

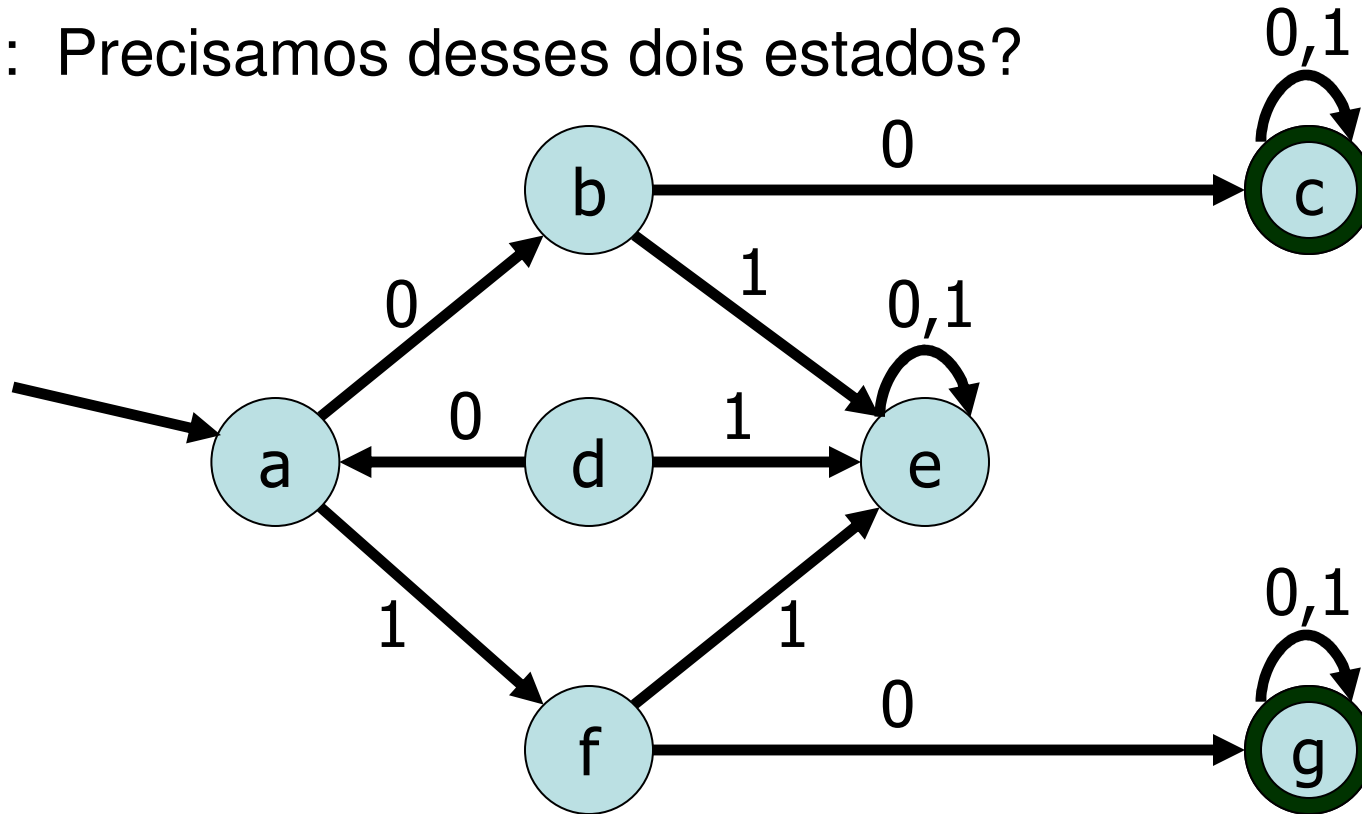
Minimização de AFD

Estados Equivalentes.

Exemplo

Considere os estados de aceitação c e g. Eles são ambos estados que, uma vez atingidos, nunca se sai deles, desde que se leia 0 ou 1.

Q: Precisamos desses dois estados?

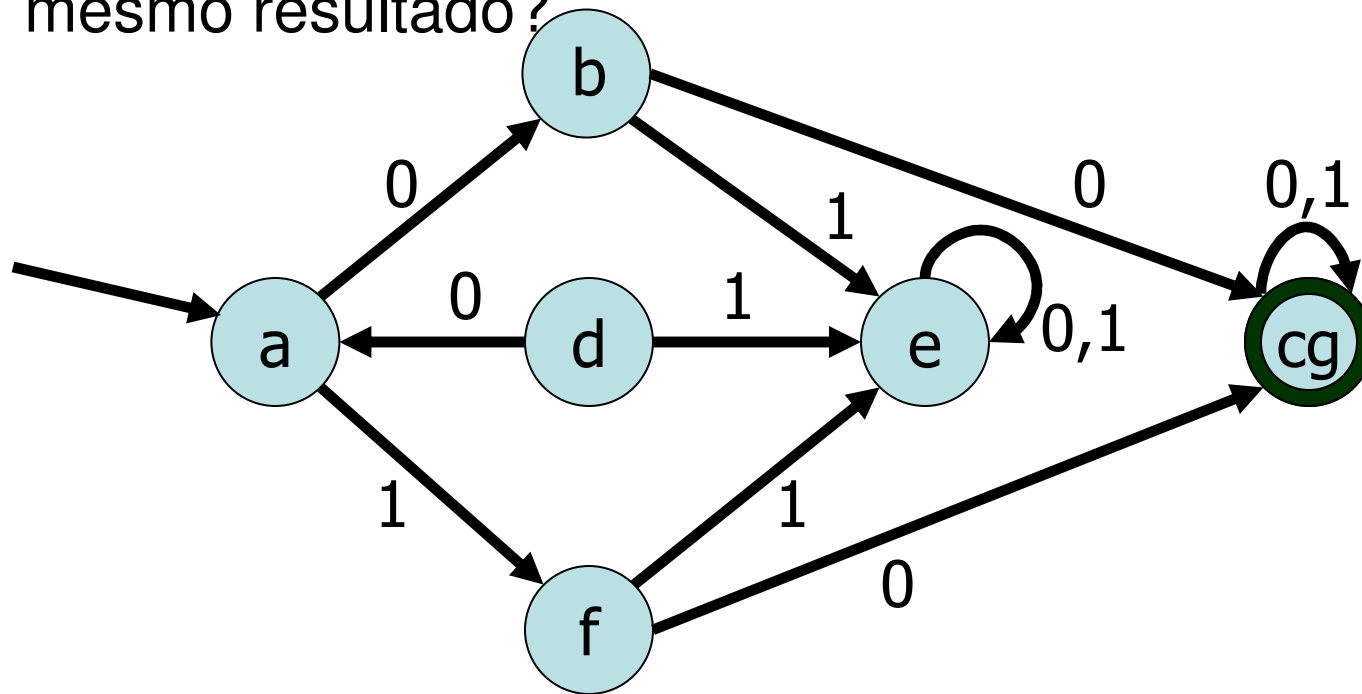


Estados Equivalentes.

Exemplo

R: Não, eles podem ser unificados como se mostra abaixo.

Q: Existem outros estados que podem ser unificados porque quaisquer sufixos subsequentes produzem o mesmo resultado?



Estados Equivalentes.

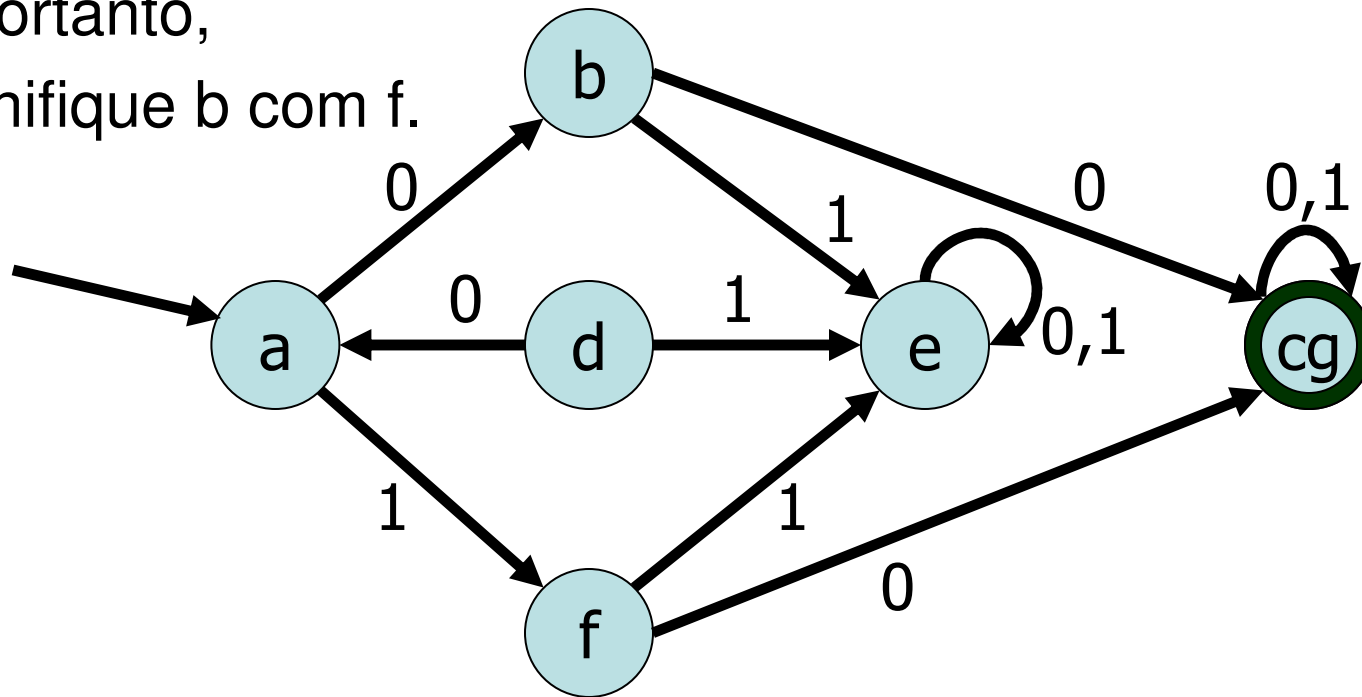
Exemplo

R: Sim, b e f. Note que se estamos em b ou f então:

1. se o string termina, é rejeitado em ambos os casos
2. se proxchar=0, aceita c/ qq sufixo em ambos os casos
3. Se proxchar=1, rejeita c/ qq sufixo em ambos os casos

Portanto,

unifique b com f.

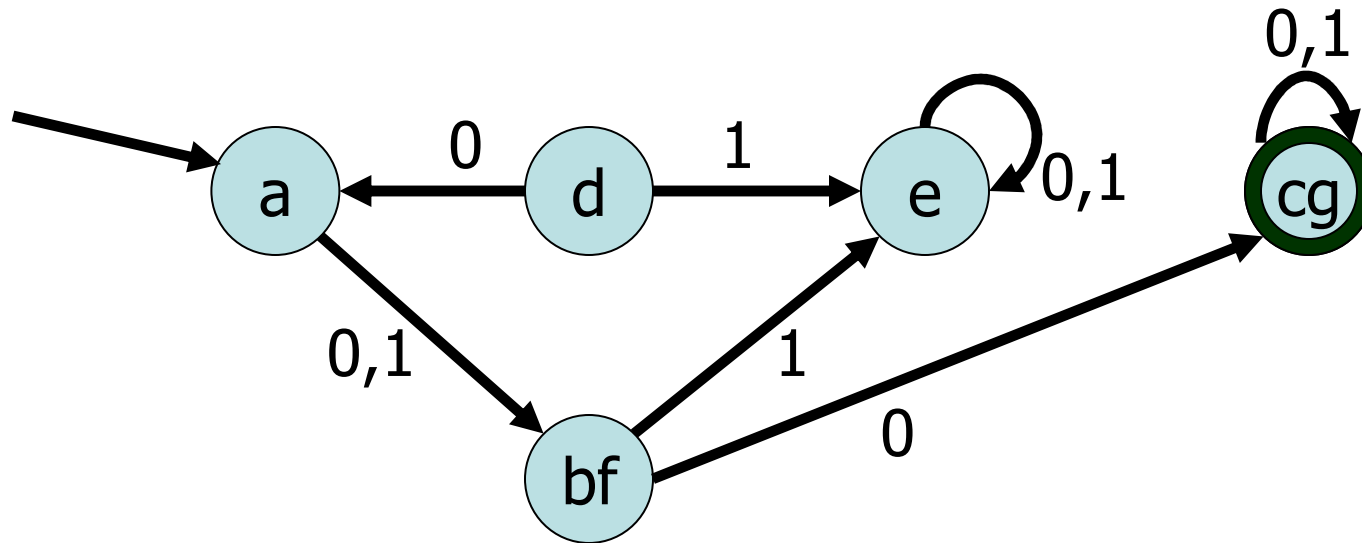


Estados Equivalentes.

Exemplo

Intuitivamente, dois estados são equivalentes se todos as computações subsequentes a partir deles são iguais.

Q: Dê uma caracterização formal de equivalência entre estados.



Estados Equivalentes.

