Linguagens Formais e Autômatos

Autômatos Finitos Determinísticos (AFDs)

Andrei Rimsa Álvares
andrei@cefetmg.br

Sumário

- Introdução
- Exemplos
- Autômatos Finitos Determinísticos (AFDs)
- Minimização de AFDs
- Propriedades de AFDs

INTRODUÇÃO

Introdução

- O que são **máquinas de estado finito**?
  - São **máquinas abstratas** que capturam as partes essenciais de algumas **máquinas concretas**
  - Essas máquinas são conhecidas como **autômatos finitos**

- Embora existam máquinas mais poderosas, as **máquinas de estado finito** são adequadas tanto do ponto de vista teórico quanto prático
  - Permitem modelar um amplo espectro de máquinas, como máquinas mecânicas, eletrônicas, de software, ...

Introdução

- Existem dois tipos de máquinas de estado finito
  - **Transdutores:** Os transdutores são máquinas com entrada e saída
  - **Reconhecedores:** Os reconhecedores (ou aceitadores) de linguagens são máquinas também com entrada e saída, mas aceita somente duas saídas possíveis: **aceitação** ou **rejeição** da entrada

Introdução

- As linguagens reconhecidas por máquinas de estado finito são denominadas **linguagens regulares**
  - Existem duas notações úteis para a especificação de linguagens regulares: **expressões regulares** e **gramáticas regulares**

- Uma máquina de estado finito possui uma memória limitada, que é organizada exclusivamente em torno do conceito de **estado**

EXEMPLOS

Exemplos

- Para introduzir os principais conceitos e a terminologia associada a autômatos finitos, serão utilizados três exemplos simples

Um quebra-cabeça

Um problema de matemática

Um elevador

Modelagem

- Mas antes de entrar na solução, é preciso fazer uma modelagem do problema, na qual

  a) As informações *relevantes* são identificadas (**abstração**)

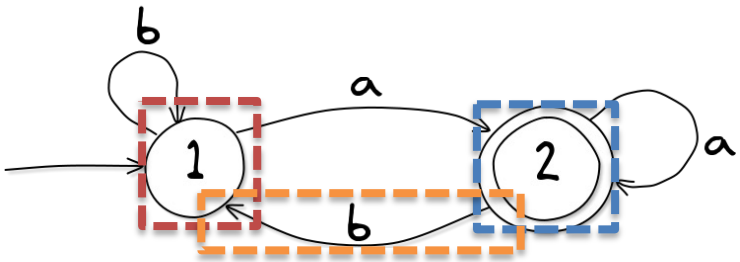
  b) As informações *relevantes* são estruturadas (**representadas**) de forma a facilitar a solução

Menos informações = solução mais fácil/eficiente

Modelagem

- O problema pode ser visto como
  - **Estado:** fotografia da realidade
  - **Transição:** mudança de um estado para outro provocado por uma ação
  - **Solução:** sequência de ações que levam de um estado inicial a um estado final

- Um **diagrama de estados** representa o espaço de estados e transições através de um grafo
  - **Estados:** vértices do grafo
    - **Inicial:** seta que o aponta
    - **Final:** oval dupla
  - **Transição:** aresta de e para e' com rótulo s



Quebra-Cabeça

- Problema: O leão, o coelho e o repolho

Um homem, um leão, um coelho e um repolho devem atravessar um rio usando uma canoa, com a restrição de que o homem deve transportar no máximo um dos três de cada vez de uma margem à outra. Além disso, o leão não pode ficar na mesma margem que o coelho sem a presença do homem, e o coelho não pode ficar com o repolho sem a presença do homem. O problema consiste em determinar se é possível fazer a travessia.

Quebra-Cabeça

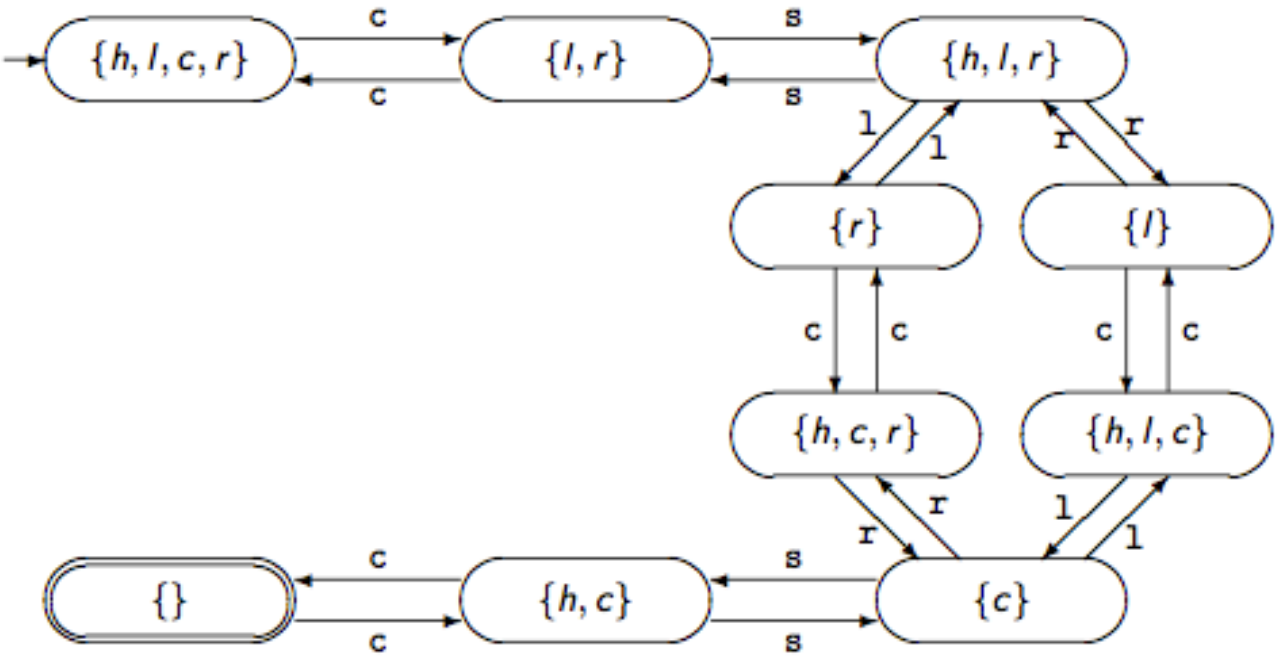
- Modelagem
  - **Abstração:** informações relevantes
    a) Em um dado instante, em que margem do rio estão o homem, o leão, o coelho e o repolho?
    b) A sequência de movimentações entre as margens que propiciou a situação indicada em (a)
  - **Representação:** das informações
    a) $A \subseteq \{h,l,c,r\}$ representa que os elementos em A estão na margem esquerda e os em $\{h,l,c,r\} - A$ na margem direita
    b) $a_0 a_1 ... a_n$ representa uma palavra em que cada a_i pode ser s,l,c,r representa a sequência de movimentos até a solução

Quebra-Cabeça

- Modelagem
  - **Estado:** um subconjunto de $A \subseteq \{h, l, c, r\}$
    - **Inicial:** $\{h, l, c, r\}$
    - **Final:** $\{\}$
  - **Transição:** Uma das ações s, l, c, r
  - **Solução:** Uma palavra do alfabeto $\Sigma = \{s, l, c, r\}$

Quebra-Cabeça

- Diagrama de estados do quebra-cabeça



- Dada $w \in \{s,l,c,r\}^*$
  - w é **reconhecida** se o caminho correspondente vai do estado inicial ao estado final
  - w é **rejeitada** caso contrário

Quebra-Cabeça

- Durante o processamento de w pode se obter uma **configuração instantânea** $[e, y]$**,** onde
  - e: o estado atual após processar um prefixo de w
  - y: o sufixo ainda não processado (Assim $w = xy$)

- Com isso, pode-se definir a relação $\vdash$, onde
  - $[e_1, w] \vdash [e_2, y]$ se existe uma transição de e_1 para e_2 sob a e $w = ay$

- Exemplo de computação

$$[\{l,r\}, sllr] \vdash [\{h,l,r\}, llr] \vdash [\{r\}, lr] \vdash [\{h,l,r\}, r] \vdash [\{l\}, \lambda]$$

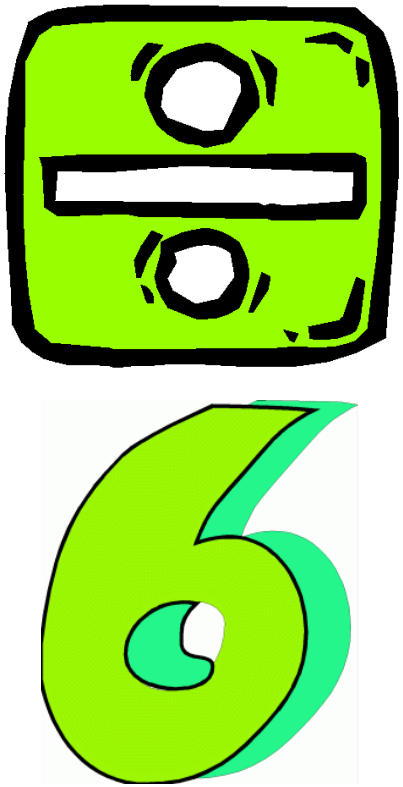
Quebra-Cabeça

- O autômato finito para esse quebra-cabeça possui as seguintes características
  - Para cada transição existe uma transição inversa
  - Há um único estado final
  - Para cada par (estado, símbolo) existe, no máximo, uma transição (**determinismo**)
  - O conjunto de estados é **finito**

O único mecanismo de memória para AFDs é o conjunto de estados

Problema da Matemática

- Problema: binário divisível por 6



Seja o problema de projetar uma máquina que, dada uma sequência de 0s e 1s, determine se o número representado por ela na base 2 é divisível por 6. O que se deseja é um projeto independente de implementação, ou seja, que capture apenas a essência de tal máquina, não importando se ela será mecânica, eletrônica, um programa ou o que quer que seja.

Problema da Matemática

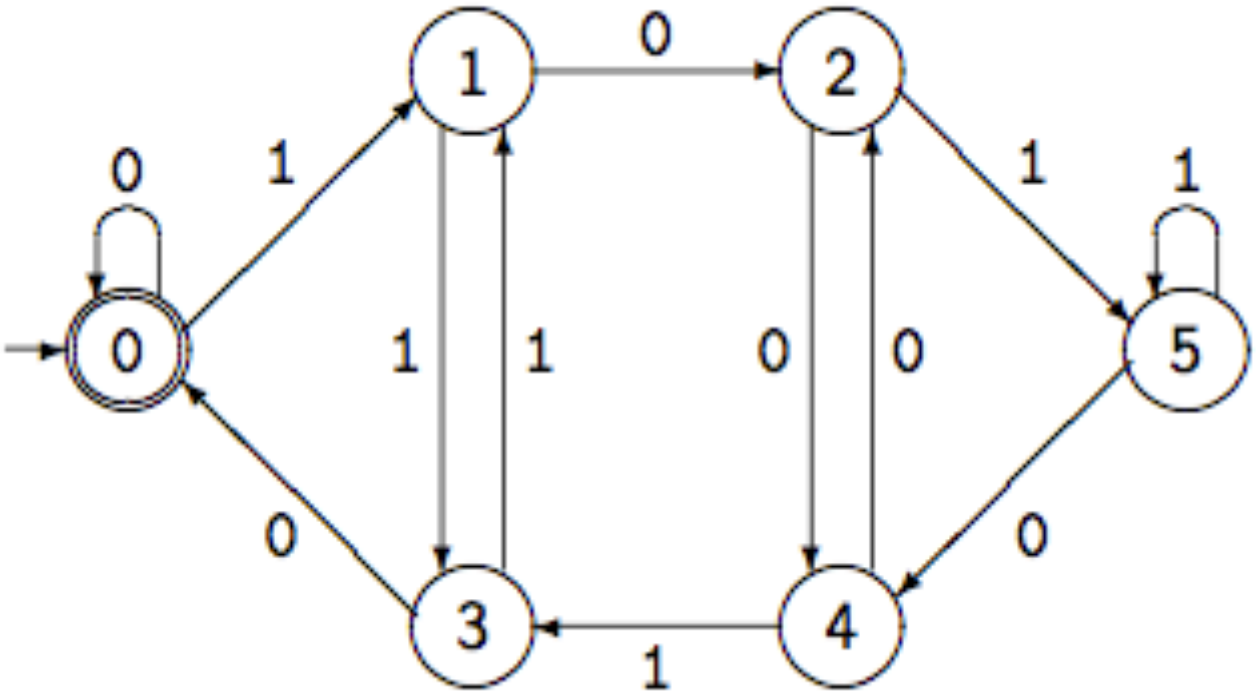
- Modelagem
  - Suponha que $\eta(x)$ é o número representado pela palavra x na base binária
    - A palavra $x0$ é $2\eta(x)$
    - A palavra $x1$ é $2\eta(x)+1$
  - Sabendo que $\eta(x) \mod 6 = r$, então
    - $\eta(x0) \mod 6 = 2r \mod 6$
    - $\eta(x1) \mod 6 = (2r+1) \mod 6$

Problema da Matemática

- Modelagem
  - **Estado:** um para cada resto (0 a 5)
    - **Inicial:** estado 0
    - **Final:** estado 0
  - **Transição:** do estado cujo resto é r emanam 2 transições
    - Sob 0: para o estado correspondente a $2r \bmod 6$
    - Sob 1: para o estado correspondente a $(2r+1) \bmod 6$
  - **Solução:** Um número na base 2

Problema da Matemática

- Diagrama de estados do problema da matemática



Elevador

- Problema: simular o funcionamento de um elevador

Modelar um elevador destinado a servir a um prédio de três andares, satisfazendo:

a) caso não haja chamada, o elevador fica parado onde estiver
b) o elevador dá prioridade ao chamado mais próximo no sentido em que estiver se movimentando
c) um chamado do elevador pode ser *desligado* manualmente. Assim, por exemplo, é possível existir uma chamada para um andar em certo instante e, logo em seguida, não existir mais, sem que o elevador se mova

Elevador

- Modelagem
  - Informações relevantes para o problema
    - O andar em que o elevador se encontra
    - O sentido em que o elevador está se movendo
    - Os andares em que o elevador está sendo solicitado
    - A ação do elevador: subir, descer ou ficar imóvel
  - Outras informações sobre o funcionamento do elevador
    - Se o elevador está sendo chamado para o andar em que já está, ele fica parado
    - O elevador dá preferência para o sentido em que está se movendo

