Fundamentos Teóricos da Computação

CIÊNCIA DA COMPUTAÇÃO

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

Sumário

* **Autômatos Finitos Determinísticos**
  * Definição
  * Equivalência
  * Minimização

Autômatos Finitos Determinísticos

* **Autômatos Finitos Determinísticos**

Autômatos Finitos Determinísticos

* Os exemplos apresentados anteriormente introduziram informalmente a noção de autômato finito na sua modalidade determinística

* Veremos agora uma definição precisa de autômato finito determinístico, um método para determinar um autômato mínimo equivalente a outro dado e algumas propriedades importantes de autômatos finitos determinísticos

Autômatos Finitos Determinísticos

* O que é autômato finito determinístico?
  * Um autômato finito determinístico é uma estrutura matemática constituída por três tipos de entidades: um conjunto de estados, um alfabeto e um conjunto de transições.
* A definição, a seguir, apresenta de forma precisa a caracterização de um AFD

Definição de AFD's

* **Definição de AFD's**

Definição de AFD's

* Um AFD é uma quíntupla: $(E, \Sigma, \delta, i, F)$ em que:
  * E é um conjunto finito de um ou mais estados;
  * Σ é um alfabeto;
  * δ : E x Σ -> E é a função de transição, uma função total;
  * i, um estado de E, é o estado inicial
  * F, um subconjunto de E, é o conjunto dos estados finais.

  * Como δ é uma função total, deve haver uma aresta, e apenas uma, sob cada símbolo de Σ, saindo de cada estado de E e levando a outro estado de E.

Definição de AFD's

* A definição modela as transições de um AFD como uma função que mapeia cada par (*estado*, *símbolo*) para **um** estado.

* Esse fato, que cada par (*estado*, *símbolo*) leva a um único estado, é que caracteriza o determinismo do AFD: a partir do estado inicial, só é possível atingir um único estado para uma dada palavra de entrada.

* E o fato de a função ser total garante que, para toda palavra de entrada, atinge-se um estado consumindo-se toda a palavra.

Homem, Leão, Coelho e Repolho

* No diagrama de estados deste problema, não se representou os estados que levaram a uma tragédia
  * O AFD foi simplificado para ser desenhado
  * Trata-se de um "diagrama de estados simplificado"

* Assume-se que existe um estado de erro (e') não mostrado no diagrama de estados para todo símbolo a não especificado no estado e, 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.

Homem, Leão, Coelho e Repolho

* Para conciliar o exemplo do "Homem, Leão, Coelho e Repolho", com a definição de AFD, basta inserir mais um estado, digamos *t* (de "tragédia").

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

* Em que E é o conjunto
$\{\{h, l, c, r\}, \{l, r\}, \{h, l, r\}, \{l\}, \{r\}, \{h, l, c\}, \{h, c, r\}, \{c\}, \{h, c\}, \{\}, \mathrm{t}\}$

* *Colinha:* $M = (E, \Sigma, \delta, i, F)$

Homem, Leão, Coelho e Repolho

* e δ é dada por:

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

Nesse formato tabular, uma função f é representada na forma que $f(i, j)$ seja exibido no cruzamento da linha i com a coluna j.

Outro exemplo
Reconhecendo número par de símbolos

* Seja o AFD M tal que:

$$L(M) = \{ w \in \{0,1\}^* \mid w \text{ tem um número par de símbolos } \}$$

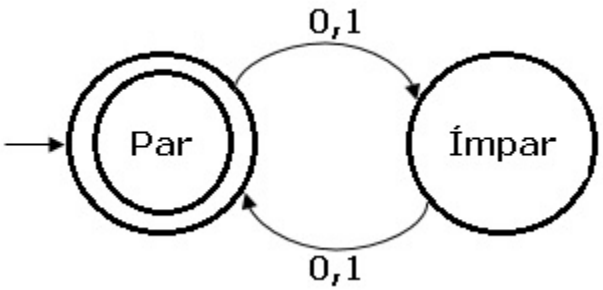
* Em linguagem matemática, o autômato seria uma quíntupla

$$M = (\{par, ímpar\}, \{0,1\}, \delta, par, \{par\})$$

* Em que δ é dada por:

δ	0	1
par	*ímpar*	*Ímpar*
ímpar	par	*par*

Outro exemplo
Reconhecendo número par de símbolos



Função de Transição Estendida

* A função de transição δ recebe somente um símbolo de Σ e computa o próximo estado

  * Pode-se definir recursivamente uma função de transição estendida, $\hat{\delta}$, que computa o estado alcançado para qualquer palavra w.

Função de Transição Estendida

* Seja um AFD $M = (E, \Sigma, \delta, i, F)$.
* A função de transição estendida para M, $\hat{\delta} : E \times \Sigma^* \rightarrow E$ é definida recursivamente como:

$$\hat{\delta}(e, \lambda) = e;$$
$$\hat{\delta}(e, ay) = \hat{\delta}(\delta(e, a), y), \forall\ a \in \Sigma \text{ e } y \in \Sigma^*.$$

Função de Transição Estendida

* Utilizando-se $\hat{\delta}$, pode-se definir a linguagem reconhecida por um AFD

* A linguagem reconhecida por um AFD $M = (E, \Sigma, \delta, i, F)$ é o conjunto $L(\mathrm{M}) = \{ w \in \Sigma^* \mid \hat{\delta}(i, w) \in F \}$

* Uma determinada palavra $w \in \Sigma^*$ é dita ser reconhecida, ou aceita por M se, e somente se, $\hat{\delta}(i, w) \in F$

Exemplo 1
Reconhecendo {0}{0,1}*

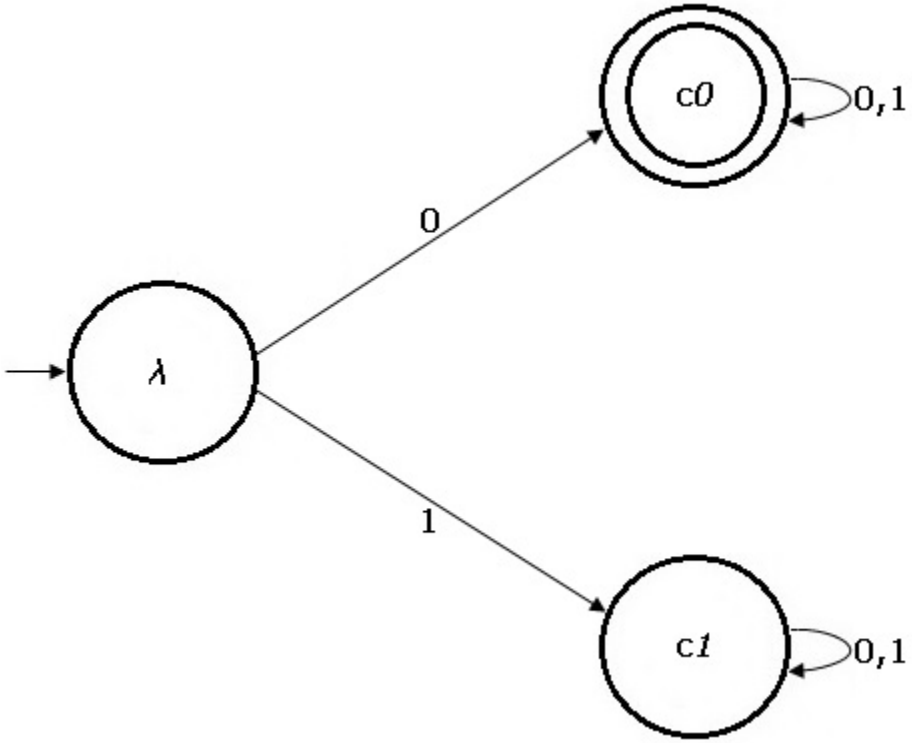
* Seja um AFD M que reconheça a linguagem {0}{0,1}*