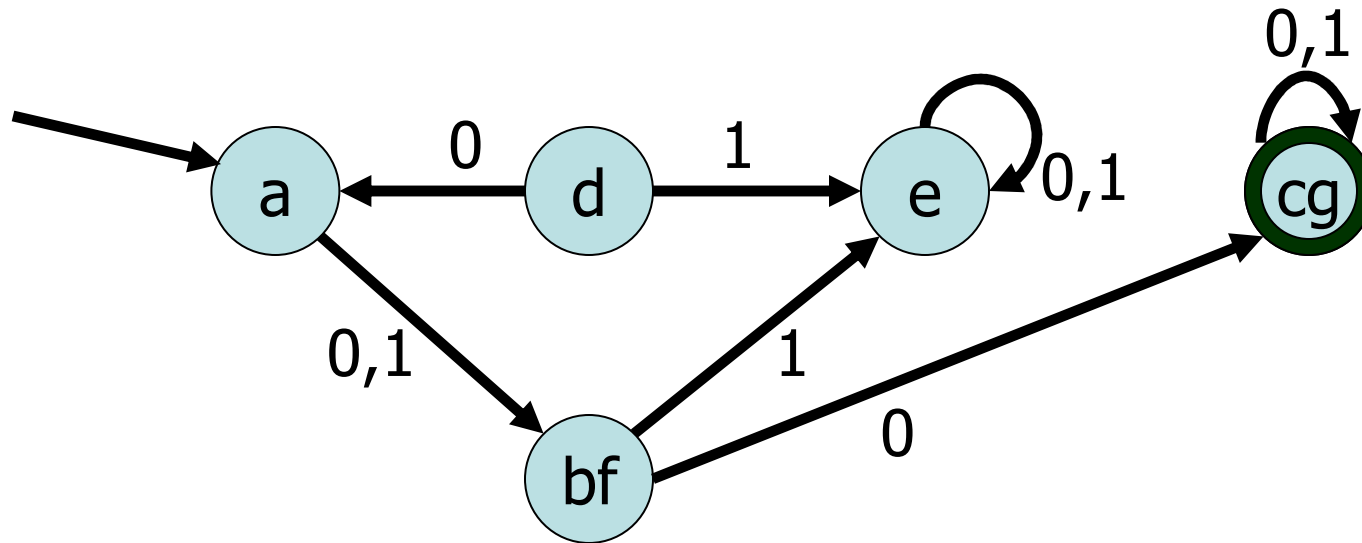
Exemplo

DEF: Dois estados q e q' em um AFD $M = (Q, \Sigma, \delta, q_0, F)$ são ***equivalentes*** (ou ***indistinguíveis***) se, para quaisquer strings $u \in \Sigma^*$, os estados a que u leva, quando lido a partir de q ou de q' são ambos de aceitação, ou ambos de rejeição.

Estados equivalentes podem ser unificados em um único, sem que isso afete o comportamento de M .

Concluindo o Exemplo

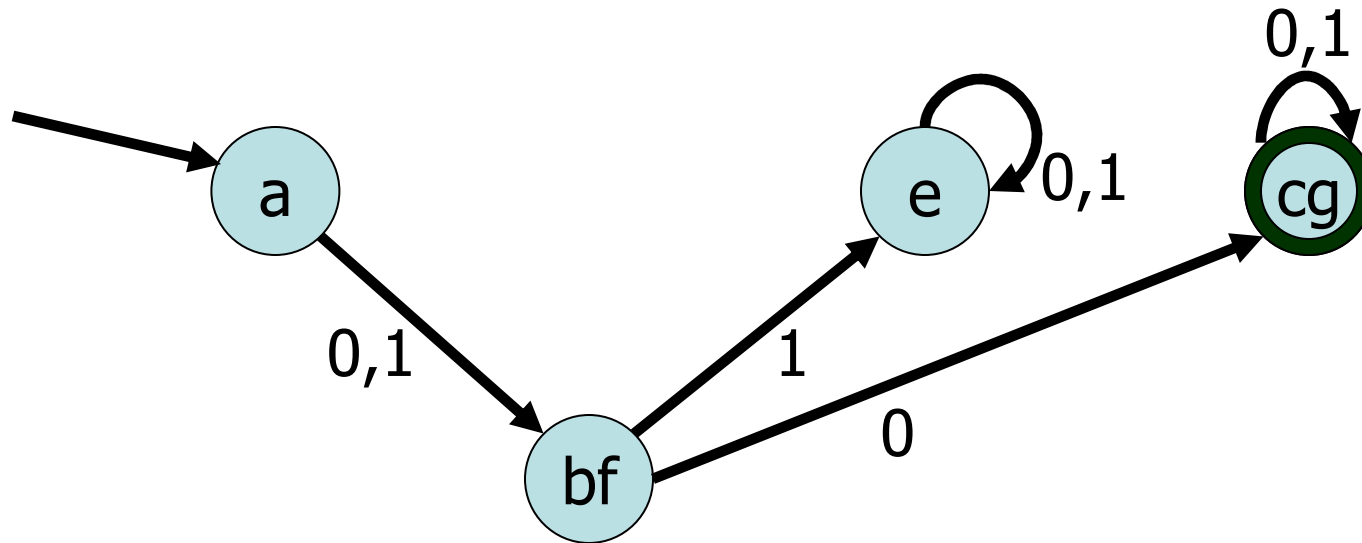
Q: Existem outras maneiras de simplificar o autômato abaixo?



Estados Inúteis

R: Sim: elimine o estado d.

A eliminação de ***estados inúteis***
(inatingíveis a partir do estado inicial)
não altera a linguagem aceita.



Algoritmo de Minimização.

DEF: Um autômato é ***irredutível*** se

- Não contém estados inúteis, e
- não contém estados distintos equivalentes.

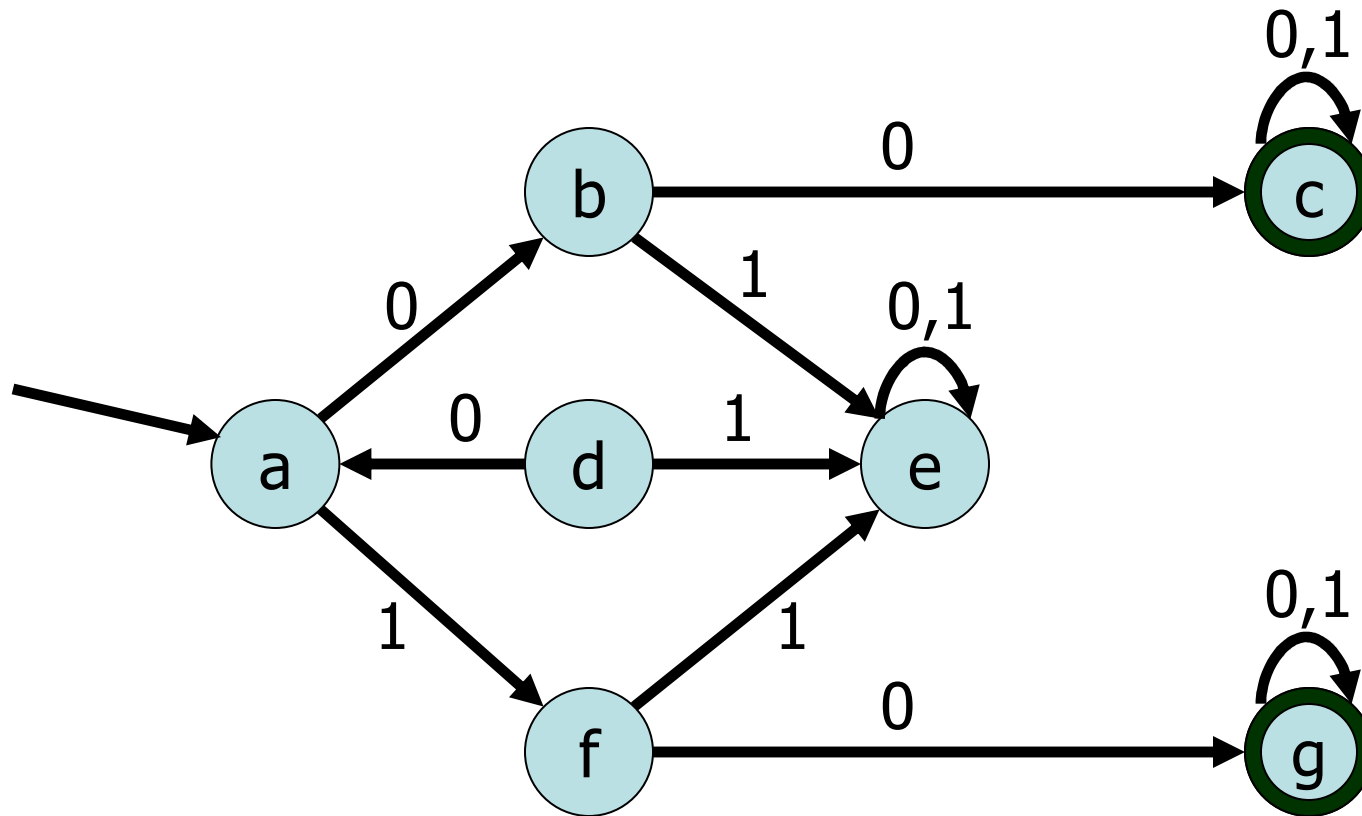
O objetivo do algoritmo de minimização é criar um autômato irredutível a partir de um autômato dado. Pode-se mostrar que esse algoritmo de fato produz o menor AFD possível equivalente ao AFD original. Portanto o nome “minimização”.

O algoritmo de minimização trabalha ao *inverso* do que vimos no exemplo anterior. Começa pelo menor número possível de estados e cria novos estados quando é forçado a isso. Vamos explicar com um jogo:

O Jogo MINIMIZAR

0. Todos os jogadores inúteis são eliminados.
1. O jogo procegue em rodadas.
2. Começa com 2 times: ACEITA vs. REJEITA.
3. Cada rodada consiste de sub-rodadas, uma para cada time.
4. Dois membros de um time **concordam** se, para um dado rótulo, passam o bastão para o mesmo time. Caso contrário, **discordam**.
5. Durante uma sub-rodada, membros que discordam são divididos em novos times maximais de membros concordantes entre si.
6. O jogo TERMINA quando se passa uma rodada sem que nenhum time seja dividido.

O Jogo MINIMIZAR



Algoritmo de Minimização. (Refinamento da Partição)

```
AFD minimize(AFD  $(Q, \Sigma, \delta, q_0, F)$  )  
  remova qq estado  $q$  não atingível a partir de  $q_0$   
  Partition  $P = \{F, Q - F\}$   
  boolean Consistent = false  
  while( Consistent == false )  
    Consistent = true  
    for(every Set  $S \in P$ , char  $a \in \Sigma$ , Set  $T \in P$  )  
      Set temp =  $\{q \in T \mid \delta(q, a) \in S\}$   
      if (temp  $\neq \emptyset$  && temp  $\neq T$  )  
        Consistent = false  
         $P = (P - T) \cup \{\text{temp}, T - \text{temp}\}$   
  return defineMinimizor(  $(Q, \Sigma, \delta, q_0, F)$ ,  $P$  )
```

Algoritmo de Minimização. (Refinamento da Partição)

AFD defineMinimizador(AFD $(Q, \Sigma, \delta, q_0, F)$, Partition P)

Set $Q' = P$

State q'_0 = the set in P which contains q_0

$F' = \{ S \in P \mid S \subseteq F \}$

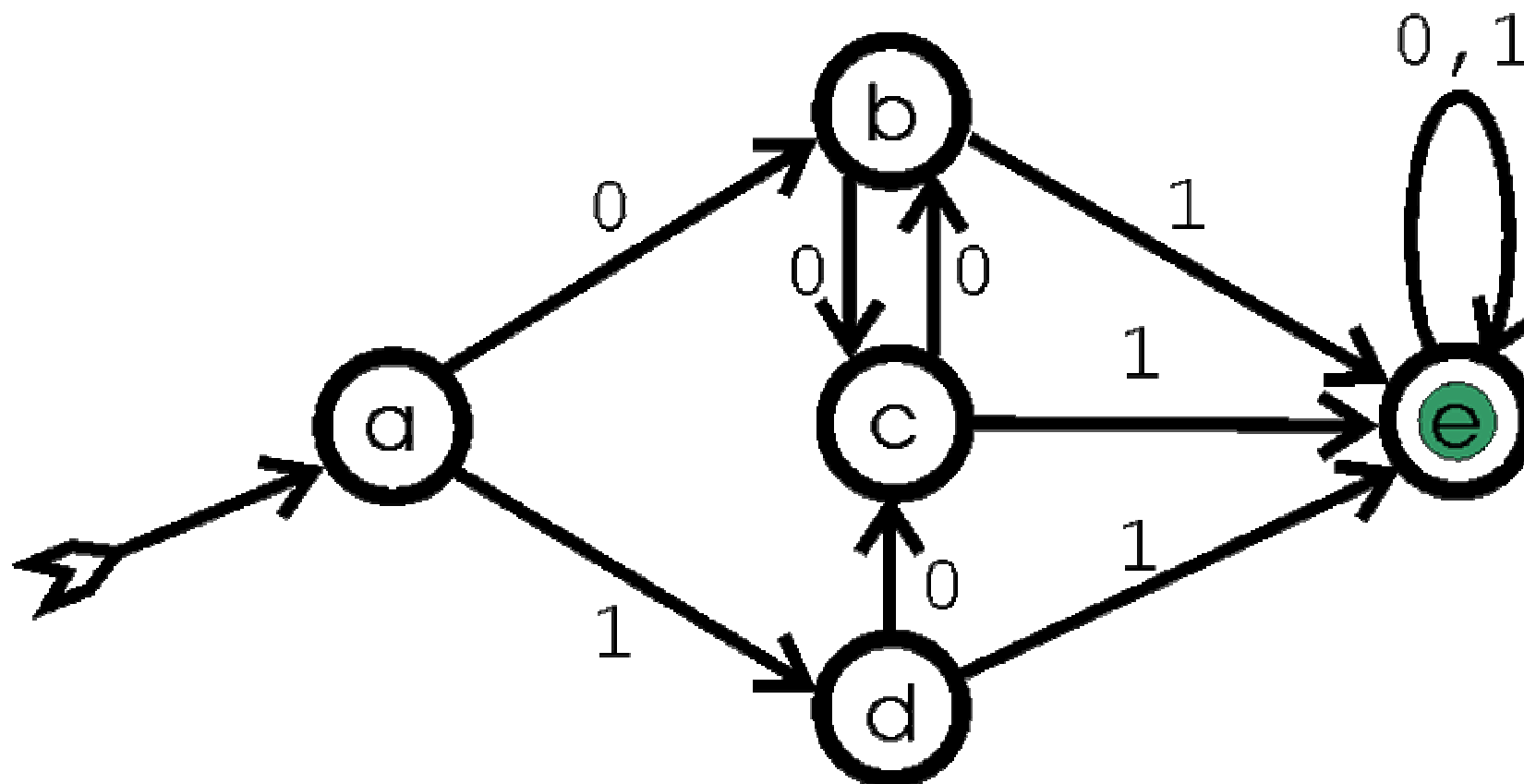
for (each $S \in P, a \in \Sigma$)