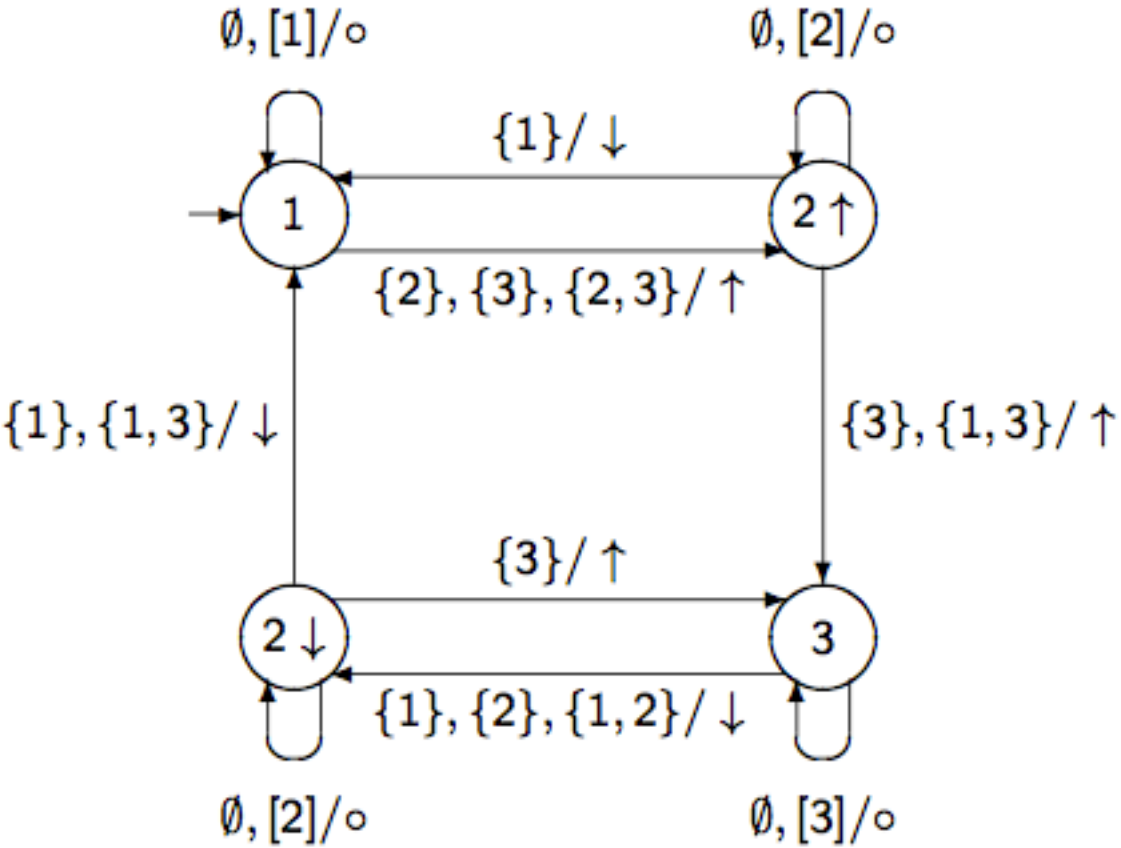
Elevador

- Modelagem
  - **Estado:** 1, 2↑, 2↓ , 3
    - Captura duas informações importantes: o andar atual do elevador e o sentido do movimento
    - Não tem o conceito de estado inicial e final
  - **Transição:** Andares que estão solicitando o elevador
  - **Solução:** Saída com todos os movimentos realizados pelo elevador ↑, ↓ e ∘

Chamada de Máquina de Mealey

Elevador

- Diagrama de estados do problema do elevador



Elevador

- O problema do elevador tem uma **configuração instantânea** $[e, x, y]$, em que
  - e: é o estado atual
  - x: é um sufixo a ser processado
  - y: é uma sequência de saídas emitidas até o momento

- Exemplo de computação

$$[1,\{2,3\}\{1,3\}\{1\},\lambda] \vdash [2\uparrow,\{1,3\}\{1\},\uparrow] \vdash [3,\{1\},\uparrow\uparrow] \vdash [2\downarrow,\lambda,\uparrow\uparrow\downarrow]$$

AUTÔMATOS FINITOS DETERMINÍSTICOS (AFDs)

Autômatos Finitos Determinísticos

- Um autômato finito determinístico (AFD) é uma quíntupla $(E, \Sigma, \delta, i, F)$ em que

  - E é um conjunto finito não vazio de estados
  - Σ é um alfabeto
  - $\delta : E \times \Sigma \rightarrow E$ é uma função de transição (função total)
  - $i \in E$ é o estado inicial
  - $F \subseteq E$ é o conjunto de estados finais

Propriedades de AFDs

- **Determinismo:** Cada par (estado, símbolo) leva a um único estado; a partir do estado inicial, só é possível atingir um único estado para uma palavra de entrada
- **Função de transição total:** para toda palavra de entrada, atinge-se um estado consumindo-se toda a palavra
- **Um único estado inicial:** ter mais de um estado inicial não aumenta o poder computacional
- **Vários estados finais:** com apenas um estado final, o poder computacional é menor
- **Conjunto finito de estados:** com conjunto infinito de estados, o poder computacional é maior

AFD de Exemplo

- AFD para o problema do quebra-cabeça

$$M = (E, \{\mathtt{s}, \mathtt{l}, \mathtt{c}, \mathtt{r}\}, \delta, \{h, l, c, r\}, \{\{\}\})$$

$$E = \{\{h, l, c, r\}, \{l, r\}, \{h, l, r\}, \{l\}, \{r\}, \{h, l, c\}, \{h, c, r\}, \{c\}, \{h, c\}, \{\}, t\}$$

δ	s	l	c	r
$\{h, l, c, r\}$	t	t	$\{l, r\}$	t
$\{l, r\}$	$\{h, l, r\}$	t	$\{h, l, c, r\}$	t
$\{h, l, r\}$	$\{l, r\}$	$\{r\}$	t	$\{l\}$
$\{l\}$	t	t	$\{h, l, c\}$	$\{h, l, r\}$
$\{r\}$	t	$\{h, l, r\}$	$\{h, c, r\}$	t
$\{h, l, c\}$	t	$\{c\}$	$\{l\}$	t
$\{h, c, r\}$	t	t	$\{r\}$	$\{c\}$
$\{c\}$	$\{h, c\}$	$\{h, l, c\}$	t	$\{h, c, r\}$
$\{h, c\}$	$\{c\}$	t	$\{\}$	t
$\{\}$	t	t	$\{h, c\}$	t
t	t	t	t	t

AFD de Exemplo

- AFD para o problema da matemática

$$M = (\{0,1,2,3,4,5\}, \{0,1\}, \delta, 0, \{0\})$$

δ	0	1
0	0	1
1	2	3
2	4	5
3	0	1
4	2	3
5	4	5

AFD Simplificados

- Se não existe transição de e sob a no diagrama de estados, então existe um estado e' (**estado de erro**) tal que
  - Existe uma transição de e para e' sob a
  - e' não é um estado final
  - Existe uma transição de e' para e' sob cada símbolo do alfabeto

Um diagrama sem estado de erro
é chamado de **diagrama simplificado**

Função de Transição Estendida

- A **função de transição estendida** $\hat{\delta} : E \times \Sigma^* \rightarrow E$ para um AFD $M = (E, \Sigma, \delta, i, F)$ definida recursivamente como
  a) $\hat{\delta}(e, \lambda) = e$
  b) $\hat{\delta}(e, ay) = \hat{\delta}(\delta(e, a), y), \forall a \in \Sigma, \forall y \in \Sigma^*$

- Exemplo de transição para o problema da matemática

$$
\begin{aligned}
\hat{\delta}(0, 1010) &= \hat{\delta}(\delta(0,1), 010) && \text{por } b, \text{ Definição 2} \\
&= \hat{\delta}(1, 010) && \text{pois } \delta(0,1) = 1 \\
&= \hat{\delta}(\delta(1,0), 10) && \text{por } b, \text{ Definição 2} \\
&= \hat{\delta}(2, 10) && \text{pois } \delta(1,0) = 2 \\
&= \hat{\delta}(\delta(2,1), 0) && \text{por } b, \text{ Definição 2} \\
&= \hat{\delta}(5, 0) && \text{pois } \delta(2,1) = 5 \\
&= \hat{\delta}(\delta(5,0), \lambda) && \text{por } b, \text{ Definição 2} \\
&= \hat{\delta}(4, \lambda) && \text{pois } \delta(5,0) = 4 \\
&= 4 && \text{por } a, \text{ Definição 2.}
\end{aligned}
$$

A função de transição estendida pode ser usada para definir a linguagem aceita ou reconhecida por um AFD

Linguagens Reconhecidas

- A **linguagem aceita** (**reconhecida**) por um AFD $M = (E, \Sigma, \delta, i, F)$ é o conjunto

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

- Uma determinada palavra $w = \Sigma^*$ é dita ser reconhecida por M se a função de transição estendida do estado i sob a palavra w chega a um estado final F:

$$\hat{\delta}(i, w) \in F$$

Algoritmo para AFDs

- Existe um algoritmo simples para simular um AFD

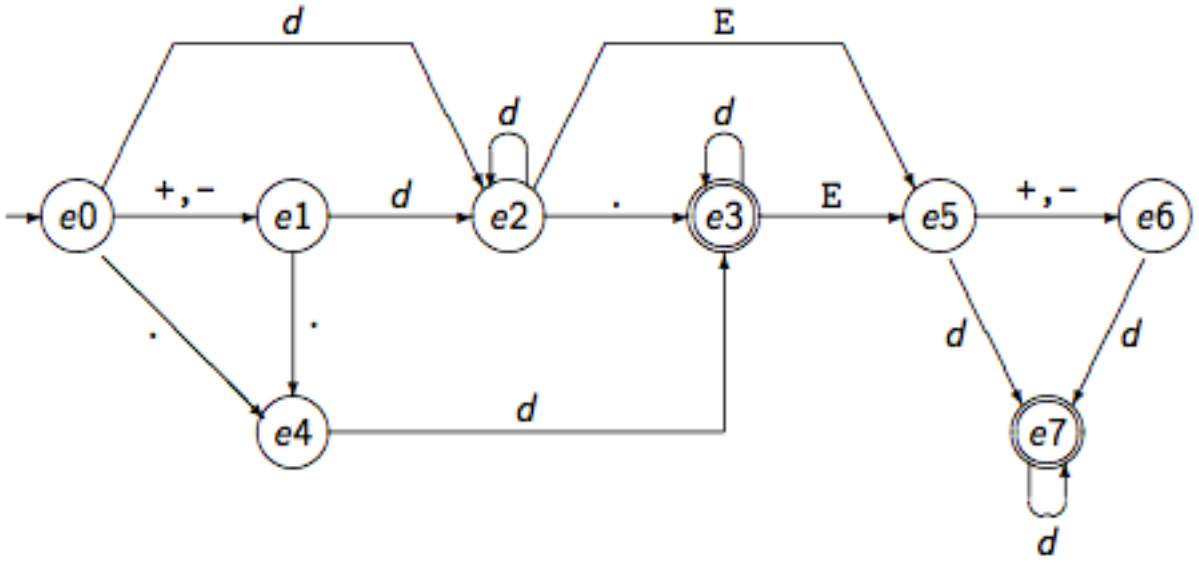
Entrada: (1) o AFD, dado por i, F e D, e
(2) a palavra de entrada, dada por *prox*.
Saída: *sim* ou *não*.
$e \leftarrow i$; $s \leftarrow prox()$;
enquanto $s \neq fs$ **faça**
    $e \leftarrow D[e, s]$; $s \leftarrow prox()$;
fimenquanto;
se $e \in F$ **então**
    retorne *sim*
senão
    retorne *não*
fimse

Qual a complexidade desse algoritmo?

Aplicação

- Uma aplicação clássica de AFDs é para a análise léxica em compiladores

- Um exemplo de autômato para literais numéricos



$$M = (E, \Sigma, \delta, e0, \{e3, e7\})$$

$$E = \{e0, e1, e2, e3, e4, e5, e6, e7, ee\}$$

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, \mathrm{E}\}$$

δ	d	.	+	−	E
e0	e2	e4	e1	e1	ee
e1	e2	e4	ee	ee	ee
e2	e2	e3	ee	ee	e5
e3	e3	ee	ee	ee	e5
e4	e3	ee	ee	ee	ee
e5	e7	ee	e6	e6	ee
e6	e7	ee	ee	ee	ee
e7	e7	ee	ee	ee	ee
ee	ee	ee	ee	ee	ee

AFDs Equivalentes

- Dois AFDs M_1 e M_2 são ditos **equivalentes** se e somente se $L(M_1) = L(M_2)$

- Logo faz sentido perguntar
  - Existe um AFD mais eficiente (em termos de?) para reconhecer uma linguagem?
  - É possível criar um AFD, partir de AFDs M_1 e M_2, que reconheça
    - $L(M_1) \cup L(M_2)$
    - $L(M_1) \cap L(M_2)$
    - $L(M_1) - L(M_2)$
    - $L(M_1)L(M_2)$
    - $L(M_1)^*$

Para ambas as perguntas a resposta é SIM!

Exercícios

- Desenvolva AFDs que reconheçam as seguintes linguagens sob $\Sigma = \{0,1\}$
  a) Toda palavra possui um número ímpar de 1s e um número ímpar de 0s

  b) Toda palavra possui pelo menos uma ocorrência de 00

  c) Toda palavra possui exatamente duas ocorrências de 00

  d) Toda palavra é tal que toda ocorrência de 1 é imediatamente seguida por um número ímpar de 0s

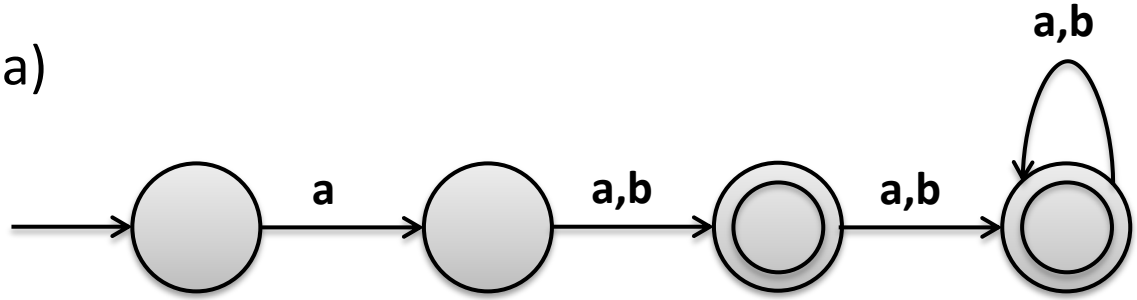
MINIMIZAÇÃO DE AFDS

Linguagens Formais e Autômatos

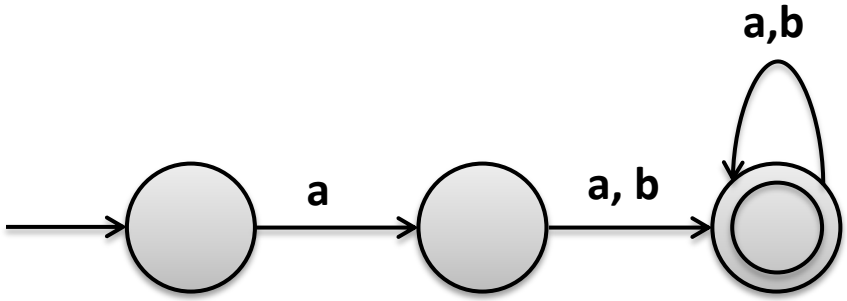
Minimização

- Considere a palavra $w = abababaaba$. Em termos de processamento, qual dos autômatos abaixo é mais eficiente?

a)



b)



Por que minimizar?