$$L(M) = \{\ 0w \mid w \in \{0,1\}^*\ \}$$

* $c0$ = estado das palavra que começam com 0 (Estado final)
* $c1$ = estado das palavra que começam com 1 (Estado de erro)

δ	0	1
λ	$c0$	$c1$
$c0$	$c0$	$c0$
$c1$	$c1$	$c1$

Exemplo 1
Reconhecendo {0}{0,1}*



Exemplo 1
Reconhecendo {0}{0,1}*

* Processar a palavra 001 a partir do estado inicial para a tabela da função de transição do autômato.

$$\hat{\delta}(\lambda, 001) = \hat{\delta}(\delta(\lambda, 0), 01)$$
$$\hat{\delta}(F, 01)$$
$$\hat{\delta}(\delta(F, 0), 1)$$
$$\hat{\delta}(F, 1)$$
$$\hat{\delta}(\delta(F, 1), \lambda)$$
$$\hat{\delta}(F, \lambda) = F$$

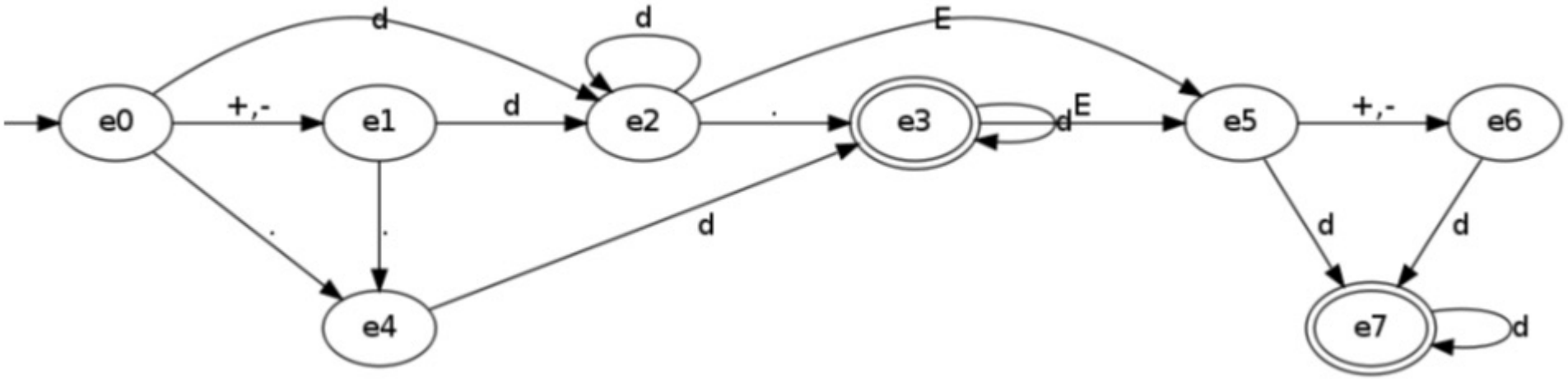
* F é estado final, portanto o autômato reconhece/aceita a palavra 001
* Obs: Para facilitar a visualização, o estado *c0* foi apelidado de F

Exemplo 2
Reconhecendo Números Reais

* Uma das aplicações de AFD's é a análise léxica de compiladores de linguagens de programação, como Pascal, C e Java. Nessa fase, são reconhecidas determinadas entidades sintáticas, como identificadores, constantes inteiras, constantes reais, palavras-chave etc.
  * Desenvolver um autômato que reconheça constantes reais.

  * E = {e0, e1, e2, e3, e4, e5, e6,e e7, ee}
  * Σ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, E}
  * F = {e3, e7}

Exemplo 2
Reconhecedor de Constantes Reais



Algoritmo para Simular AFDs

* Veja a seguir o algoritmo para simulação de um AFD

* A função AFD-RECONHECE
  * retorna “sim” caso o AFD reconheça a palavra
  * retorna “não” caso o AFD não reconheça a palavra

* Ao terminar de consumir a palavra, o algoritmo verificará se o estado onde foi consumido o último símbolo da palavra, pertence ou não ao conjunto de estados finais do AFD

```
função AFD-RECONHECE(i, F) retorna sim ou não
  entradas: i, Estado inicial do AFD
            F, Conjunto de estados finais
  saídas: sim ou não
  usa: PRÓXIMO(), retorna o próximo símbolo de entrada (fs se atingiu o fim)
       FUNÇÃO-TRANSIÇÃO(estado, símbolo), retorna o estado atingido a
                                          partir de estado sob símbolo
  estado ← i
  s ← PRÓXIMO()
  enquanto s ≠ fs faça
      estado ← FUNÇÃO-TRANSIÇÃO(estado, s)
      s ← PRÓXIMO()
  fim enquanto
  se estado ∈ F então
      retorne sim
  senão
      retorne não
  fim se
```

Autômatos Equivalentes

* Dois AFD's M_1 e M_2 , são ditos equivalentes se, e somente se, $L(M_1) = L(M_2)$.

* Se mais de um AFD pode reconhecer uma mesma linguagem, como se obter o AFD mínimo que reconhece uma linguagem?
  * O que é um AFD mínimo?

Definição

* Um AFD M é dito ser um AFD mínimo para a linguagem $L(M)$ se nenhum AFD para $L(M)$ contém um número menor de estados que M.
  * Como a função de transição é total e, considerando um alfabeto mínimo (sem símbolos inúteis), o número de transições é função, somente, do número de estados.

Minimização de AFDs

* Passo 1: Eliminar estados não alcançáveis a partir do estado inicial
  * Qualquer AFD que possua estados não alcançáveis a partir do estado inicial não pode ser mínimo

Minimização de AFDs

* Passo 2: Substituir estados equivalentes por um único estado

Estados Equivalentes

* Seja um AFD $M = (E, \Sigma, \delta, i, F)$. Dois estados $e, e' \in \mathrm{E}$ são equivalentes, $e \approx e'$, se, e somente se:
  * Para todo $y \in \Sigma^*$, $\hat{\delta}(e, y) \in F$, se e somente se, $\hat{\delta}(e', y) \in F$.
* Ou seja, qualquer palavra y que for reconhecida passando-se por um estado e até chegar a um estado final, também passará por e' até chegar a um estado final.

