Teoria da Computação

Prof. Sergio D. Zorzo
Departamento de Computação – UFSCar

03

Propriedades das Linguagens Regulares

- Propriedade do Bombeamento
- Linguagem Regular Vazia, Finita ou Infinita
- Operações Fechadas sobre as Linguagens Regulares (união, interseção, concatenação e complemento)
- Linguagens Regulares expressas por Autômatos Finitos com número mínimo de estados
- Expressões Regulares

Bombeamento para as Linguagens Regulares

Linguagens não regulares

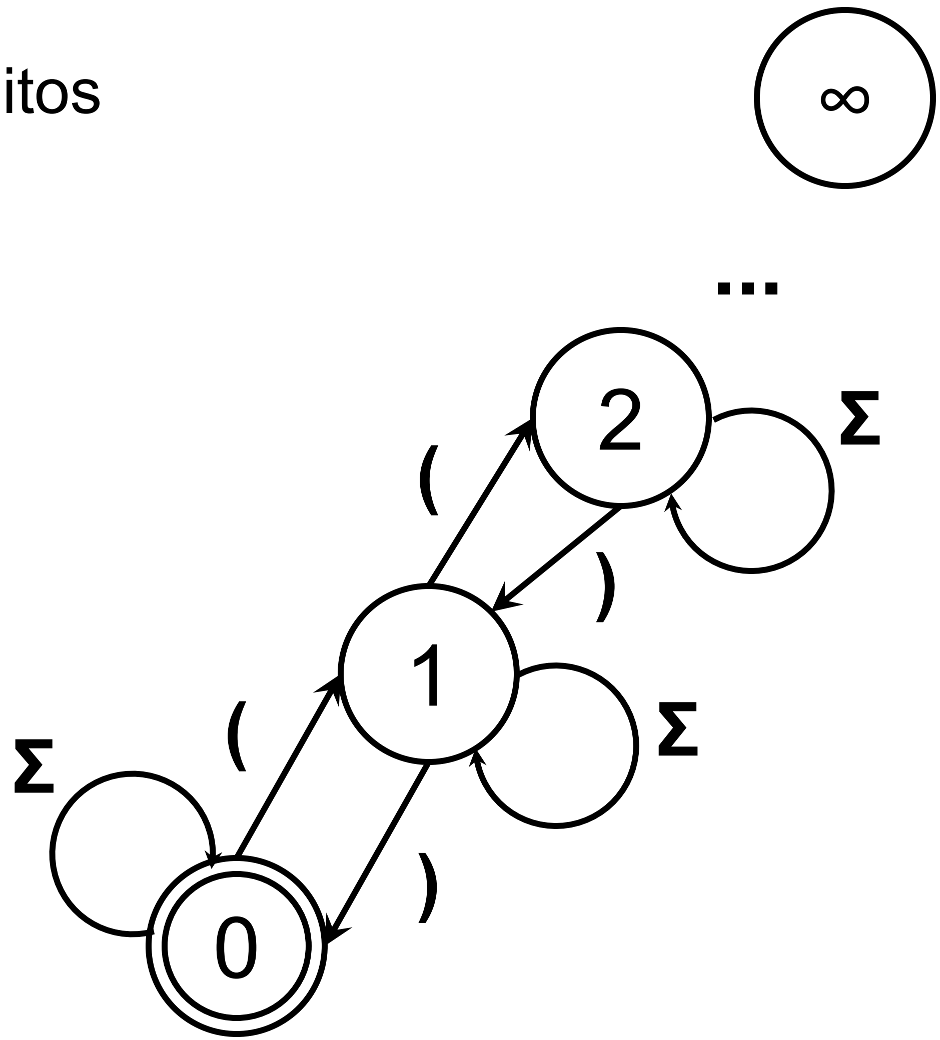
- Até agora vimos que: linguagens regulares são aquelas reconhecidas por autômatos finitos
  - Não foi feita nenhuma definição do que é uma linguagem regular
  - Um ser humano, ao olhar para uma linguagem, dificilmente consegue dizer se é ou não regular
  - Na verdade, não existe tal definição
- Mas existe uma distinção
  - Linguagens regulares vs não-regulares
  - A linha divisória é o fato de que
    - Autômatos finitos não conseguem contar

Linguagens não regulares

- Linguagens que exigem um contador
  - Ex: comentários dentro de comentários, escopos aninhados em uma linguagem, parêntesis aninhados, etc
  - Ex:
    - (1+2*(3-5+(7*7))-6)
    - É preciso contar quantos parêntesis são abertos e quantos são fechados
- Tente imaginar um autômato que reconheça tais cadeias
  - Estados são a “memória” do autômato

Linguagens não regulares

- Para reconhecer infinitos níveis de parêntesis aninhados, seriam necessários infinitos estados



Linguagens não regulares

Outros exemplos:

- $\{0^n1^n | n \geq 0\}$
- {w|w tem número igual de 0s e 1s}
- {ww| w seja uma cadeia sobre qualquer alfabeto}

Mas veja esse exemplo:

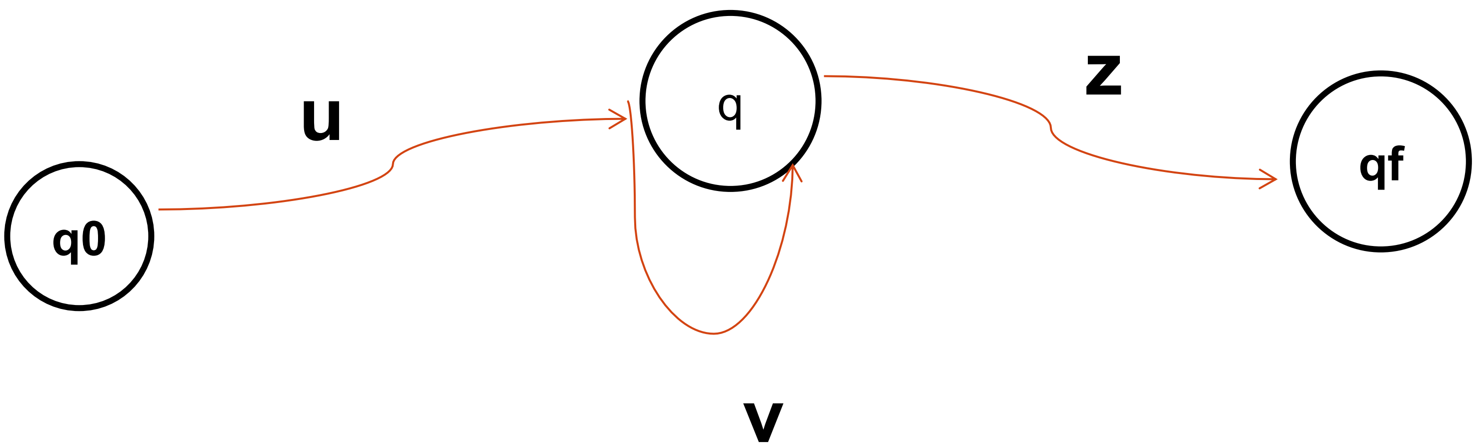
- {w|w tem um número igual de ocorrências de 01 e 10 como subcadeias} Aparentemente, precisa contar
  - Mas essa linguagem é regular! (faça depois como exercício a prova, se duvidar)

Formalmente:

- Lema do bombeamento para linguagens regulares
- Permite definir exatamente quais linguagens não são regulares

Lema do Bombeamento

- Se a linguagem é regular, então é aceita por um autômato finito determinístico que possui um número finito n de estados;
- Se o DFA aceita uma cadeia w de comprimento maior que n, obrigatoriamente o autômato tem algum estado q que é percorrido mais de uma vez (forma um ciclo)
- Logo, w = uvz e tem-se que uv^iz pertencerá à linguagem, para todo i>=0



Lema do bombeamento para linguagens regulares

Se L é uma linguagem regular,

então existe uma constante n (o comprimento de bombeamento) tal que,

Para qualquer cadeia w de L de comprimento no mínimo n ($|w|>=n$) , então w pode ser dividida em três partes, $w=uvz$, satisfazendo as seguintes condições:

- Para cada $i \geq 0$, $uv^iz \in L$
- $|v| > 0$
- $|uv| \leq n$

Informalmente:

Toda cadeia da linguagem contém uma parte que pode ser repetida um número qualquer de vezes (bombeada), com a cadeia resultante permanecendo na linguagem

Lema do bombeamento para linguagens regulares

Essa repetição ou bombeamento é a característica que faz com que seja sempre possível definir um número finito de estados para um autômato que reconheça a linguagem

Uso do lema do bombeamento:

Provar que B não é regular

Contradição: suponha que B seja regular

1. Encontre um p de forma que todas as cadeias de comprimento p ou maiores possam ser bombeadas
2. Encontre uma cadeia s em B que tenha comprimento p ou mais, mas que não possa ser bombeada
3. Demonstre que s não pode ser bombeada considerando todas as maneiras de dividir s em x,y e z, conforme o lema

Lema do bombeamento para linguagens regulares

Ex: $\{0^n1^n | n \geq 0\}$

1. Seja p o comprimento de bombeamento
2. Escolha $s = 0^p1^p$

   s é maior que p (conforme o lema)

   Portanto, o lema diz que s pode ser dividida em 3 partes, $s=xyz$, onde para qualquer $i \geq 0$, xy^iz está em B

   Ou seja, deve ser possível "bombear" y
3. Mas é impossível!!

   Primeira possibilidade: Suponha que y contém apenas 0s

   Ex: s = 000111, x=0, y=00, z=111

   Sempre que bombearmos y, teremos como resposta uma cadeia que não pertence à linguagem

   Pois teremos como resultado mais 0s do que 1s

Lema do bombeamento para linguagens regulares

Ex: $\{0^n 1^n | n \geq 0\}$

3. Mas é impossível!! (continuação)

   - Segunda possibilidade: Suponha que y contém apenas 1s
     - Ex: s = 000111, x=000, y=11, z=1
     - Sempre que bombearmos y, teremos como resposta uma cadeia que não pertence à linguagem
     - Pois teremos como resultado mais 1s do que 0s
   - Terceira possibilidade: y contém 0s e 1s
     - Ex: s = 000111, x=00, y=01, z=11
     - Sempre que bombearmos y, teremos como resposta uma cadeia que não pertence à linguagem
     - Pois teremos como resultado a presença de 0s e 1s alternados

Lema do bombeamento para linguagens regulares

Ex: $\{0^n1^n | n \geq 0\}$

Ou seja, é impossível existir uma divisão de w de acordo com o lema do bombeamento

Isso é uma contradição!

Ou seja, se não fizemos nada de errado, a suposição de que B é regular é falsa

Portanto, B não é regular

Lema do bombeamento para linguagens regulares

O “truque” é encontrar o w

Requer um pouco de pensamento criativo

Tentativa e erro

Busca pela “essência” da não-regularidade de B

Conhecimento das restrições do lema

($|v| > 0$, $|uv| \leq n$, etc)

Linguagens não regulares

- Ok, descobri que uma linguagem não é regular
  - Como resolver o problema?
  - Como obter uma implementação?
- Bom, se o problema é que um autômato finito não consegue contar...
  - ... basta adicionar um contador!
- Essa é exatamente a solução
  - Mais poder aos autômatos
  - Classe maior de linguagens
  - Mais detalhes nas próximas aulas!

Operações Fechadas sobre as Linguagens Regulares

Operações Fechadas sobre as Linguagens Regulares

Útil para construir novas linguagens regulares a partir de linguagens regulares conhecidas;

Provar Propriedades;

Construir algoritmos.

A classe das linguagens regulares é fechada para diversas operações, com destaque para:

- União
- Concatenação
- Complemento
- Intersecção

Linguagem Regular
Vazia, Finita ou Infinita

Linguagem Regular Vazia, Finita ou Infinita

Uma linguagem regular L aceita por um autômato Finito $M=(Q,\Sigma,\delta,q_0,F)$ com n estados , então L é:

a) Vazia se e somente se M não aceita qualquer palavra w tal que $|w| < n$;
b) Finita se e somente se M não aceita alguma palavra w tal que $n <= |w| <= 2n$;
c) Infinita se e somente se M aceita uma palavra w tal que $n <= |w| <= 2n$.

(prova)

Igualdade de Linguagens Regulares

Igualdade de linguagens Regulares

Se M1 e M2 são autômatos finitos, então existe um algoritmo para determinar se:

$$L(M1) = L(M2)$$

Prova:

Suponha que M1 e M2 são DFAs que aceitam L1 e L2 respectivamente, ou seja, L1=L(M1) e L2=L(M2).

É possivel construir o DFA M3 que aceita L3, onde:

$$L3 = (L1 \cap L2') \cup (L1' \cap L2)$$

Claramente, L1 = L2 se e somente se L3 for vazia.

E existe um algoritmo para determinar se uma linguagem regular é vazia ou não.

Minimização de AFs Determinísticos

Minimização de DFAs

- Existe um procedimento que minimiza um DFA
  - Ou seja, dado um DFA, ele permite encontrar um DFA equivalente que tenha o número mínimo de estados.
- De fato, esse DFA é mínimo:
  - Teorema: Se A é um DFA e M é o DFA construído a partir de A pelo algoritmo descrito a seguir, então M tem tão poucos estados quanto qualquer DFA equivalente a A
  - Em outras palavras, podemos testar a equivalência entre DFAs
    - Minimizando os dois e verificando se são iguais (com exceção, possivelmente, dos nomes dos estados)

Minimização de DFAs

Conceito de estados equivalentes

- Objetivo: entender quando dois estados distintos p e q podem ser substituídos por um único estado que se comporte como p e q

Formalmente:

- Dois estados p e q são equivalentes se:
  - Para todas as cadeias de entrada w, $\delta^*(p, w)$ é um estado de aceitação se e somente se $\delta^*(q, w)$ é um estado de aceitação

Minimização de DFAs

Menos formalmente:

- Existe uma cadeia w que leva p à aceitação e w à não-aceitação (ou vice-versa)?
- Se existir pelo menos uma cadeia assim, os estados são distinguíveis
  - Caso contrário, são equivalentes!