Os dois possuem a mesma eficiência, já que o processamento depende do tamanho da palavra e não do tamanho do autômato

AFD Mínimo

- Um AFD M é dito ser **mínimo** para a linguagem $L(M)$ se nenhum AFD para $L(M)$ contém menor número de estados que M

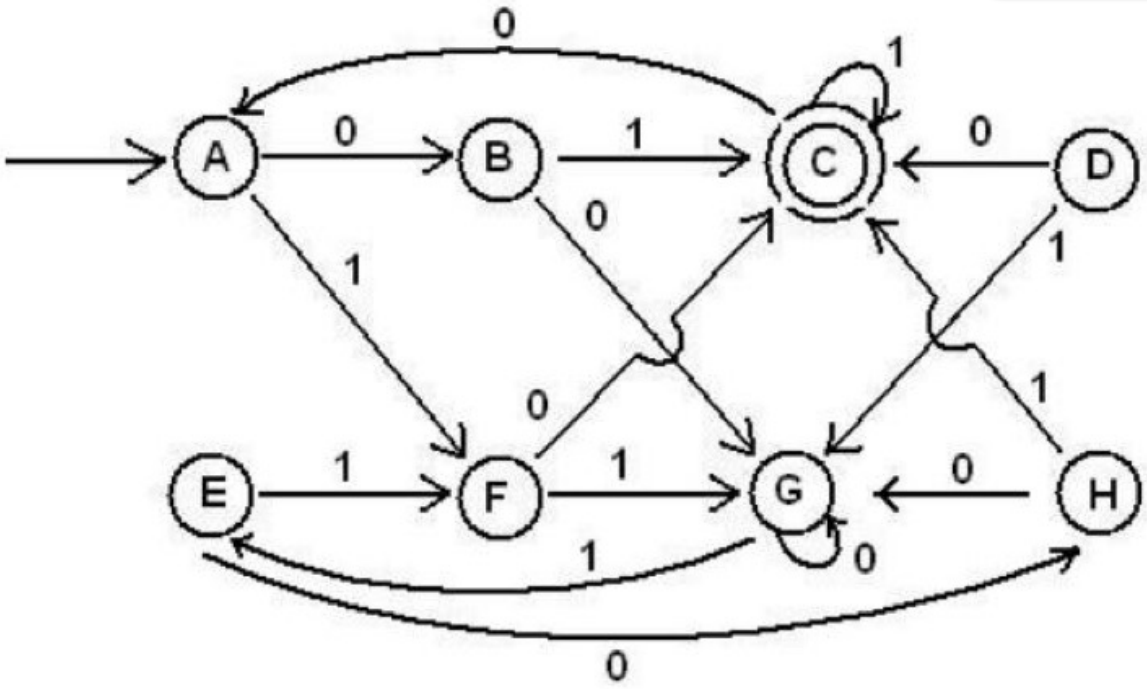
- Para obter um AFD mínimo, deve-se
  a) Eliminar **estados não alcançáveis** a partir do estado inicial
  b) Substituir cada grupo de **estados equivalentes** por um único estado

Estados Não Alcançáveis

- Para eliminar estados não alcançáveis, pode-se utilizar qualquer algoritmo de busca em grafos, como por exemplo, **busca em profundidade**

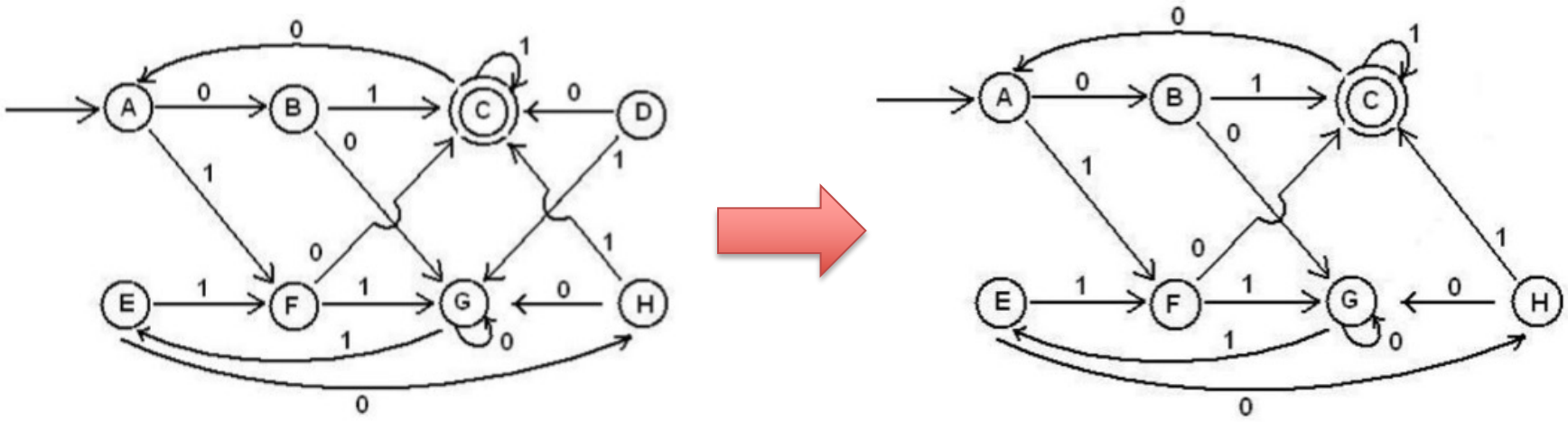
- Exemplo

Quais estados são inalcançáveis?



Estados Não Alcançáveis

- Autômato sem estados não alcançáveis



Estados Equivalentes

- Seja um AFD $M = (E, \Sigma, \delta, i, F)$. Então $e, e' \in E$ são ditos **equivalentes**, $e \approx e'$ se e somente se

$$\forall y \in \Sigma^*, \hat{\delta}(e, y) \in F \leftrightarrow \hat{\delta}(e', y) \in F$$

A relação ≈ é de equivalência, já que é reflexiva, simétrica e transitiva

- O objetivo é determinar grupos de estados equivalentes e substituir cada grupo por um único estado

Por que reduzir estados equivalentes a um só?

Estados Equivalentes

- Seja um AFD $M = (E, \Sigma, \delta, i, F)$

  - Se $e \approx e'$: um sufixo y é reconhecido passando-se por e se e somente se ele é reconhecido passando por e'; logo, e e e' podem se tornar um só

  - Se $e \not\approx e'$ ($\exists y \in \Sigma^*, \hat{\delta}(e, y) \in F$ e $\hat{\delta}(e', y) \notin F$ ou vice-versa): se um sufixo y levar a um estado de F, a palavra é aceita, caso contrário, não é; logo, e e e' não podem se tornar um só

$[e] = \{e_1, e_2, \ldots, e_n\}$ é a classe de equivalência de e na partição induzida por $\approx$

AFD Reduzido

- Seja um AFD $M = (E, \Sigma, \delta, i, F)$, um AFD **reduzido** correspondente a M é o AFD $M' = (E', \Sigma, \delta', i', F')$

  - $E' = \{[e] \mid e \in E\}$
  - $\delta'([e], a) = [\delta(e, a)] \quad \forall e \in E, \forall a \in \Sigma$
  - $i' = [i]$
  - $F' = \{[e] \mid e \in F\}$

Como mostrar que o AFD M' é equivalente ao AFD M?

AFD Reduzido

- Para mostrar que um AFD M' reduzido é equivalente ao AFD M, basta mostrar que

a) O processamento é equivalente

$$\hat{\delta}'([e], w) = [\hat{\delta}(e, w)] \quad \forall w \in \Sigma^* \text{ por indução sobre } |w|$$

b) Reconhecem a mesma linguagem ($L(M') = L(M)$), $\forall w \in \Sigma^*$

$$\hat{\delta}'(i', w) \in F \leftrightarrow \hat{\delta}(i, w) \in F \quad \forall w \in \Sigma^*$$

O problema do algoritmo de minimização é encontrar as classes de equivalência induzidas pela relação $\approx$

Classes de Equivalência Induzidas por ≈

- A relação $\approx$ pode ser definida como uma série de refinamentos ($\approx_0, \approx_1, \approx_2 \ldots$), em que $\approx_{n+1}$ refina $\approx_n$

- A definição de $\approx_i$ ($i \geq 0$) para um AFD $M = (E, \Sigma, \delta, i, F)$ é dada da seguinte forma

  a) $e \approx_0 e'$ se e somente se $e, e' \in F$ ou $e, e' \in E - F$

  b) $e \approx_{n+1} e'$ ($n \geq 0$) se e somente se $e \approx e'$ e $\delta(e, a) \approx_n \delta(e', a) \quad \forall a \in \Sigma$

Classes de Equivalência Induzidas por ≈

- A definição anterior pode ser reformulada usando a notação $[e]_n$, onde ela é usada para denotar a classe de equivalência que pertence o estado *e* na partição induzida por $\approx_n$

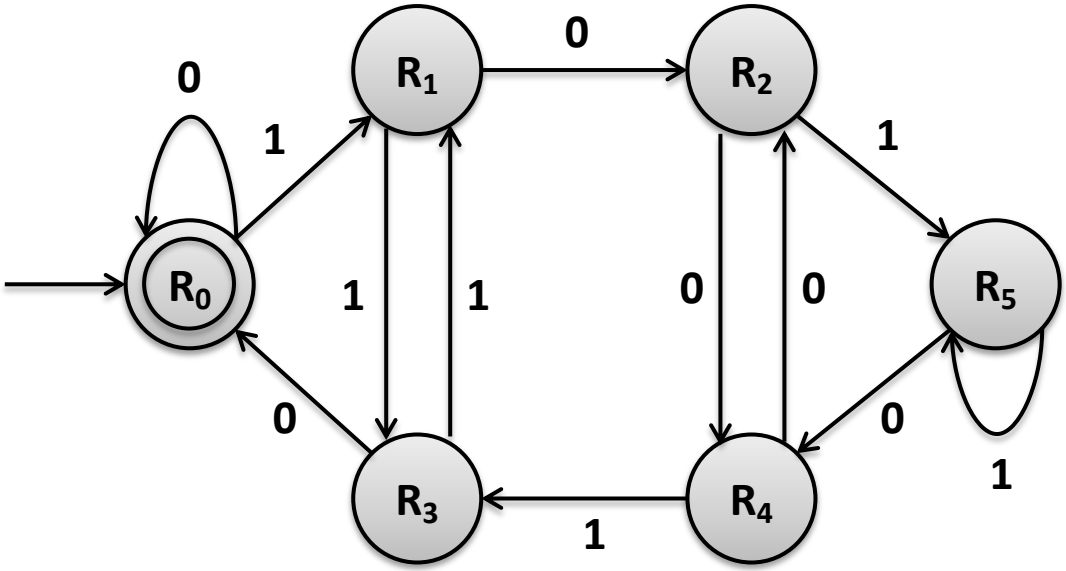
- A definição de $[e]_i$ ($i \geq 0$) para um AFD $M = (E, \Sigma, \delta, i, F)$ é dada da seguinte forma

a) $$[e]_0 = \begin{cases} F & \text{se } e \in F \\ E - F & \text{se } e \in E - F \end{cases}$$

b) $$[e]_{n+1} = \{e' \in [e]_n \mid [\delta(e', a)]_n = [\delta(e, a)]_n, \forall a \in \Sigma\}$$
$$(n \geq 0)$$

Exemplo

- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)



O objetivo é particionar o conjunto de estados em conjuntos de equivalência

Exemplo

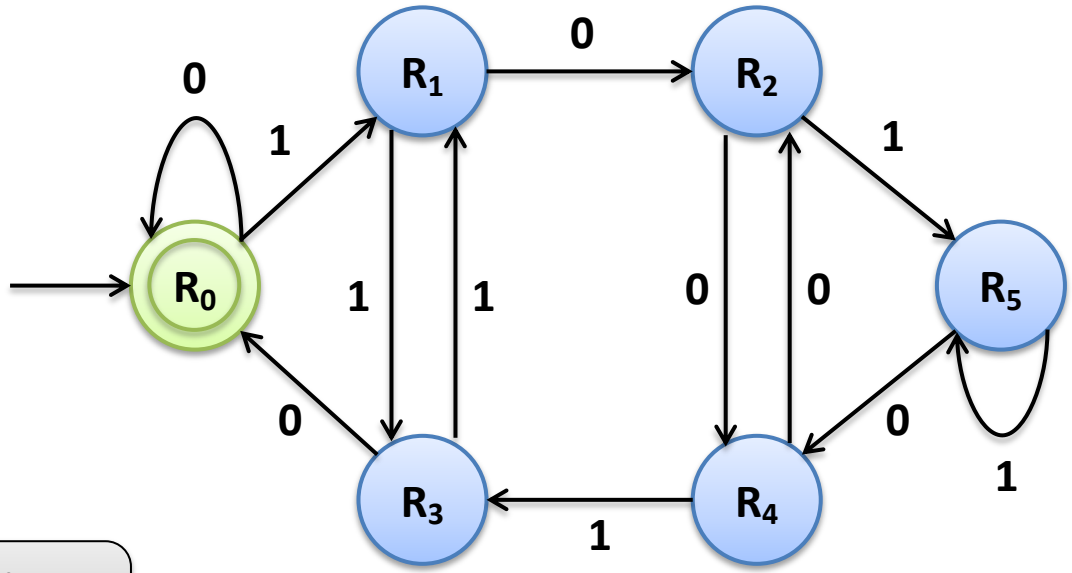
- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

Grupos

$G_1 = \{R_0\}$

$G_2 = \{R_1, R_2, R_3, R_4, R_5\}$

Partição dos estados finais e não finais



As transições de cada estado levam a qual grupo?

Exemplo

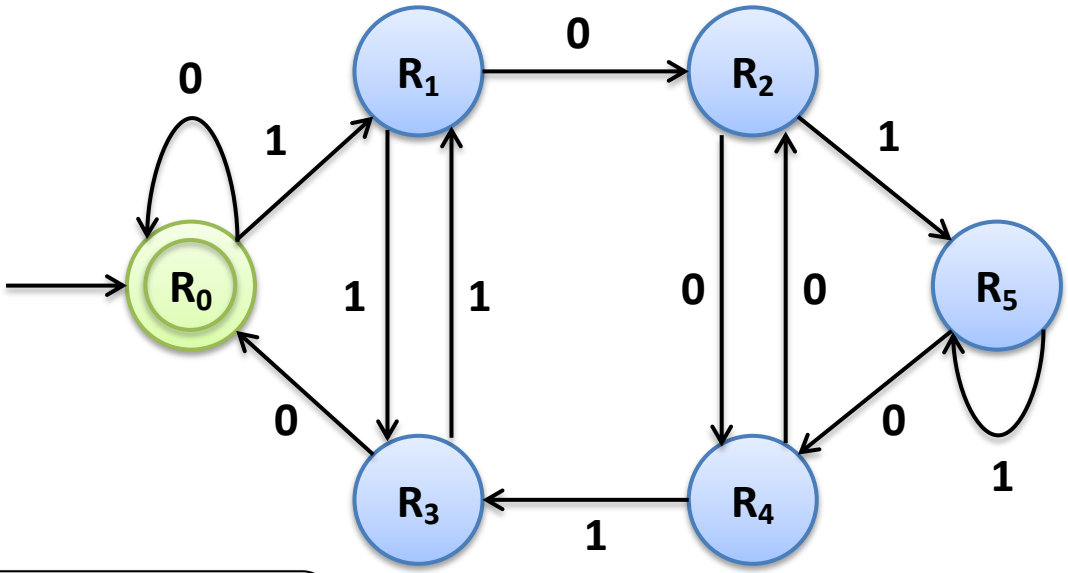
- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

Grupos

$G_1 = \{R_0\}$

$G_2 = \{R_1, R_2, R_3, R_4, R_5\}$

	0	1
R_0	G_1	G_2
R_1	G_2	G_2
R_2	G_2	G_2
R_3	G_1	G_2
R_4	G_2	G_2
R_5	G_2	G_2



Quais os grupos equivalentes?