Relação "≈"

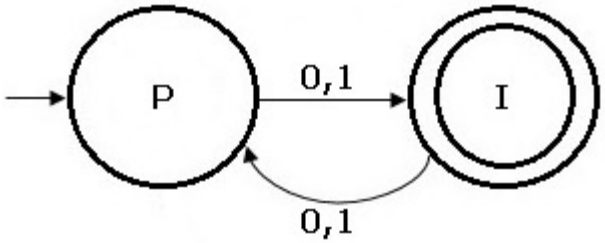
* A relação "≈" é uma relação de equivalência (reflexiva, transitiva, simétrica)
  * Pode induzir classes de equivalência
    * Também chamadas de partições
  * $[e] = \{e_1, e_2, ..., e_n\}$
    * Todos os estados da partição podem ser substituídos por um único estado

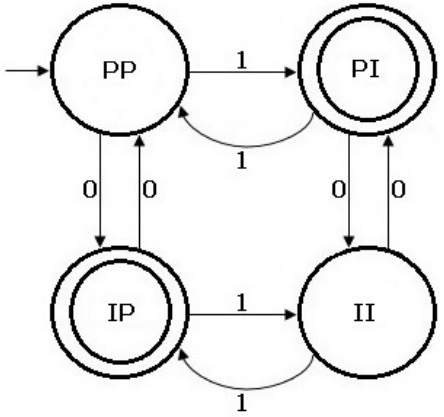
Autômato Reduzido

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

  * $E' = \{[e] \mid e \in E\}$;
  * $\delta'([e], a) = [\delta(e, a)]$ para todo $e \in E$ e $a \in \Sigma$
  * $i' = [i]$;
  * $F' = \{[e] \mid e \in F\}$.

* São Equivalentes?

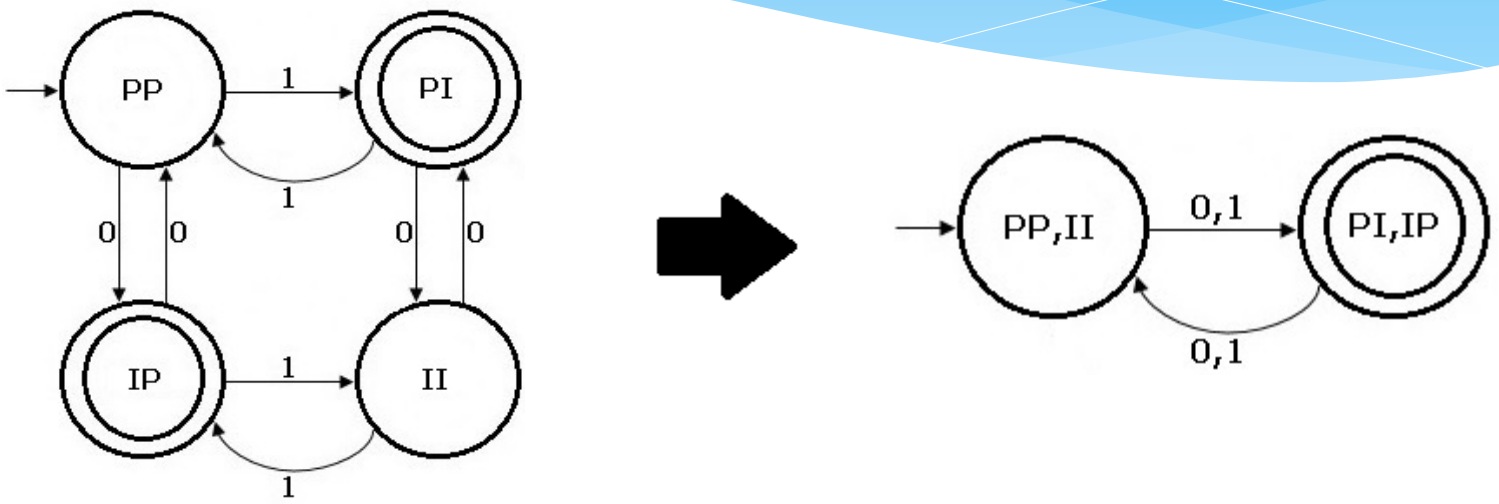




* [P] = { PP, II }
  * Qualquer palavra com tamanho <u>ímpar</u> leva a um estado final
  * Exemplos: 0, 1, 000, 001, ... (Reconhece)
* [I] = { PI, IP }
  * Qualquer palavra com tamanho <u>par</u> leva a um estado que **não** é final
  * Exemplos: 00, 11, 0001, 1110, ... (**Não** reconhece)

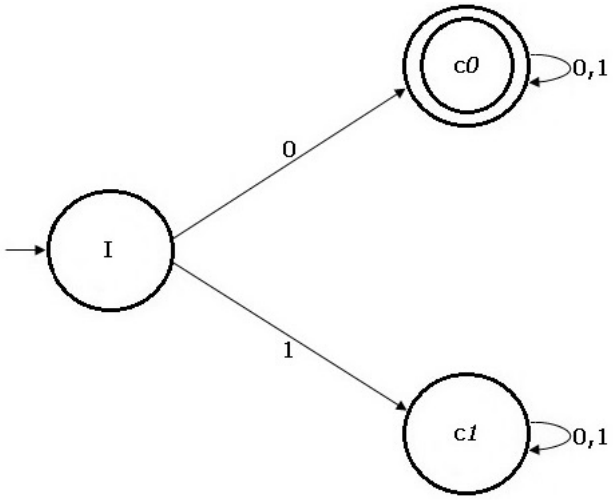
* <u>Sim, são equivalentes</u>. Ambos os autômatos reconhecem número ímpar de símbolos.

Minimizando



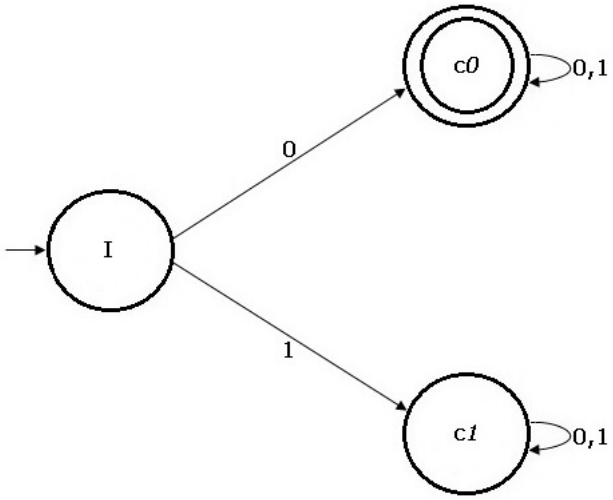
* Como fazer isto automaticamente?