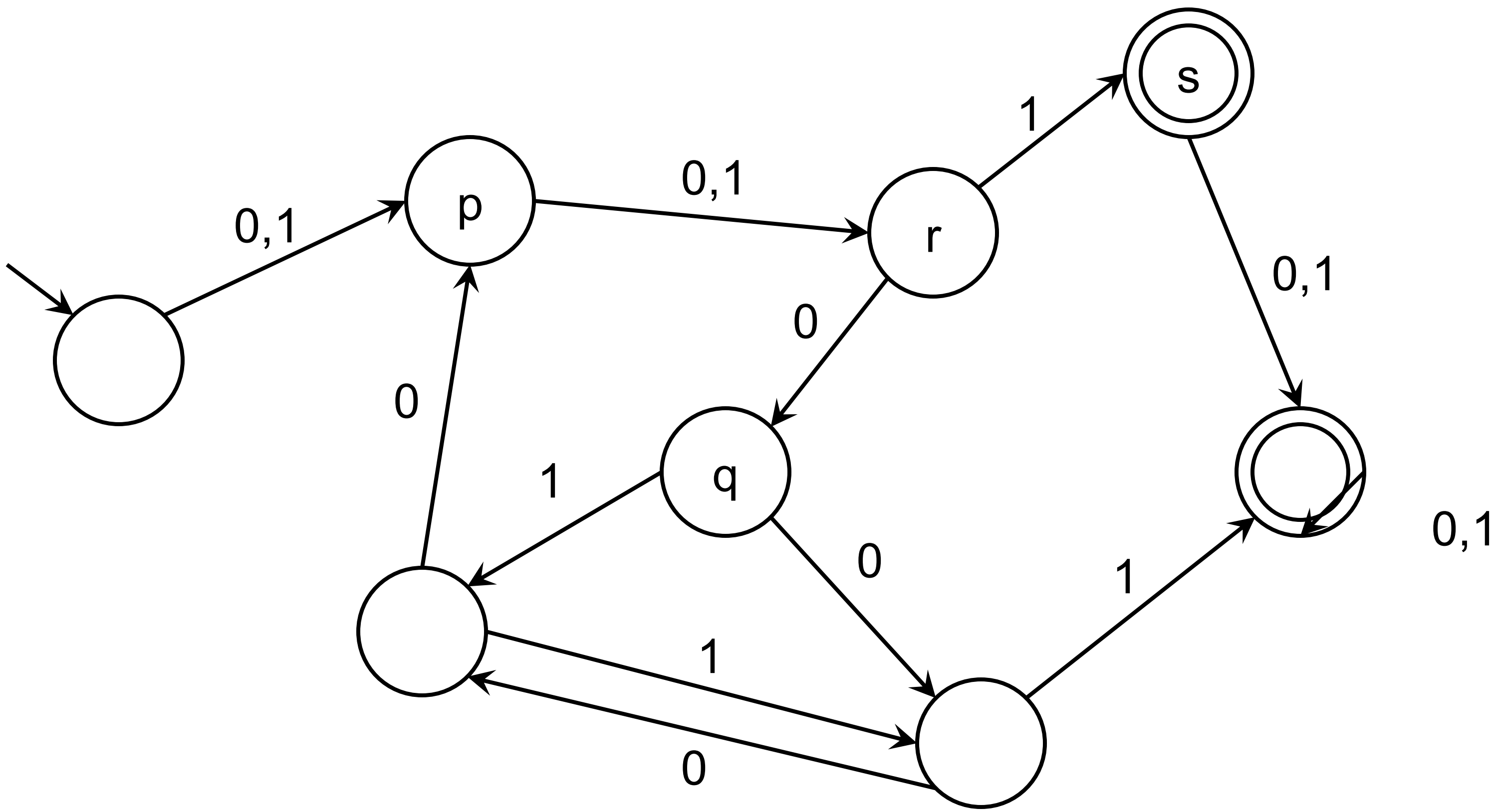
Minimização de DFAs

Ilustrando:

0,1, 010, 111 não distingue p e q

11 distingue p e q

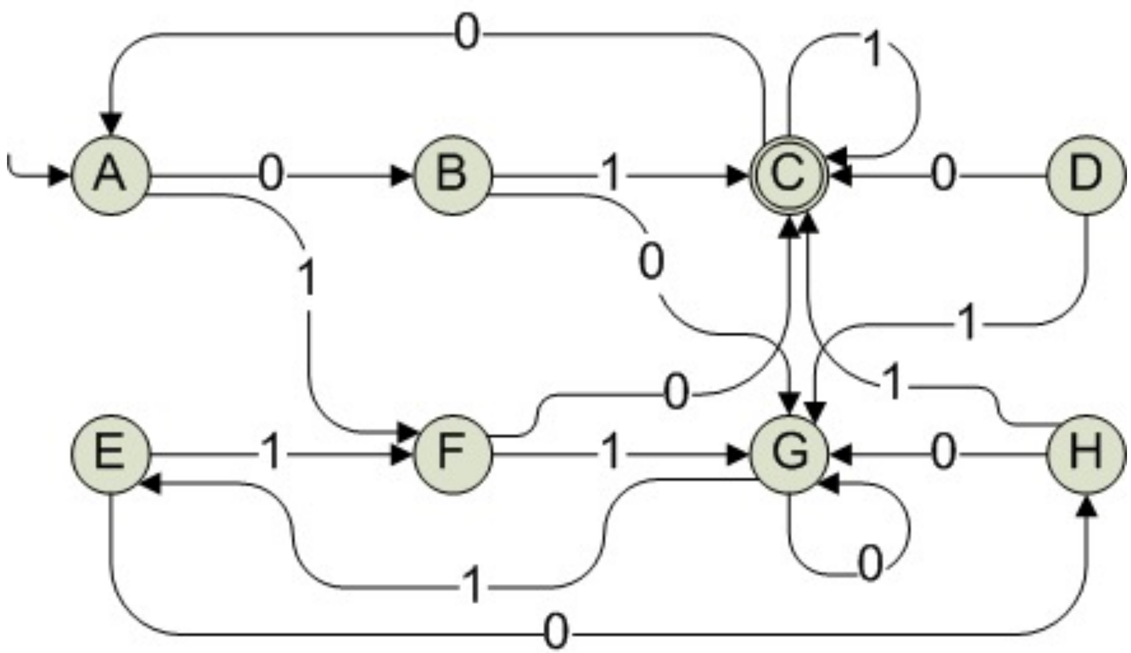
r e s são distinguíveis (ε os distingue)



Minimização de DFAs

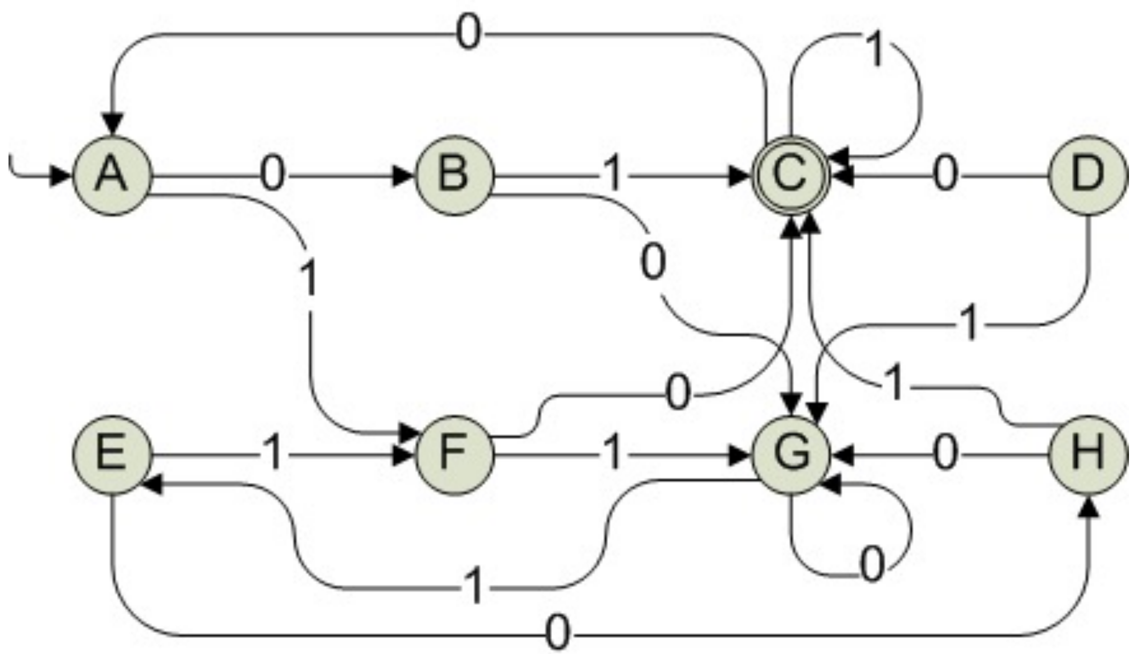
- Difícil encontrar estados equivalentes apenas “olhando” para o DFA
  - Muitas combinações, fácil se perder
- Estratégia sistemática: encontrar todos os pares de estados que sejam distinguíveis
  - Se fizermos o melhor possível
    - Qualquer par de estados que não considerarmos distinguíveis serão equivalentes
- Algoritmo de preenchimento de tabela
  - Descoberta recursiva de pares distinguíveis
  - Cada célula da tabela marca um par distinguível
  - Células em branco marcam pares equivalentes

Minimização de DFAs



	A	B	C	D	E	F	G
B							
C							
D							
E							
F							
G							
H							

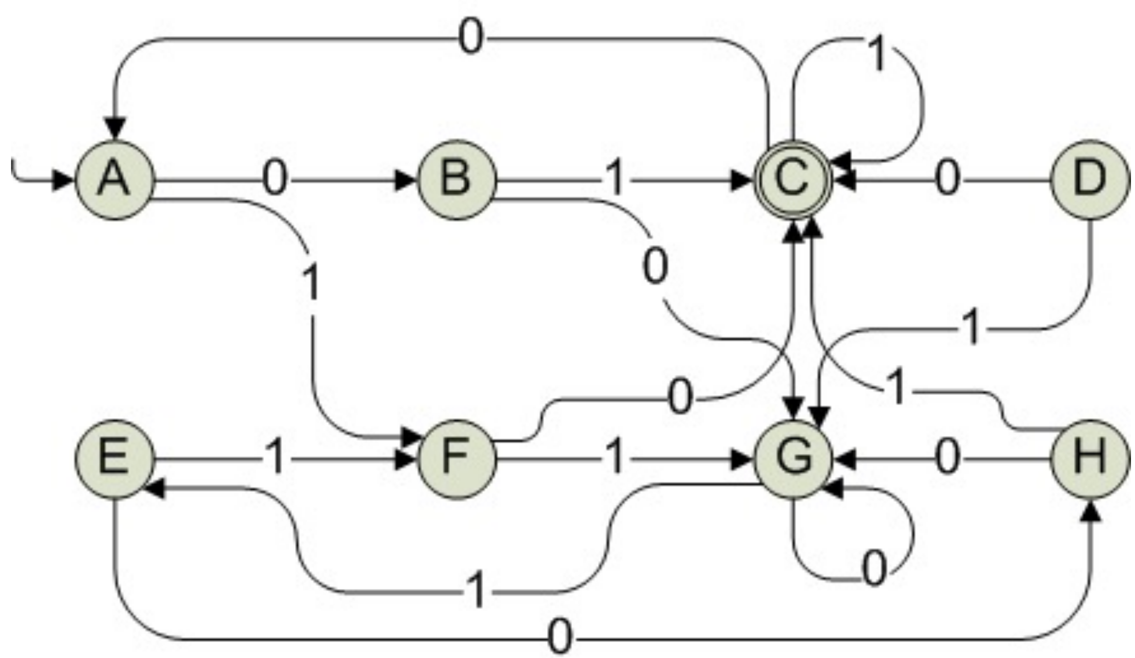
Minimização de DFAs



	A	B	C	D	E	F	G
B							
C	x	x					
D			x				
E			x				
F			x				
G			x				
H			x				

Começamos pelos estados de aceitação/não-aceitação. São obviamente pares distinguíveis pela cadeia vazia

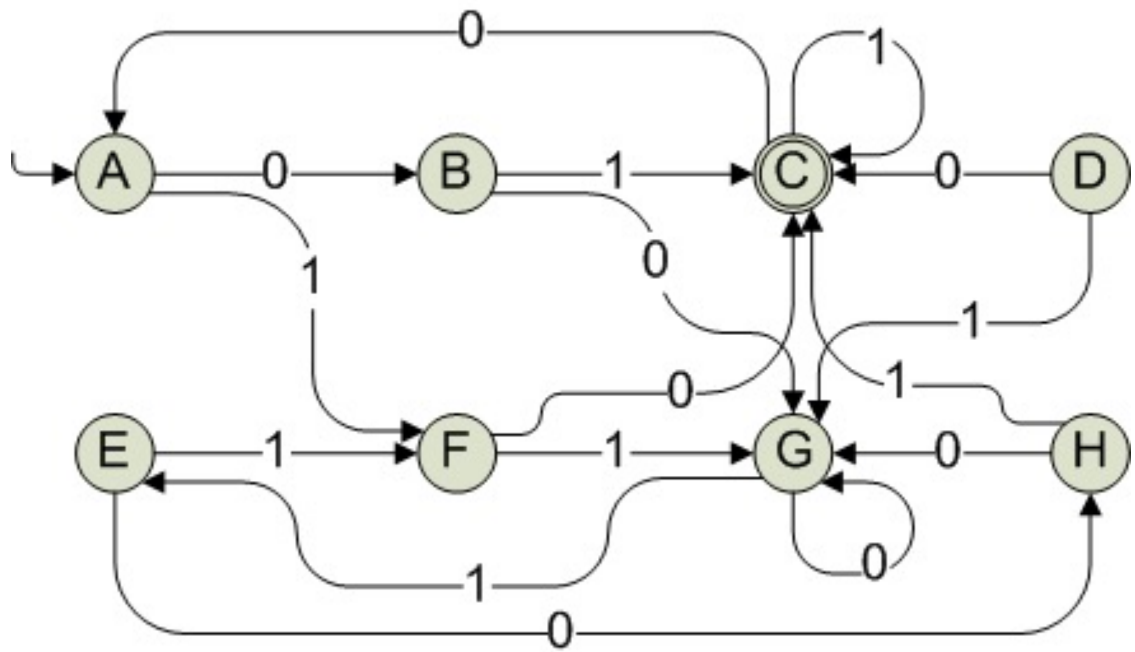
Minimização de DFAs



	A	B	C	D	E	F	G
B							
C	x	x					
D			x				
E			x				
F			x				
G			x				
H			x				

Agora tentamos encontrar outros estados que "chegam" em um par conhecido, dada uma mesma entrada.
A técnica é seguir, para cada par distinguível, as setas pelo lado inverso, com um mesmo rótulo
Fica mais fácil se marcar as células já analisadas

