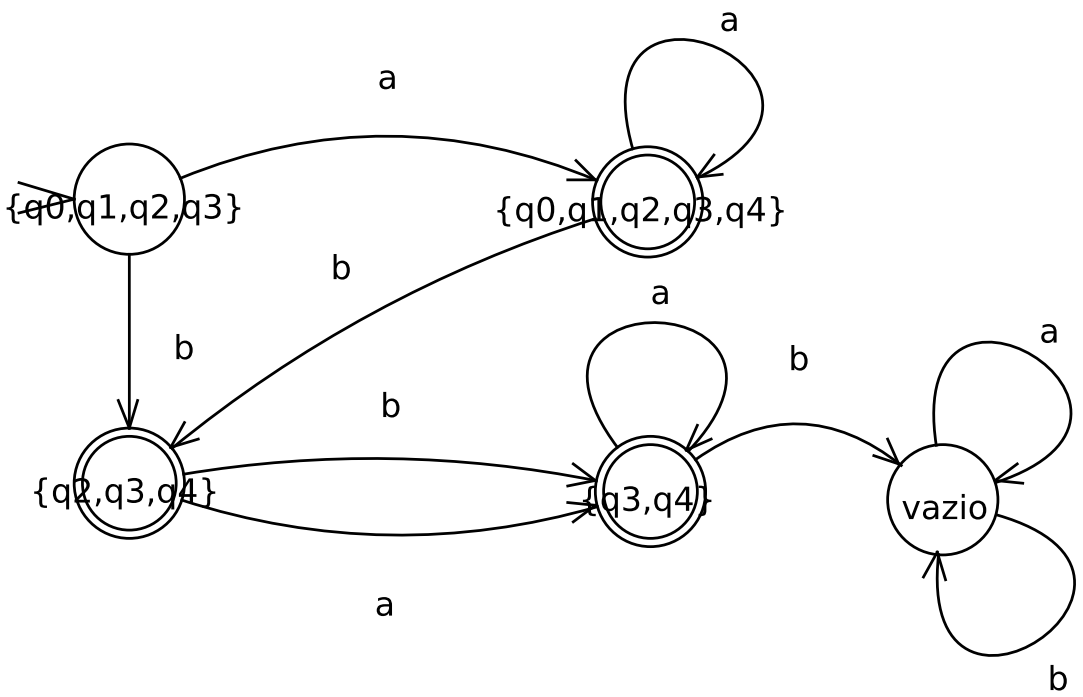
- F', o conjunto dos estados finais, contém cada um dos conjuntos dos quais q_4 é membro, já que $F = \{q_4\}$.

Exemplo

- F', o conjunto dos estados finais, contém cada um dos conjuntos dos quais q_4 é membro, já que $F = \{q_4\}$.
- Assim, $\{q_0, q_1, q_2, q_3, q_4\}$, $\{q_2, q_3, q_4\}$ e $\{q_3, q_4\}$ de M' são estados finais.

Exemplo

- F', o conjunto dos estados finais, contém cada um dos conjuntos dos quais q_4 é membro, já que $F = \{q_4\}$.
- Assim, $\{q_0, q_1, q_2, q_3, q_4\}$, $\{q_2, q_3, q_4\}$ e $\{q_3, q_4\}$ de M' são estados finais.



Autômatos finitos e expressões regulares

- A classe de linguagens reconhecidas por autômatos finitos (AFs) apresenta uma certa *estabilidade*: dois esquemas, um aparentemente mais poderoso que o outro, acabam definindo a mesma classe de linguagens.

Autômatos finitos e expressões regulares

- A classe de linguagens reconhecidas por autômatos finitos (AFs) apresenta uma certa *estabilidade*: dois esquemas, um aparentemente mais poderoso que o outro, acabam definindo a mesma classe de linguagens.
- A classe de linguagens aceitas por AFs é a classe das linguagens regulares, a mesma descrita por expressões regulares.

Autômatos finitos e expressões regulares

- A classe de linguagens reconhecidas por autômatos finitos (AFs) apresenta uma certa *estabilidade*: dois esquemas, um aparentemente mais poderoso que o outro, acabam definindo a mesma classe de linguagens.
- A classe de linguagens aceitas por AFs é a classe das linguagens regulares, a mesma descrita por expressões regulares.
- A classe das linguagens regulares é o fechamento de certas linguagens finitas sob as operações de união, concatenação e estrela de Kleene.