* Voltando ao exemplo 1



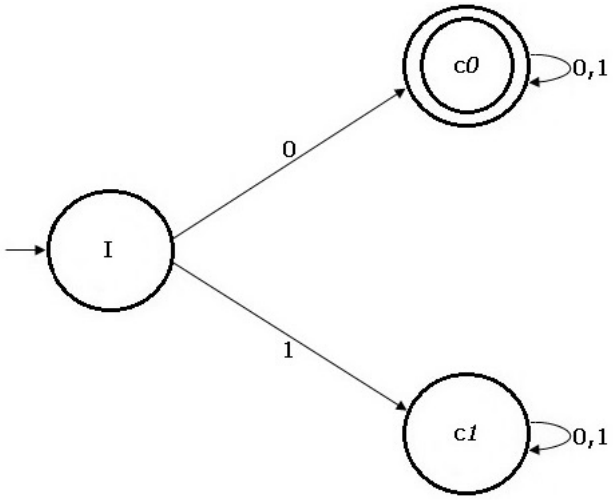
* Encontre o AFD mínimo equivalente ao AFD acima

* Resolução



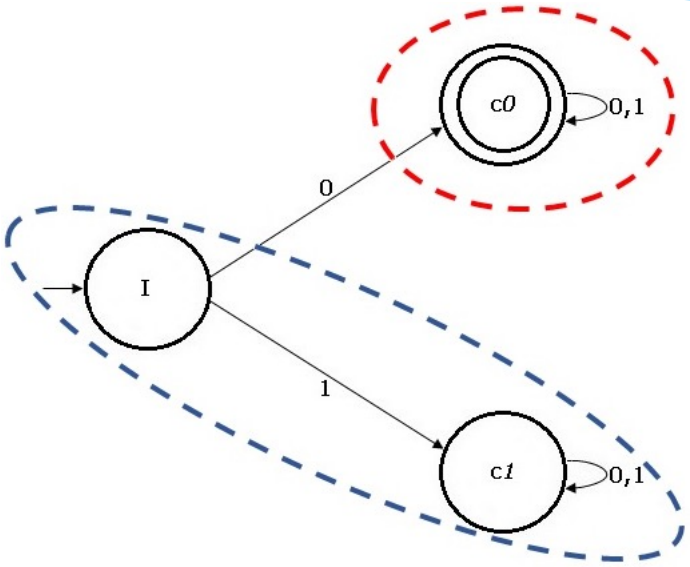
* Elimine estados não alcançáveis a partir do inicial
  * Todos os estados são alcançáveis

* Resolução



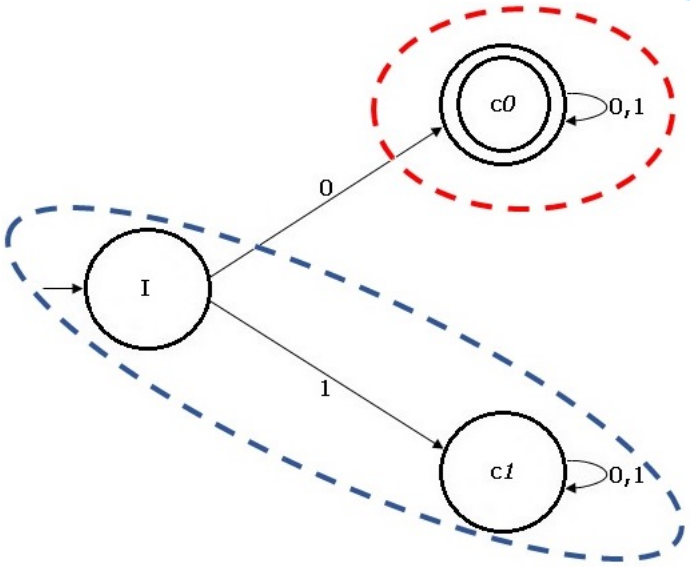
* Encontre as classes de equivalência por refinamentos
  * Inicialmente duas classes de equivalência
    * Estados finais
    * Estados não finais

* Resolução



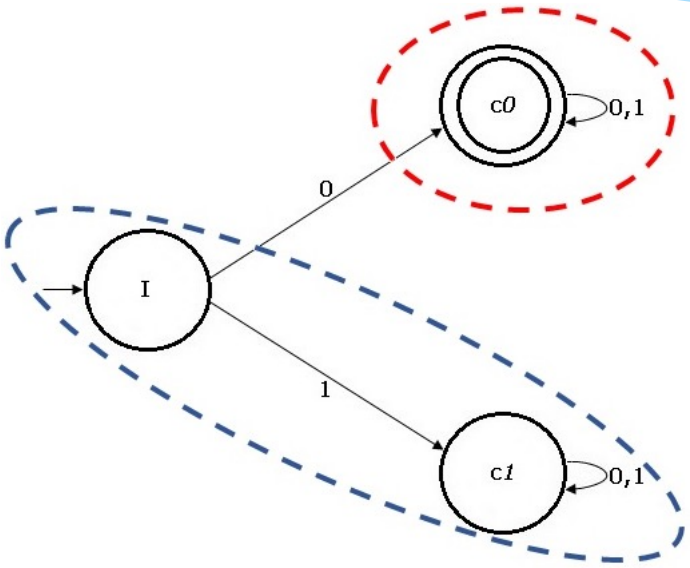
* Encontre as classes de equivalência por refinamentos
  * Inicialmente duas classes de equivalência
    * Estados finais
    * Estados não finais

* Resolução



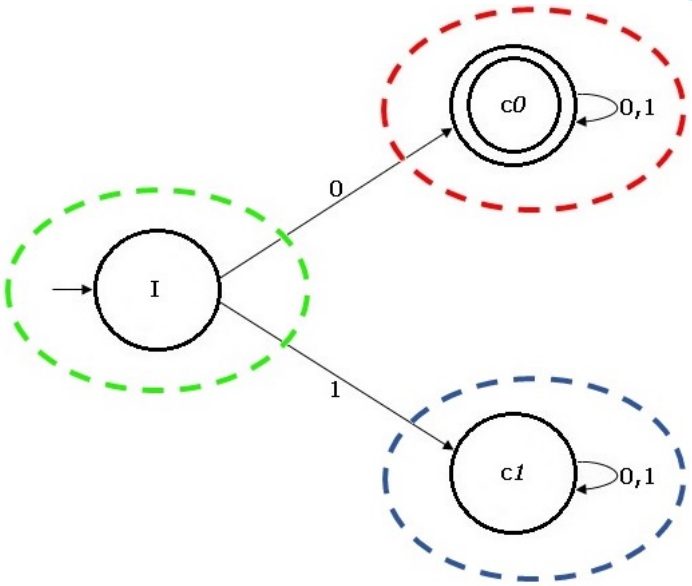
* Refine as Classes de equivalência
  * Cada estado dentro de uma classe deve ter arestas sob o símbolo a somente para uma mesma classe de equivalência
    * Sempre que esta regra não for obedecida, crie novas classes de equivalência para satisfazê-la