define $\delta'(S, a)$ = the set $T \in P$ which contains
the states $\delta(S, a)$

return $(Q', \Sigma, \delta', q'_0, F')$

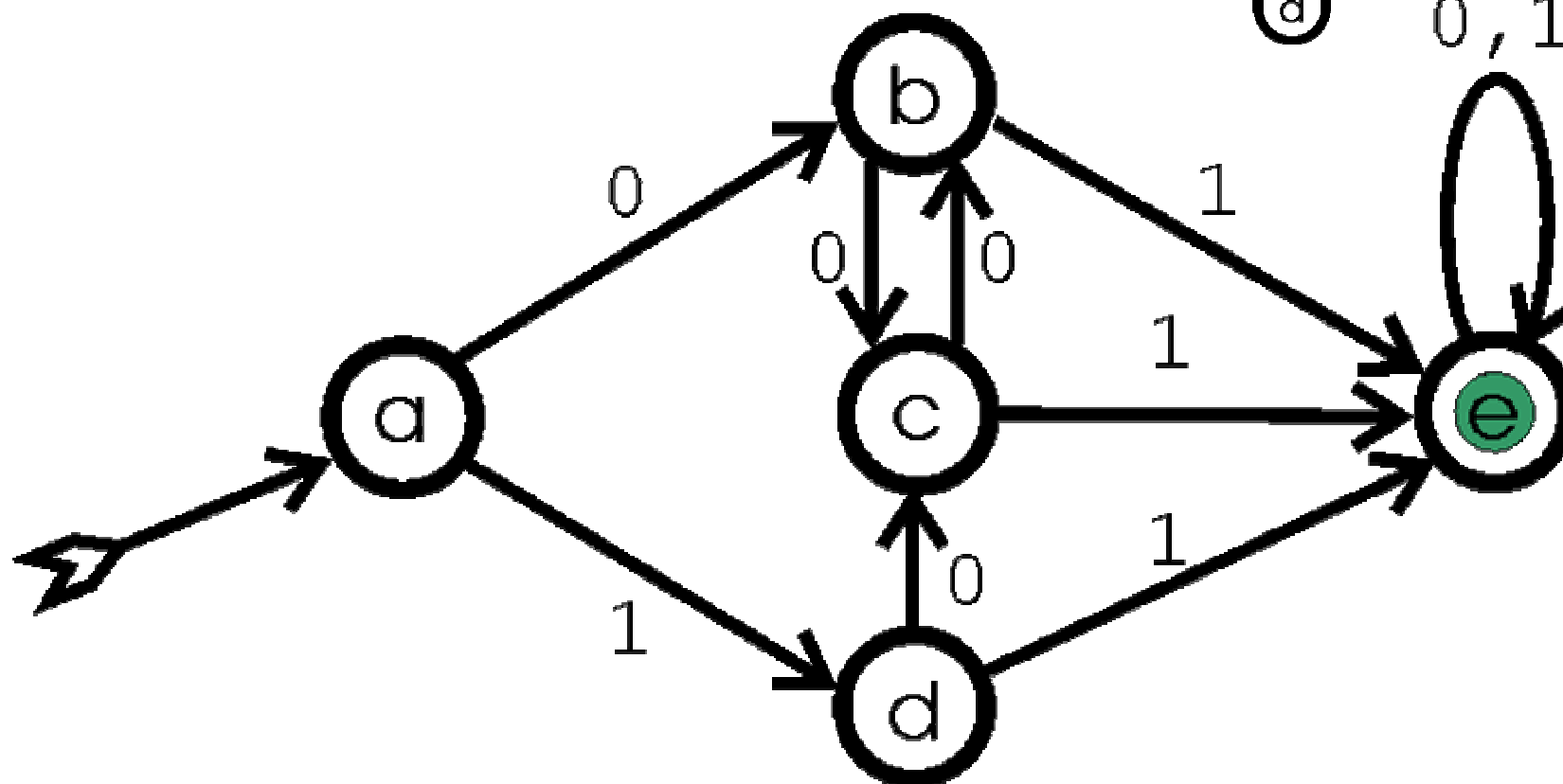
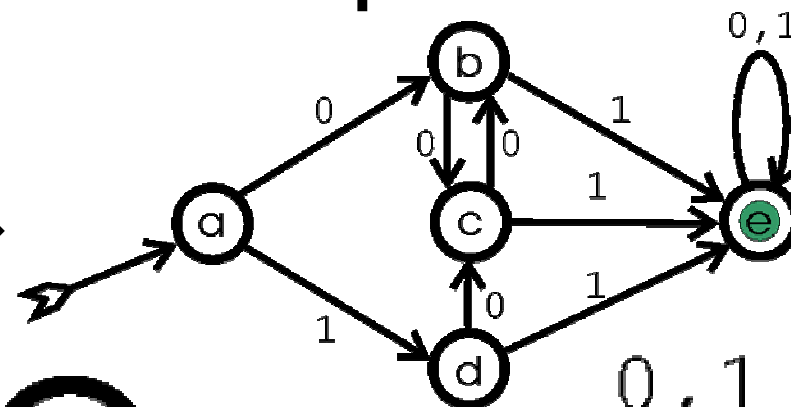
Minimização: Exemplo

Considere o AFD



Minimização: Exemplo

Versão miniatura →



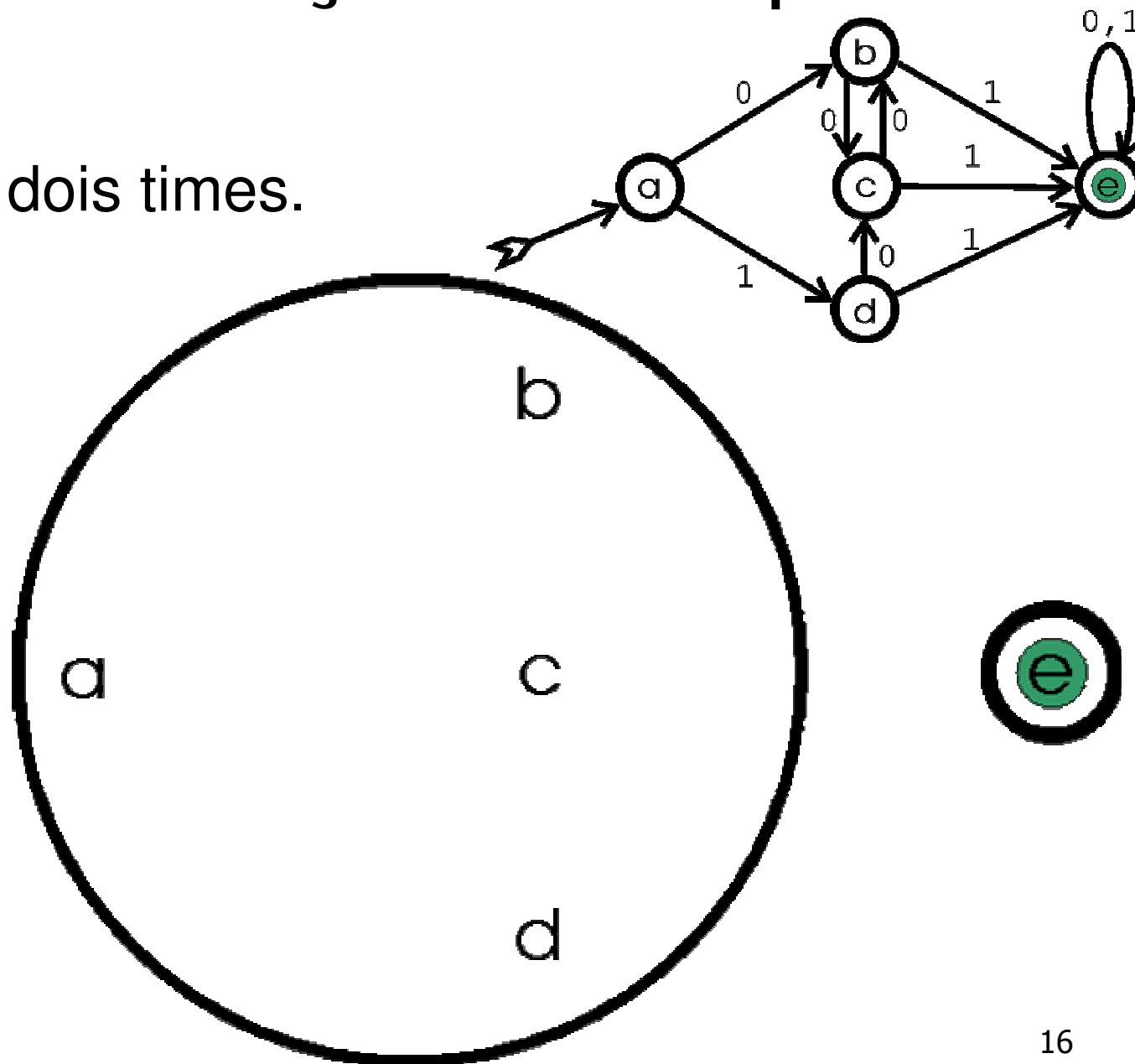
Minimização: Exemplo

Divida em dois times.

ACEITA

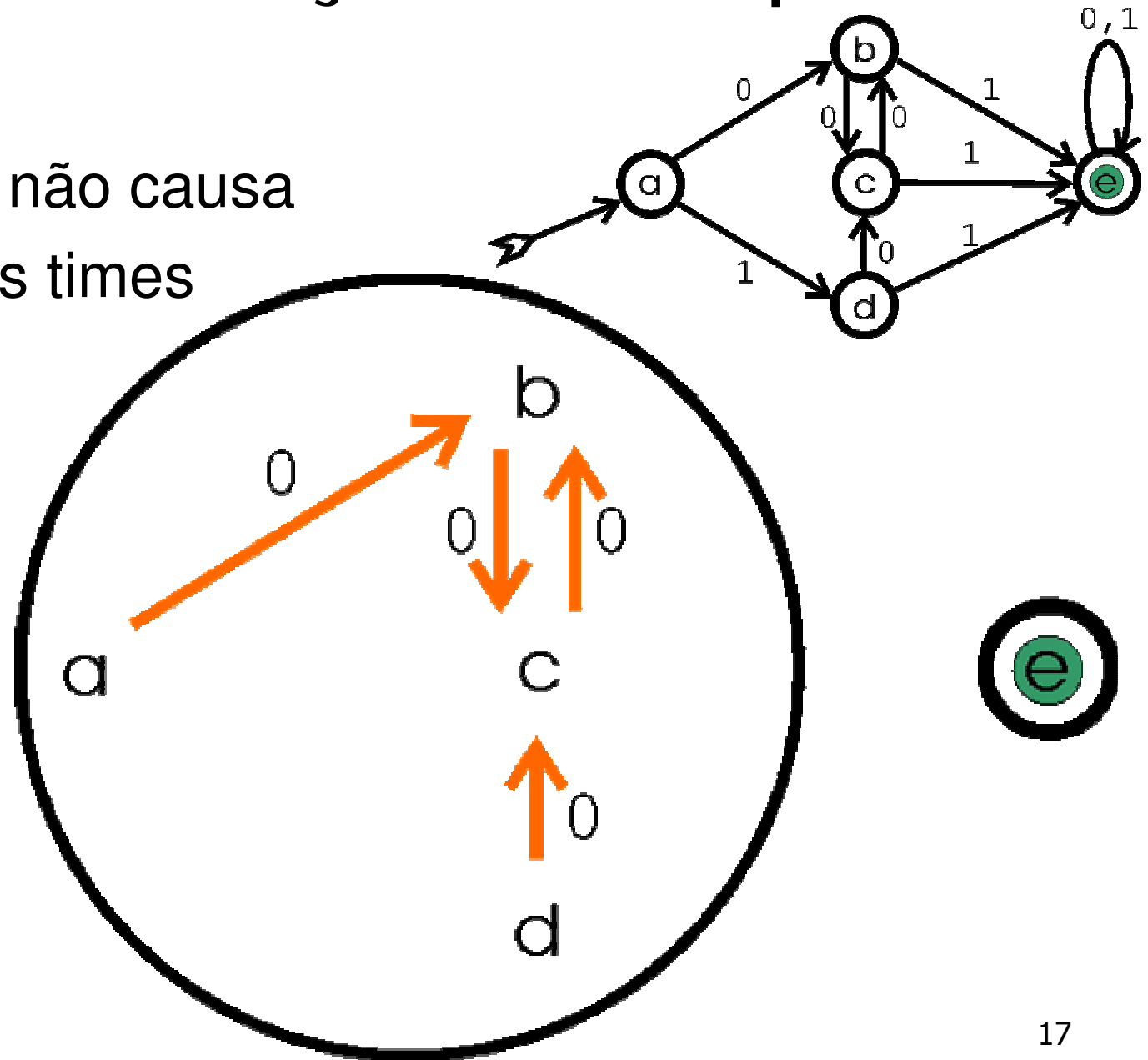
vs.