Autômatos finitos e expressões regulares

- A classe de linguagens reconhecidas por autômatos finitos (AFs) apresenta uma certa *estabilidade*: dois esquemas, um aparentemente mais poderoso que o outro, acabam definindo a mesma classe de linguagens.
- A classe de linguagens aceitas por AFs é a classe das linguagens regulares, a mesma descrita por expressões regulares.
- A classe das linguagens regulares é o fechamento de certas linguagens finitas sob as operações de união, concatenação e estrela de Kleene.

Autômatos finitos e expressões regulares

- Devemos então mostrar propriedades de fechamento para a classe de linguagens aceitas por AFs.

Teorema

A classe de linguagens aceitas por AFs é fechada em relação às operações de

1. união,
2. concatenação,
3. estrela de Kleene,
4. complementação e
5. intersecção.

- **Prova:** Em cada caso, mostraremos que dados dois AFs, M_1 e M_2 (apenas M_1 para a estrela de Kleene e complementação), é possível construir a linguagem correspondente.

Prova

1. **União.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \cup L(M_2)$.

Prova

1. **União.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \cup L(M_2)$.
   1.1 M lança mão do não-determinismo para adivinhar se a cadeia de entrada está em $L(M_1)$ ou $L(M_2)$, então processa a cadeia exatamente como o AF correspondente o faria.

Prova

1. **União.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \cup L(M_2)$.
   1.1 M lança mão do não-determinismo para adivinhar se a cadeia de entrada está em $L(M_1)$ ou $L(M_2)$, então processa a cadeia exatamente como o AF correspondente o faria.
   1.2 Sem perda de generalidade, admitimos que K_1 e K_2 são disjuntos.

Prova

1. **União.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \cup L(M_2)$.
   1.1 M lança mão do não-determinismo para adivinhar se a cadeia de entrada está em $L(M_1)$ ou $L(M_2)$, então processa a cadeia exatamente como o AF correspondente o faria.
   1.2 Sem perda de generalidade, admitimos que K_1 e K_2 são disjuntos.
   1.3 s é um novo estado que não está em K_1 ou K_2.

Prova

1. **União.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \cup L(M_2)$.
   1.1 M lança mão do não-determinismo para adivinhar se a cadeia de entrada está em $L(M_1)$ ou $L(M_2)$, então processa a cadeia exatamente como o AF correspondente o faria.
   1.2 Sem perda de generalidade, admitimos que K_1 e K_2 são disjuntos.
   1.3 s é um novo estado que não está em K_1 ou K_2.
   1.4 $K = K_1 \cup K_2 \cup \{s\}$,

Prova

1. **União.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \cup L(M_2)$.
   - 1.1 M lança mão do não-determinismo para adivinhar se a cadeia de entrada está em $L(M_1)$ ou $L(M_2)$, então processa a cadeia exatamente como o AF correspondente o faria.
   - 1.2 Sem perda de generalidade, admitimos que K_1 e K_2 são disjuntos.
   - 1.3 s é um novo estado que não está em K_1 ou K_2.
   - 1.4 $K = K_1 \cup K_2 \cup \{s\}$,
   - 1.5 $F = F_1 \cup F_2$,

Prova

1. **União.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \cup L(M_2)$.
   - 1.1 M lança mão do não-determinismo para adivinhar se a cadeia de entrada está em $L(M_1)$ ou $L(M_2)$, então processa a cadeia exatamente como o AF correspondente o faria.
   - 1.2 Sem perda de generalidade, admitimos que K_1 e K_2 são disjuntos.
   - 1.3 s é um novo estado que não está em K_1 ou K_2.
   - 1.4 $K = K_1 \cup K_2 \cup \{s\}$,
   - 1.5 $F = F_1 \cup F_2$,
   - 1.6 $\Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \epsilon, s_1), (s, \epsilon, s_2)\}$,

Prova

1. **União.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \cup L(M_2)$.
   - 1.1 M lança mão do não-determinismo para adivinhar se a cadeia de entrada está em $L(M_1)$ ou $L(M_2)$, então processa a cadeia exatamente como o AF correspondente o faria.
   - 1.2 Sem perda de generalidade, admitimos que K_1 e K_2 são disjuntos.
   - 1.3 s é um novo estado que não está em K_1 ou K_2.
   - 1.4 $K = K_1 \cup K_2 \cup \{s\}$,
   - 1.5 $F = F_1 \cup F_2$,
   - 1.6 $\Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \epsilon, s_1), (s, \epsilon, s_2)\}$,
   - 1.7 Se $w \in \Sigma^*$, então $(s, w) \vdash_M^* (q, \epsilon)$ para algum $q \in F$, *see* $(s_1, w) \vdash_{M_1}^* (q, \epsilon)$ para algum $q \in F_1$ ou $(s_2, w) \vdash_{M_2}^* (q, \epsilon)$ para algum $q \in F_2$.