Minimização de DFAs



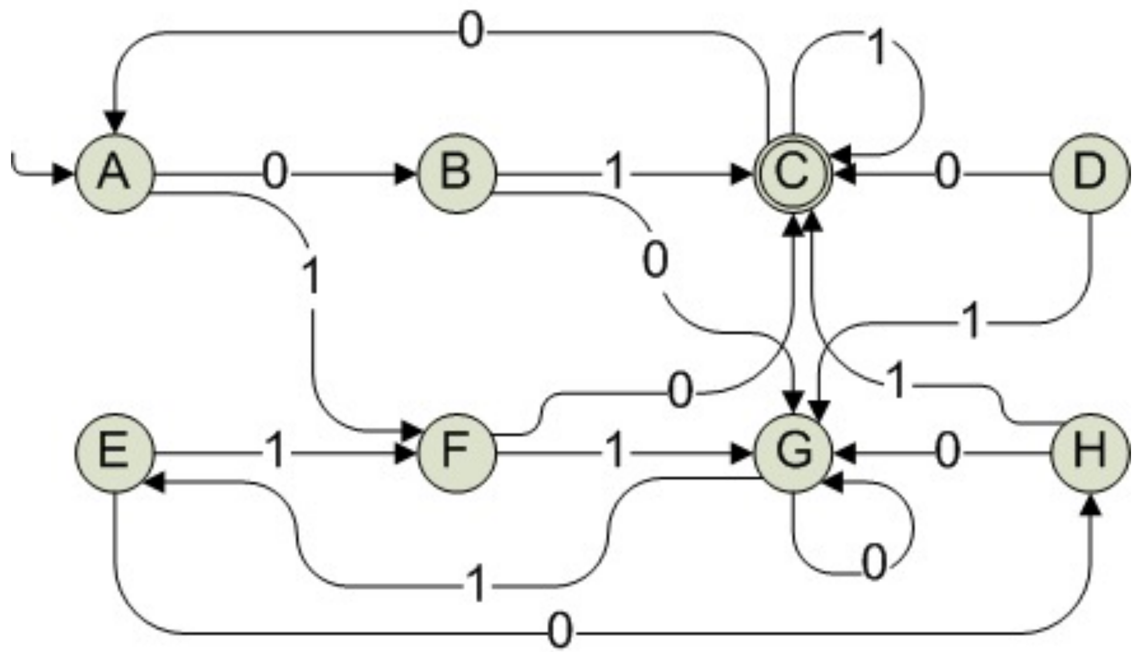
	A	B	C	D	E	F	G
B							
C	x	x					
D			x				
E			x				
F			x				
G			x				
H			x				

Exs:

(a) Seguindo as setas que “chegam” em A e C (um par distinguível), mediante entrada 0, temos:

- SetasA_0:{C}, SetasC_0:{D,F}
- Novos pares = SetasA_0 x SetasC_0 = {(C,D), (C,F)}
- Estes pares já estão marcados na tabela, com um x
- Analisando para entrada 1, temos:
- SetasA_1: {}, SetasC_1: {B,C,H}
- Novos pares = SetasA_1 X SetasC_1 = {} (nenhum novo par)
- Uma vez que já analisamos as entradas 0 e 1, a célula (A,C) foi analisada e é marcada

Minimização de DFAs



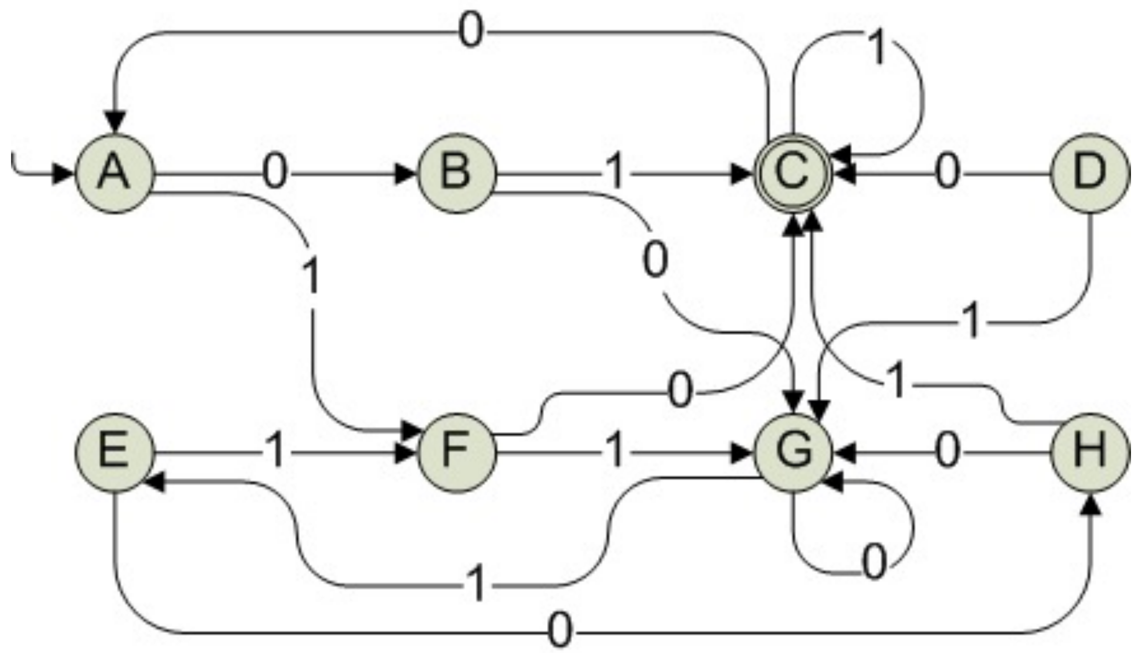
	A	B	C	D	E	F	G
B							
C	x	x					
D	x		x				
E			x				
F	x		x				
G			x				
H			x				

Exs:

(b) Continuando para par (B,C):

- SetasB_0:{A}, SetasC_0:{D,F}
- Novos pares = SetasB_0 x SetasC_0 = {(A,D), (A,F)}
- Esses pares ainda não foram marcados, e portanto a tabela precisa ser atualizada
- SetasB_1: {}, SetasC_1: {B,C,H}
- Novos pares = SetasB_1 X SetasC_1 = {} (nenhum novo par)
- Uma vez que já analisamos as entradas 0 e 1, a célula (B,C) foi analisada e é marcada

Minimização de DFAs



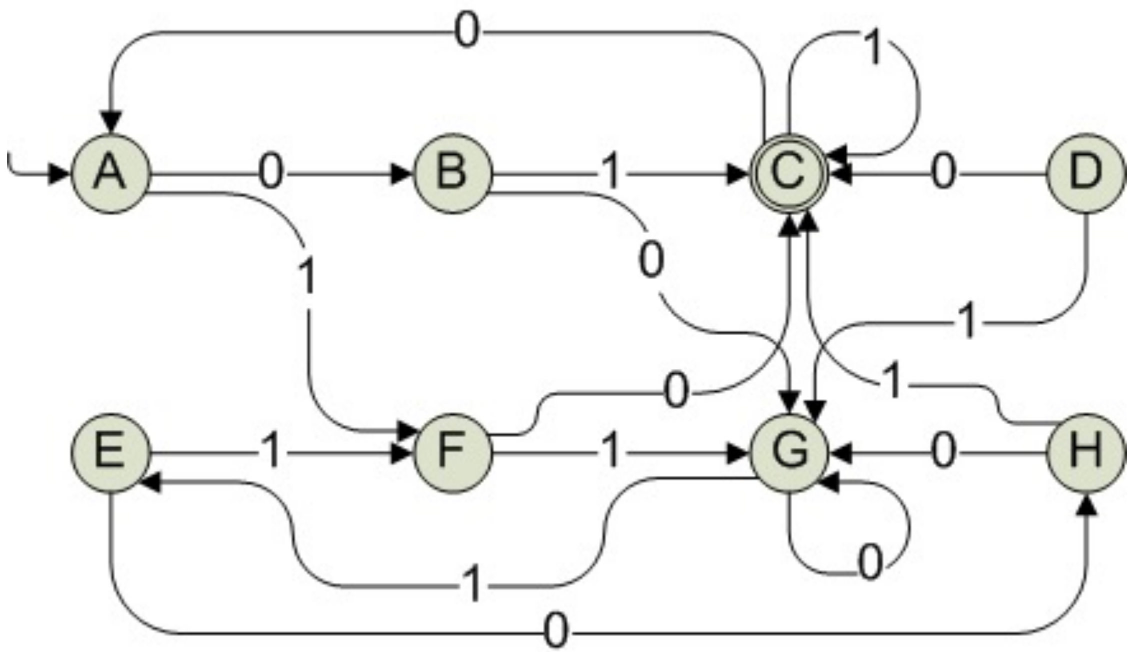
	A	B	C	D	E	F	G
B							
C	x	x					
D	x		x				
E			x				
F	x		x				
G			x				
H			x				

Exs:

(c) Continuando para par (C,D):

- SetasC_0:{D,F}, SetasD_0:{}
- Novos pares = {}
- SetasC_1:{B,C,H}, SetasD_1: {}
- Novos pares = {}
- Nenhum novo par

Minimização de DFAs



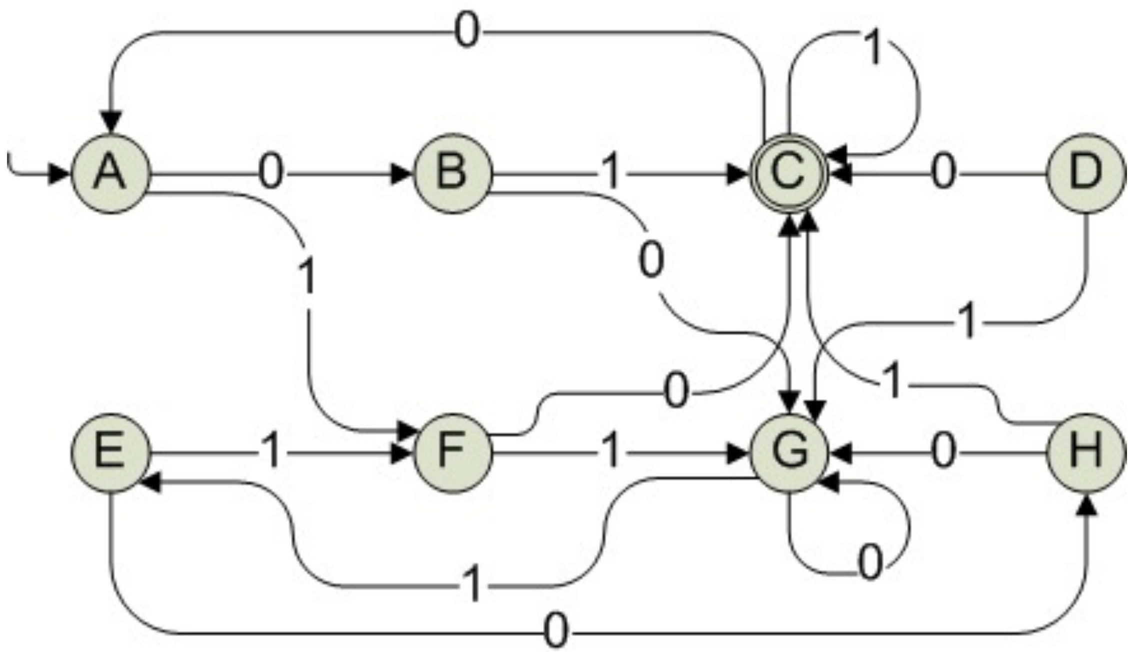
	A	B	C	D	E	F	G
B							
C	x	x					
D	x		x				
E			x				
F	x		x				
G		x	x				
H			x				x

Exs:

(d) Continuando para par (C,E):

- SetasC_0:{D,F}, SetasE_0:{}
- Novos pares = {}
- SetasC_1:{B,C,H}, SetasE_1: {G}
- Novos pares = {(B,G),(C,G),(H,G)}
- Novos pares e a célula (C,E) são marcados