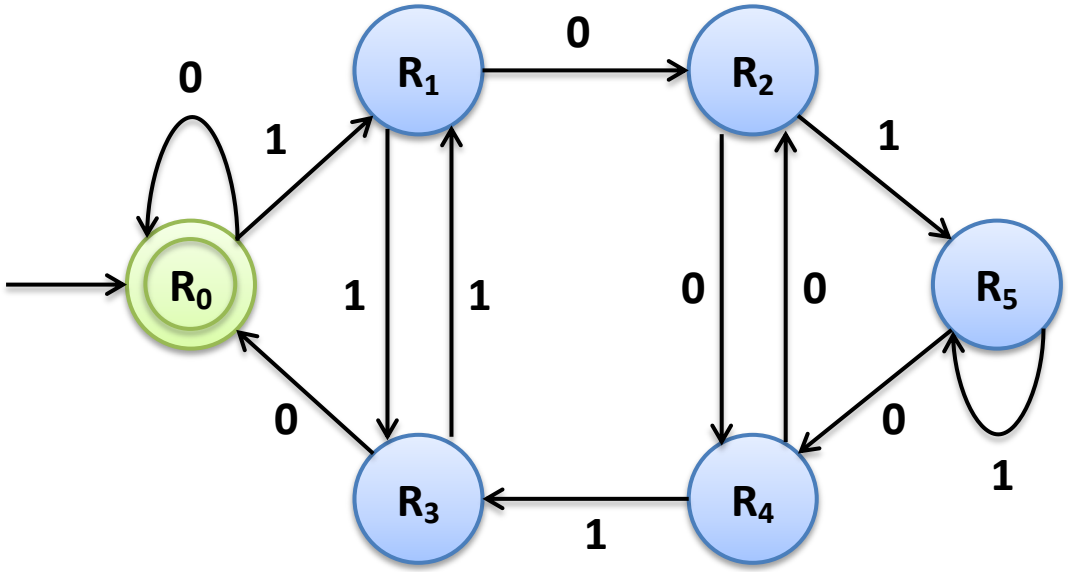
Exemplo

- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

Grupos

$G_1 = \{R_0\}$

$G_2 = \{R_1, R_2, R_3, R_4, R_5\}$



		0	1
G_1	R_0	G_1	G_2
G_2	R_1	G_2	G_2
G_2	R_2	G_2	G_2
G_3	R_3	G_1	G_2
G_2	R_4	G_2	G_2
G_2	R_5	G_2	G_2

Exemplo

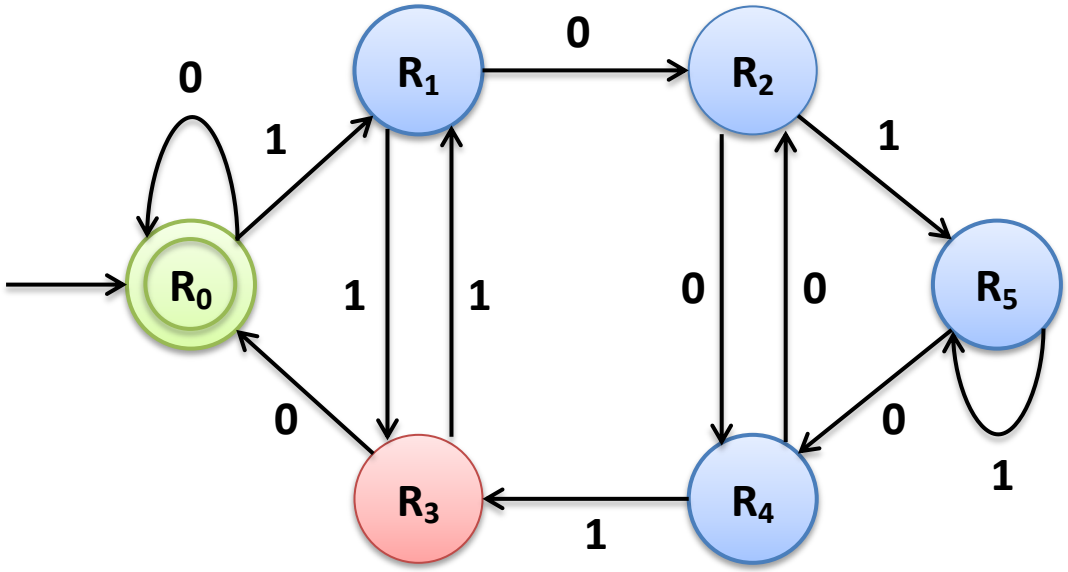
- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

Grupos

$G_1 = \{R_0\}$

$G_2 = \{R_1, R_2, R_4, R_5\}$

$G_3 = \{R_3\}$



Qual o próximo particionamento?

Exemplo

- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

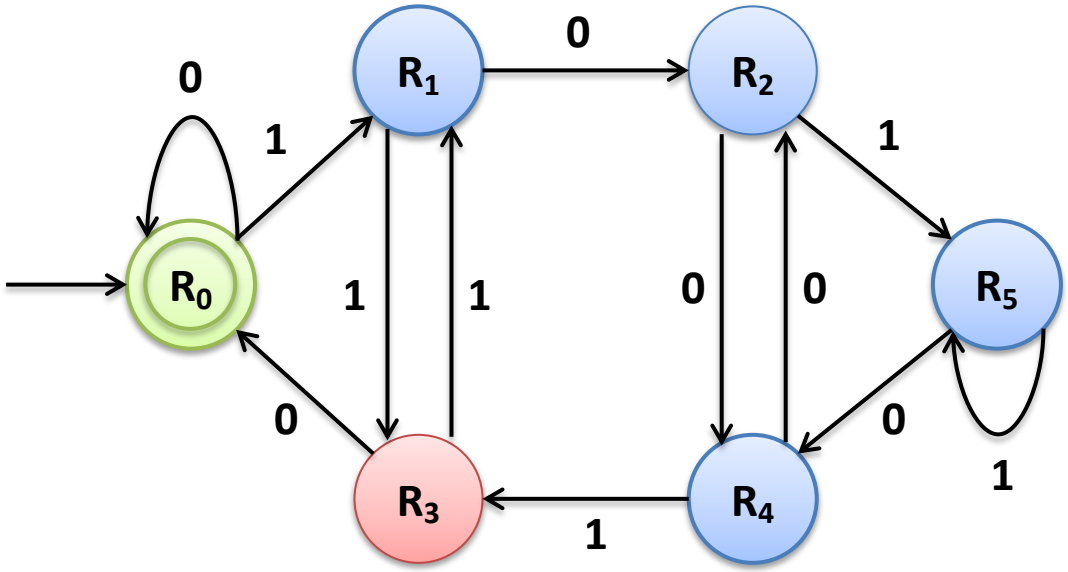
Grupos

$G_1 = \{R_0\}$

$G_2 = \{R_1, R_2, R_4, R_5\}$

$G_3 = \{R_3\}$

		0	1
G_1	R_0	G_1	G_2
G_2	R_1	G_2	G_3
G_4	R_2	G_2	G_2
G_3	R_3	G_1	G_2
G_2	R_4	G_2	G_3
G_4	R_5	G_2	G_2



Exemplo

- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

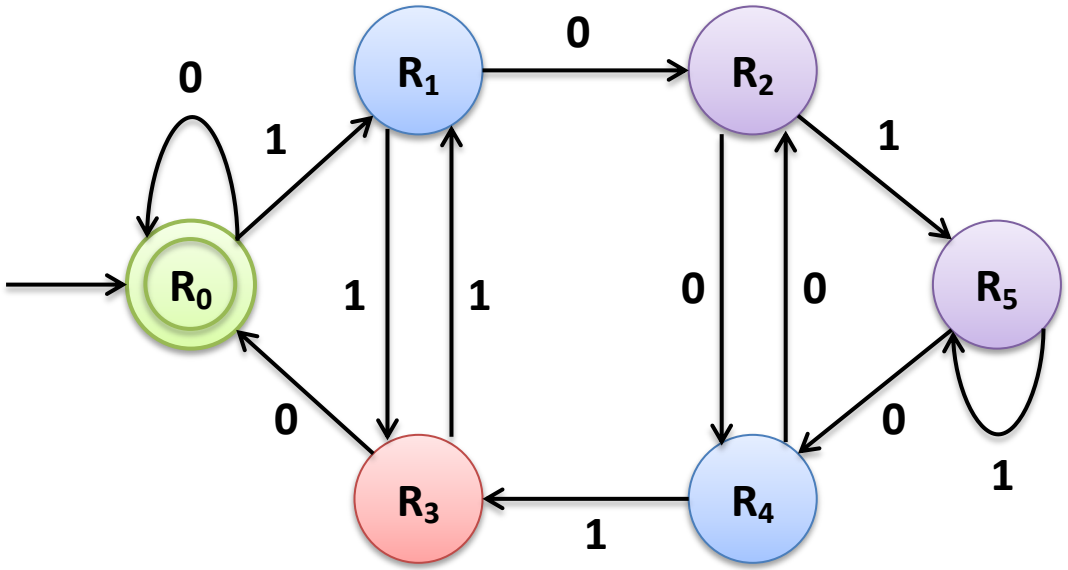
Grupos

$G_1 = \{R_0\}$

$G_2 = \{R_1, R_4\}$

$G_3 = \{R_3\}$

$G_4 = \{R_2, R_5\}$



Tem mais partições?

Exemplo

- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

Grupos

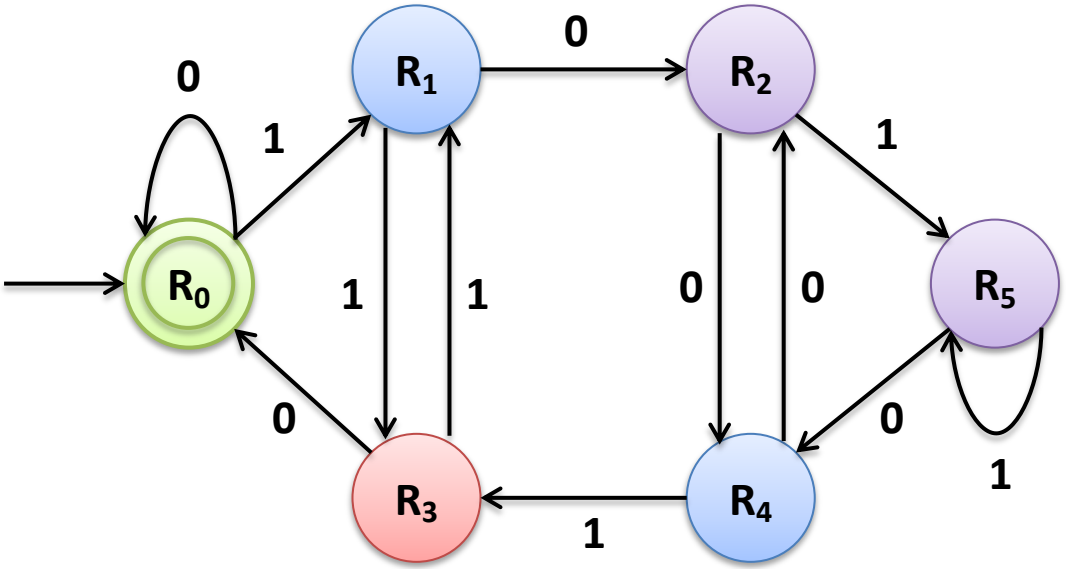
$G_1 = \{R_0\}$

$G_2 = \{R_1, R_4\}$

$G_3 = \{R_3\}$

$G_4 = \{R_2, R_5\}$

		0	1
G_1	R_0	G_1	G_2
G_2	R_1	G_2	G_3
G_4	R_2	G_2	G_2
G_3	R_3	G_1	G_2
G_2	R_4	G_2	G_3
G_4	R_5	G_2	G_2



Como gerar o AFD reduzido?

Exemplo

- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

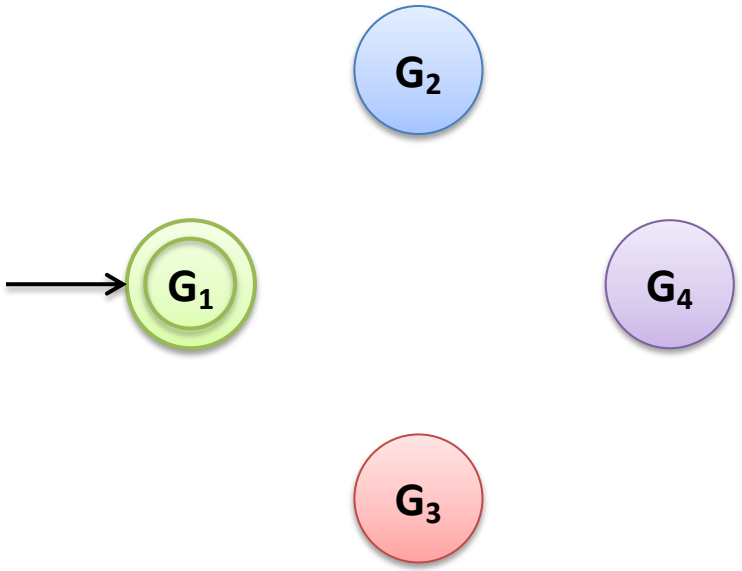
Grupos

$G_1 = \{R_0\}$

$G_2 = \{R_1, R_4\}$

$G_3 = \{R_3\}$

$G_4 = \{R_2, R_5\}$



Criar um vértice para cada grupo

Exemplo

- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)

Grupos

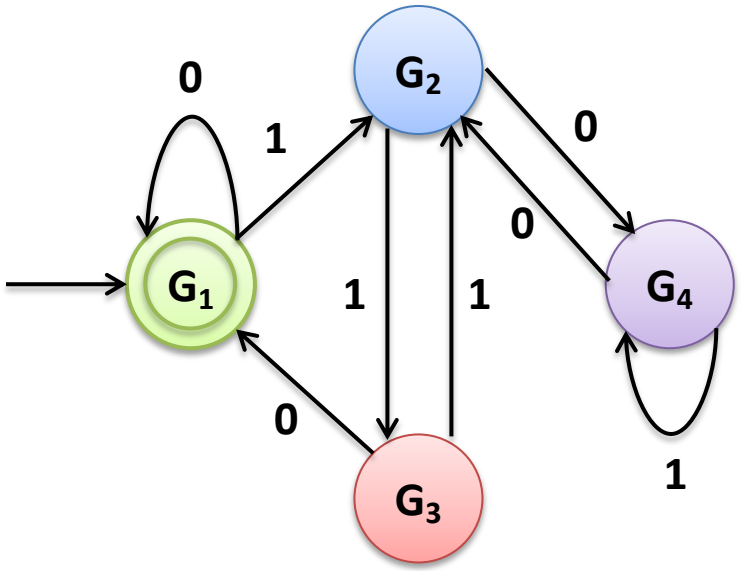
$G_1 = \{R_0\}$

$G_2 = \{R_1, R_4\}$

$G_3 = \{R_3\}$

$G_4 = \{R_2, R_5\}$

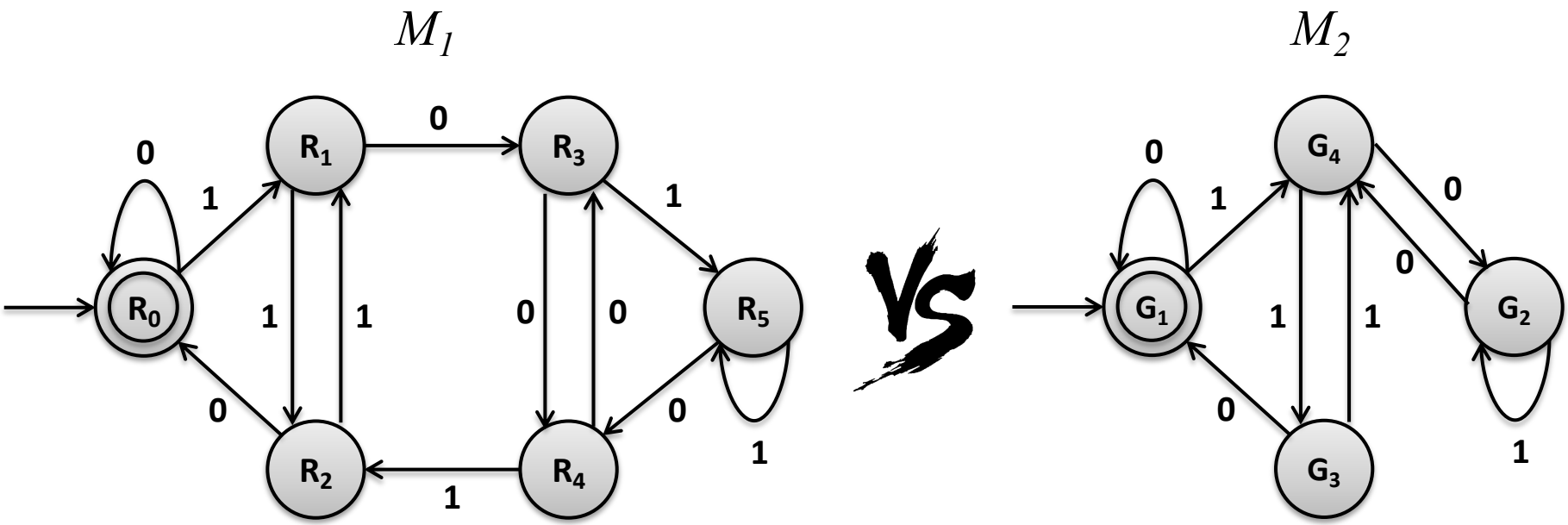
		0	1
G_1	R_0	G_1	G_2
G_2	R_1	G_4	G_3
G_4	R_2	G_2	G_4
G_3	R_3	G_1	G_2
G_2	R_4	G_4	G_3
G_4	R_5	G_2	G_4