* Resolução



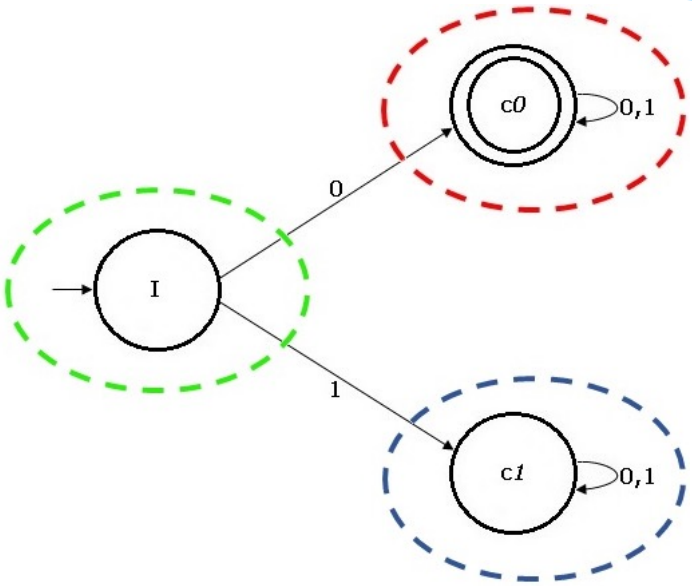
* Refine as Classes de equivalência
  * Sob a = 1, tanto I quanto $c1$, vão para a classe de equivalência azul
  * Sob a = 0, I vai para a classe vermelha e $c1$ vai para a classe azul
    * A regra não foi satisfeita!

* Resolução



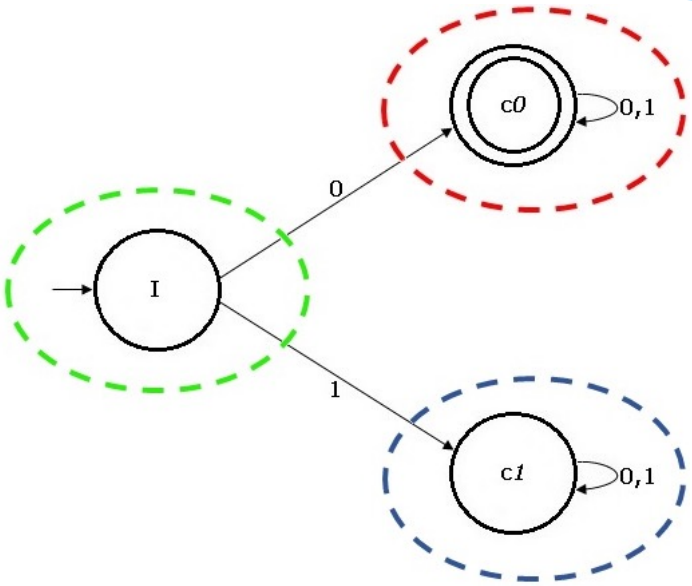
* Refine as Classes de equivalência
  * Todos os estados cujas arestas levam às mesmas classes de equivalência de I vão para a nova classe
  * Os demais permanecem na classe antiga

* Resolução



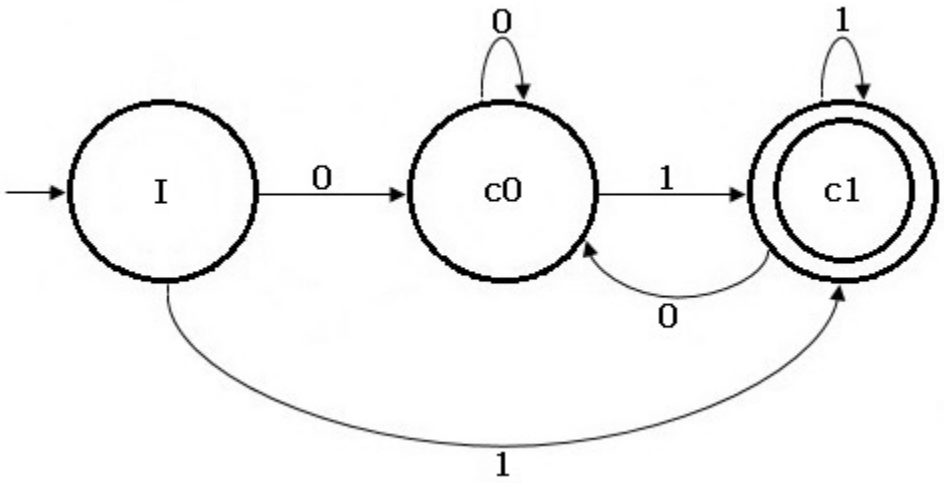
* Continue o refinamento até a regra não ser mais violada
  * Neste caso, existe uma classe para cada estado
    * O autômato já era mínimo

* Resolução



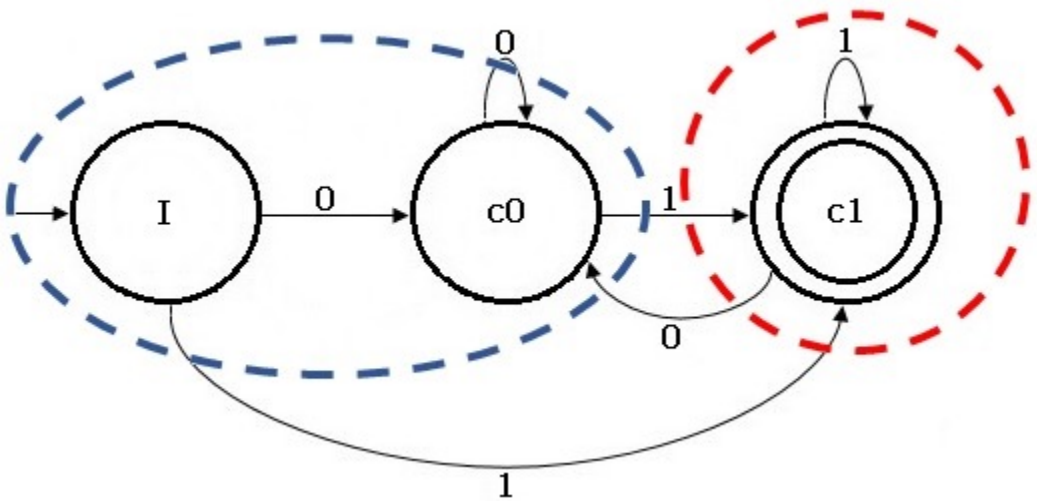
* Faça o estado inicial igual à classe de equivalência que contenha o estado inicial
* Faça os estados finais iguais às classes de equivalência cujos estados são finais