Minimização de DFAs



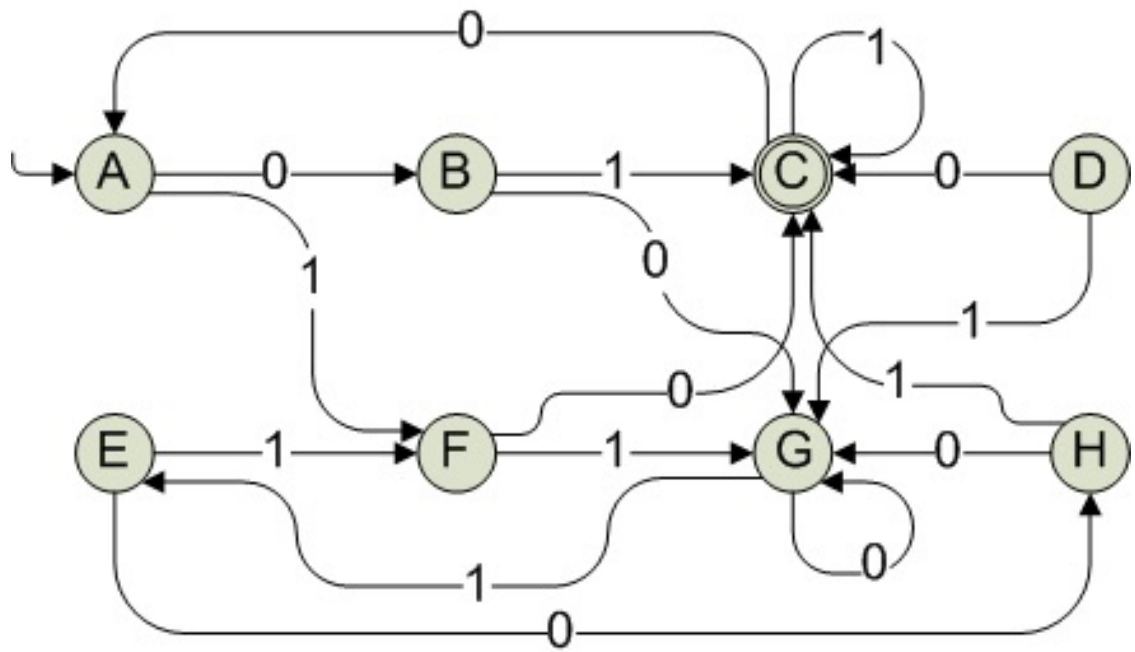
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x		x				
E		x	x				
F	x		x				
G		x	x				
H	x		x		x		x

Exs:

(e) Continuando para par (C,F):

- SetasC_0:{D,F}, SetasF_0:{}
- Novos pares = {}
- SetasC_1:{B,C,H}, SetasF_1: {A,E}
- Novos pares = {(B,A),(B,E),(C,A),(C,E),(H,A),(H,E)}
- Novos pares e a célula (C,F) são marcados

Minimização de DFAs



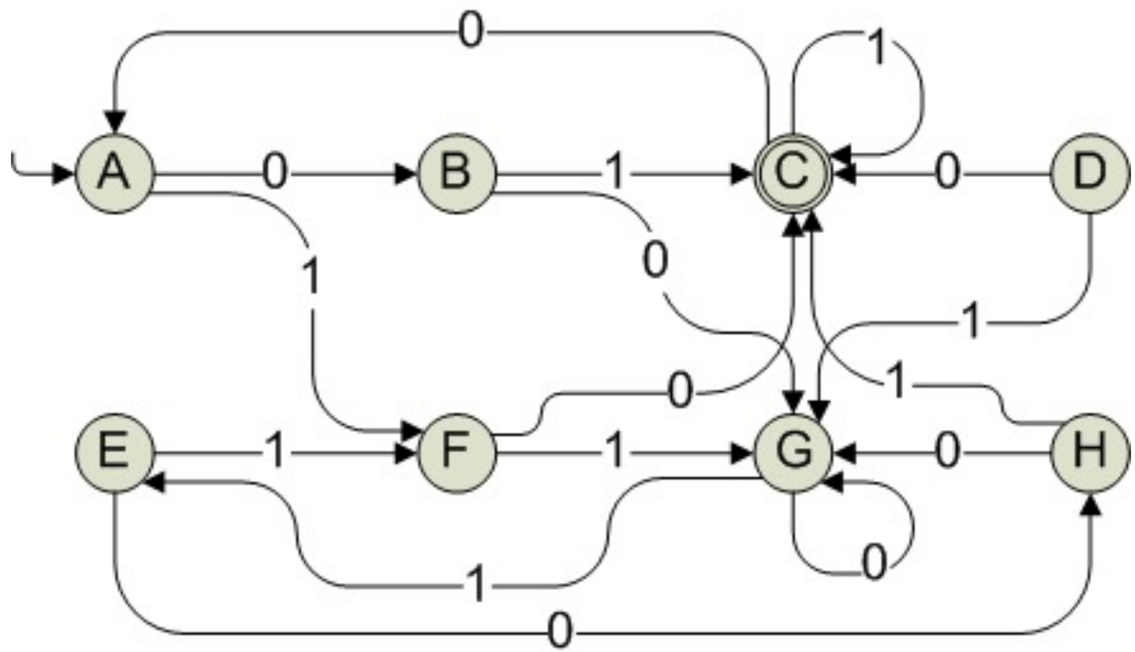
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x				
F	x	x	x				
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(f) Continuando para par (C,G):

- SetasC_0:{D,F}, SetasG_0:{B,G,H}
- Novos pares = {(D,B),(D,G),(D,H),(F,B),(F,G),(F,H)}
- SetasC_1:{B,C,H}, SetasG_1: {D,F}
- Novos pares = {(B,D),(B,F),(C,D),(C,F),(H,D),(H,F)}
- Novos pares e a célula (C,G) são marcados

Minimização de DFAs



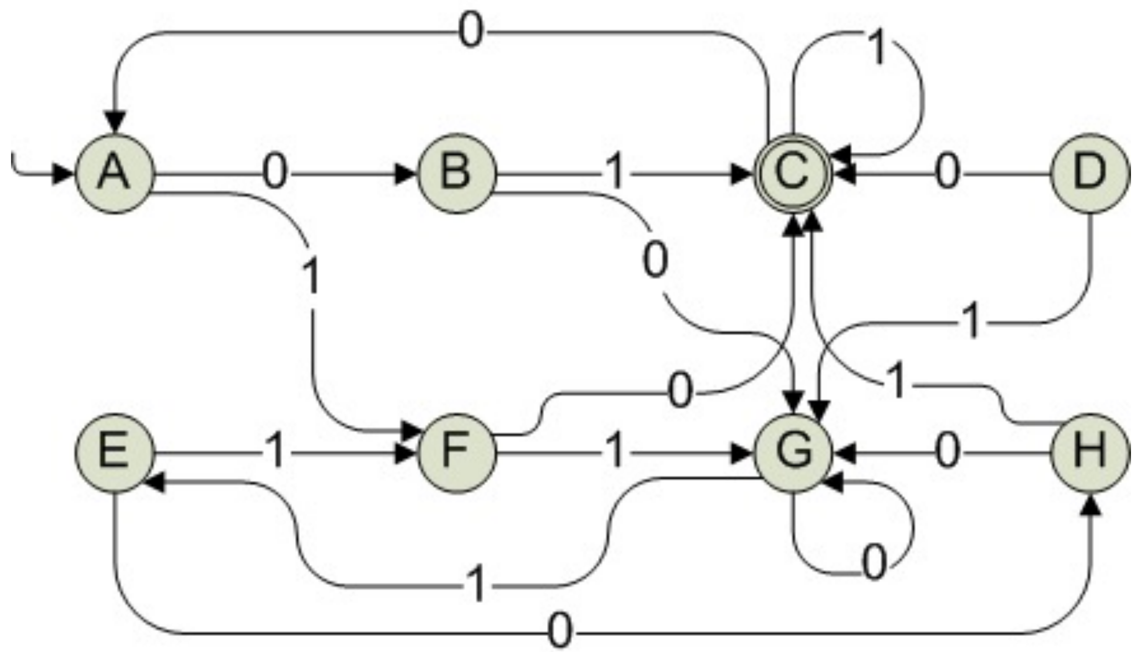
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(g) Continuando para par (C,H):

- SetasC_0:{D,F}, SetasH_0:{E}
- Novos pares = {(D,E),(F,E)}
- SetasC_1:{B,C,H}, SetasH_1: {}
- Novos pares = {}
- Novos pares e a célula (C,H) são marcados

Minimização de DFAs



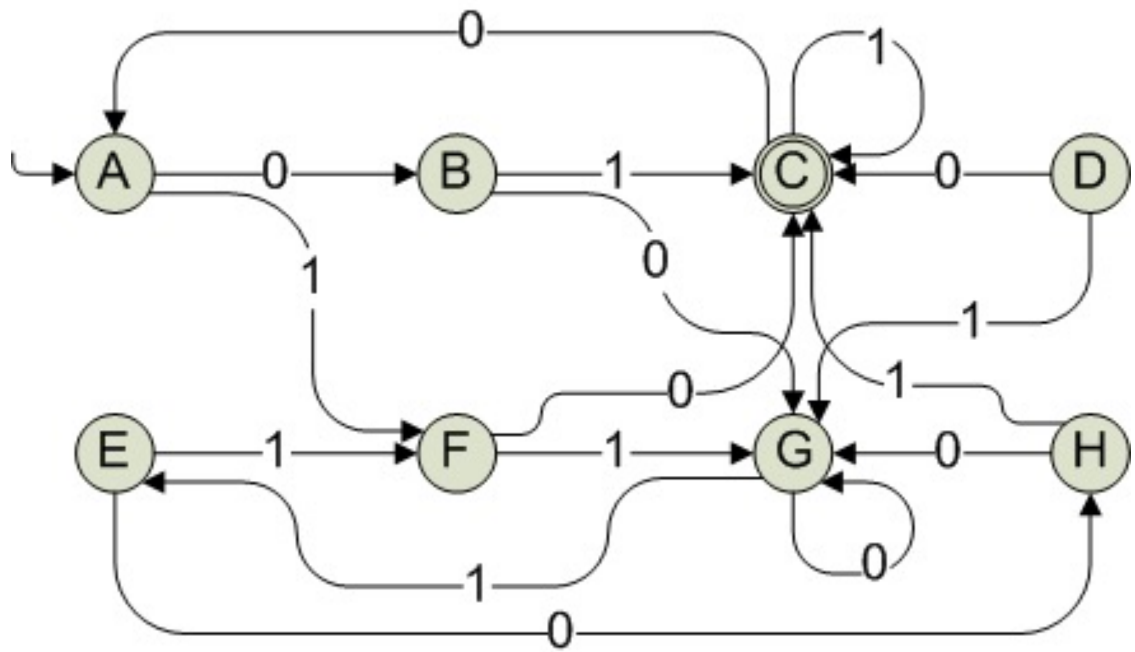
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(h) Continuando para par (A,B):

- SetasA_0:{C}, SetasB_0:{A}
- Novos pares = {(A,C)}
- SetasA_1:{}, SetasB_1: {}
- Novos pares = {}
- Novos pares e a célula (A,B) são marcados

Minimização de DFAs



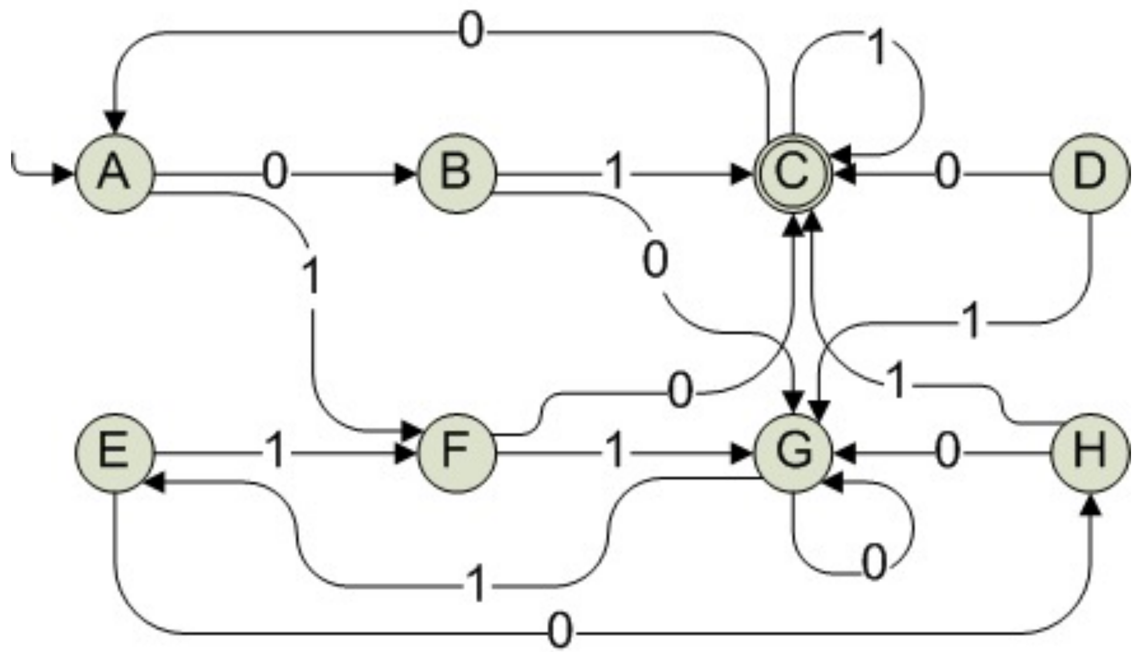
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(h) Continuando para par (A,D):

- SetasA_0:{C}, SetasD_0:{}
- Novos pares = {(A,C)}
- SetasA_1:{}, SetasD_1: {}
- Novos pares = {}
- Célula (A,D) é marcada