Prova

2. **Concatenação.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \circ L(M_2)$.

Prova

2. **Concatenação.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \circ L(M_2)$.
   2.1 M opera como M_1 por alguns momentos e então "salta" não-deterministicamente de um estado final de M_1 para o estado inicial de M_2.

Prova

2. **Concatenação.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \circ L(M_2)$.
   - 2.1 M opera como M_1 por alguns momentos e então "salta" não-deterministicamente de um estado final de M_1 para o estado inicial de M_2.
   - 2.2 Os estados finais de M_1 deixam de ser finais em M, pois essa reconhece apenas $L(M_1) \circ L(M_2)$.

Prova

2. **Concatenação.** Sejam $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ e $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ dois AFNs, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1) \circ L(M_2)$.
   - 2.1 M opera como M_1 por alguns momentos e então "salta" não-deterministicamente de um estado final de M_1 para o estado inicial de M_2.
   - 2.2 Os estados finais de M_1 deixam de ser finais em M, pois essa reconhece apenas $L(M_1) \circ L(M_2)$.
   - 2.3 Os estados finais de M são os mesmos de M_2.

Prova

3. **Estrela de Kleene.** Seja $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ um AFN, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1)^*$.

Prova

3. **Estrela de Kleene.** Seja $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ um AFN, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1)^*$.
   3.1 M é composto pelos estados e todas as transições de M_1.

Prova

3. **Estrela de Kleene.** Seja $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ um AFN, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1)^*$.
   - 3.1 M é composto pelos estados e todas as transições de M_1.
   - 3.2 Qualquer estado final de M_1 é também estado final em M.

Prova

3. **Estrela de Kleene.** Seja $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ um AFN, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1)^*$.
   - 3.1 M é composto pelos estados e todas as transições de M_1.
   - 3.2 Qualquer estado final de M_1 é também estado final em M.
   - 3.3 M tem um novo estado inicial s. Esse novo estado também é final, assim a cadeia vazia ϵ é aceita.

Prova

3. **Estrela de Kleene.** Seja $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ um AFN, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1)^*$.
   - 3.1 M é composto pelos estados e todas as transições de M_1.
   - 3.2 Qualquer estado final de M_1 é também estado final em M.
   - 3.3 M tem um novo estado inicial s. Esse novo estado também é final, assim a cadeia vazia ϵ é aceita.
   - 3.4 De s há uma transição em vazio para o estado inicial s_1 de M_1, de tal modo que a operação de M_1 possa ser iniciada logo após M ter sido iniciada no estado s.

Prova

3. **Estrela de Kleene.** Seja $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ um AFN, devemos construir um AFN $M = (K, \Sigma, \Delta, s, F)$ tal que $L(M) = L(M_1)^*$.
   - 3.1 M é composto pelos estados e todas as transições de M_1.
   - 3.2 Qualquer estado final de M_1 é também estado final em M.
   - 3.3 M tem um novo estado inicial s. Esse novo estado também é final, assim a cadeia vazia ϵ é aceita.
   - 3.4 De s há uma transição em vazio para o estado inicial s_1 de M_1, de tal modo que a operação de M_1 possa ser iniciada logo após M ter sido iniciada no estado s.
   - 3.5 Outras transições em vazio são adicionadas a partir de cada estado final de M_1 de volta para s_1, desse modo, uma vez que uma cadeia de $L(M_1)$ tenha sido lida, a computação poderá prosseguir a partir do estado inicial de M_1.

Prova

4. **Complementação.** Seja $M = (K, \Sigma, \delta, s, F)$ um AFD. Então, a linguagem complementar $L = \Sigma^* - L(M)$ é aceita pelo AFD $\bar{M} = (K, \Sigma, \delta, s, K - F)$. Isto é, $\bar{M}$ é idêntico a M, exceto que estados finais e não-finais são intercambiados.