Colocar as arestas baseado na tabela

Exemplo

- Para exemplificar o algoritmo de minimização, será utilizado o exemplo do problema da matemática (divisão por 6)



$L(M_1) = L(M_2)$

PROPRIEDADES DE AFDS

Propriedades de AFDs

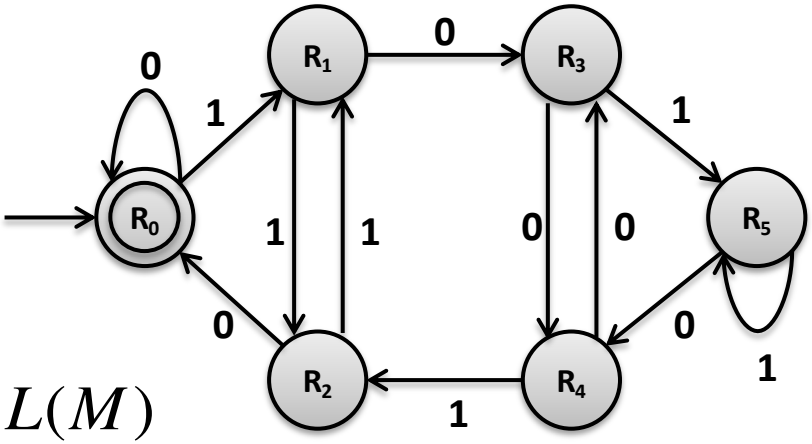
- Se existe um AFD M_1 para uma linguagem e um AFD M_2 para outra linguagem, então é possível criar AFDs para as seguintes linguagens
  - **Complemento:** $\overline{L(M_1)}$
  - **União:** $L(M_1) \cup L(M_2)$
  - **Interseção:** $L(M_1) \cap L(M_2)$

Complemento

- Seja o AFD $M = (E, \Sigma, \delta, i, F)$ para linguagem $L(M)$, então o AFD para $\overline{L(M)}$ é dado por $(E, \Sigma, \delta, i, (E - F))$

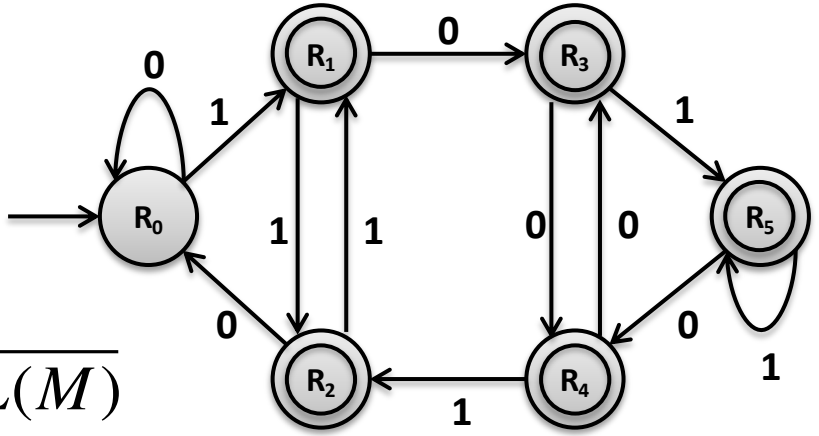
- Exemplo

Inverter os estados finais



$L(M)$

{ números na base 2 divisíveis por 6 }



$\overline{L(M)}$

{ números na base 2 **não** divisíveis por 6 }

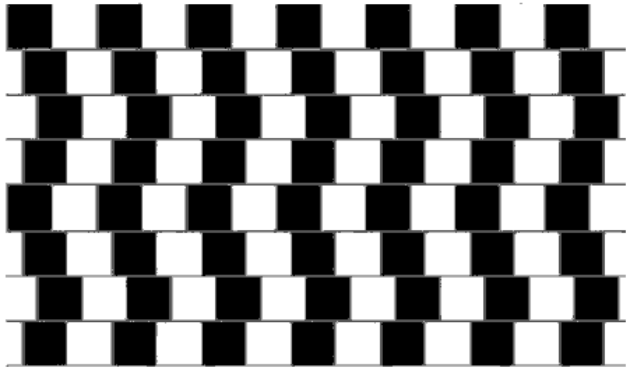
Cuidado: o autômato deve estar completo

União e Interseção

- A união ou a interseção de dois AFDs $M_1 = (E_1, \Sigma, \delta_1, i_1, F_1)$ e $M_2 = (E_2, \Sigma, \delta_2, i_2, F_2)$ é realizada através do **produto** de M_1 por M_2

$$M_1 \times M_2$$

- O produto é obtido simulando a execução em paralelo de dois AFDs



Produto de AFDs

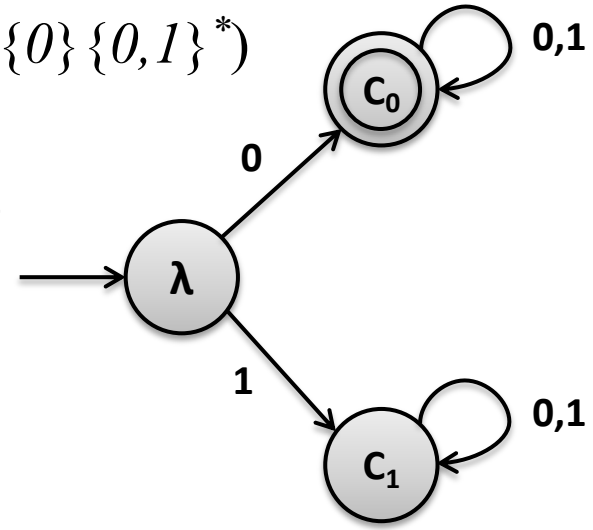
- Sejam os AFDs $M_1 = (E_1, \Sigma, \delta_1, i_1, F_1)$ e $M_2 = (E_2, \Sigma, \delta_2, i_2, F_2)$ o AFD $M_3 = (E_3, \Sigma, \delta_3, i_3, F_3)$ pode ser construído simulando a execução dos AFDs M_1 e M_2 em paralelo, onde
  - $E_3 = E_1 \times E_2$
  - $\delta_3([e_1, e_2], a) = [\delta_1(e_1, a), \delta_2(e_2, a)]$
  - $i_3 = [i_1, i_2]$
  - $F_3 = \begin{cases} F_1 \times F_2 & \text{(Interseção)} \\ (F_1 \times E_2) \cup (E_1 \times F_2) & \text{(União)} \end{cases}$
- Sejam dois estados $e_1 \in E_1$ e $e_2 \in E_2$ do AFD M_3, então

$$\hat{\delta}_3([e_1, e_2], w) = [\hat{\delta}_1(e_1, w), \hat{\delta}_2(e_2, w)], \forall w \in \Sigma^*$$

Produto de AFDs

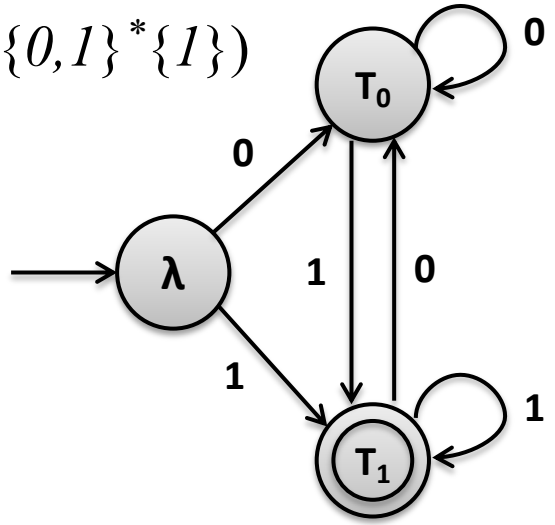
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1

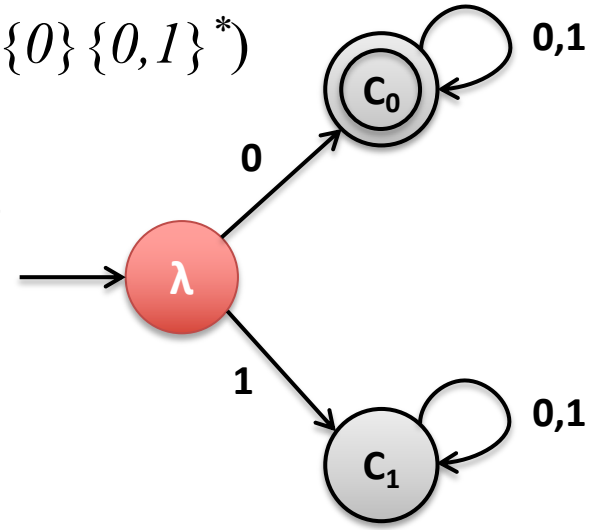


Como simular os dois em paralelo?

Produto de AFDs

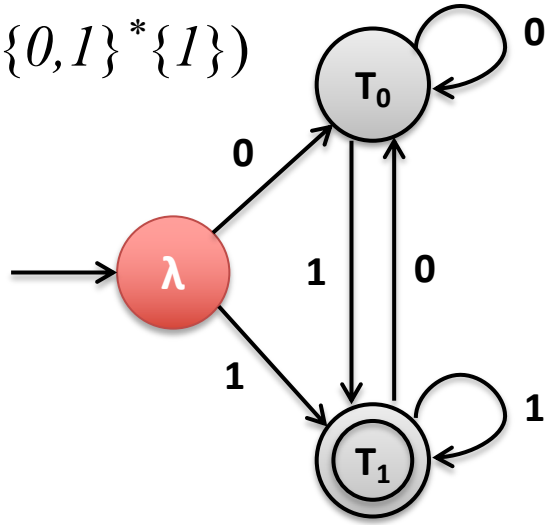
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]		

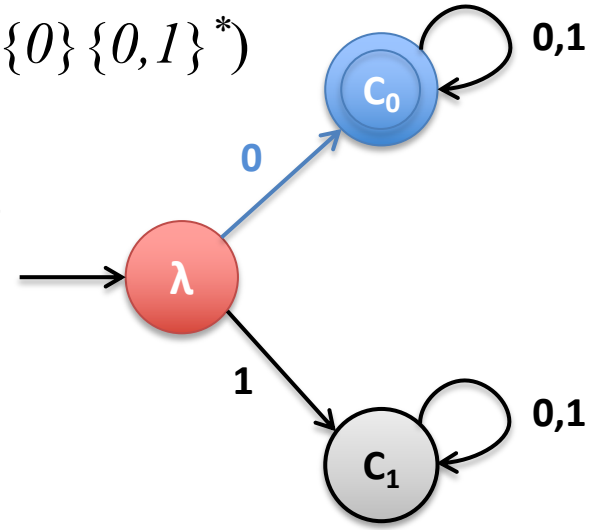
M_3

Criar um estado inicial no AFD M_3 com a junção dos estados iniciais dos AFDs M_1 e M_2

Produto de AFDs

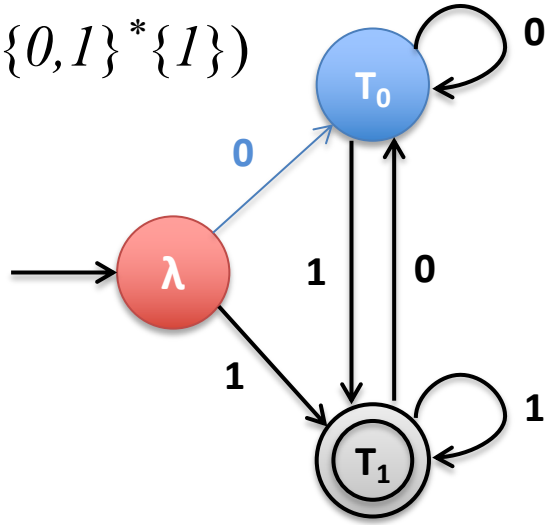
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



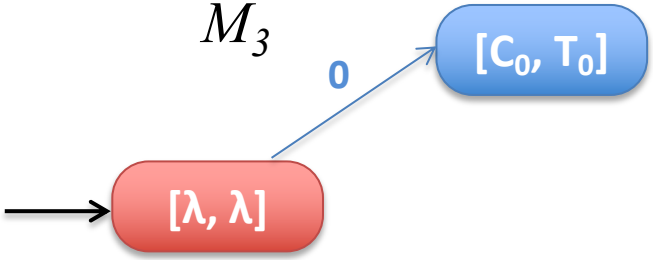
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	**[C_0,T_0]**	
[C_0,T_0]		

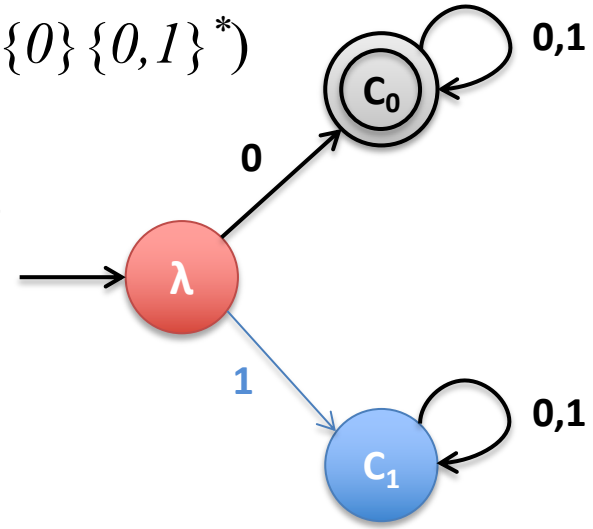
M_3



Produto de AFDs

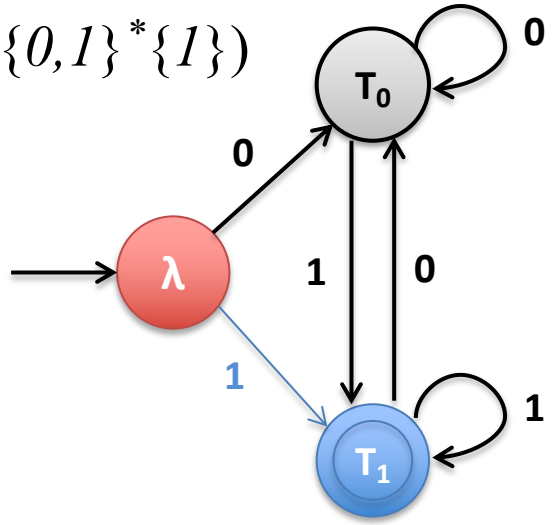
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