Minimização de DFAs



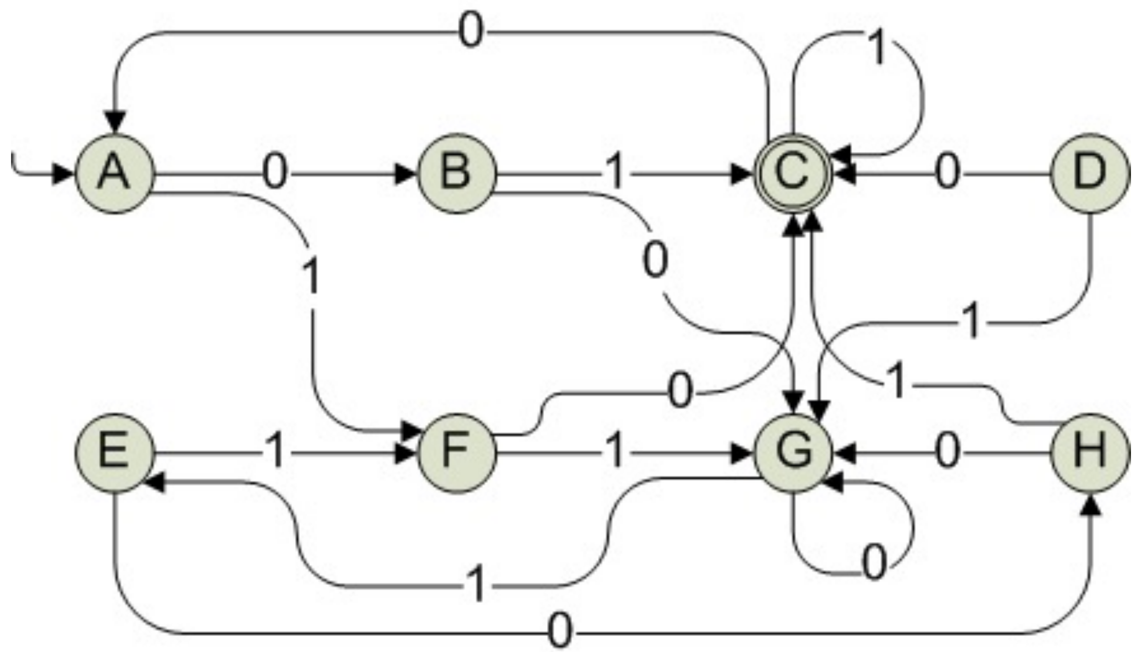
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(h) Continuando para par (A,F):

- SetasA_0:{C}, SetasF_0:{}
- Novos pares = {(A,C)}
- SetasA_1:{}, SetasF_1: {A,E}
- Novos pares = {}
- Célula (A,F) é marcada

Minimização de DFAs



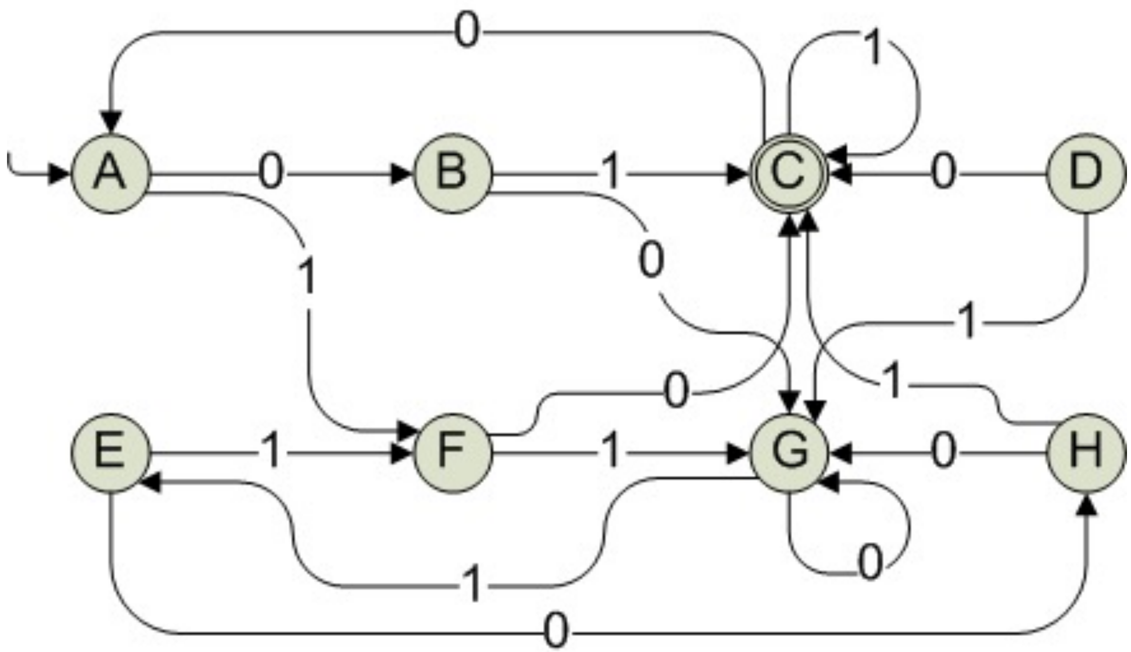
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(h) Continuando para par (A,H):

- SetasA_0:{C}, SetasH_0:{E}
- Novos pares = {(C,E)}
- SetasA_1:{}, SetasH_1: {}
- Novos pares = {}
- Célula (A,H) é marcada

Minimização de DFAs



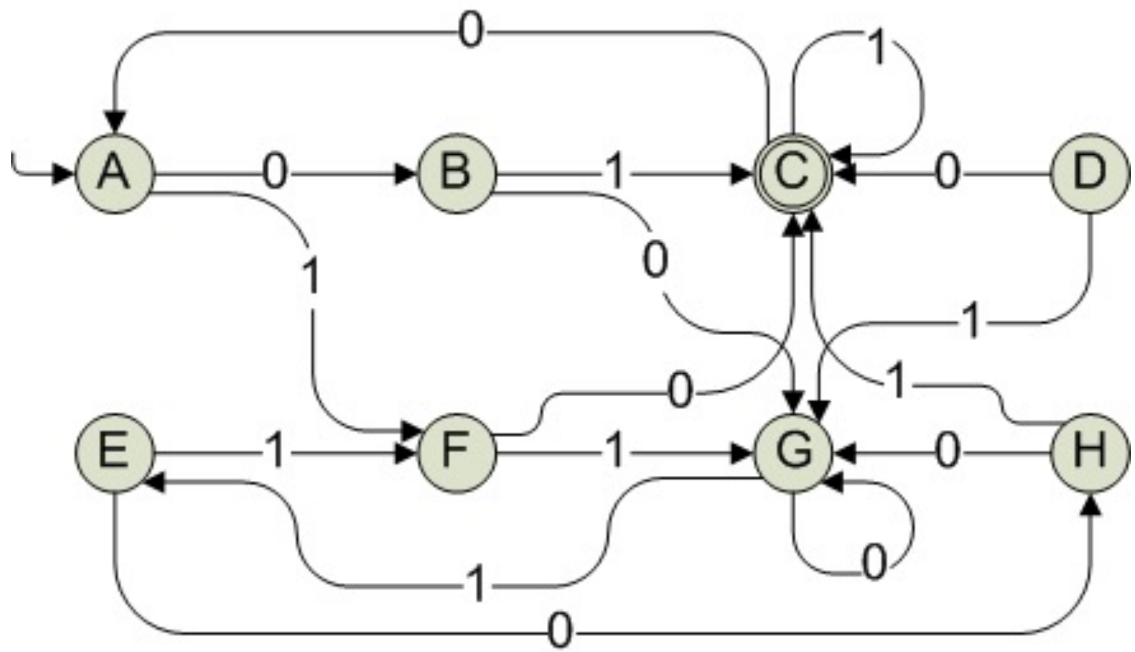
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(i) Continuando para par (B,D):

- SetasB_0:{A}, SetasD_0:{}
- Novos pares = {}
- SetasB_1:{}, SetasD_1: {}
- Novos pares = {}
- Célula (B,D) é marcada

Minimização de DFAs



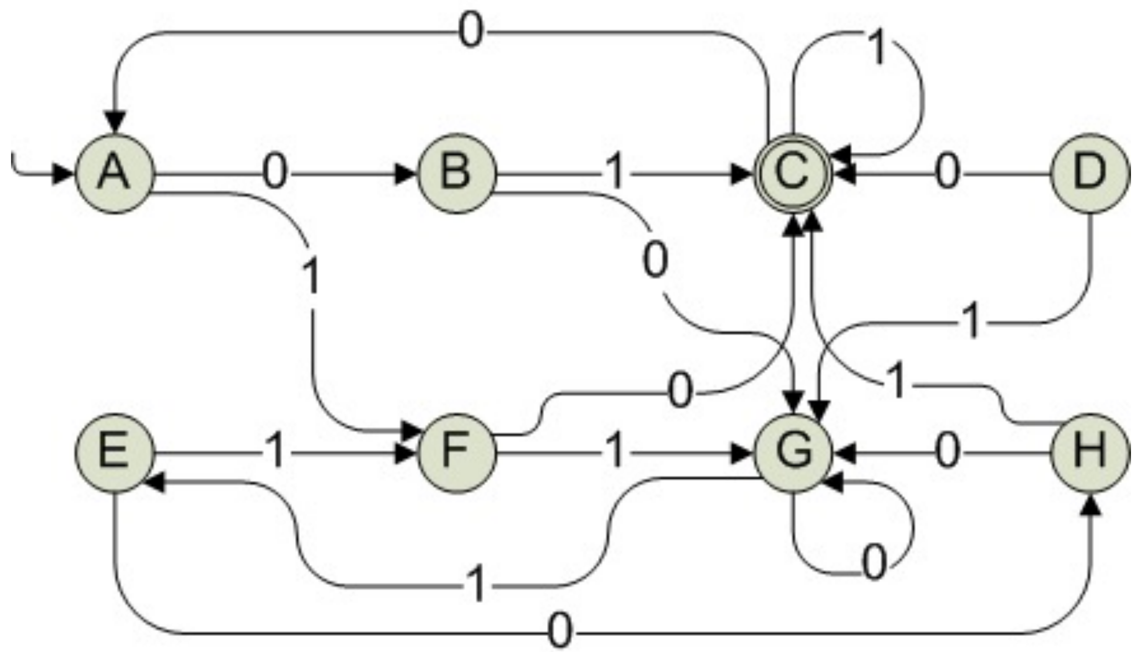
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(i) Continuando para par (B,E):

- SetasB_0:{A}, SetasE_0:{}
- Novos pares = {}
- SetasB_1:{}, SetasE_1: {G}
- Novos pares = {}
- Célula (B,E) é marcada

Minimização de DFAs



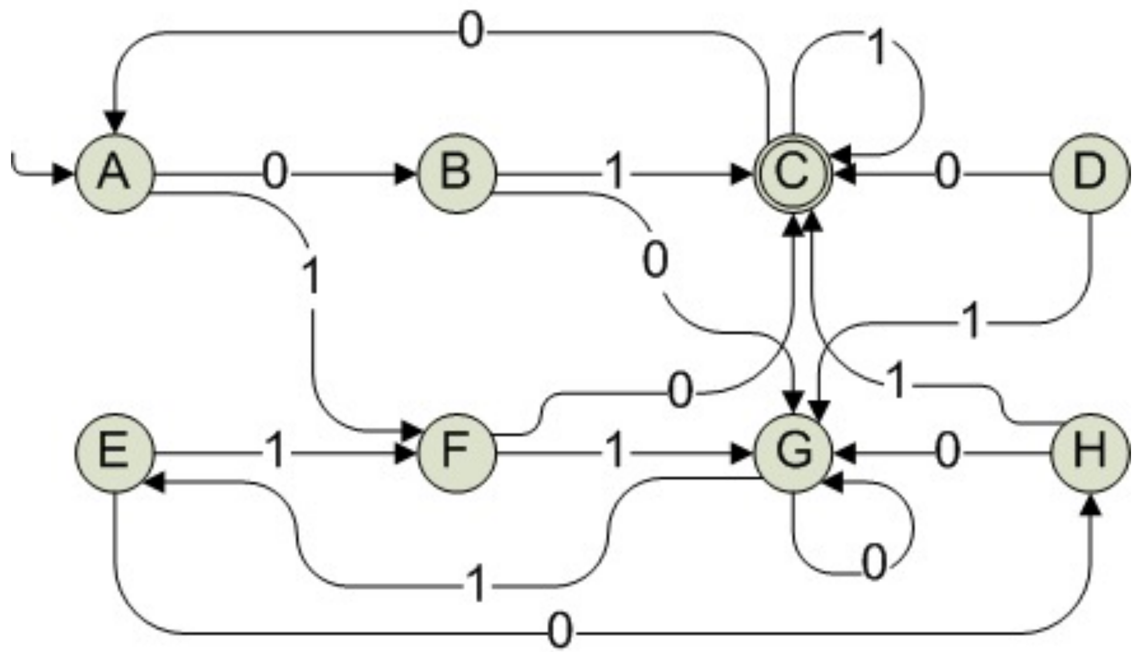
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

Exs:

(j) Continuando para par (B,F):

- SetasB_0:{A}, SetasF_0:{}
- Novos pares = {}
- SetasB_1:{}, SetasF_1: {A,E}
- Novos pares = {}
- Célula (B,F) é marcada

Minimização de DFAs



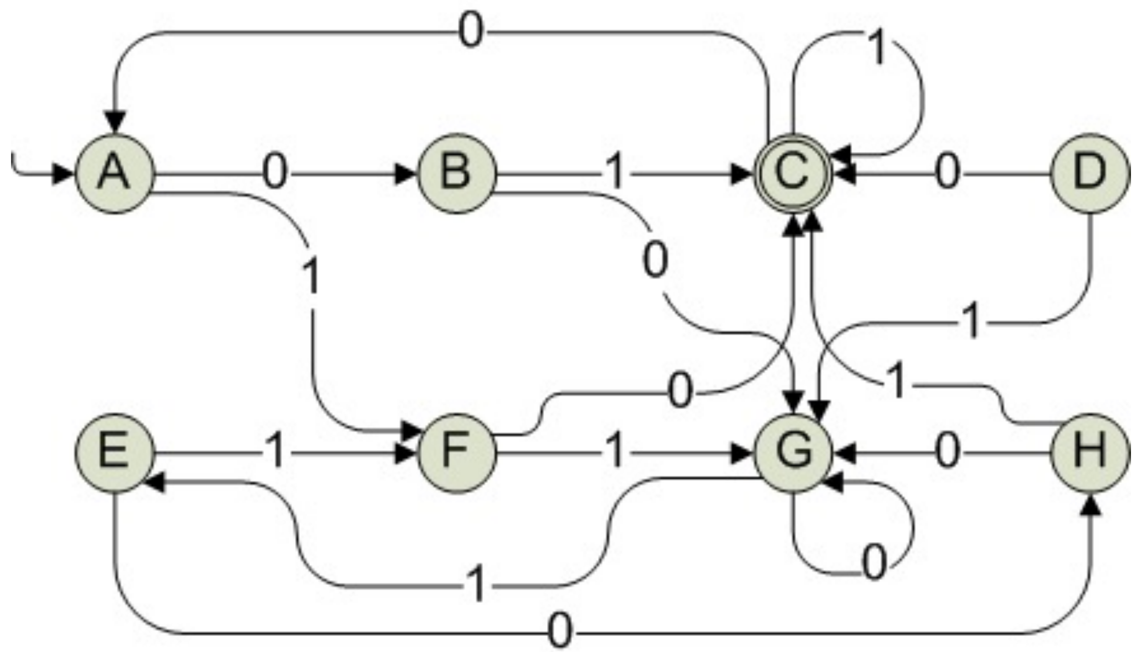
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