Prova

4. **Complementação.** Seja $M = (K, \Sigma, \delta, s, F)$ um AFD. Então, a linguagem complementar $L = \Sigma^* - L(M)$ é aceita pelo AFD $\bar{M} = (K, \Sigma, \delta, s, K - F)$. Isto é, $\bar{M}$ é idêntico a M, exceto que estados finais e não-finais são intercambiados.
5. **Intersecção.** Basta considerar que

$$L_1 \cap L_2 = \Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2)).$$

Assim, o fechamento em relação à intersecção pode ser obtido a partir do fechamento sob união e complementação mostrados anteriormente.

Autômatos finitos e expressões regulares

Identificamos a equivalência de duas importantes técnicas de especificação finita de linguagens - um gerador e um reconhecedor de linguagem.

Autômatos finitos e expressões regulares

Identificamos a equivalência de duas importantes técnicas de especificação finita de linguagens - um gerador e um reconhecedor de linguagem.

Teorema

Uma linguagem é regular se e somente se ela for aceita por um autômato finito.

Prova

- (*Somente se*) A classe das linguagens regulares é a menor classe de linguagens contendo $\emptyset$ e os conjuntos unitários $\{a\}$, onde a é um símbolo do alfabeto e é fechada em relação às operações de união, concatenação e estrela de Kleene.

Prova

- (*Somente se*) A classe das linguagens regulares é a menor classe de linguagens contendo $\emptyset$ e os conjuntos unitários $\{a\}$, onde a é um símbolo do alfabeto e é fechada em relação às operações de união, concatenação e estrela de Kleene.
- É evidente que $\emptyset$ e os conjuntos unitários são aceitos por AFs.

Prova

- (*Somente se*) A classe das linguagens regulares é a menor classe de linguagens contendo $\emptyset$ e os conjuntos unitários $\{a\}$, onde a é um símbolo do alfabeto e é fechada em relação às operações de união, concatenação e estrela de Kleene.
- É evidente que $\emptyset$ e os conjuntos unitários são aceitos por AFs.
- As linguagens representadas por AFs são fechadas em relação à união, concatenação e estrela de Kleene.

Prova

- (*Somente se*) A classe das linguagens regulares é a menor classe de linguagens contendo $\emptyset$ e os conjuntos unitários $\{a\}$, onde a é um símbolo do alfabeto e é fechada em relação às operações de união, concatenação e estrela de Kleene.
- É evidente que $\emptyset$ e os conjuntos unitários são aceitos por AFs.
- As linguagens representadas por AFs são fechadas em relação à união, concatenação e estrela de Kleene.
- Portanto, todas as linguagens regulares são aceitas por AFs.

Prova

- (*Somente se*) A classe das linguagens regulares é a menor classe de linguagens contendo $\emptyset$ e os conjuntos unitários $\{a\}$, onde a é um símbolo do alfabeto e é fechada em relação às operações de união, concatenação e estrela de Kleene.
- É evidente que $\emptyset$ e os conjuntos unitários são aceitos por AFs.
- As linguagens representadas por AFs são fechadas em relação à união, concatenação e estrela de Kleene.
- Portanto, todas as linguagens regulares são aceitas por AFs.
- **Exemplo:** Um AF que aceita a linguagem gerada por $(ab \cup abb)^*$ pode ser construído de acordo com os métodos mostrados anteriormente.

Linguagens regulares e não-regulares

- Mostramos que as linguagens regulares são fechadas em relação a diversas operações, e que podem ser especificadas por expressões regulares ou por AFDs ou AFNs.

Linguagens regulares e não-regulares

- Mostramos que as linguagens regulares são fechadas em relação a diversas operações, e que podem ser especificadas por expressões regulares ou por AFDs ou AFNs.
- Esses fatos nos oferecem uma diversidade de técnicas para mostrar que uma linguagem é regular.

Exemplo

- Seja $\Sigma = \{0, 1, \ldots, 9\}$ e $L \subseteq \Sigma^*$ o conjunto de representações decimais para inteiros não-negativos (sem 0s redundantes) divisíveis por 2 e por 3.