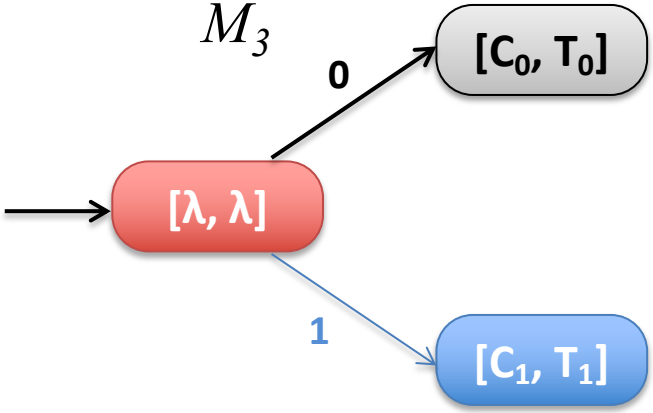
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	$[C_0,T_0]$	**$[C_1,T_1]$**
$[C_0,T_0]$		
$[C_1,T_1]$		

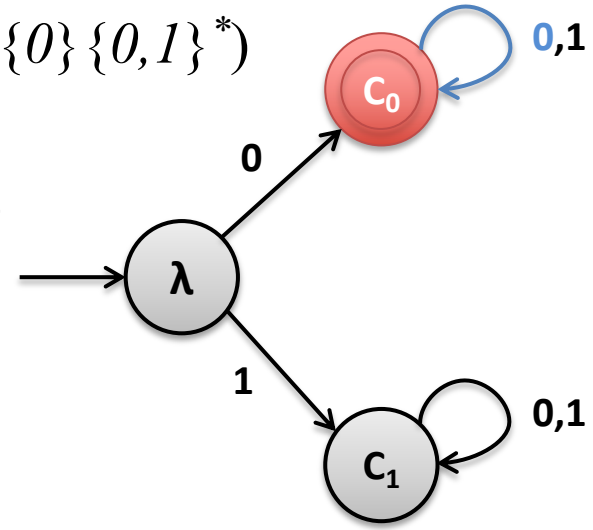
M_3



Produto de AFDs

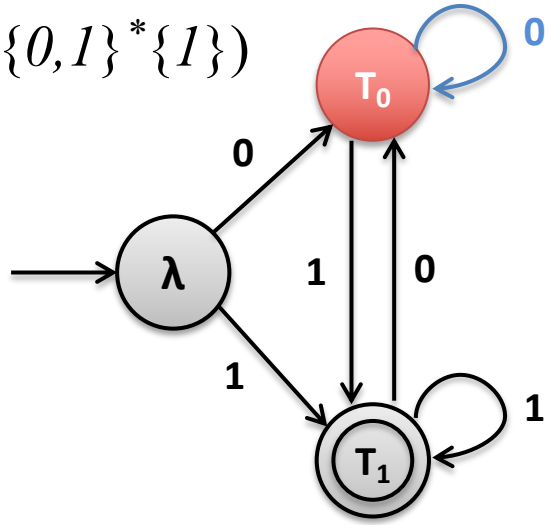
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



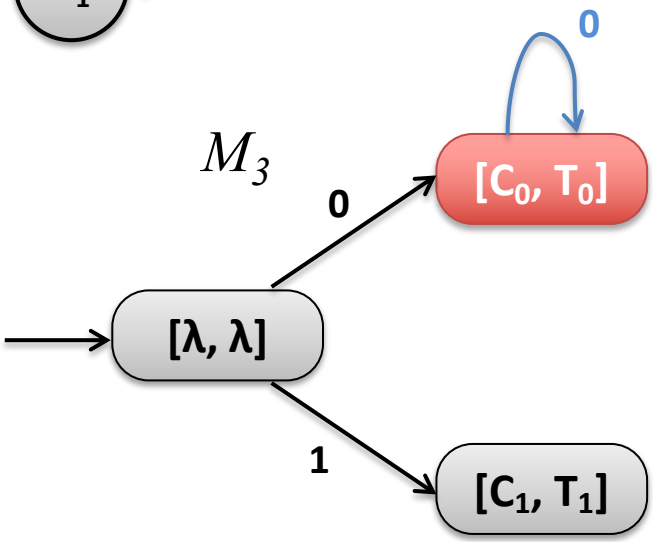
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	$[C_0,T_0]$	$[C_1,T_1]$
$[C_0,T_0]$	**$[C_0,T_0]$**	
$[C_1,T_1]$		

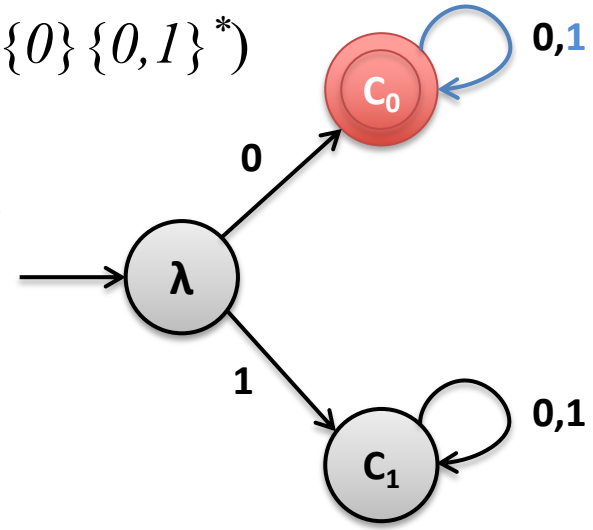
M_3



Produto de AFDs

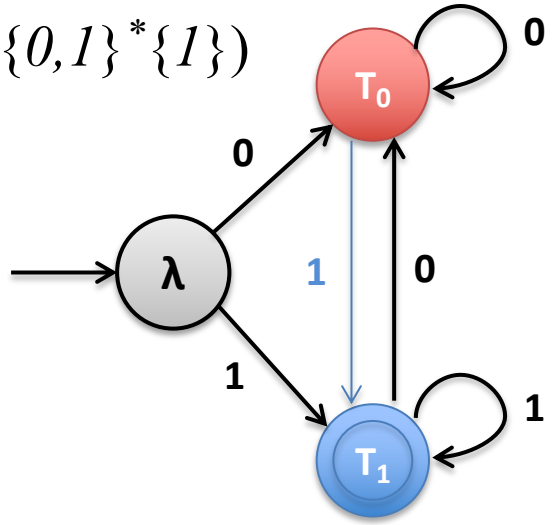
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



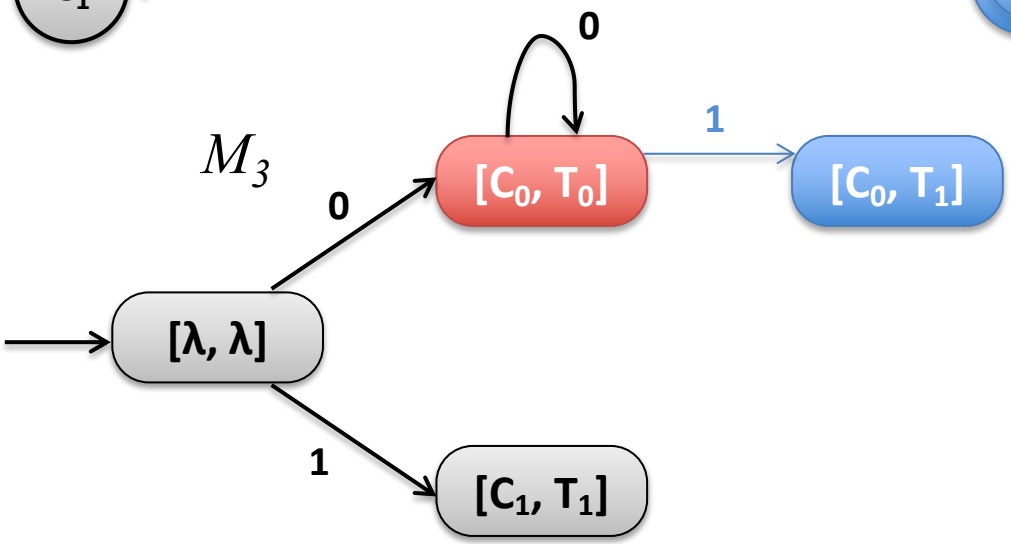
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	$[C_0,T_0]$	$[C_1,T_1]$
$[C_0,T_0]$	$[C_0,T_0]$	**$[C_0,T_1]$**
$[C_1,T_1]$		
$[C_0,T_1]$		

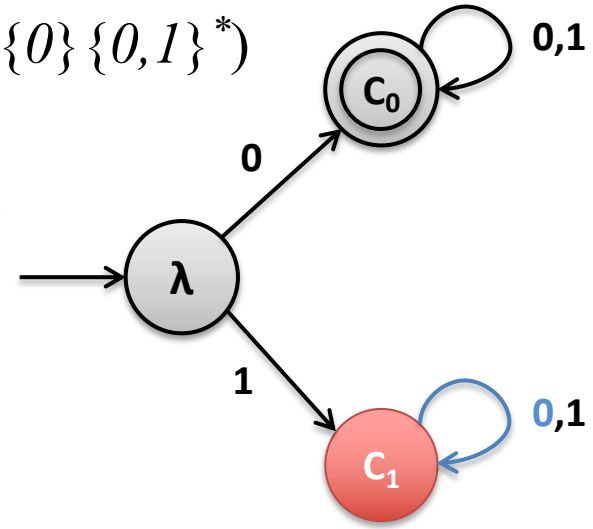
M_3



Produto de AFDs

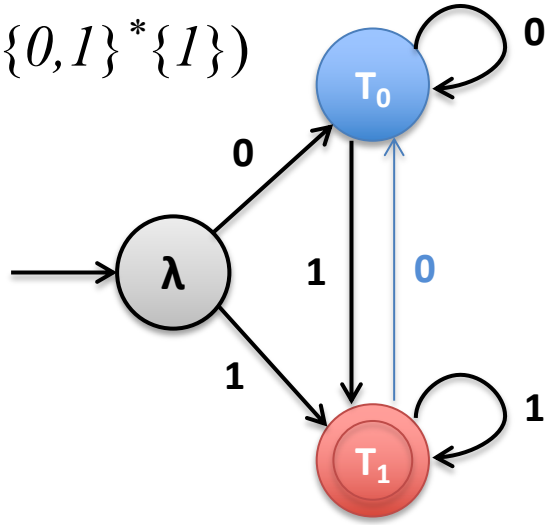
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



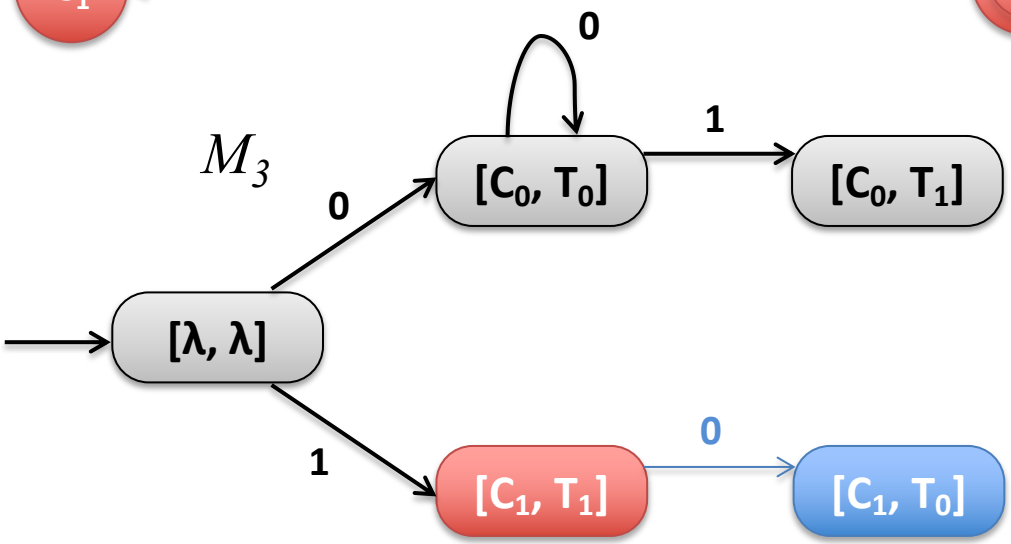
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	$[C_0,T_0]$	$[C_1,T_1]$
$[C_0,T_0]$	$[C_0,T_0]$	$[C_0,T_1]$
$[C_1,T_1]$	**$[C_1,T_0]$**	
$[C_0,T_1]$		
$[C_1,T_0]$		

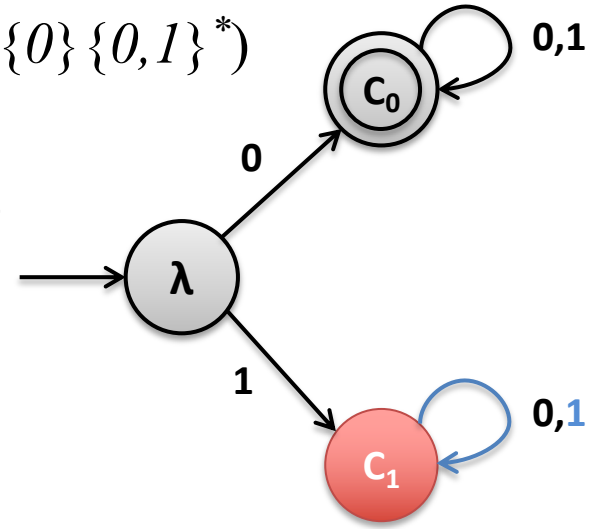
M_3



Produto de AFDs

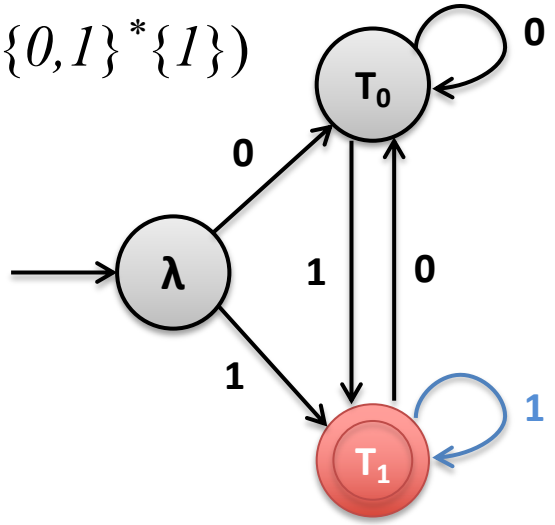
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



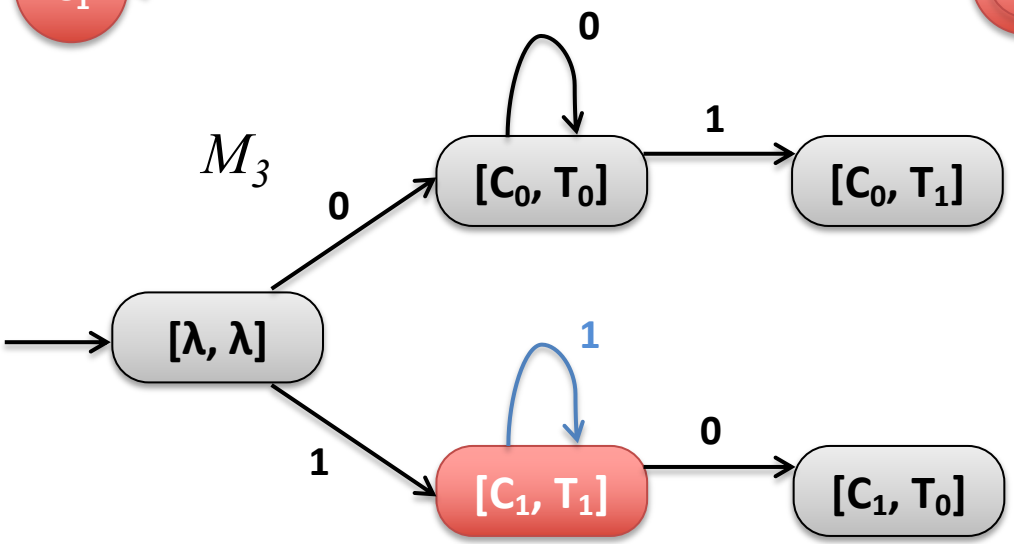
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	[C_0,T_0]	[C_1,T_1]
[C_0,T_0]	[C_0,T_0]	[C_0,T_1]
[C_1,T_1]	[C_1,T_0]	**[C_1,T_1]**
[C_0,T_1]		
[C_1,T_0]		