REJEITA



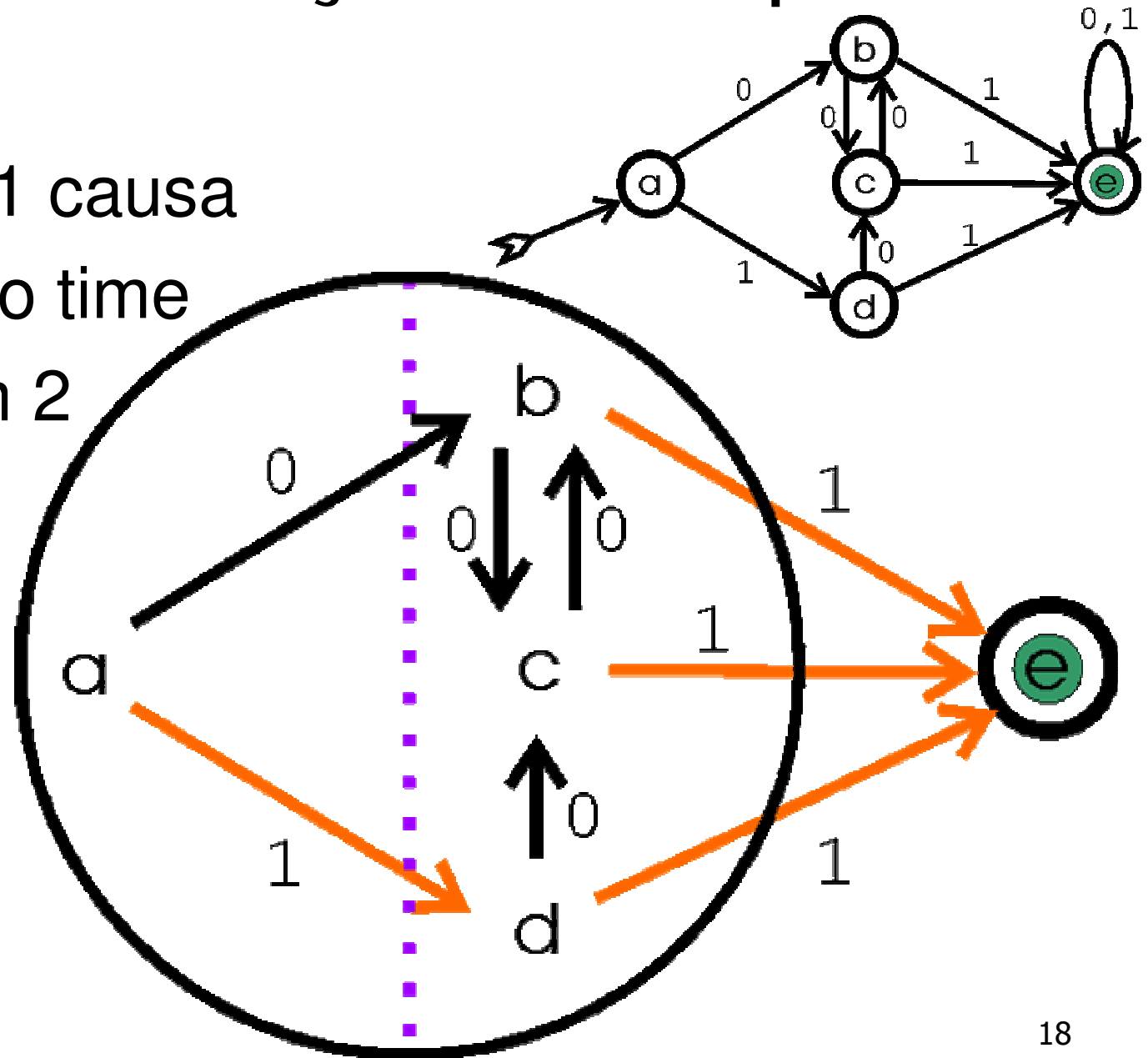
Minimização: Exemplo

O rótulo 0 não causa
divisão nos times



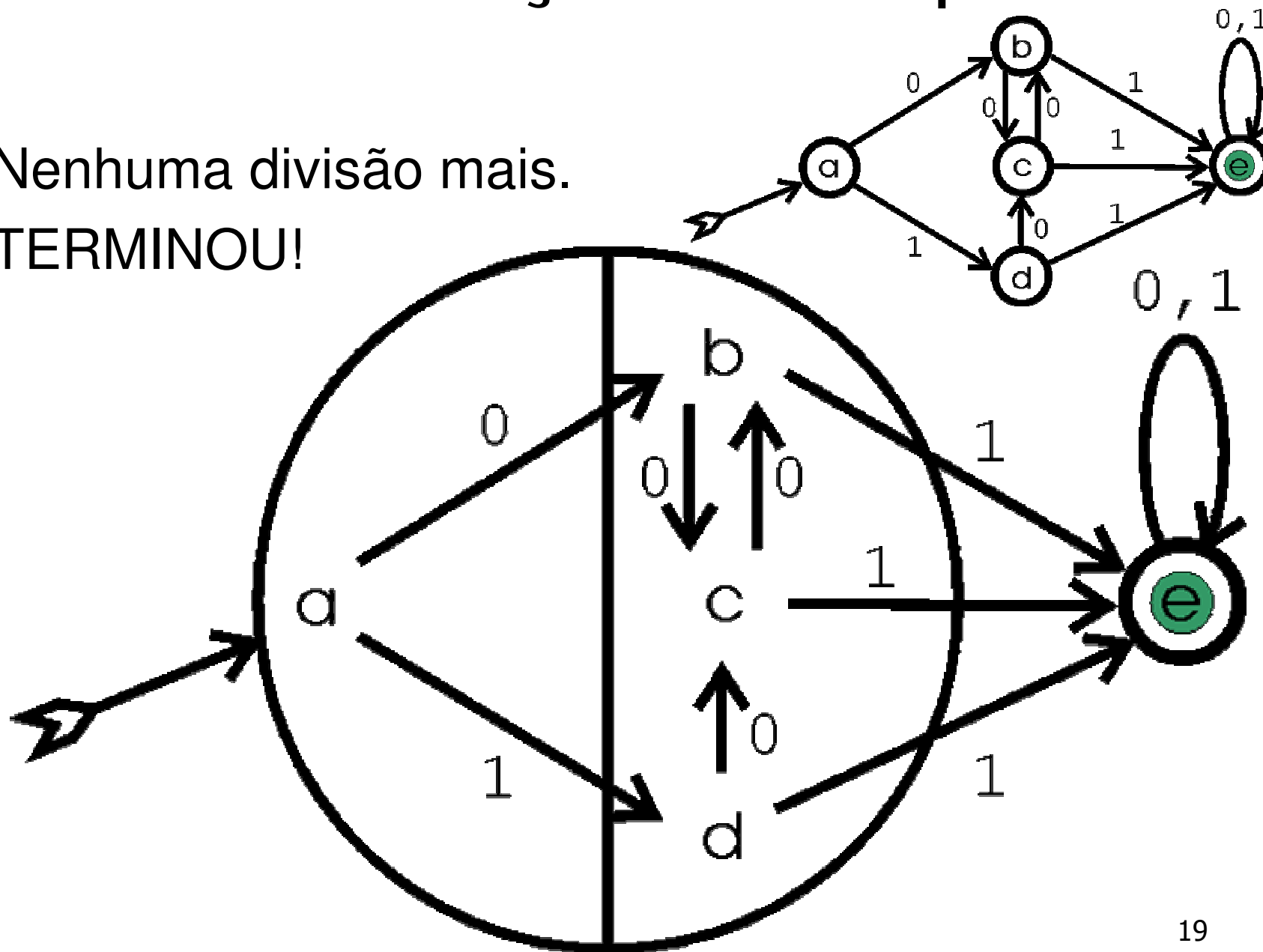
Minimização: Exemplo

O rótulo 1 causa
divisão do time
maior em 2



Minimização: Exemplo

Nenhuma divisão mais.
TERMINOU!

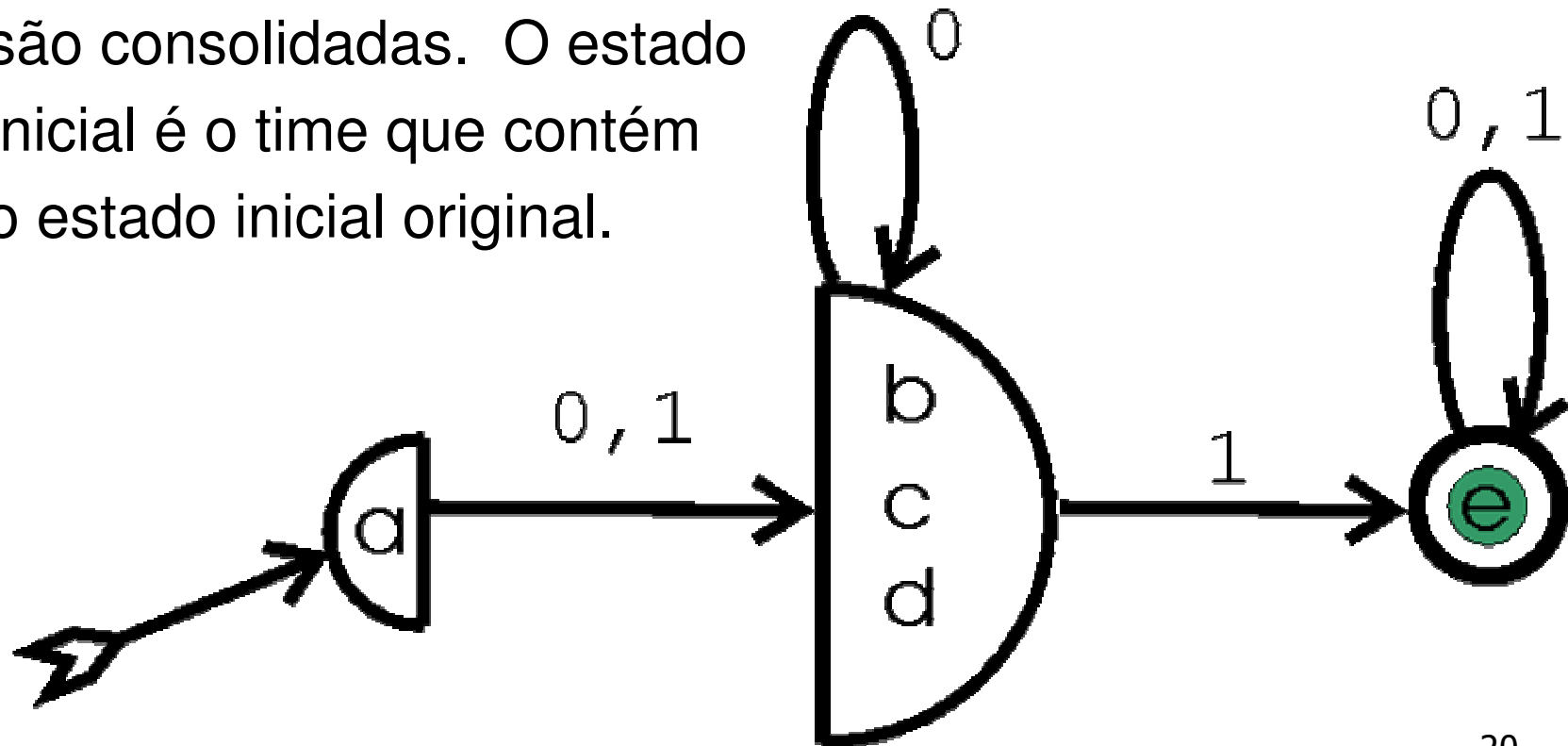
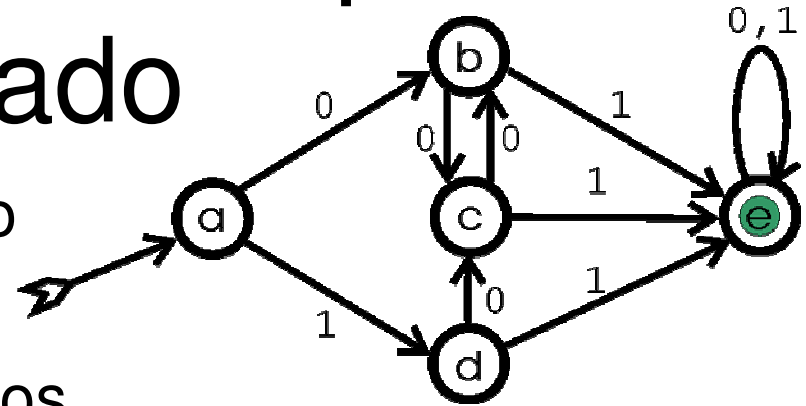


Minimização: Exemplo.

Resultado

Os estados do autômato mínimo são os times resultantes.