Exs:

(k) Continuando para par (B,G):

- SetasB_0:{A}, SetasG_0:{B,G,H}
- Novos pares = {(A,B),(A,G),(A,H)}
- SetasB_1:{}, SetasG_1: {D,F}
- Novos pares = {}
- Novo par e célula (B,G) são marcados

Minimização de DFAs



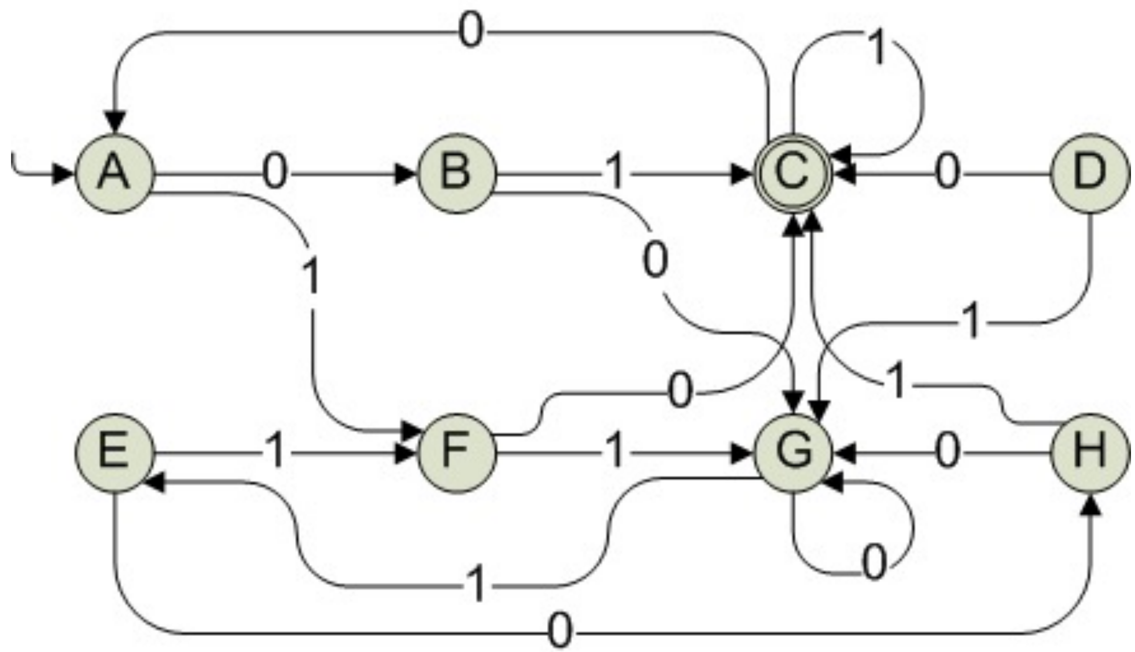
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

Exs:

(I) Continuando para par (A,G):

- SetasA_0:{C}, SetasG_0:{B,G,H}
- Novos pares = {(C,B),(C,G),(C,H)}
- SetasA_1:{}, SetasG_1: {D,F}
- Novos pares = {}
- Célula (A,G) é marcada

Minimização de DFAs



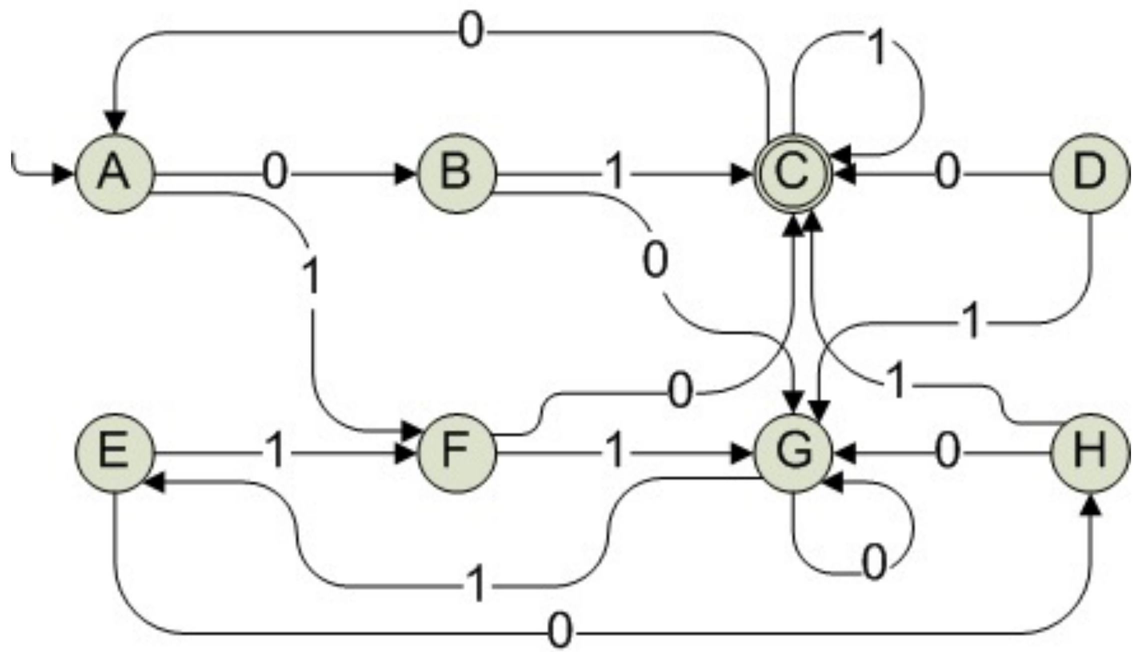
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

Exs:

(m) Continuando para par (D,E):

- SetasD_0:{}, SetasE_0:{}
- Novos pares = {}
- SetasD_1:{}, SetasE_1: {G}
- Novos pares = {}
- Células (D,E), (D,G) e (D,H) são marcadas (pois SetasD_0 e SetasD_1 são vazios)

Minimização de DFAs



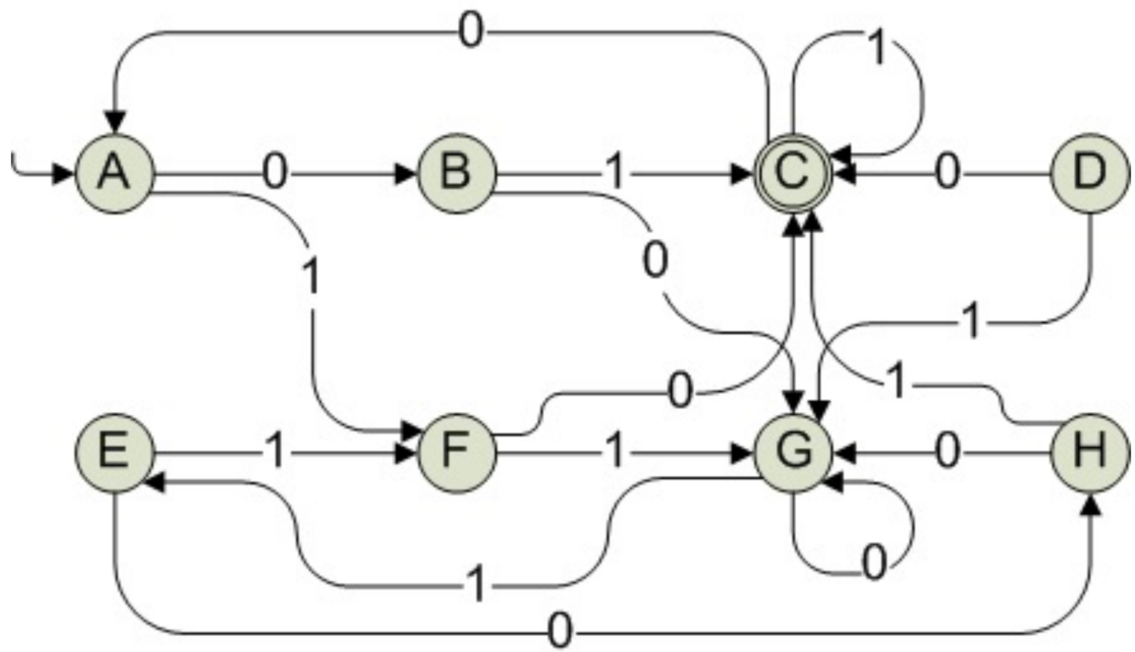
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

Exs:

(n) Continuando para par (E,F):

- SetasE_0:{}, SetasF_0:{}
- Novos pares = {}
- SetasE_1:{G}, SetasF_1: {A,G}
- Novos pares = {(A,G)}
- Célula (E,F) é marcada

Minimização de DFAs



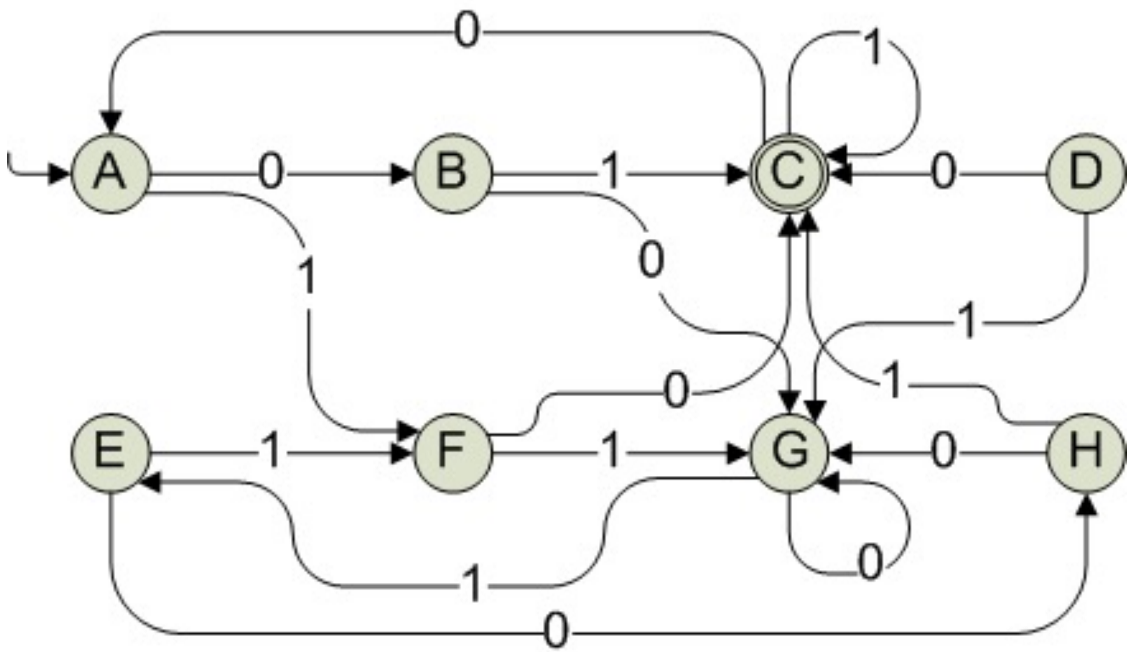
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

Exs:

(o) Continuando para par (E,H):

- SetasE_0:{}, SetasH_0:{E}
- Novos pares = {}
- SetasE_1:{G}, SetasH_1: {}
- Novos pares = {}
- Célula (E,H) é marcada

Minimização de DFAs



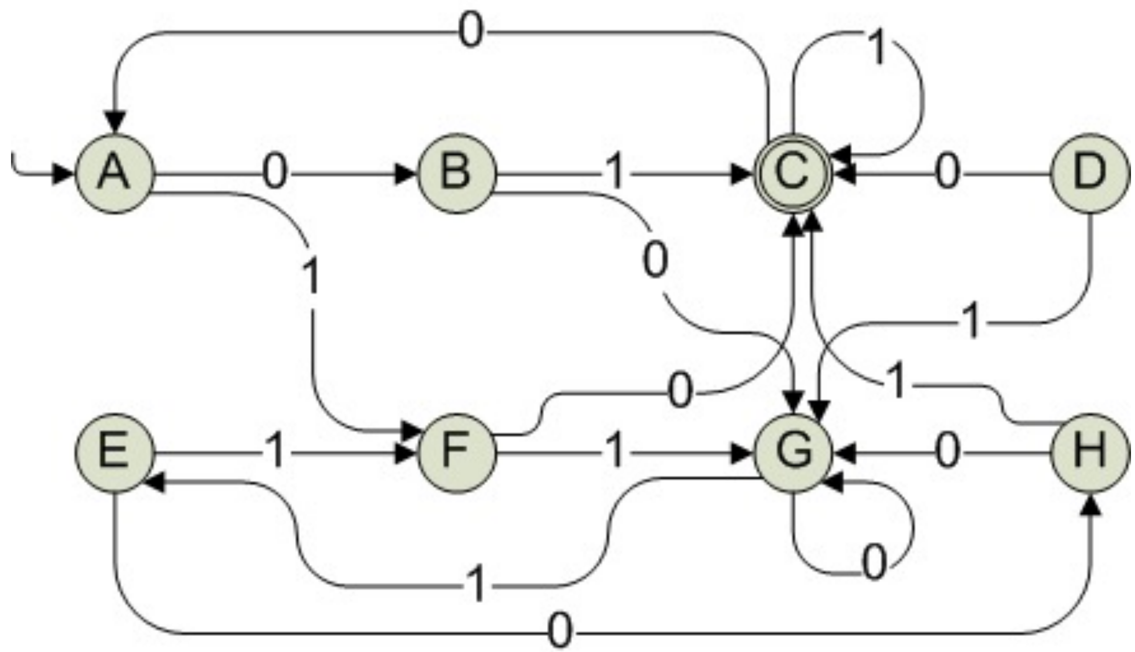
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