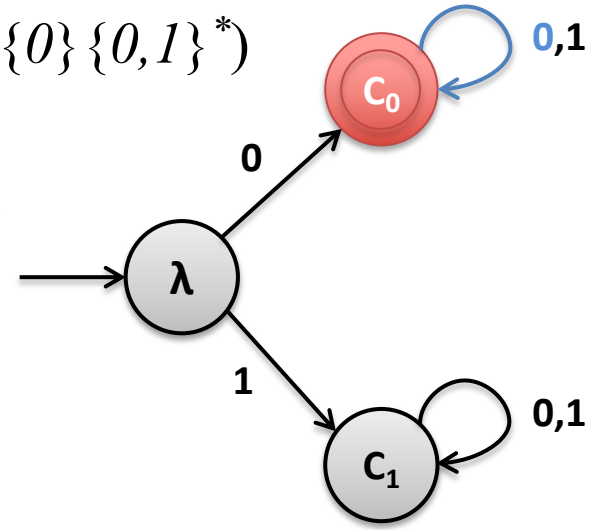
M_3



Produto de AFDs

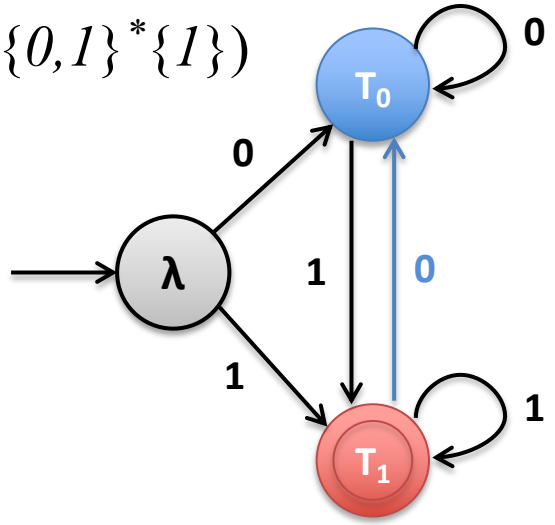
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



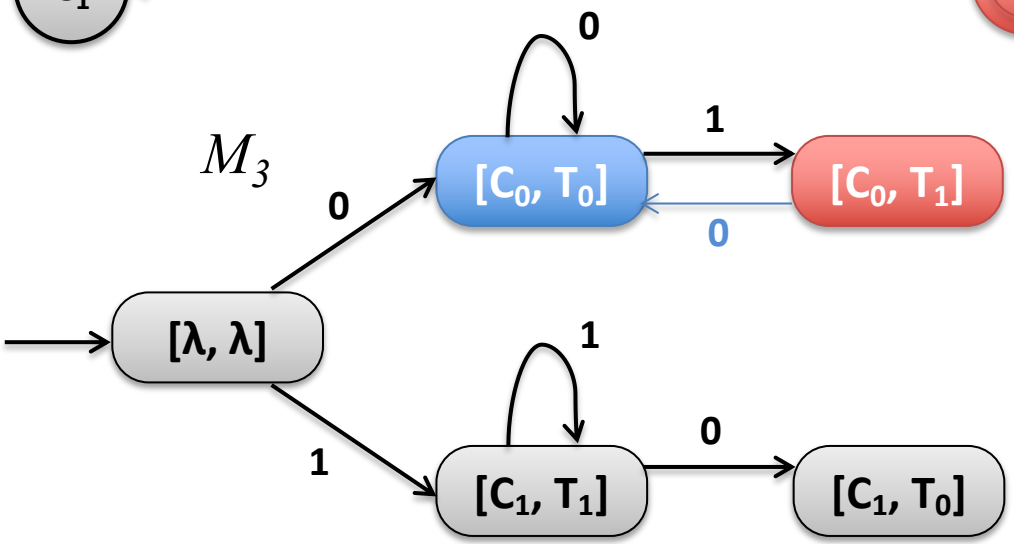
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	$[C_0,T_0]$	$[C_1,T_1]$
$[C_0,T_0]$	$[C_0,T_0]$	$[C_0,T_1]$
$[C_1,T_1]$	$[C_1,T_0]$	$[C_1,T_1]$
$[C_0,T_1]$	**$[C_0,T_0]$**	
$[C_1,T_0]$		

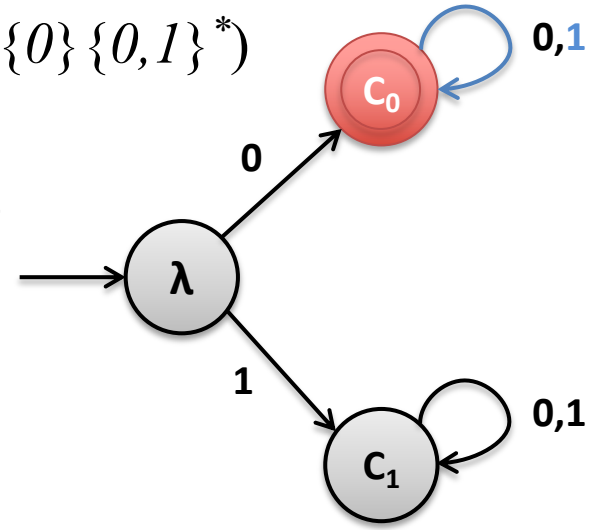
M_3



Produto de AFDs

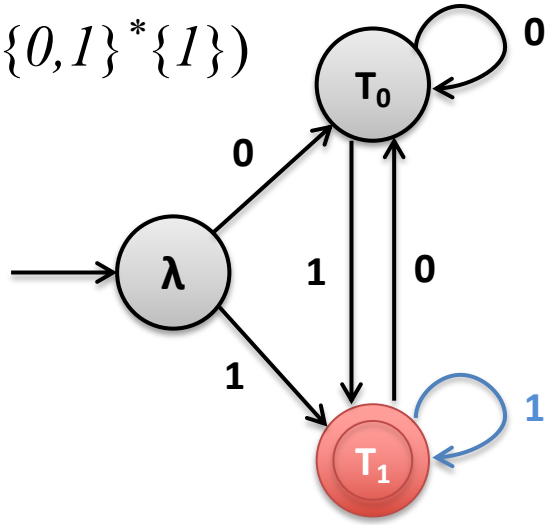
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



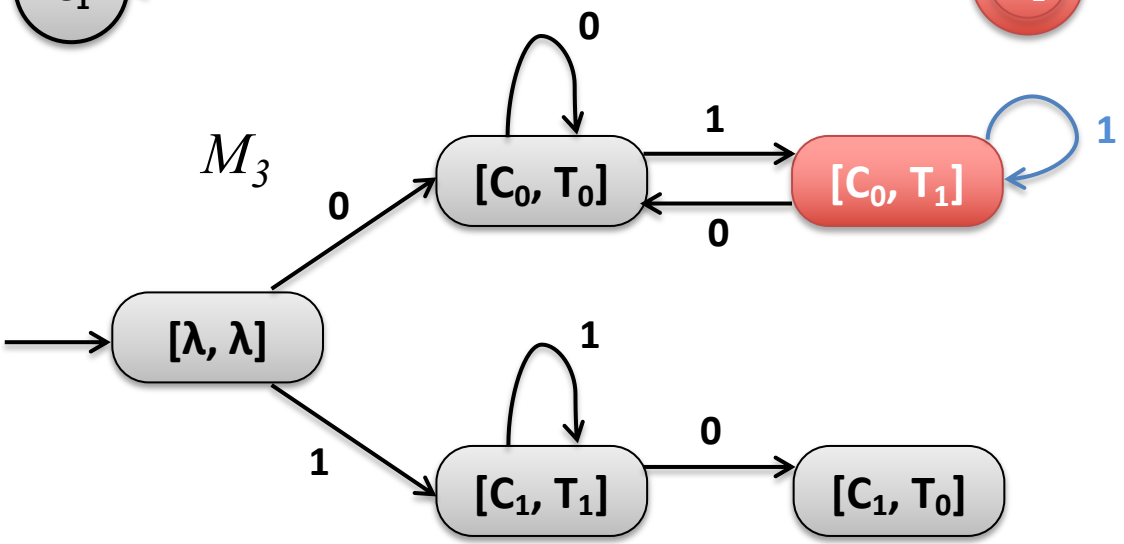
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	$[C_0,T_0]$	$[C_1,T_1]$
$[C_0,T_0]$	$[C_0,T_0]$	$[C_0,T_1]$
$[C_1,T_1]$	$[C_1,T_0]$	$[C_1,T_1]$
$[C_0,T_1]$	$[C_0,T_0]$	**$[C_0,T_1]$**
$[C_1,T_0]$		

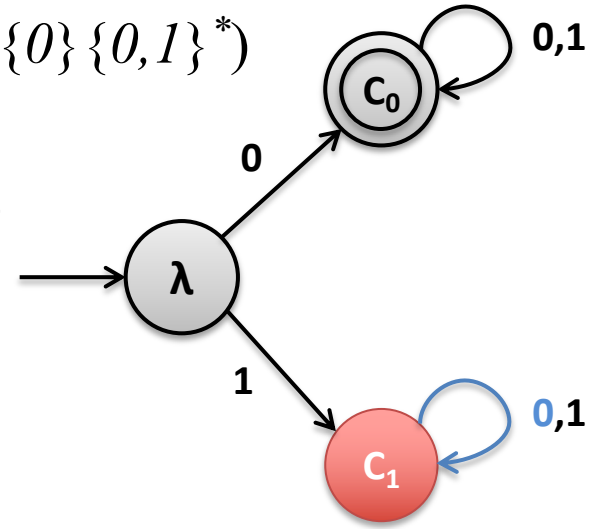
M_3



Produto de AFDs

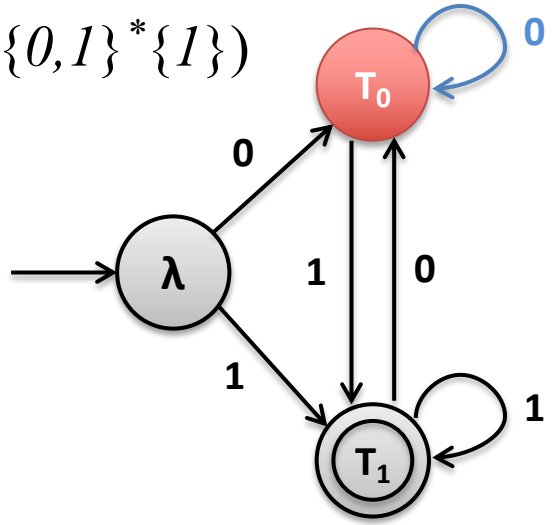
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



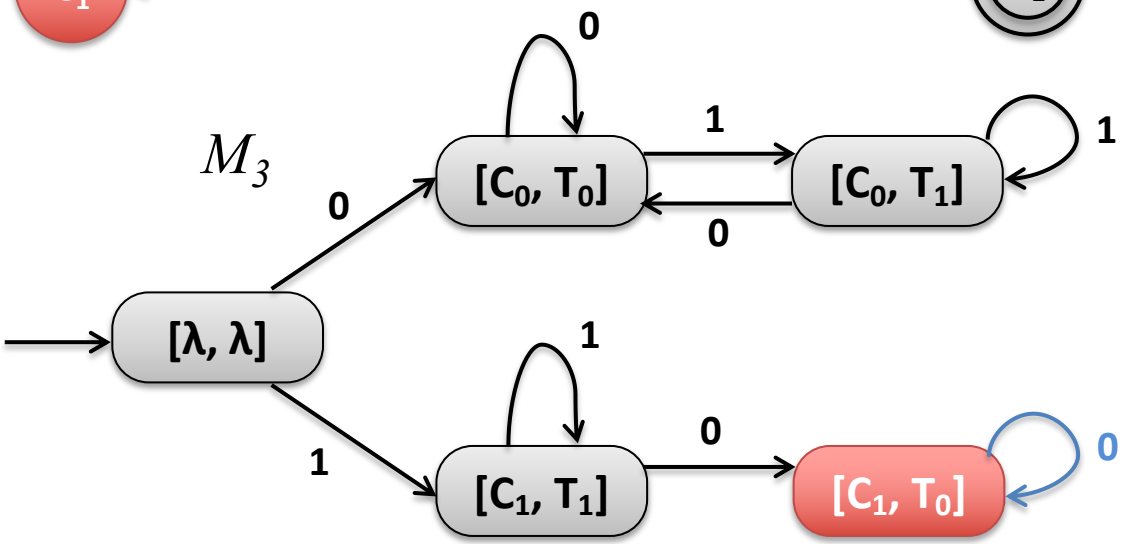
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1



δ_3	0	1
[λ, λ]	$[C_0,T_0]$	$[C_1,T_1]$
$[C_0,T_0]$	$[C_0,T_0]$	$[C_0,T_1]$
$[C_1,T_1]$	$[C_1,T_0]$	$[C_1,T_1]$
$[C_0,T_1]$	$[C_0,T_0]$	$[C_0,T_1]$
$[C_1,T_0]$	**$[C_1,T_0]$**	