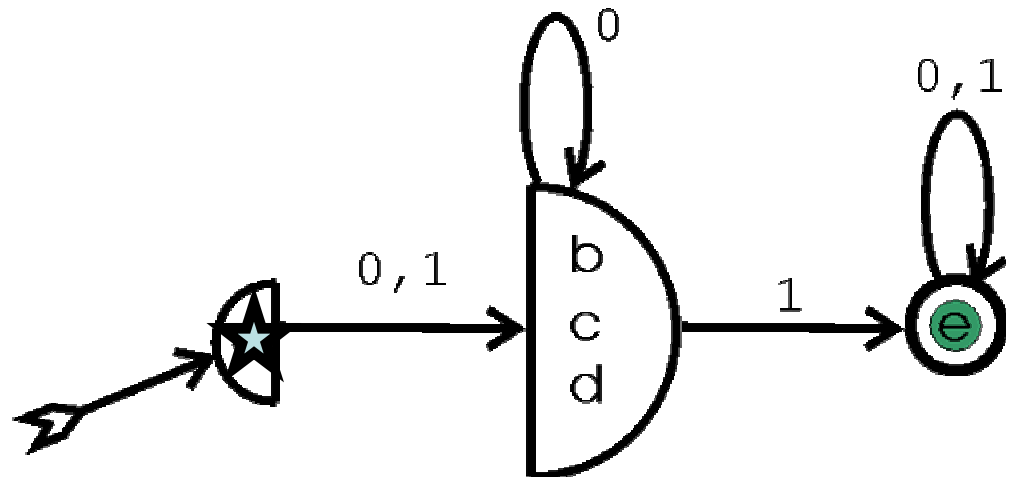
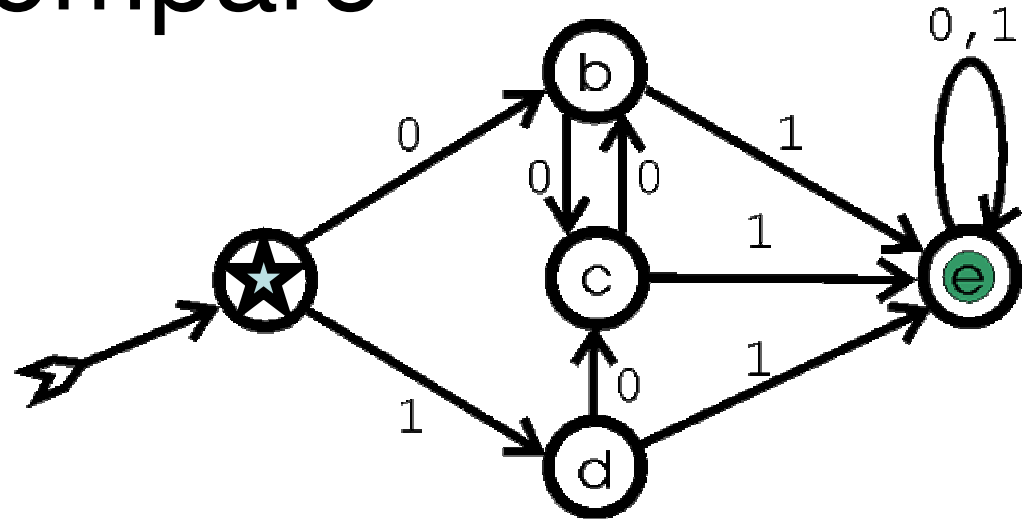
As transições entre esses estados são consolidadas. O estado inicial é o time que contém o estado inicial original.



Minimização: Exemplo.

Compare

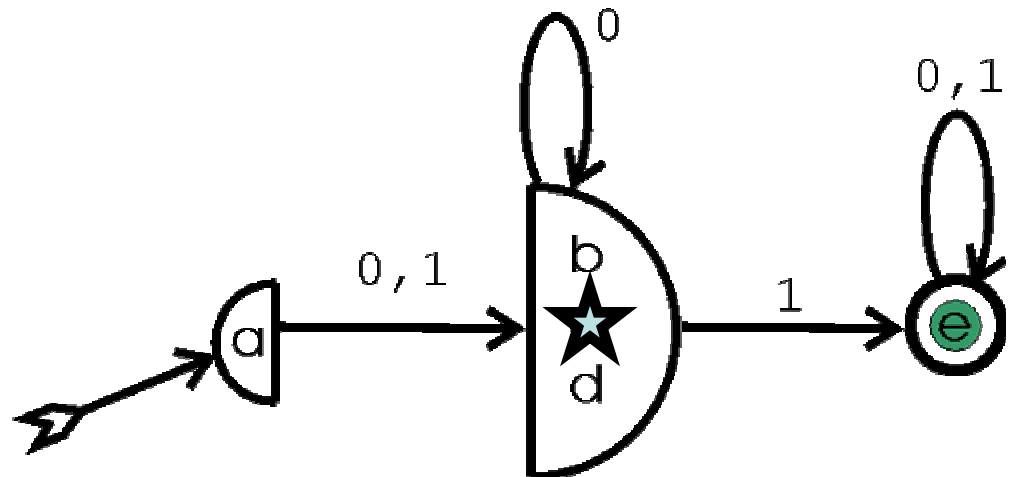
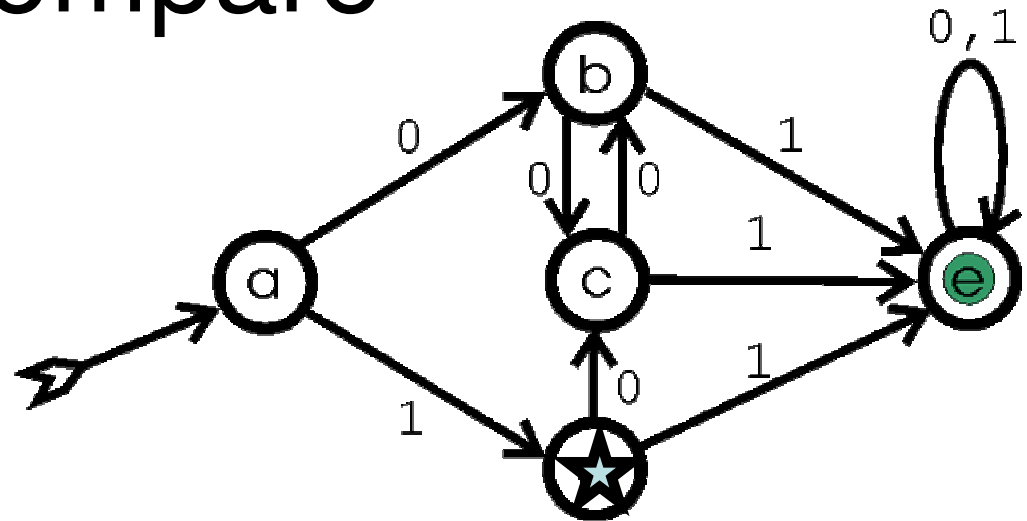
↑ 100100101



Minimização: Exemplo.

Compare

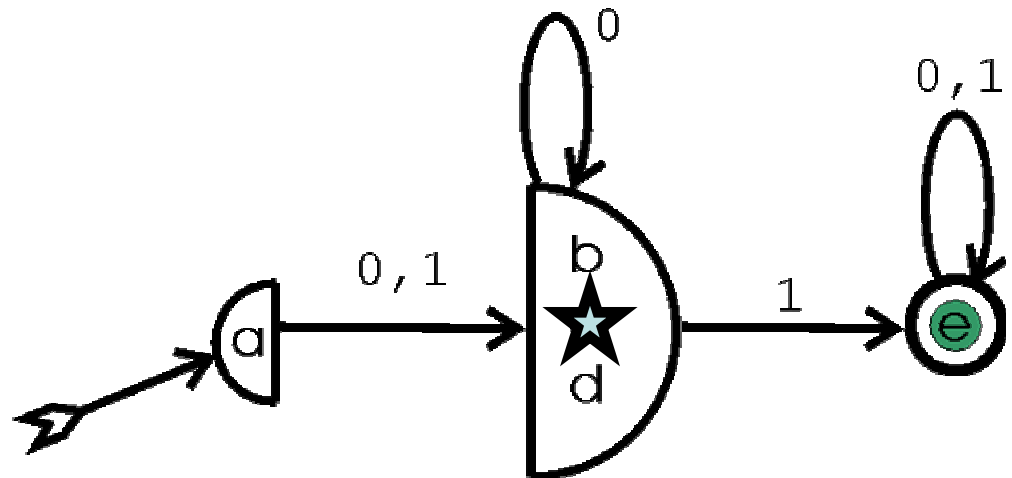
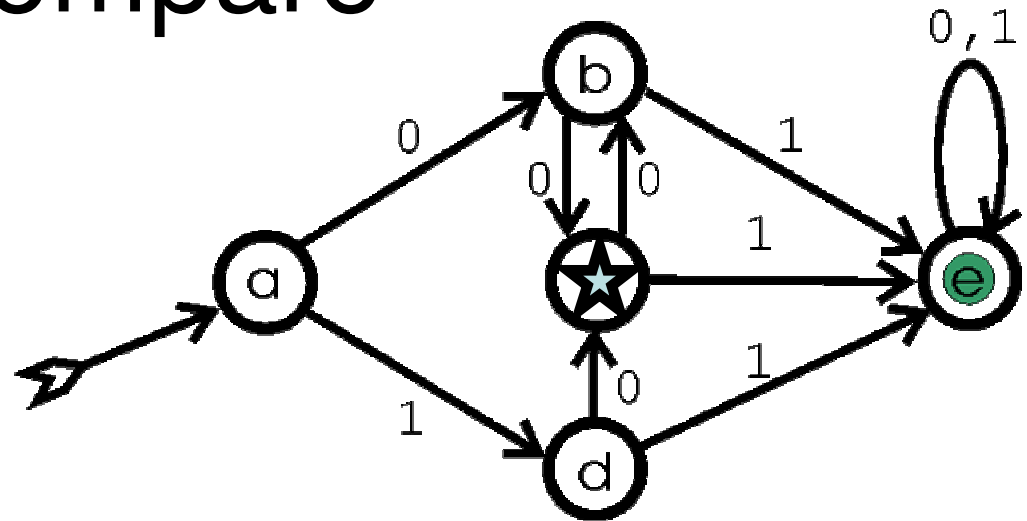
100100101
↑



Minimização: Exemplo.

Compare

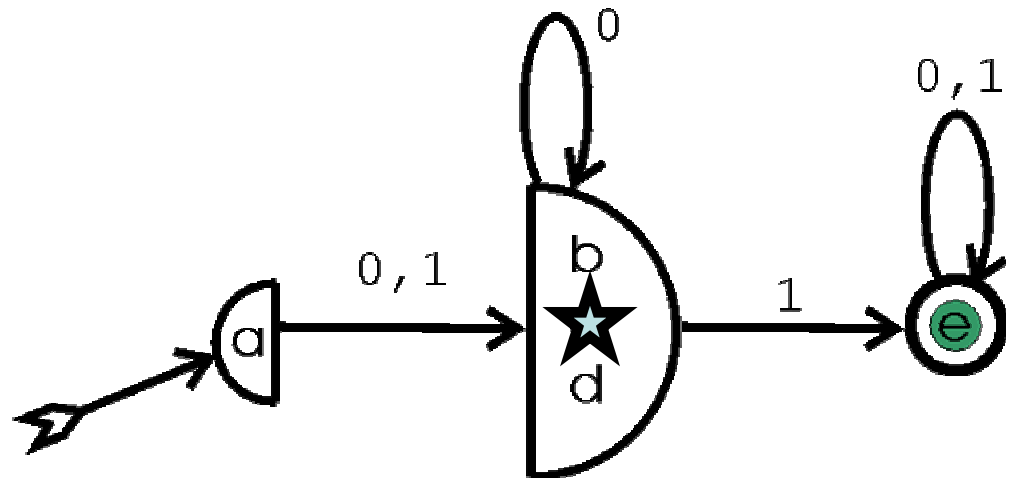
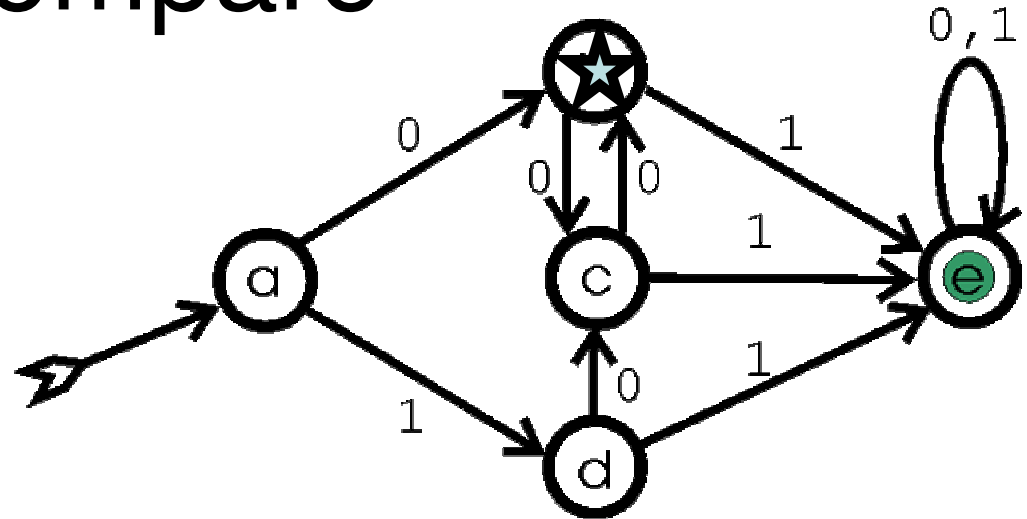
100100101
↑



Minimização: Exemplo.