Exemplo

- Seja $\Sigma = \{0, 1, \ldots, 9\}$ e $L \subseteq \Sigma^*$ o conjunto de representações decimais para inteiros não-negativos (sem 0s redundantes) divisíveis por 2 e por 3.
  1. Seja L_1 o conjunto de representações decimais de inteiros não-negativos, assim,

     $$L_1 = 0 \cup \{1, 2, \ldots, 9\}\Sigma^*$$

     é regular, pois é denotada por uma expressão regular.

Exemplo

- Seja $\Sigma = \{0, 1, \ldots, 9\}$ e $L \subseteq \Sigma^*$ o conjunto de representações decimais para inteiros não-negativos (sem 0s redundantes) divisíveis por 2 e por 3.
  1. Seja L_1 o conjunto de representações decimais de inteiros não-negativos, assim,

     $$L_1 = 0 \cup \{1, 2, \ldots, 9\}\Sigma^*$$

     é regular, pois é denotada por uma expressão regular.
  2. Seja L_2 o conjunto de representações decimais de números inteiros não-negativos divisíveis por 2. Então, L_2 é apenas o conjuntos dos membros de L_1 terminados em 0, 2, 4, 6, 8, ou seja,

     $$L_2 = L_1 \cap \Sigma^*\{0, 2, 4, 6, 8\}.$$

Exemplo

- Seja $\Sigma = \{0, 1, \ldots, 9\}$ e $L \subseteq \Sigma^*$ o conjunto de representações decimais para inteiros não-negativos (sem 0s redundantes) divisíveis por 2 e por 3.
  1. Seja L_1 o conjunto de representações decimais de inteiros não-negativos, assim,

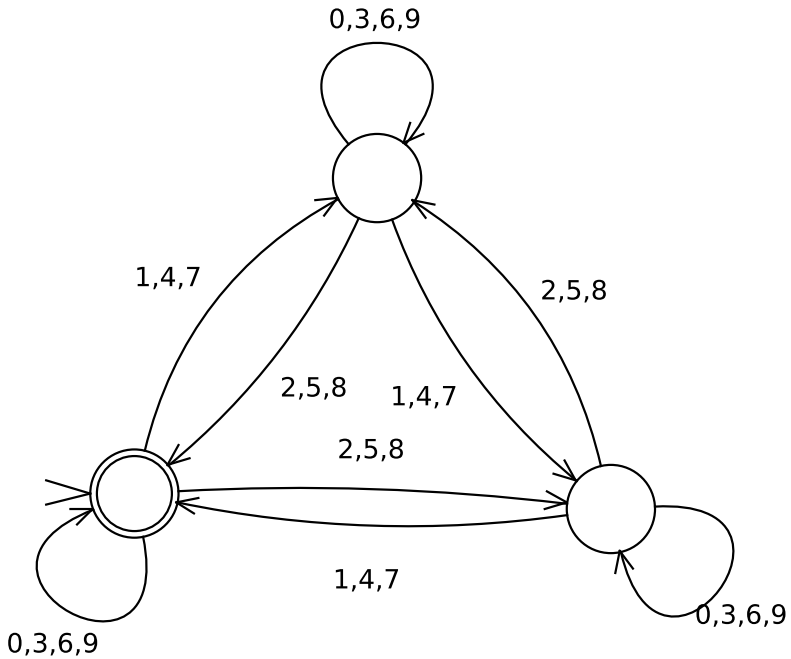
     $$L_1 = 0 \cup \{1, 2, \ldots, 9\}\Sigma^*$$

     é regular, pois é denotada por uma expressão regular.
  2. Seja L_2 o conjunto de representações decimais de números inteiros não-negativos divisíveis por 2. Então, L_2 é apenas o conjuntos dos membros de L_1 terminados em 0, 2, 4, 6, 8, ou seja,