* Outro exemplo
  * Reconhecendo: {0,1}*{1}



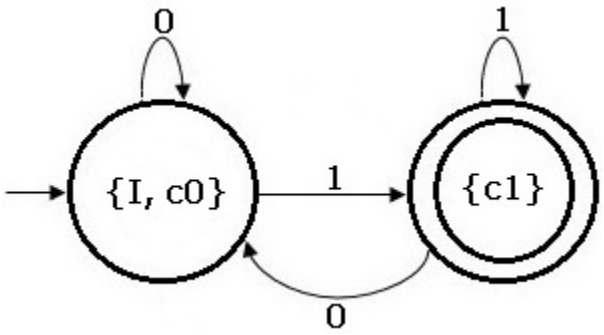
* Encontre o AFD mínimo equivalente ao AFD acima

* Resolução



* Classes de Equivalência
  * A classe vermelha só tem um estado
  * Sob $a = 0$
    * I e $c0$ vão para a classe azul
  * Sob $a = 1$
    * I e $c0$ vão para a classe vermelha

* Resolução



* Autômato mínimo
  * Os estados passam a ser classes de equivalência
  * As transições, as arestas entre as classes
    * Todas as transições sob um mesmo símbolo vão para uma mesma classe de equivalência

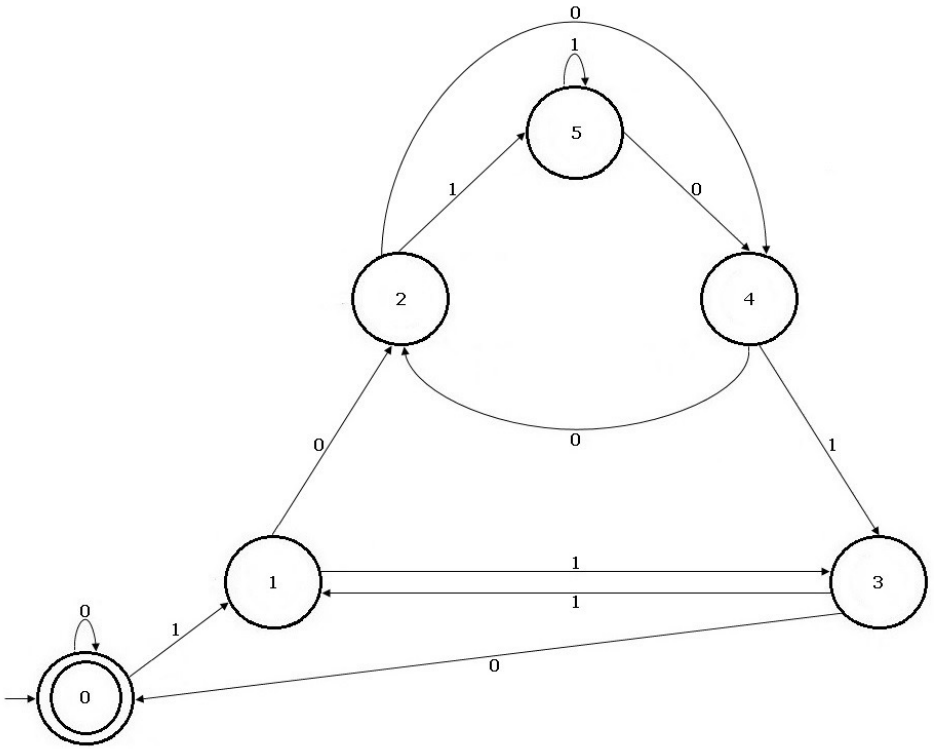
Vejamos mais um exemplo

* Seja o problema de projetar uma máquina que, dada uma sequencia de 0s e 1s, determine se o número **decimal** representado por ela na base 2 é divisível por 6.

* Para esse caso, não vamos focar no projeto do autômato, mas sim em sua minimização.

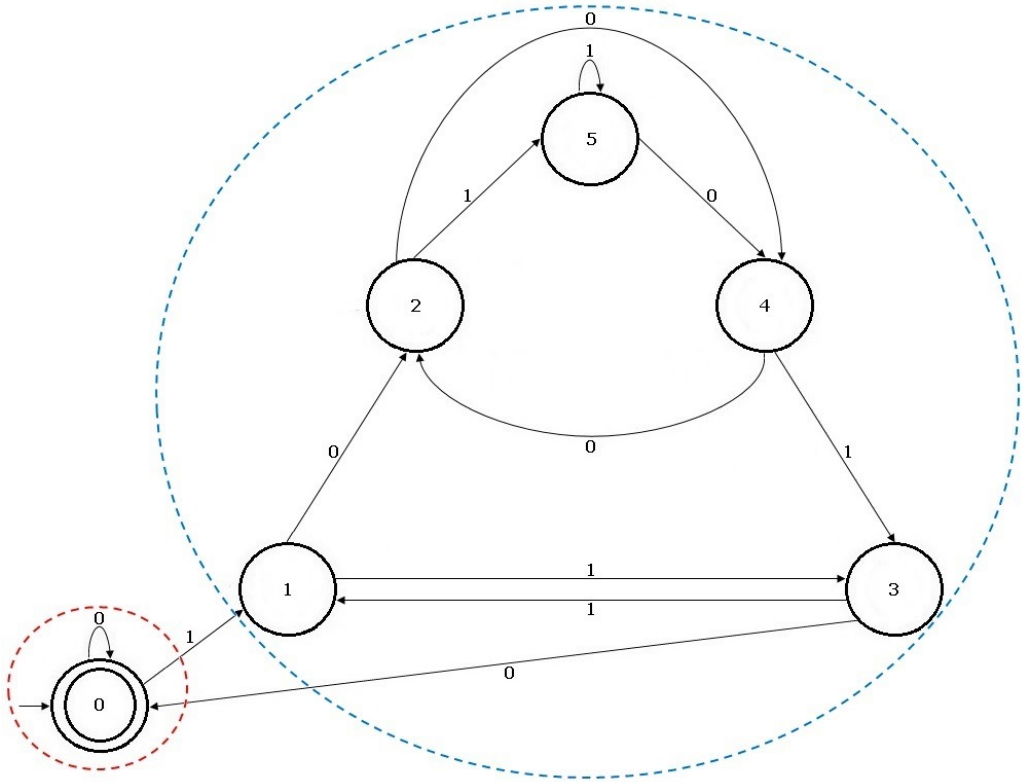
* Inicialmente, pode-se imaginar uma máquina com seis estados, um para cada resto de 0 a 5.