Compare

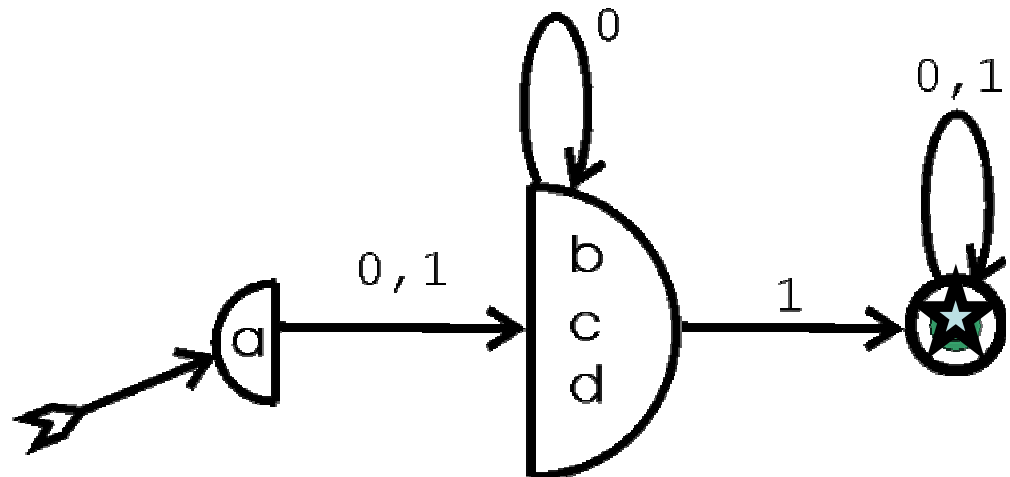
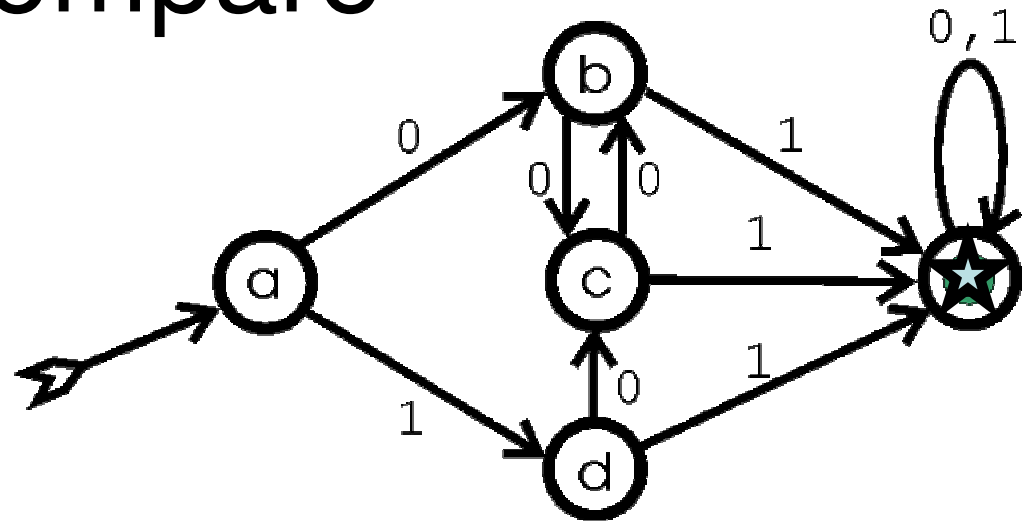
100100101
↑



Minimização: Exemplo.

Compare

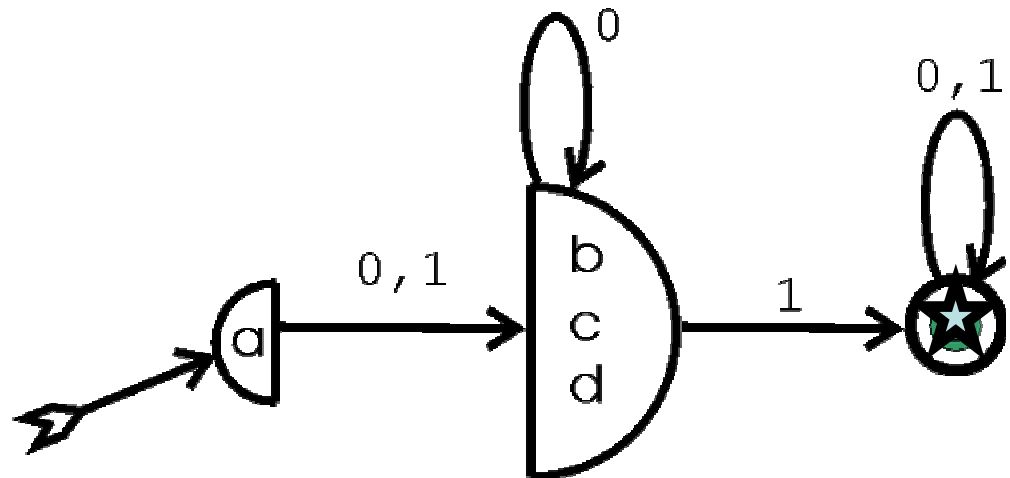
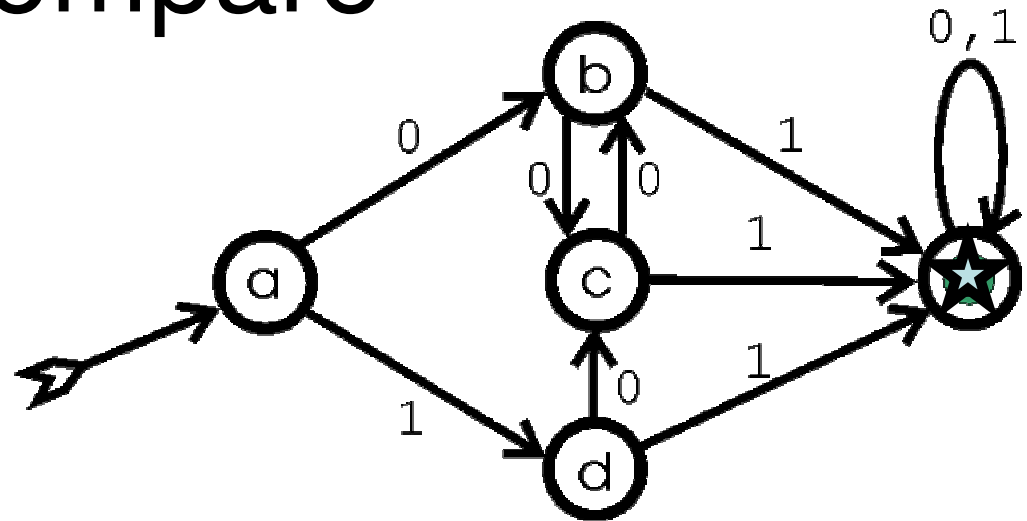
100100101
↑



Minimização: Exemplo.

Compare

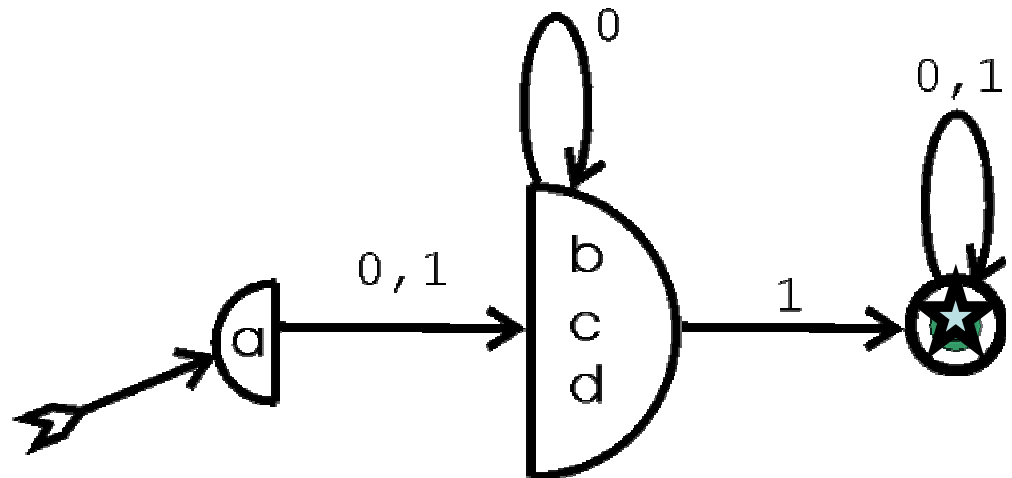
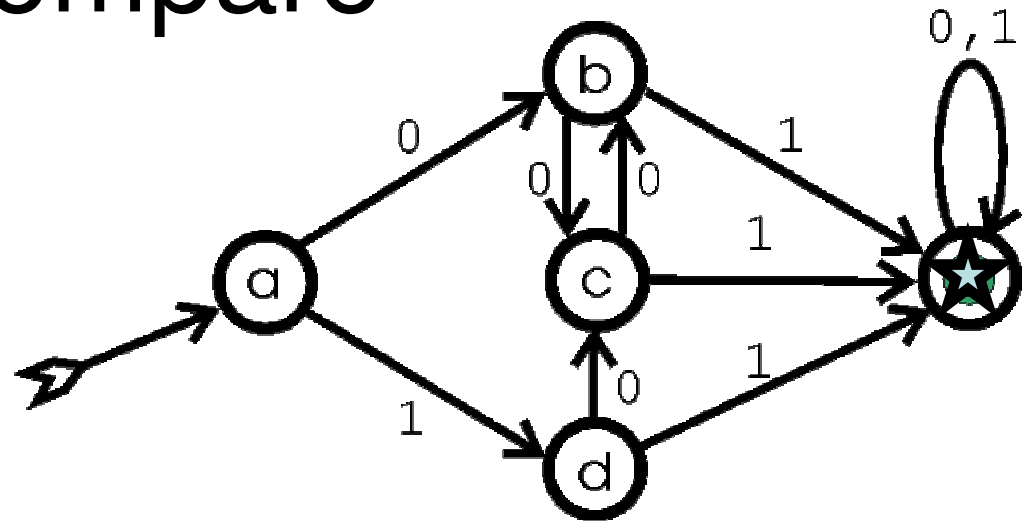
100100101
↑



Minimização: Exemplo.

Compare

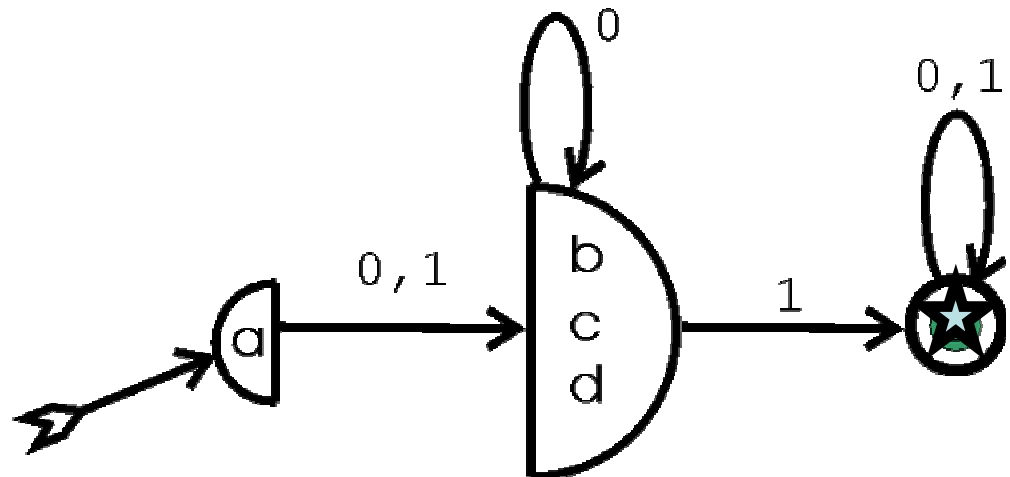
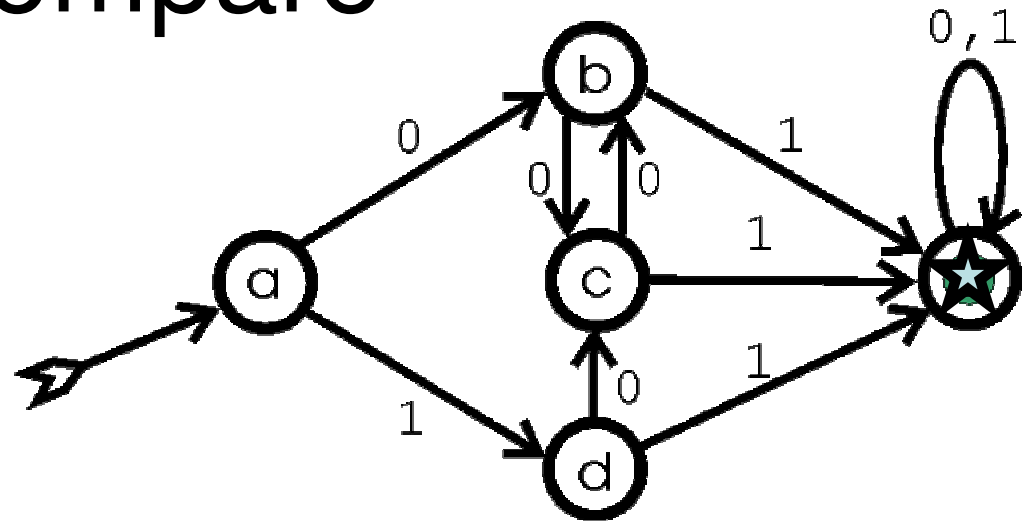
100100101
↑



Minimização: Exemplo.