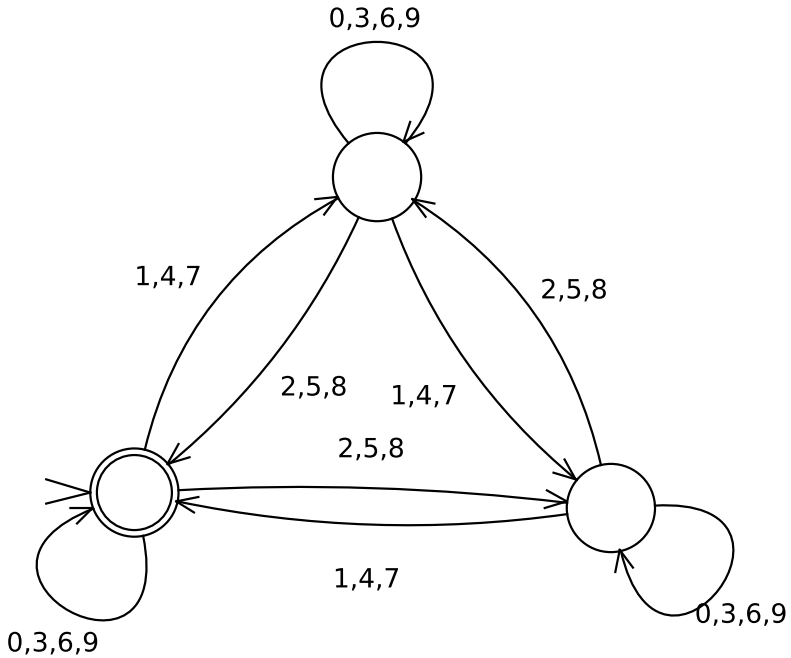
     $$L_2 = L_1 \cap \Sigma^*\{0, 2, 4, 6, 8\}.$$

  3. Seja L_3 o conjunto de representações decimais de números inteiros não-negativos divisíveis por 3. Construimos o seguinte AF para tal fim.

Exemplo

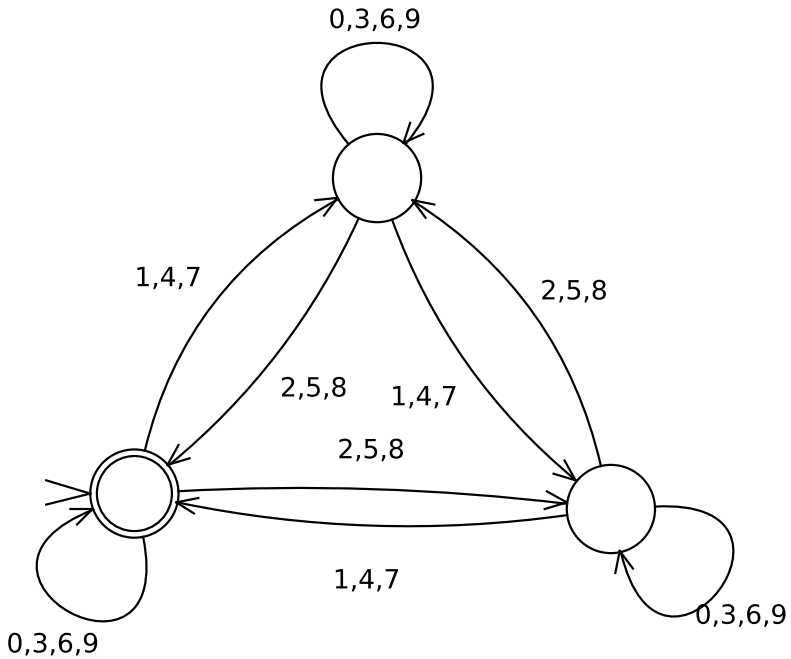


Exemplo



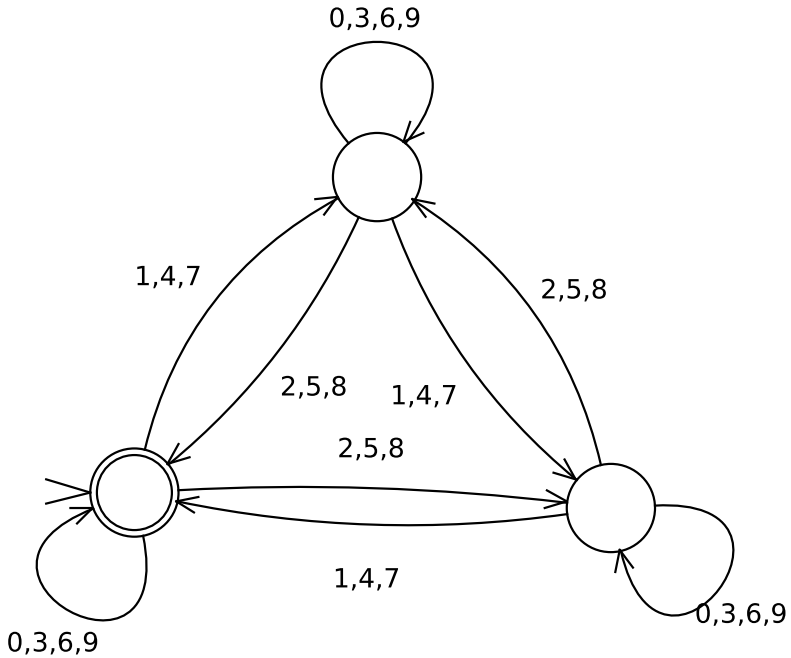
- Então, $L = L_1 \cup L_2$ é regular.