* Diagrama de estados



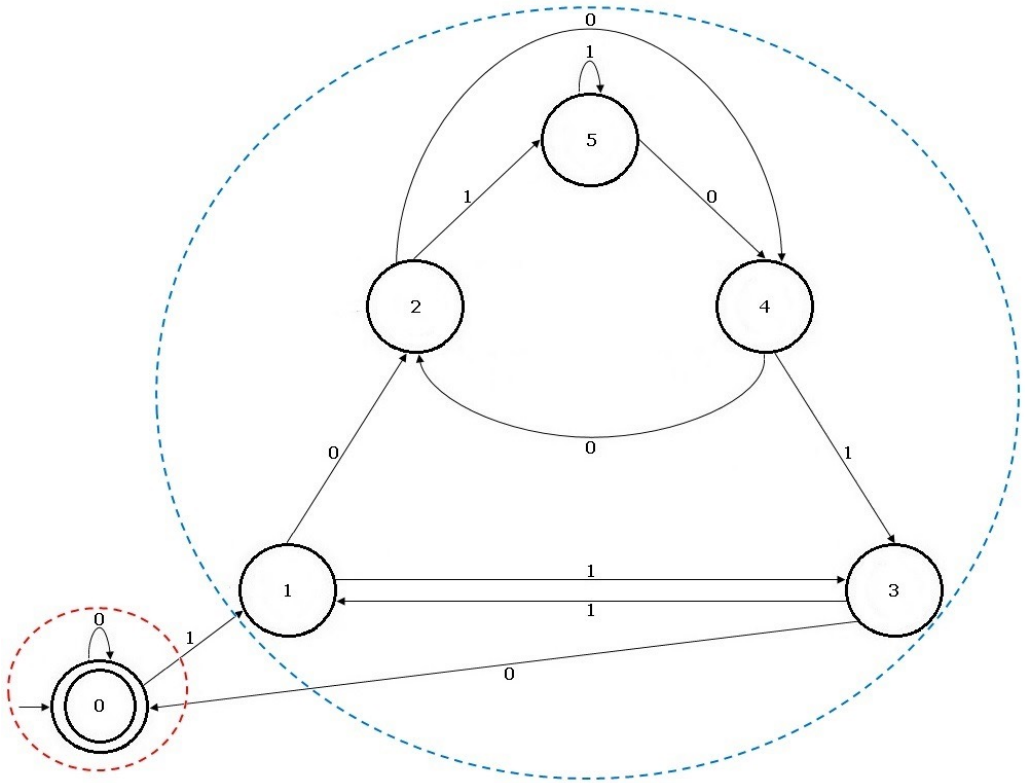
* Encontre o AFD mínimo equivalente ao AFD acima.

* Resolução



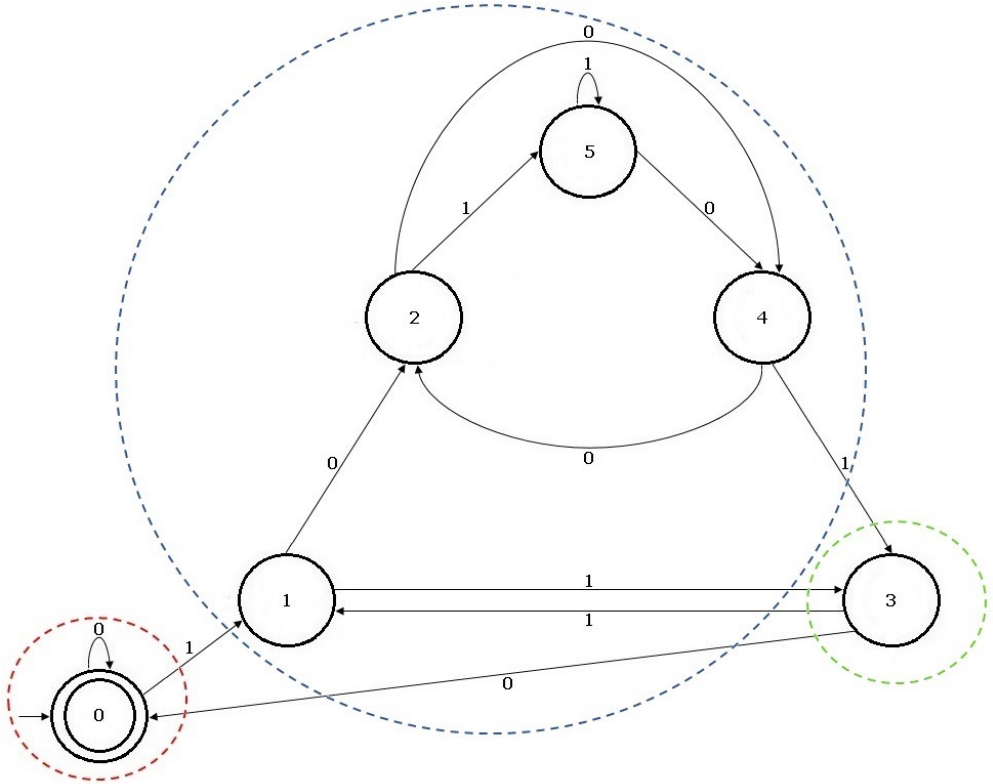
* Inicialmente duas classes de equivalência, estados finais e estados não finais

* Diagrama de estados



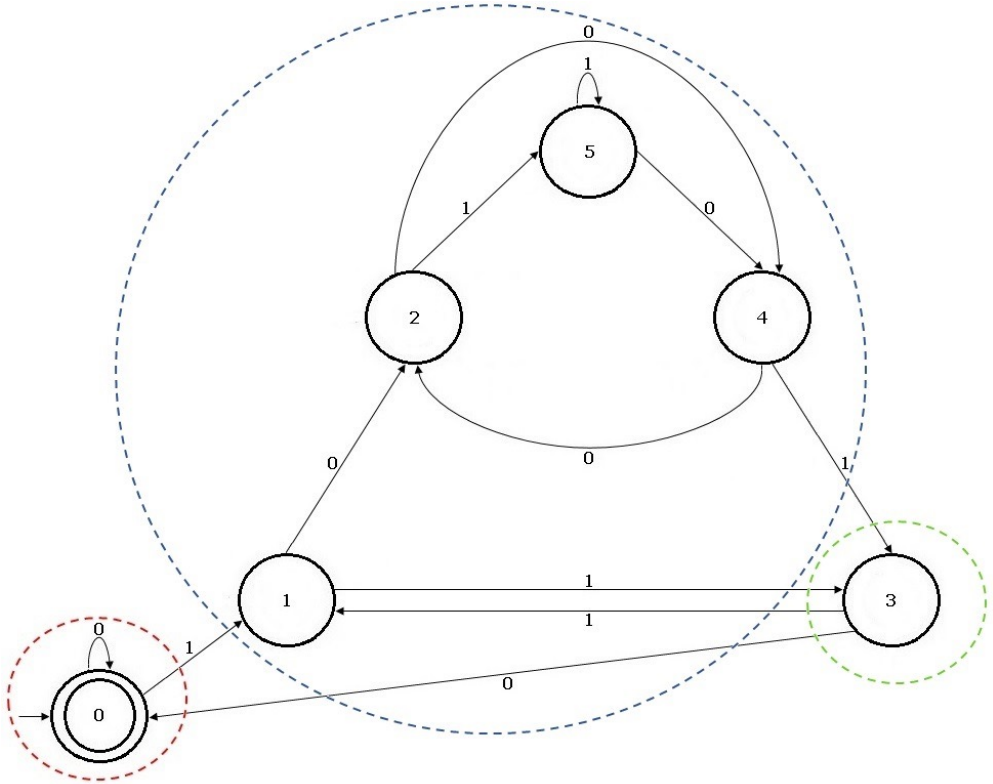
* O estado 3 tem uma transição para uma classe diferente que todos os outros