Compare

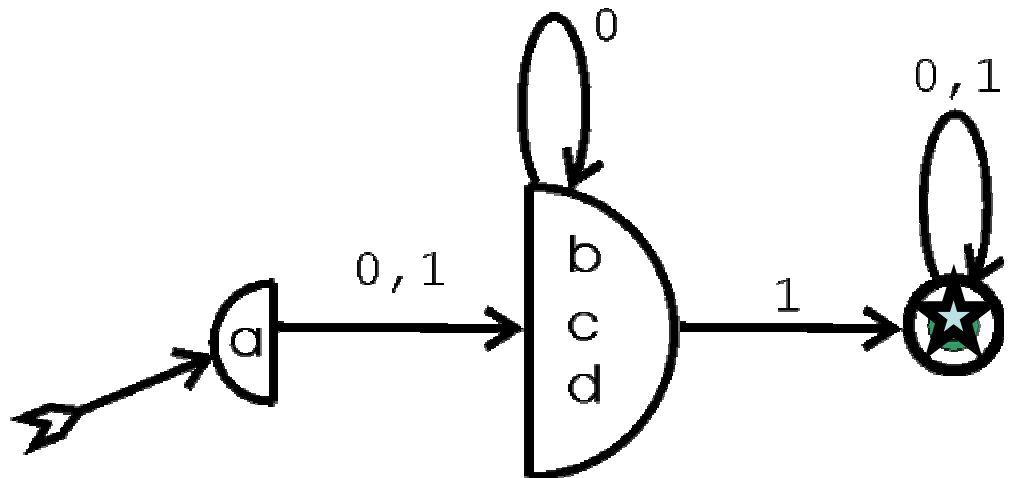
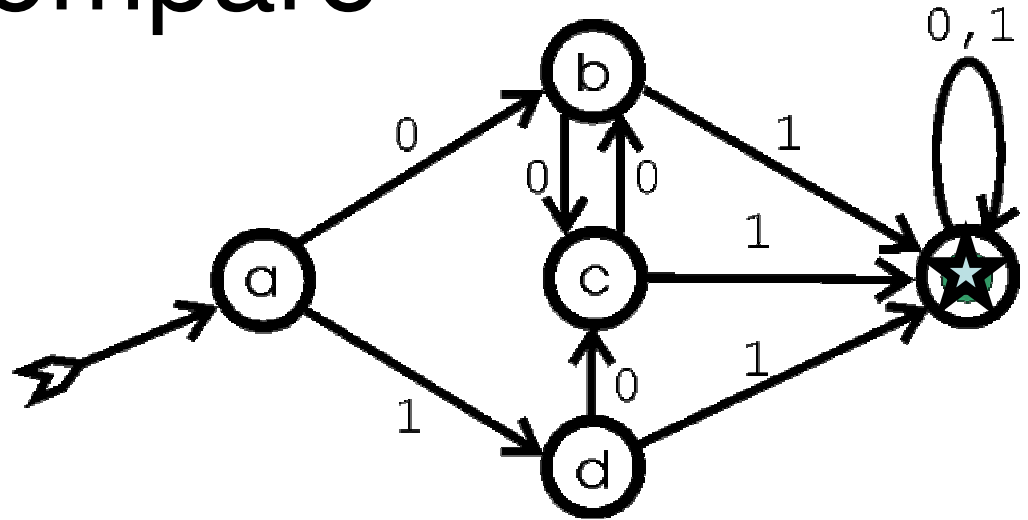
100100101
↑



Minimização: Exemplo.

Compare

100100101
↑

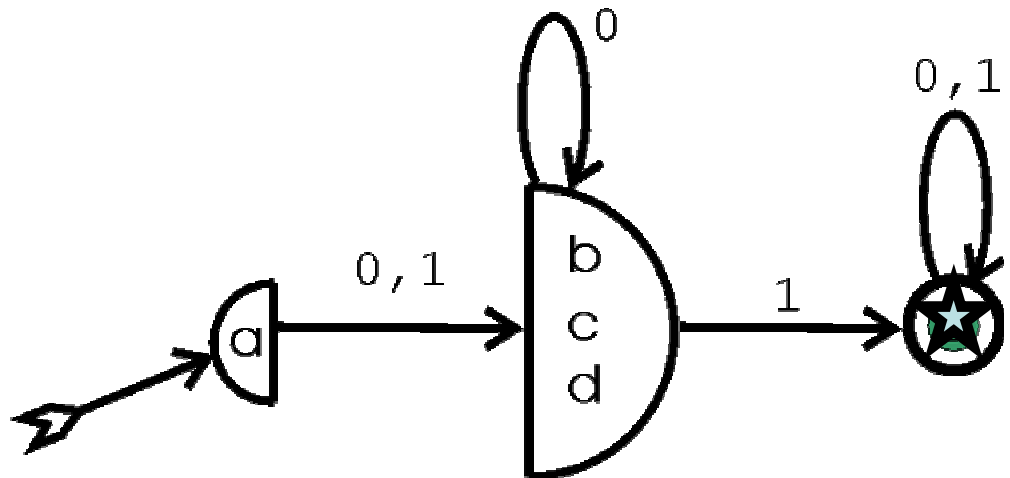
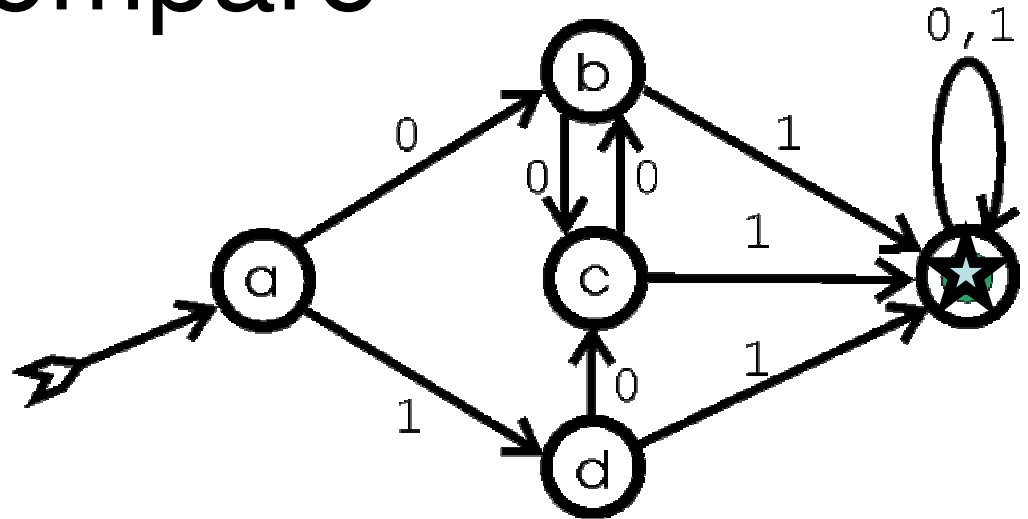


Minimização: Exemplo.

Compare

100100101↑

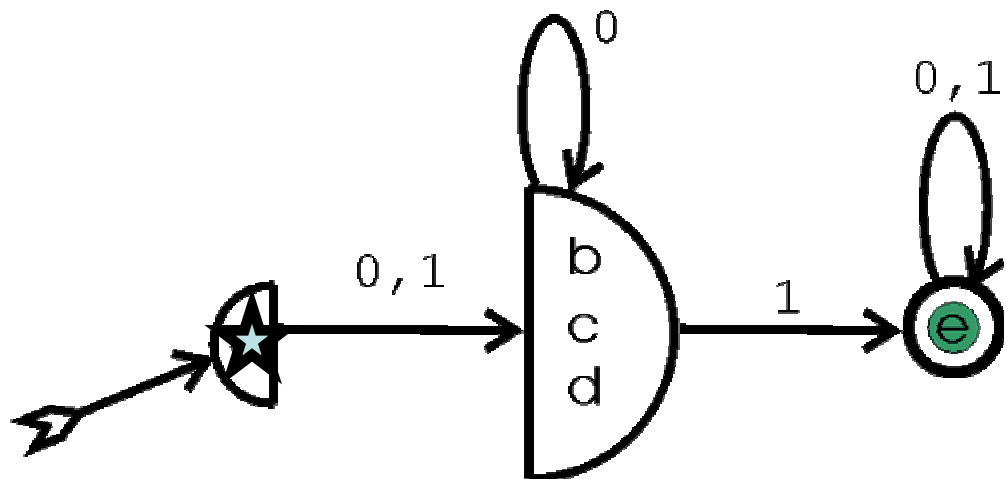
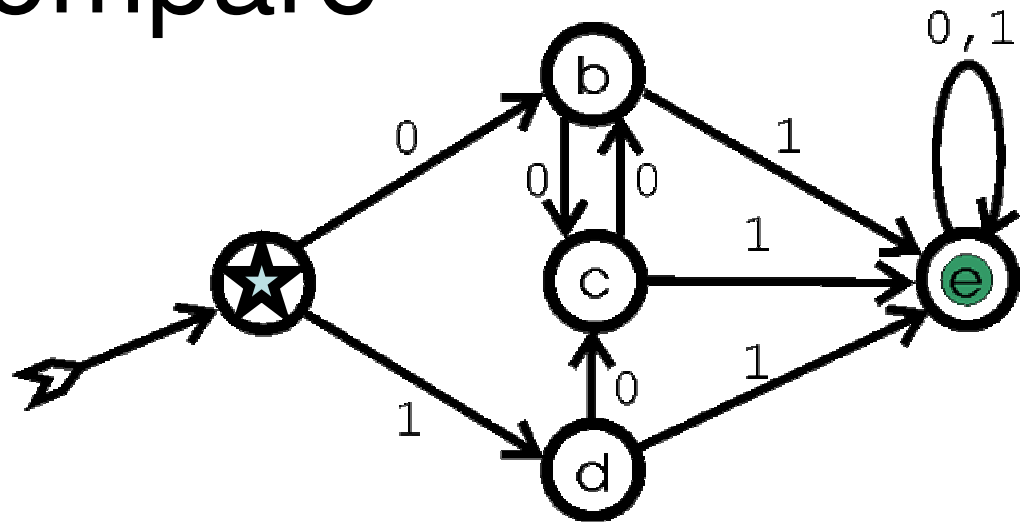
ACEITA.



Minimização: Exemplo.

Compare

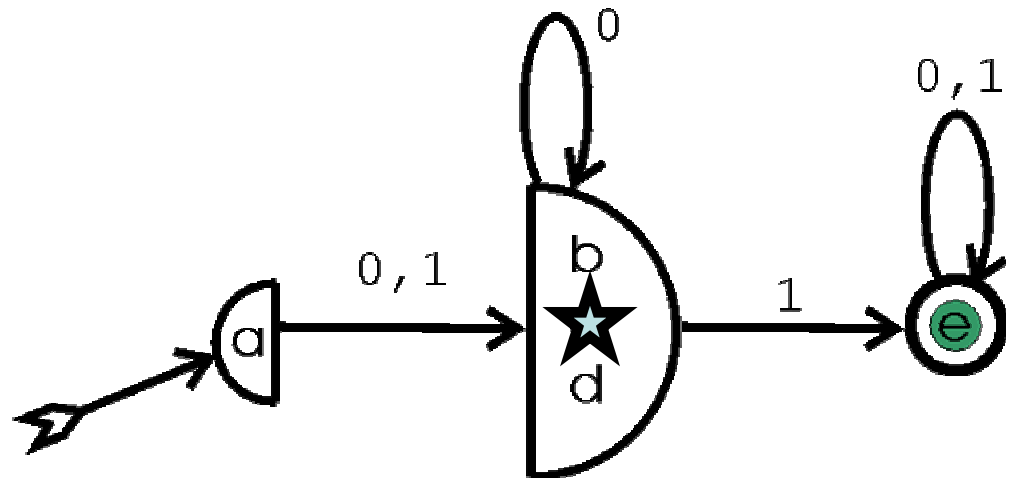
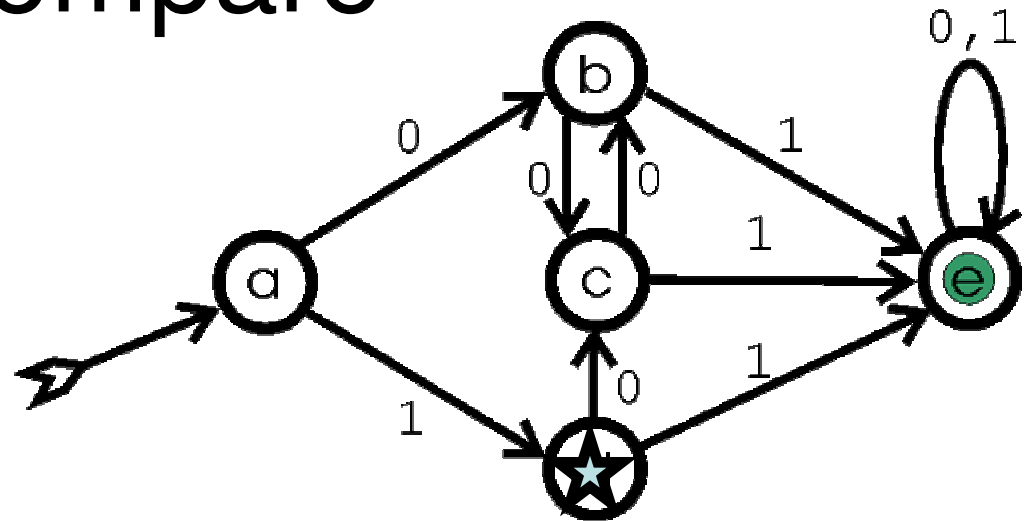
↑ 10000



Minimização: Exemplo.