Exemplo



- Então, $L = L_1 \cup L_2$ é regular.
- Sabemos que o conjuntos de todas as expressões regulares (e de AFs) é contável, porém o conjunto de todas as linguagens não é. Assim, aprendemos que existem linguagens não-regulares.

Exemplo



- Então, $L = L_1 \cup L_2$ é regular.
- Sabemos que o conjuntos de todas as expressões regulares (e de AFs) é contável, porém o conjunto de todas as linguagens não é. Assim, aprendemos que existem linguagens não-regulares.
- Demonstrar que uma linguagem não é regular requer técnicas especiais.

Teorema de bombeamento

Duas propriedades são compartilhadas por linguagens regulares, mas não por certas linguagens não-regulares.

Teorema de bombeamento

Duas propriedades são compartilhadas por linguagens regulares, mas não por certas linguagens não-regulares.

1. Como uma cadeia é lida da esquerda para direita, a quantidade de memória exigida para determinar se a cadeia pertence ou não à linguagem deve ser limitada, fixada previamente e dependente da linguagem e não da cadeia em particular.
   - Exemplo: esperamos que $\{a^n b^n : n \geq 0\}$ não seja regular.

Teorema de bombeamento

Duas propriedades são compartilhadas por linguagens regulares, mas não por certas linguagens não-regulares.

1. Como uma cadeia é lida da esquerda para direita, a quantidade de memória exigida para determinar se a cadeia pertence ou não à linguagem deve ser limitada, fixada previamente e dependente da linguagem e não da cadeia em particular.
   - Exemplo: esperamos que $\{a^n b^n : n \geq 0\}$ não seja regular.
2. Linguagens regulares com um número infinito de cadeias são representadas por AFs com ciclos e por ERs com estrela de Kleene.
   - Tais linguagens devem ter subconjuntos infinitos com uma certa estrutura simples e repetitiva associada à estrela de Kleene em uma ER correspondente, ou a um circuito no diagrama de estados do AF.
   - Assim, suspeitamos que $\{a^n : n \geq 1$ é um primo$\}$ não é regular, pois não há periodicidade simples no conjunto dos primos.

Teorema de bombeamento

Teorema

Seja L uma linguagem regular. Há um inteiro $n \geq 1$ tal que qualquer cadeia $w \in L$ com $|w| \geq n$ pode ser reescrita como $w = xyz$, tal que $y \neq \epsilon$, $|xy| \leq n$, e $xy^i z \in L$ para cada $i \geq 0$.

Teorema de bombeamento

Teorema

Seja L uma linguagem regular. Há um inteiro $n \geq 1$ tal que qualquer cadeia $w \in L$ com $|w| \geq n$ pode ser reescrita como $w = xyz$, tal que $y \neq \epsilon$, $|xy| \leq n$, e $xy^i z \in L$ para cada $i \geq 0$.

- Esse teorema pertence a uma classe geral, chamada de teoremas de bombeamento (*pumping theorems*), porque afirmam a existência de certos pontos em determinadas cadeias, onde uma subcadeia pode ser repetidamente inserida sem afetar a aceitação da mesma.

Teorema de bombeamento

Às vezes, é útil pensar na aplicação desse teorema como um jogo entre você, que tentará provar que uma linguagem L não é regular, e um adversário, que insiste que L é regular.

Teorema de bombeamento

Às vezes, é útil pensar na aplicação desse teorema como um jogo entre você, que tentará provar que uma linguagem L não é regular, e um adversário, que insiste que L é regular.

1. O adversário deve iniciar estabelecendo um número n.

Teorema de bombeamento

Às vezes, é útil pensar na aplicação desse teorema como um jogo entre você, que tentará provar que uma linguagem L não é regular, e um adversário, que insiste que L é regular.