* Diagrama de estados



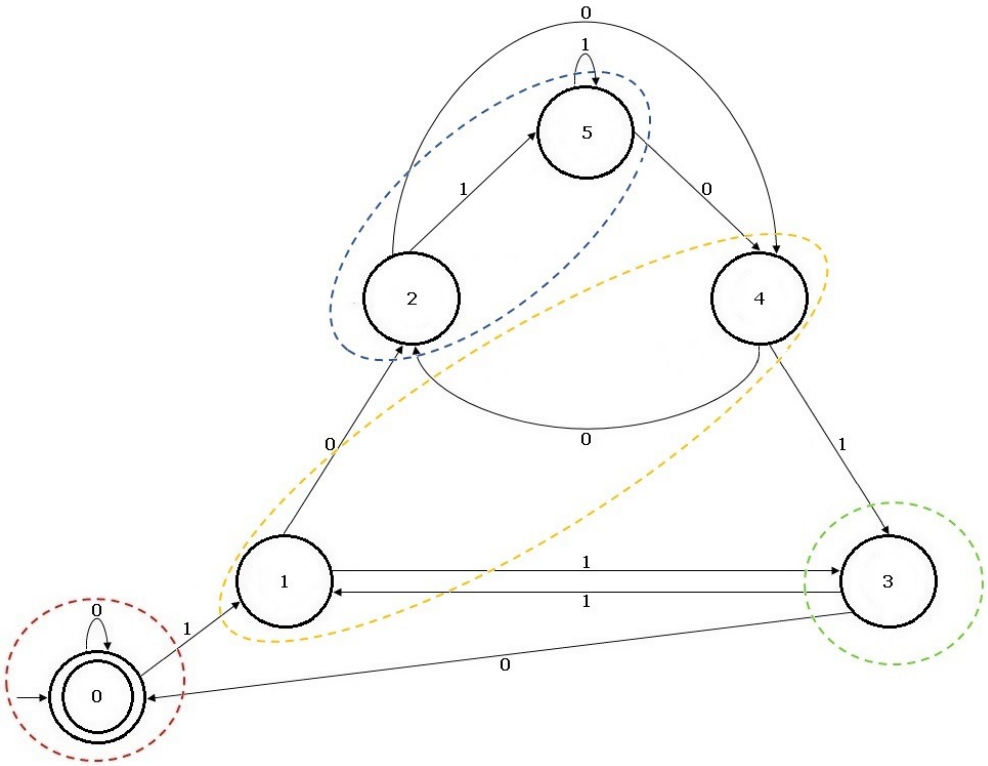
* O estado 3 tem uma transição para uma classe diferente que todos os outros

* Diagrama de estados



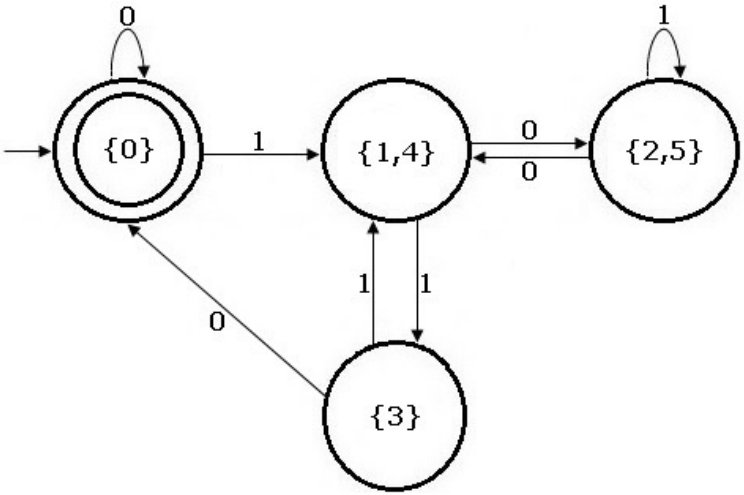
* O estado 1 e 4 transitam para a classe verde sob $a = 1$ e permanecem na azul caso contrário

* Diagrama de estados



* Não existem mais violações

* Diagrama de estados para binário módulo 6 minimizado



* {0} é o estado inicial e final

Algoritmo de minimização de AFDs

* Veja a seguir o algoritmo para minimização de um AFD

* A função AFD-MINIMIZA
  * Tem como entrada um AFD *P*
  * retorna um AFD mínimo equivalente a *P*

```
função AFD-MINIMIZA(P) retorna um AFD mínimo equivalente a P
  entradas: P, um AFD P = (E,Σ,δ,i,F)
  saídas: um AFD mínimo equivalente a P

  Elimine de P todo estado na alcançável a partir de i
  Se F = ∅ então
    retorne ({i},Σ,δ',i,∅), sendo δ'(i,a) = i para todo a ∈ Σ
  Senão E - F = ∅ então
    retorne ({i},Σ,δ',i,{i}), sendo δ'(i,a) = i para todo a ∈ Σ
  Fim se
  S0 ← {E - F, F}; n ← 0;
  repita
    n ← n + 1; Sn ← ∅;
    para cada X ∈ S(n-1) faça
      repita
        Selecione um estado e ∈ X;
        para cada a ∈ Σ:
          seja [δ(e,a)] o conjunto que contém δ(e,a) em S(n-1);
        seja Y = {e' ∈ X | δ(e',a) ∈ [δ(e,a)] para todo a ∈ Σ};
        X ← X - Y;
        Sn ← Sn ∪ {Y}
      até X = ∅
    fim para;
  até Sn = S(n-1)
  i' ← conjunto em Sn que contém i;
  F' ← {X ∈ Sn| X ⊆ F}
  para cada X ∈ Sn-1 e a ∈ Σ:
    δ'(X,a) = conjunto em Sn que contém δ(e,a), para qualquer e ∈ X;
  retorne (Sn,Σ,δ',i',F')
```

Atenção!

* Para fazer a minimização do autômato, deve-se considerar todos os seus estados

  * No caso de autômatos simplificados, onde o estado de erro é omitido no desenho, este estado DEVE ser considerado na minimização!

Obrigado.

joaopauloaramuni@gmail.com
joaopauloaramuni@fumec.br

UNIVERSIDADE FUMEC
50 Anos