Exs:

(p) Continuando para par (F,G):

- SetasF_0:{}, SetasG_0:{B,G,H}
- Novos pares = {}
- SetasF_1:{A,E}, SetasG_1: {D,F}
- Novos pares = {(A,D),(A,F),(E,D),(E,F)}
- Célula (F,G) é marcada

Minimização de DFAs



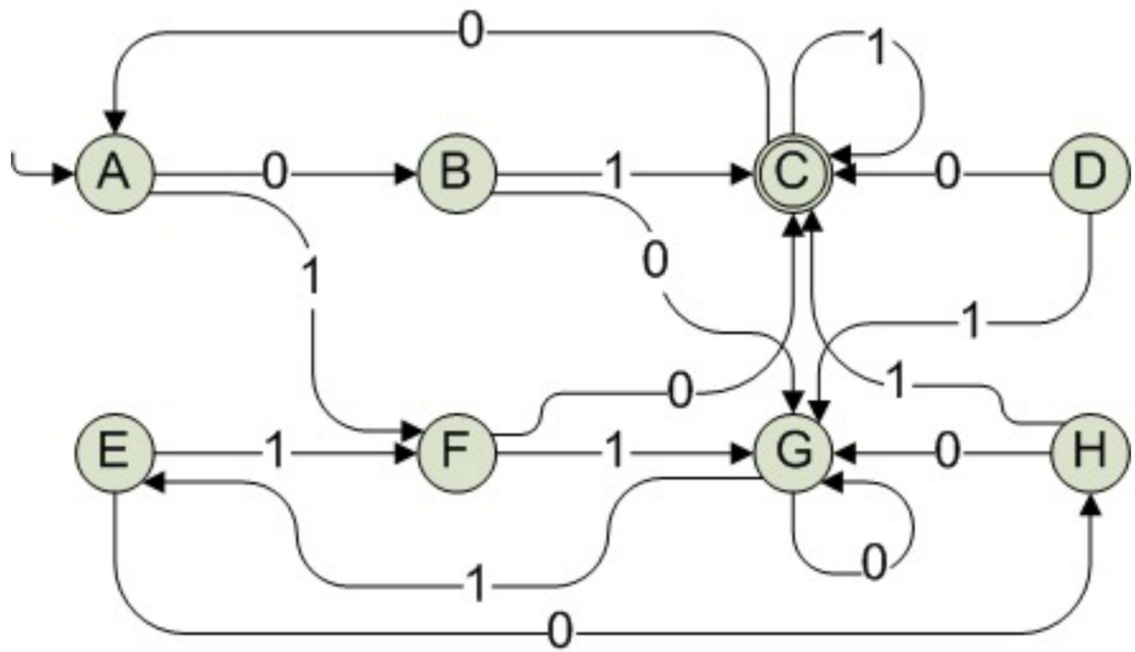
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

Exs:

(q) Continuando para par (F,H):

- SetasF_0:{}, SetasH_0:{E}
- Novos pares = {}
- SetasF_1:{A,E}, SetasH_1: {}
- Novos pares = {}
- Célula (F,H) é marcada

Minimização de DFAs



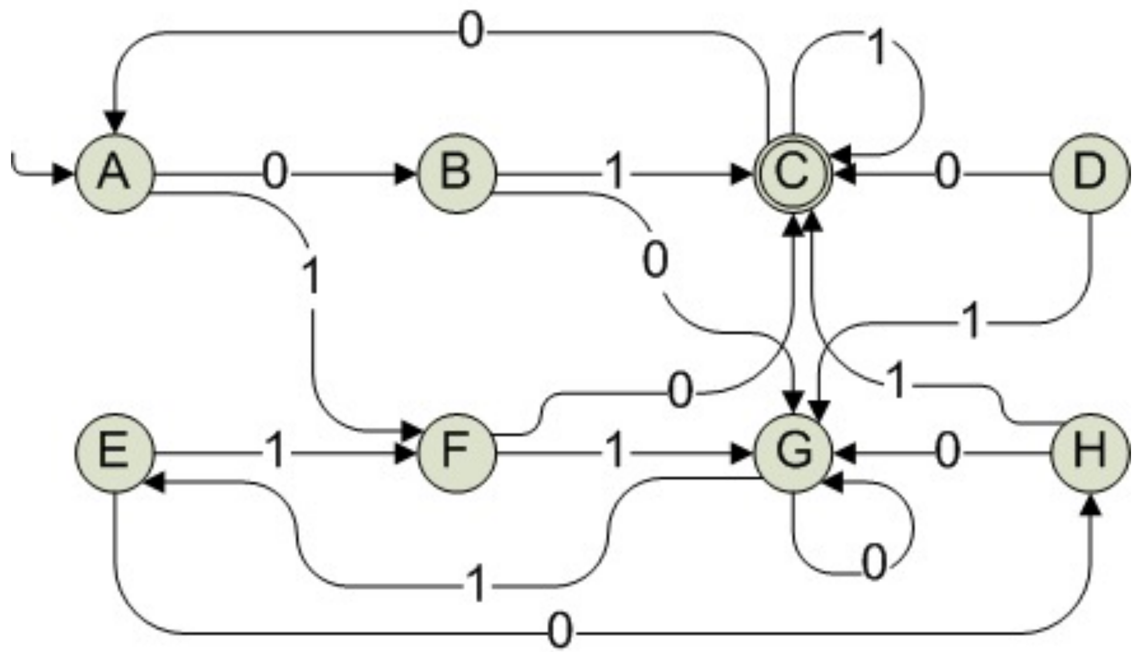
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Exs:

(q) Continuando para par (G,H):

- SetasG_0:{B,G,H}, SetasH_0:{E}
- Novos pares = {(B,E),(G,E),(H,E)}
- SetasG_1:{D,F}, SetasH_1: {}
- Novos pares = {}
- Novo par e célula (G,H) são marcados

Minimização de DFAs



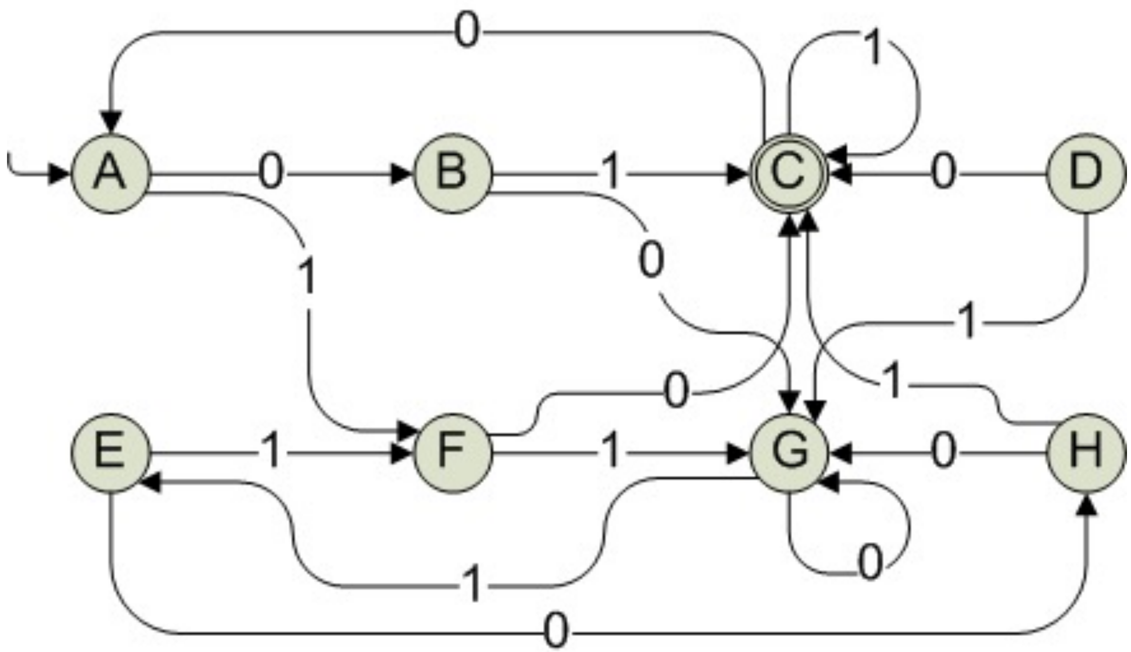
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Exs:

(r) Continuando para par (G,E):

- SetasG_0:{B,G,H}, SetasE_0:{}
- Novos pares = {}
- SetasG_1:{D,F}, SetasE_1: {G}
- Novos pares = {(D,G),(F,G)}
- Célula (G,E) é marcada

Minimização de DFAs



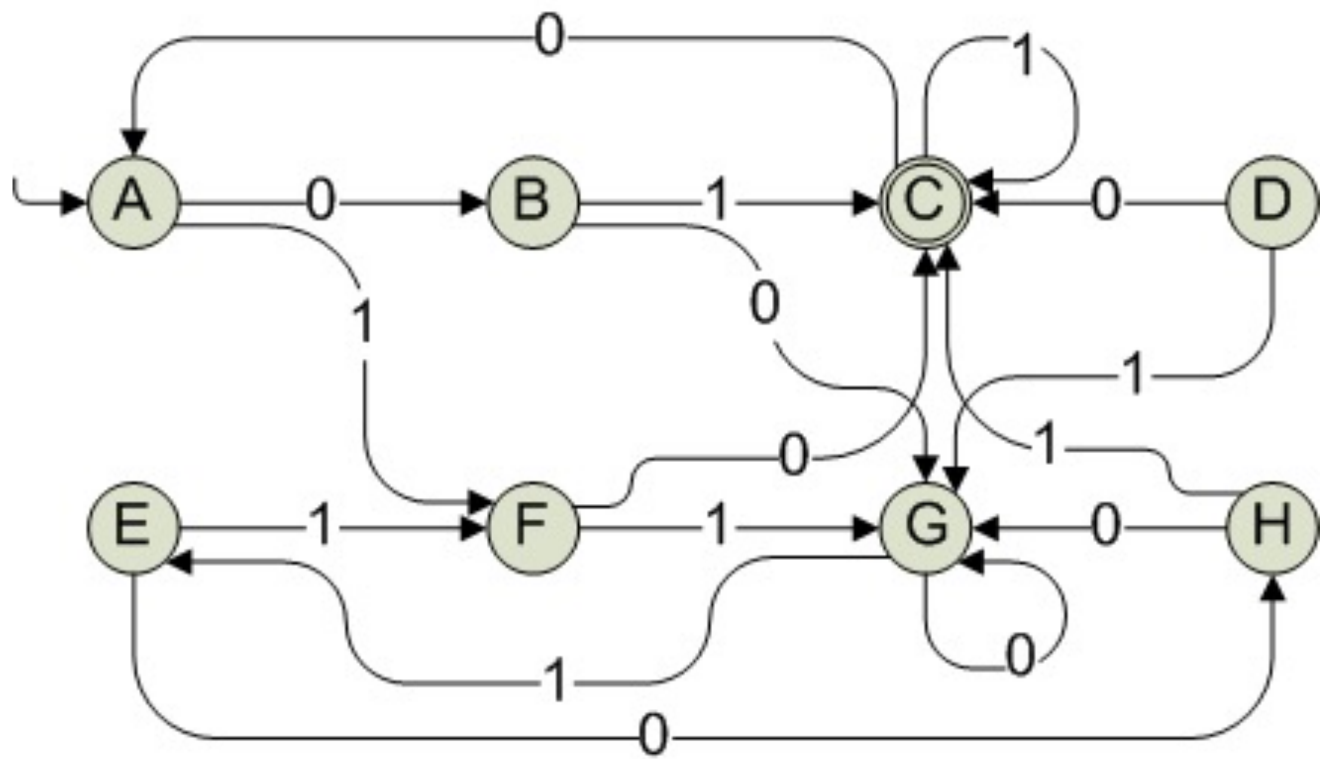
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Resultado:
- São pares equivalentes: (A,E), (B,H) e (D,F)

Minimização de DFAs

- Algoritmo em duas etapas:
  - a. Eliminar estados inalcançáveis
    - Reduz o trabalho do algoritmo de preenchimento de tabela
  - b. Particionar os estados restantes em blocos de estados equivalentes
    - Primeiro deve-se identificar os pares equivalentes
    - Depois formar os grupos de estados equivalentes