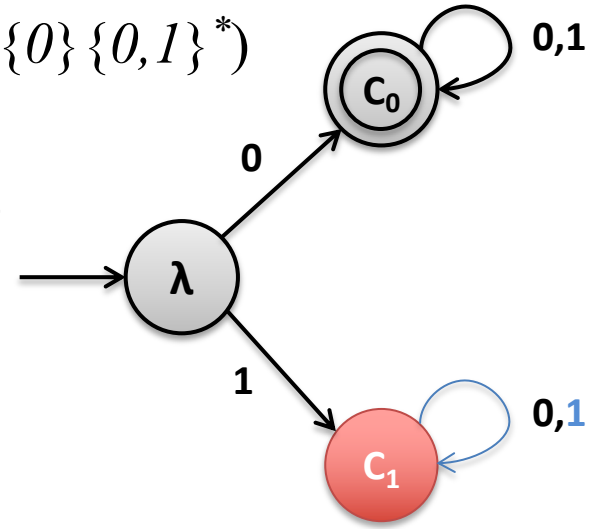
M_3



Produto de AFDs

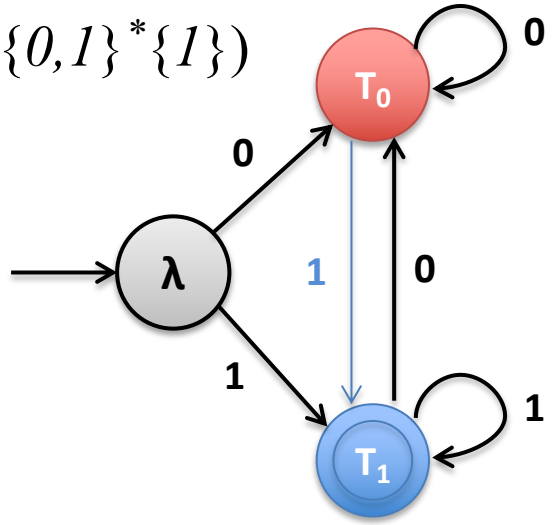
$M_1(\{0\}\{0,1\}^*)$

δ_1	0	1
λ	C_0	C_1
C_0	C_0	C_0
C_1	C_1	C_1



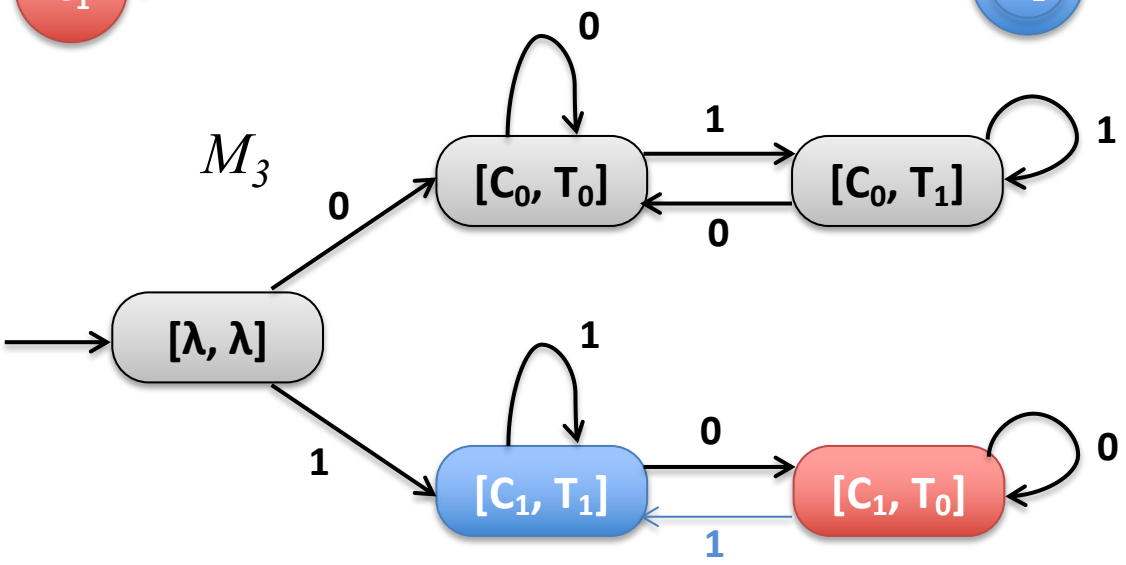
$M_2(\{0,1\}^*\{1\})$

δ_2	0	1
λ	T_0	T_1
T_0	T_0	T_1
T_1	T_0	T_1

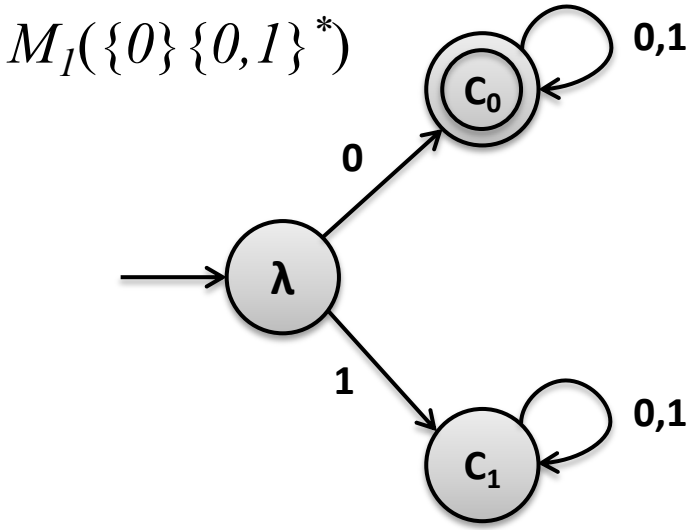


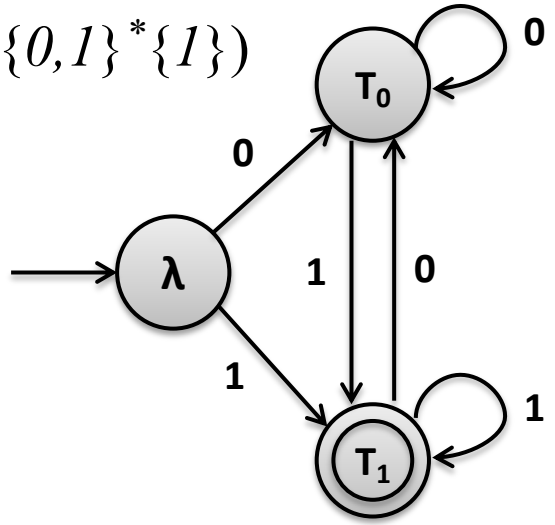
δ_3	0	1
[λ, λ]	$[C_0,T_0]$	$[C_1,T_1]$
$[C_0,T_0]$	$[C_0,T_0]$	$[C_0,T_1]$
$[C_1,T_1]$	$[C_1,T_0]$	$[C_1,T_1]$
$[C_0,T_1]$	$[C_0,T_0]$	$[C_0,T_1]$
$[C_1,T_0]$	$[C_1,T_0]$	**$[C_1,T_1]$**

M_3

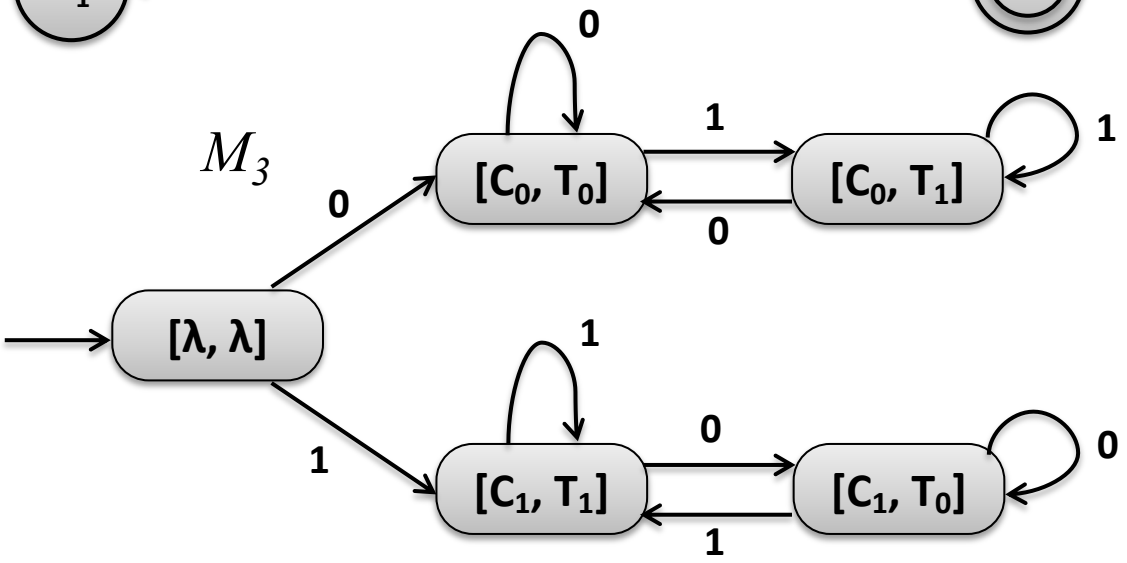


Produto de AFDs



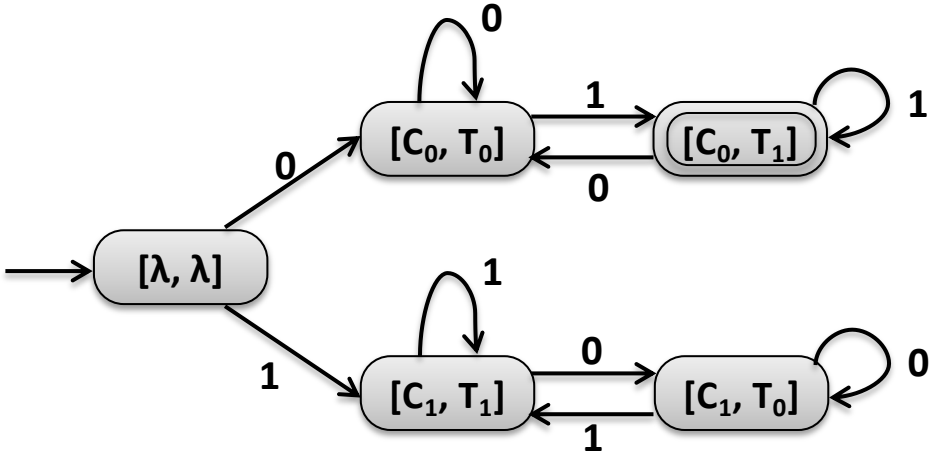


Onde colocar os estados finais?

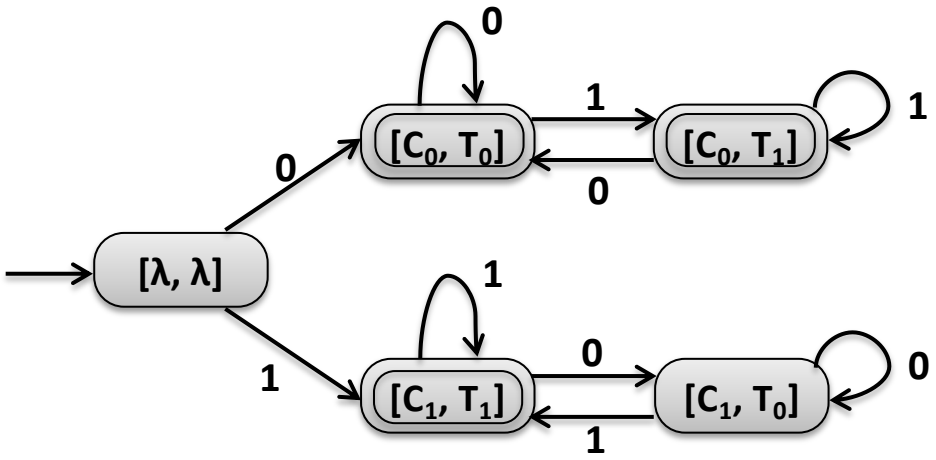


Produto de AFDs

- Reconhecendo $\{0\}\{0,1\}^* \cap \{0,1\}^*\{1\}$



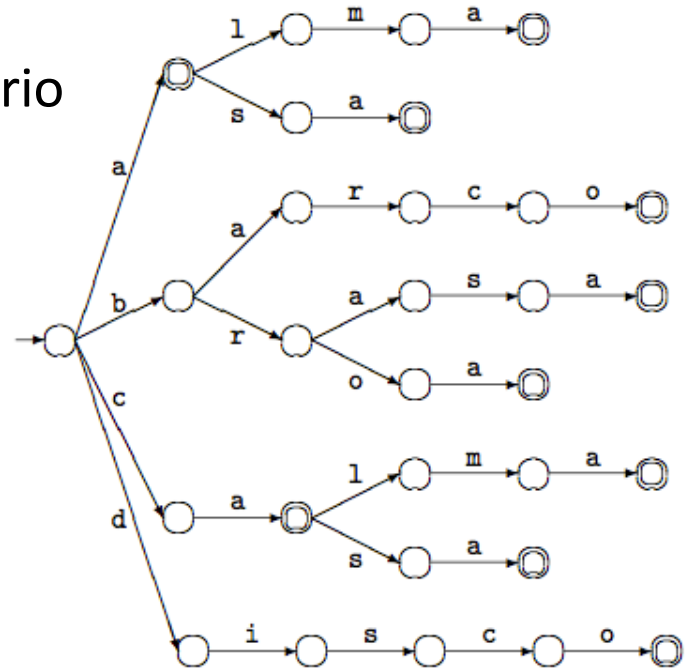
- Reconhecendo $\{0\}\{0,1\}^* \cup \{0,1\}^*\{1\}$



Linguagens Finitas

- Para toda **linguagem finita** existe um AFD
- Uma linguagem finita possui um AFD com diagrama de estados simplificado que **não possui ciclos**
  - O autômato é uma **árvore** cuja raiz é o estado inicial
- Exemplo: Um pequeno dicionário

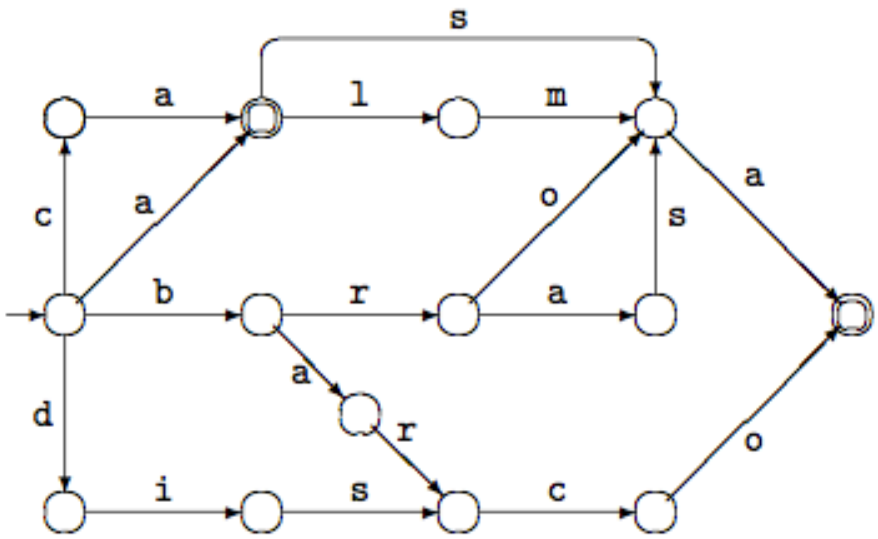
K = { a, alma, asa, barco, brasa, broa, ca, calma, casa, disco }



Linguagens Finitas

- Um AFD mais conciso pode ser construído a partir de um diagrama de estados simplificado sem ciclos – basta aplicar o algoritmo de minimização

- Alguns sufixos são comuns, como no exemplo do dicionário K



Linguagens Finitas

- Propriedades de linguagens finitas
  - Se uma linguagem é finita, então existe um AFD que a reconhece cujo diagrama de estados simplificado não contém ciclos
  - Se um AFD com diagrama de estados simplificado não contém ciclo, então a linguagem que ele reconhece é finita

Linguagens Infinitas

- Se um AFD reconhece uma linguagem infinita, então seu AFD com diagrama de estados simplificados **tem ciclos**
  - Já que uma linguagem infinita possui palavras de todos os tamanhos

- Seja uma linguagem infinita
  - Como saber se existe ou não AFD que a reconhece?
  - Como mostrar que não existe AFD para uma determinada linguagem?

Linguagens Infinitas

- Suponha que existe um AFD M para reconhecer a linguagem $L = \{a^n b^n \mid n \geq 0\}$. Como L é infinita, o AFD possui ciclos. Seja v ($v \neq \lambda$) uma subsequência de z consumida ao se percorrer o ciclo, onde $z = uvw$ para algum prefixo u e sufixo w. Como o ciclo pode ser percorrido várias vezes, tem-se

$$uv^i w \in L, \forall i \geq 0$$

- Seja $z = a^k b^k$ para algum k tal que $|z|$ é maior ou igual ao número de estados de M. Então $uv^2 w \notin L$, pois
  a) se v só contém a's: $uv^2 w = a^{k+|v|} b^k$
  b) se $v = a^i b^j$ para $1 \leq i, j \leq k$: $uv^2 w = a^{k-i}(a^i b^j)^2 b^{k-j} = a^k b^j a^i b^k$
  c) se v só contém b's: $uv^2 w = a^k b^{k+|v|}$

Contradição! A existência de ciclos implicaria no reconhecimento de palavras que não pertencem a L

Linguagens Infinitas

- A técnica empregada no exemplo anterior leva a um teorema para provar que não existe AFD para L

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

- $z = uvw$
- $v \neq \lambda$
- $uv^i w \in L(M), \forall i \geq 0$

Problemas de Decisão para AFDs

- Existem procedimentos de decisão para determinar, para qualquer AFD M, se
  - $L(M) = \emptyset$
  - $L(M)$ é finita

- Seja M' um AFD mínimo equivalente à M, então
  - $L(M) = \emptyset$ se e somente se M' não tiver estados finais
  - $L(M)$ é finita se e somente se o diagrama de estados de M' não possui ciclo

ISSO É TUDO, PESSOAL!