Compare

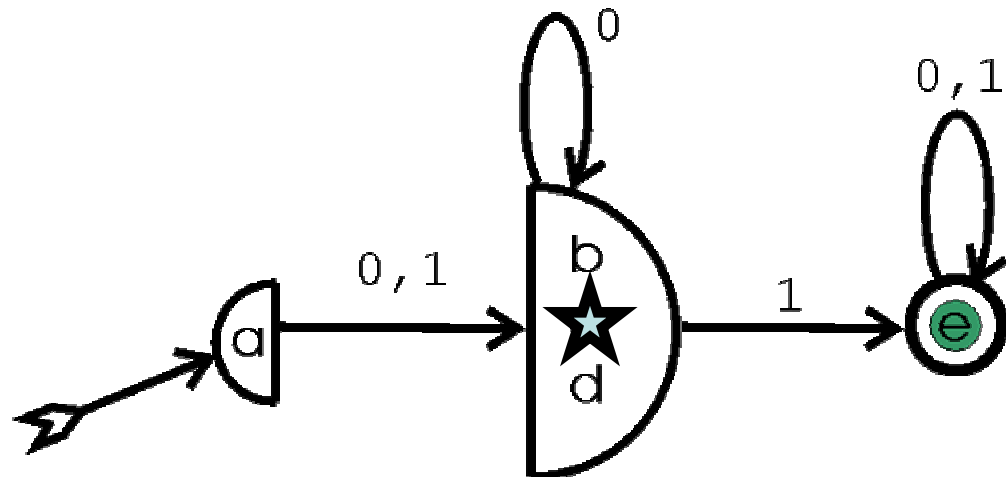
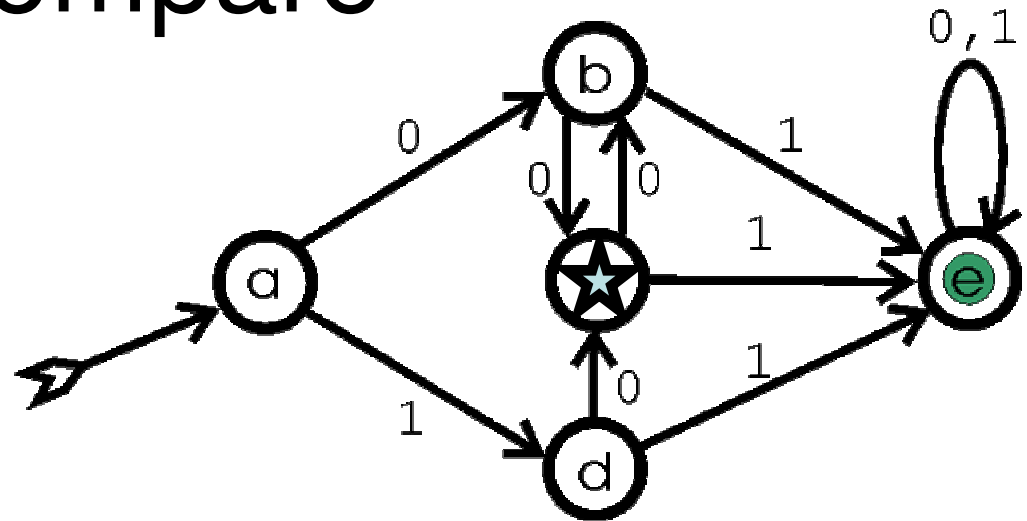
10000
↑



Minimização: Exemplo.

Compare

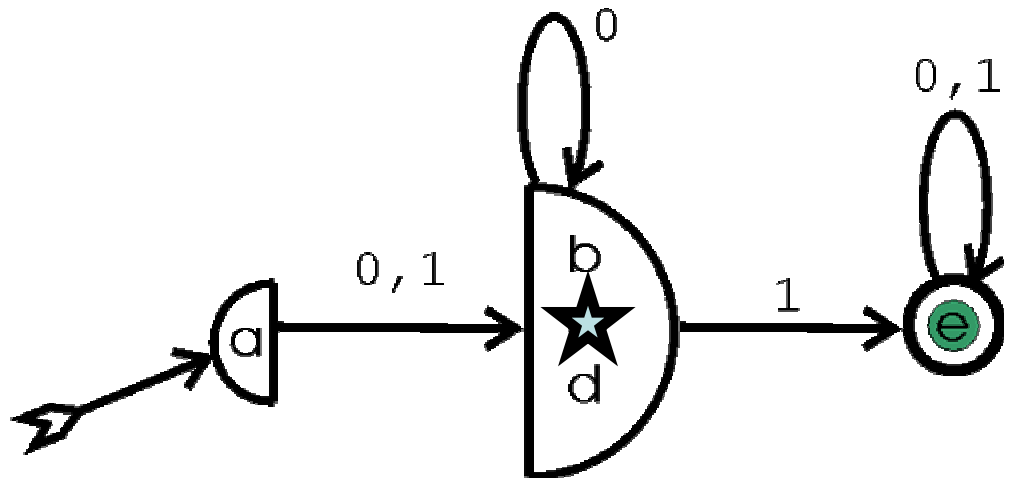
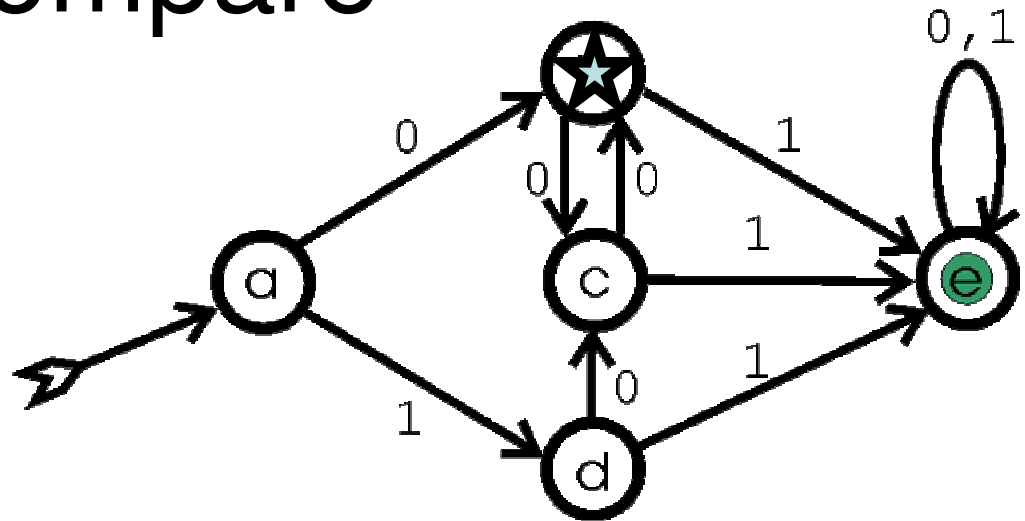
10000
↑



Minimização: Exemplo.

Compare

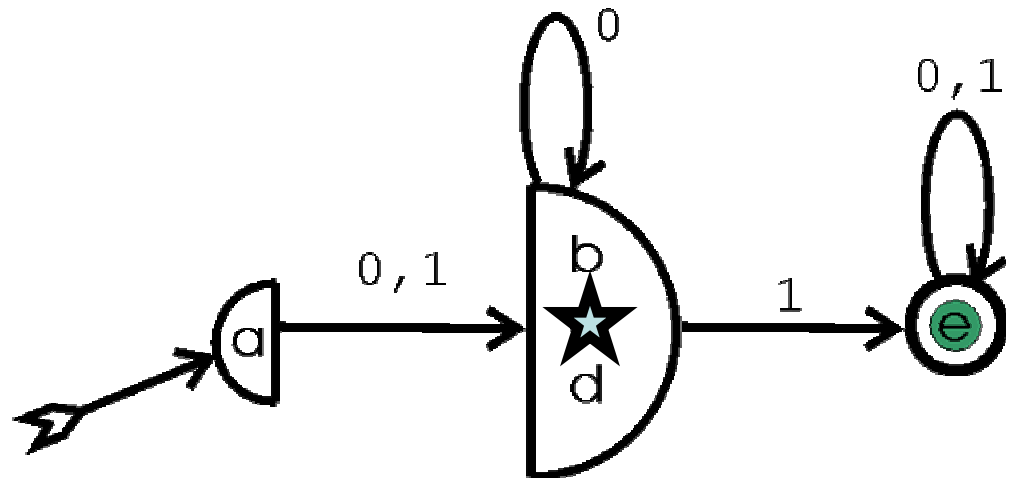
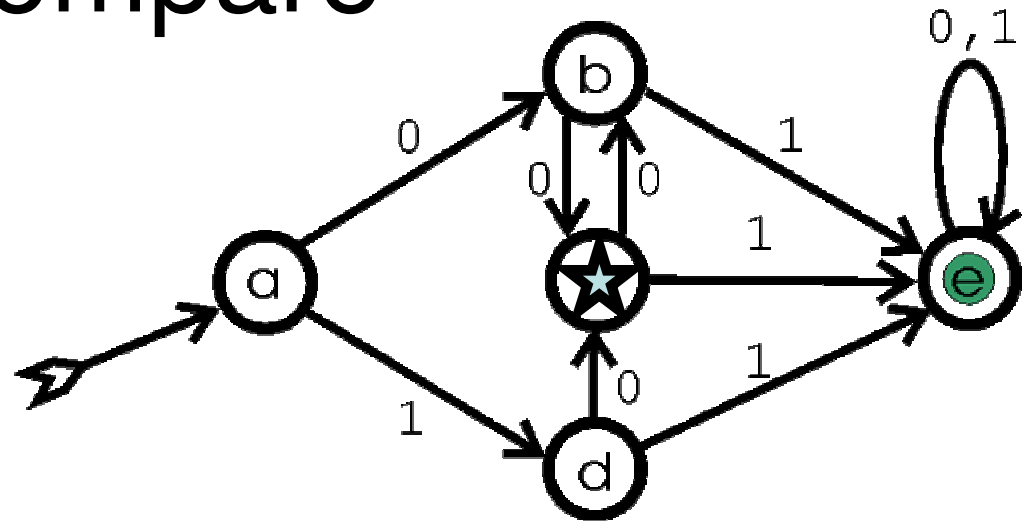
10000
↑



Minimização: Exemplo.

Compare

10000

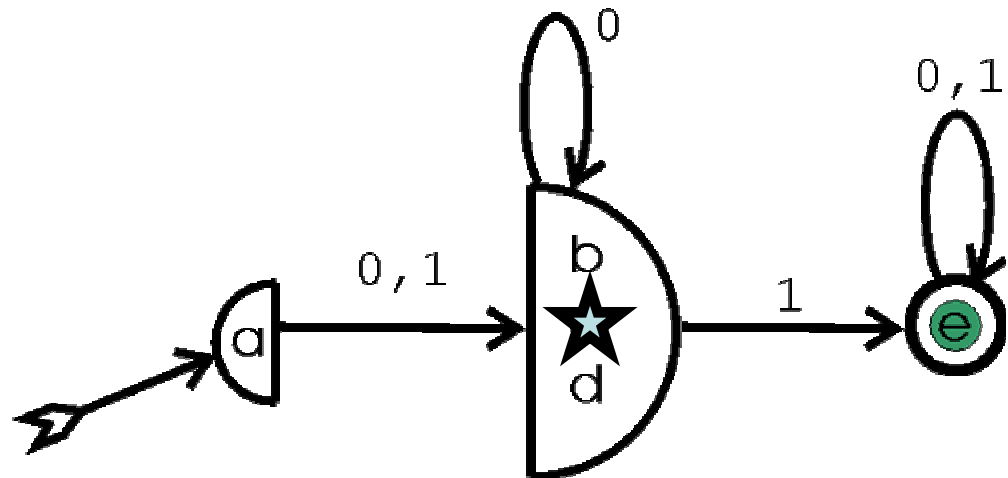
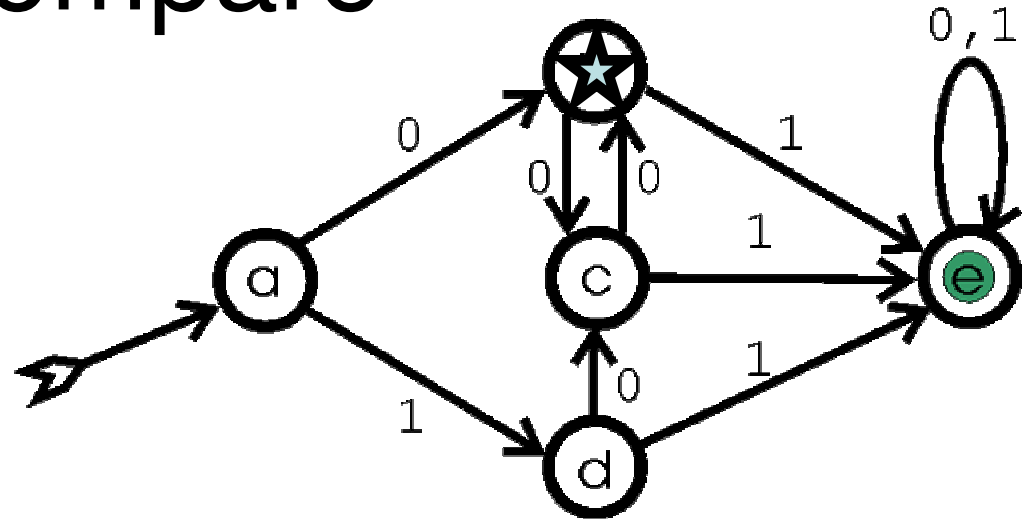


Minimização: Exemplo.

Compare

10000

REJEITA.



Exercício

- Sobre o alfabeto $\{0,1\}$, forneça o AFD mínimo de:
 - Linguagem dos strings binários que representam números decimais divisíveis por 6.
 - Linguagem dos strings binários que possuem tamanho par e não possuem 00.

Minimização: Prova

O algoritmo apresentado produz um AF irreduzível. Porque esse AF seria o menor AF possível que reconhece a linguagem aceita pelo AF original?

Questão análoga em cálculo: Por que um mínimo local seria um mínimo global?
Nem sempre é o caso!

Minimização: Prova

THM (Myhill-Nerode): O algoritmo de minimização produz um autômato mínimo equivalente.

Proof. Vamos mostrar que qq autômato irreduzível é mínimo para a linguagem L que ele aceita:

Dizemos que strings $u, v \in \Sigma^*$ são **indistinguíveis** se para todo sufixo x , $ux \in L$ sse $vx \in L$.

Note que se u e v são distinguíveis, os caminhos, a partir do estado inicial, rotulados por u e por v devem terminar em estados diferentes.

Minimização: Prova

Consequentemente, o número de estados em um AFD para L deve ser pelo menos igual ao ao número de strings mutuamente distinguíveis de L .

Mas um AFD irreduzível tem a propriedade de que todo estado dá origem a outro string mutuamente distinguível do anterior!

Portanto, qq outro AFD para L deve ter pelo menos tantos estados quantos os de um AFD irreduzível.

Vejamos como essa prova funciona:

Minimização: Prova. Exemplo

A “spanning tree” dos strings $\{\varepsilon, 0, 01, 00\}$ é um conjunto de estados mutuamente distinguíveis (caso contrário ocorreria redundância, AFD teria sido reduzido).

Qualquer outro AFD para L tem ≥ 4 estados.