1. O adversário deve iniciar estabelecendo um número n.
2. Então, você apresenta uma cadeia w que tenha comprimento de, no mínimo, n.

Teorema de bombeamento

Às vezes, é útil pensar na aplicação desse teorema como um jogo entre você, que tentará provar que uma linguagem L não é regular, e um adversário, que insiste que L é regular.

1. O adversário deve iniciar estabelecendo um número n.
2. Então, você apresenta uma cadeia w que tenha comprimento de, no mínimo, n.
3. O adversário deve agora produzir uma decomposição apropriada de w na forma $w = xyz$,

Teorema de bombeamento

Às vezes, é útil pensar na aplicação desse teorema como um jogo entre você, que tentará provar que uma linguagem L não é regular, e um adversário, que insiste que L é regular.

1. O adversário deve iniciar estabelecendo um número n.
2. Então, você apresenta uma cadeia w que tenha comprimento de, no mínimo, n.
3. O adversário deve agora produzir uma decomposição apropriada de w na forma $w = xyz$,
4. Até que você, triunfante, indique i para o qual xy^iz não pertença a L.

Teorema de bombeamento

Às vezes, é útil pensar na aplicação desse teorema como um jogo entre você, que tentará provar que uma linguagem L não é regular, e um adversário, que insiste que L é regular.

1. O adversário deve iniciar estabelecendo um número n.
2. Então, você apresenta uma cadeia w que tenha comprimento de, no mínimo, n.
3. O adversário deve agora produzir uma decomposição apropriada de w na forma $w = xyz$,
4. Até que você, triunfante, indique i para o qual xy^iz não pertença a L.
5. Se você tiver uma estratégia sempre vitoriosa, então provou que L não é regular.

Teorema de bombeamento

1. $L = \{a^i b^i : i \geq 0\}$ não é regular.

Teorema de bombeamento

1. $L = \{a^i b^i : i \geq 0\}$ não é regular.
   - Considere $w = a^n b^n \in L$. Pelo teorema, ela pode ser reescrita como $w = xyz$, tal que $|xy| \leq n$ e $y \neq \epsilon$.
   - Isto é, $y = a^i$ para algum $i > 0$. Mas então $xz = a^{n-i} b^n \notin L$, contradizendo o teorema.

Teorema de bombeamento

1. $L = \{a^i b^i : i \geq 0\}$ não é regular.
   - Considere $w = a^n b^n \in L$. Pelo teorema, ela pode ser reescrita como $w = xyz$, tal que $|xy| \leq n$ e $y \neq \epsilon$.
   - Isto é, $y = a^i$ para algum $i > 0$. Mas então $xz = a^{n-i} b^n \notin L$, contradizendo o teorema.
2. $\{a^n : n$ é primo$\}$ não é regular.

Teorema de bombeamento

1. $L = \{a^i b^i : i \geq 0\}$ não é regular.
   - Considere $w = a^n b^n \in L$. Pelo teorema, ela pode ser reescrita como $w = xyz$, tal que $|xy| \leq n$ e $y \neq \epsilon$.
   - Isto é, $y = a^i$ para algum $i > 0$. Mas então $xz = a^{n-i} b^n \notin L$, contradizendo o teorema.
2. $\{a^n : n$ é primo$\}$ não é regular.
   - Conforme o teorema, teríamos $x = a^p$, $y = a^q$ e $z = a^r$, onde $p, r \geq 0$ e $q > 0$.
   - Ainda pelo teorema, $xy^n z \ in L$ para cada $n \geq 0$, isto é, $p + nq + r$ é primo para cada $n \geq 0$.
   - Isso é impossível, supondo que $n = p + 2q + r + 2$, então $p + nq + r = (q + 1) * (p + 2q + r)$, que é o produto de dois números naturais maiores que 0, daí n não pode ser primo.

Teorema de bombeamento

3. Às vezes, é útil usar as propriedades de fechamento para a prova.

   $L = \{w \in \{a, b\}^* : w \text{ tem o mesmo número de a's e b's}\}$

   não é regular, pois se assim fosse, $L \cap a^*b^*$ também seria regular, por fechamento das linguagens regulares em relação à intersecção. Entretanto essa última linguagem é precisamente $\{a^i b^i : i \geq 0\}$, que acabamos de mostrar que não é regular.

Minimização de estados