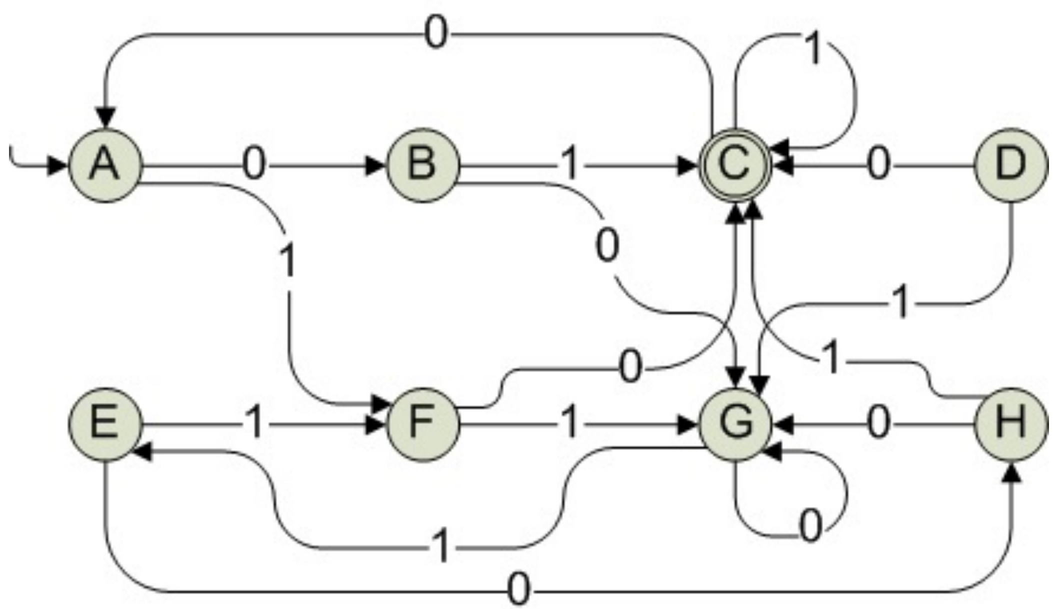
Estados inalcançáveis



Estados alcançáveis devem ter um caminho a partir do estado inicial

Neste exemplo, o estado D é inalcançável

Particionamento em grupos de estados equivalentes



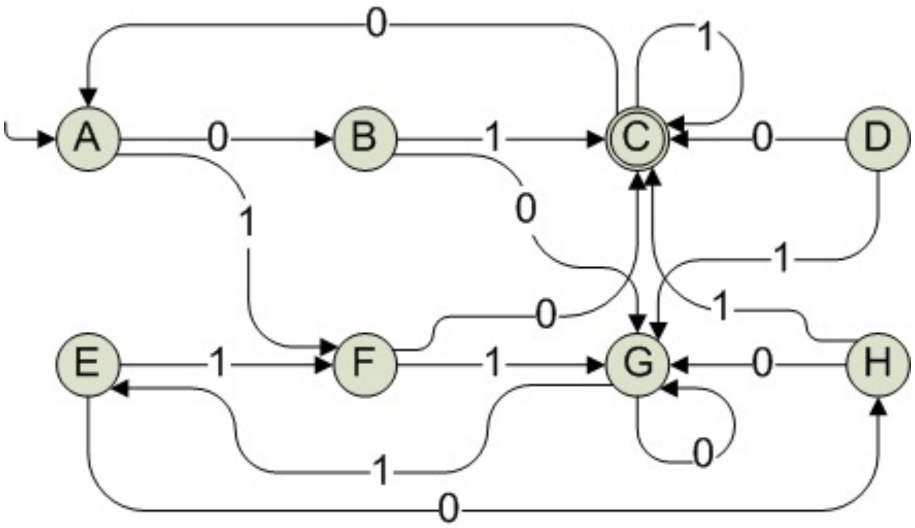
Neste exemplo, são pares equivalentes: (A,E), (B,H) e (D,F)
- Partição: {A,E},{B,H},{D,F},{C},{G}

Importante: deve-se considerar o caráter transitivo da equivalência. Por exemplo, se os pares equivalentes fossem: (A,E), (E,H), (D,F)
- A partição seria: {A,E,H},{D,F},{B},{C},{G}

(Ou seja, A é equivalente a E, E é equivalente a H, portanto A é equivalente a H, e os três formam um único grupo)

Particionamento em grupos de estados equivalentes

Para concluir a minimização, basta definir a nova função de transição



	0	1
{A,E}	{B,H}	{F}
{B,H}	{G}	{C}
{D,F}	{C}	{G}
{C}	{A}	{C}
{G}	{G}	{E}

Para isso, monta-se uma tabela vazia, onde cada estado é um grupo da partição

As transições são definidas como a união das transições no autômato original

Particionamento em grupos de estados equivalentes

	0	1
{A,E}	{B,H}	{F}
{B,H}	{G}	{C}
{D,F}	{C}	{G}
{C}	{A}	{C}
{G}	{G}	{E}

	0	1
{A,E}	{B,H}	{D,F}
{B,H}	{G}	{C}
{D,F}	{C}	{G}
{C}	{A,E}	{C}
{G}	{G}	{A,E}

De {F} para {D,F}

De {A} para {A,E}

De {E} para {A,E}

Agora, basta substituir os valores das células por grupos que representam estados válidos (a primeira coluna da tabela)

Nunca haverá conflito, devido ao algoritmo de preenchimento da tabela

Particionamento em grupos de estados equivalentes

Estados iniciais e de aceitação são os grupos que contém os estados iniciais e de aceitação do DFA original

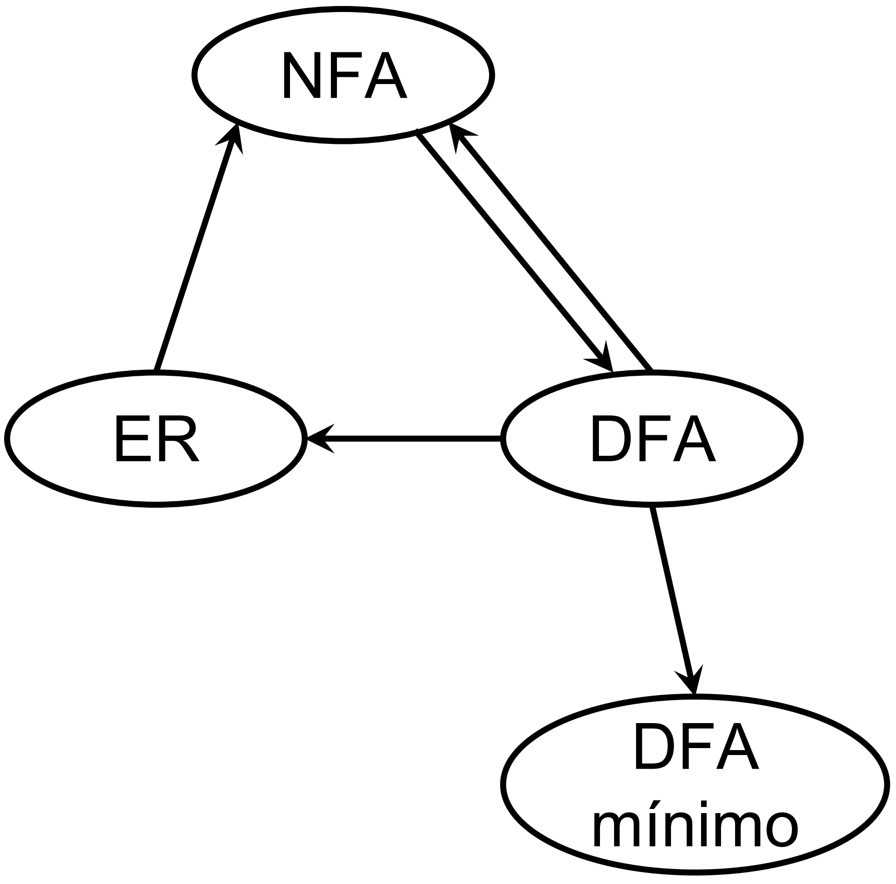
Para concluir, renomeie os estados para ficar mais legível

	0	1
→ {A,E}	{B,H}	{D,F}
{B,H}	{G}	{C}
{D,F}	{C}	{G}
* {C}	{A,E}	{C}
{G}	{G}	{A,E}

	0	1
→ Q1	Q2	Q3
Q2	Q5	Q4
Q3	Q4	Q5
* Q4	Q1	Q4
Q5	Q5	Q1

Resumo

- Minimização proporciona execução mais rápida
- Especialmente útil em compiladores, pois uma vez implementado, o DFA não irá mudar
  - Vale a pena o esforço extra



Expressões regulares

Expressões regulares

Autômatos denotam linguagens

Autômatos possuem duas notações

- Diagrama de estados
- Tabela de transições

Vimos apenas uma notação para linguagens

- Notações de conjuntos / formadores de conjuntos
- Ex: {w | regra sobre w}

Existe outra notação

- Expressões regulares

Expressões regulares

Definição de álgebra:

Um conjunto A e

uma coleção de operações sobre A

Operações que podem ser k-árias:

0-árias: ex: constantes, 2, x, y

1-árias: ex: -10

2-árias: ex: 2+2, 3*y

Expressões regulares

Na teoria da computação

Álgebra envolve

- Conjunto A = alfabeto
- Operações = operações regulares

Operações regulares

- Operações sobre membros de um alfabeto

- Linguagens regulares são fechadas sob as operações regulares

Conjuntos fechados sob uma operação

Exemplo:

N = {1,2,3,...} (conjunto de números naturais)

N é fechado sob multiplicação

Ou seja: para quaisquer x e y em N

x * y também está em N

N não é fechado sob divisão

Contra-exemplo: 1 e 2 estão em N, mas ½ não está!

Definição:

Uma coleção de objetos é fechada sob alguma operação se, aplicando-se essa operação a membros da coleção, recebe-se um objeto ainda na coleção

Operações regulares

Operações que:

- Aplicadas sobre elementos de linguagens regulares
- Resultam em linguagens regulares

Em outras palavras:

- Sejam L1 e L2 duas linguagens regulares
- L1 op_{reg} L2 é regular

Operações regulares

São 3 as operações regulares:

- União: $A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$
- Concatenação: $A.B = \{xy \mid x \in A \text{ e } y \in B\}$
- Estrela (ou fechamento, ou fechamento de Kleene):
  $A^* = \{x_1x_2...x_k \mid k \geq 0 \text{ e cada } x_i \in A\}$

União e concatenação são operações binárias

Estrela é uma operação unária

Operações regulares

União

$L = \{001, 10, 111\}$ e $M = \{\varepsilon, 001\}$

$L \cup M = \{\varepsilon, 10, 001, 111\}$

Concatenação

$L = \{001, 10, 111\}$ e $M = \{\varepsilon, 001\}$

L.M (ou LM) = $\{001,10,111,001001,10001,111001\}$

Estrela

$L = \{0, 11\}$

$L^* = \{\varepsilon$, 0, 00, 000, 000, 11, 011, 1111, 00011011, ... (não há uma ordem lógica aqui)$\}$

Operações regulares

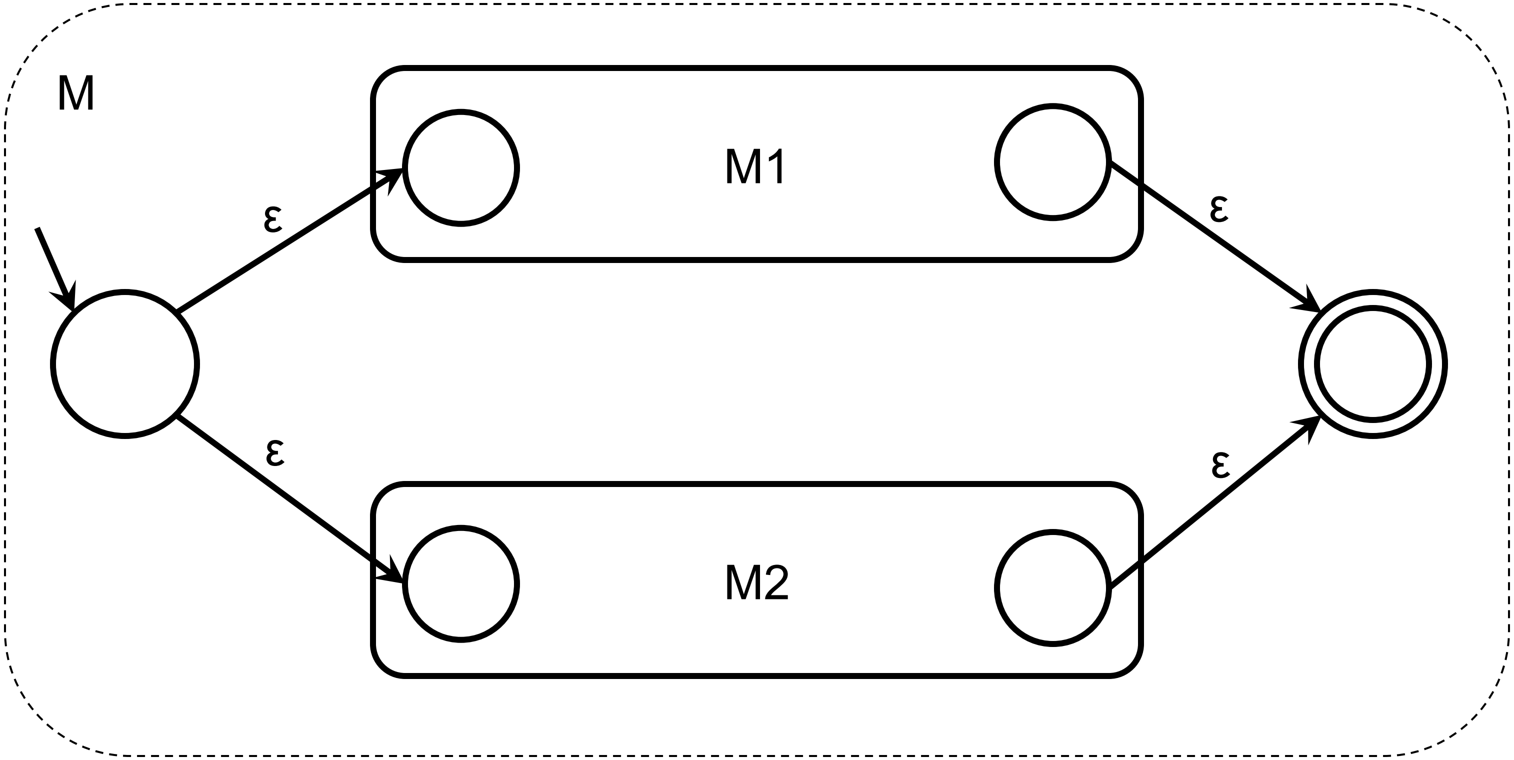
Teorema: A classe de linguagens regulares é fechada sob a operação de união

- Em outras palavras: se A1 e A2 são linguagens regulares, A1 ∪ A2 é regular

Prova por construção:

- A1 é regular, existe um autômato M1
- A2 é regular, existe um autômato M2
- Construímos um autômato M que simula M1 e M2, aceitando se uma das simulações aceita

Operações regulares



$$L(M) = L(M1) \cup L(M2)$$

Operações regulares

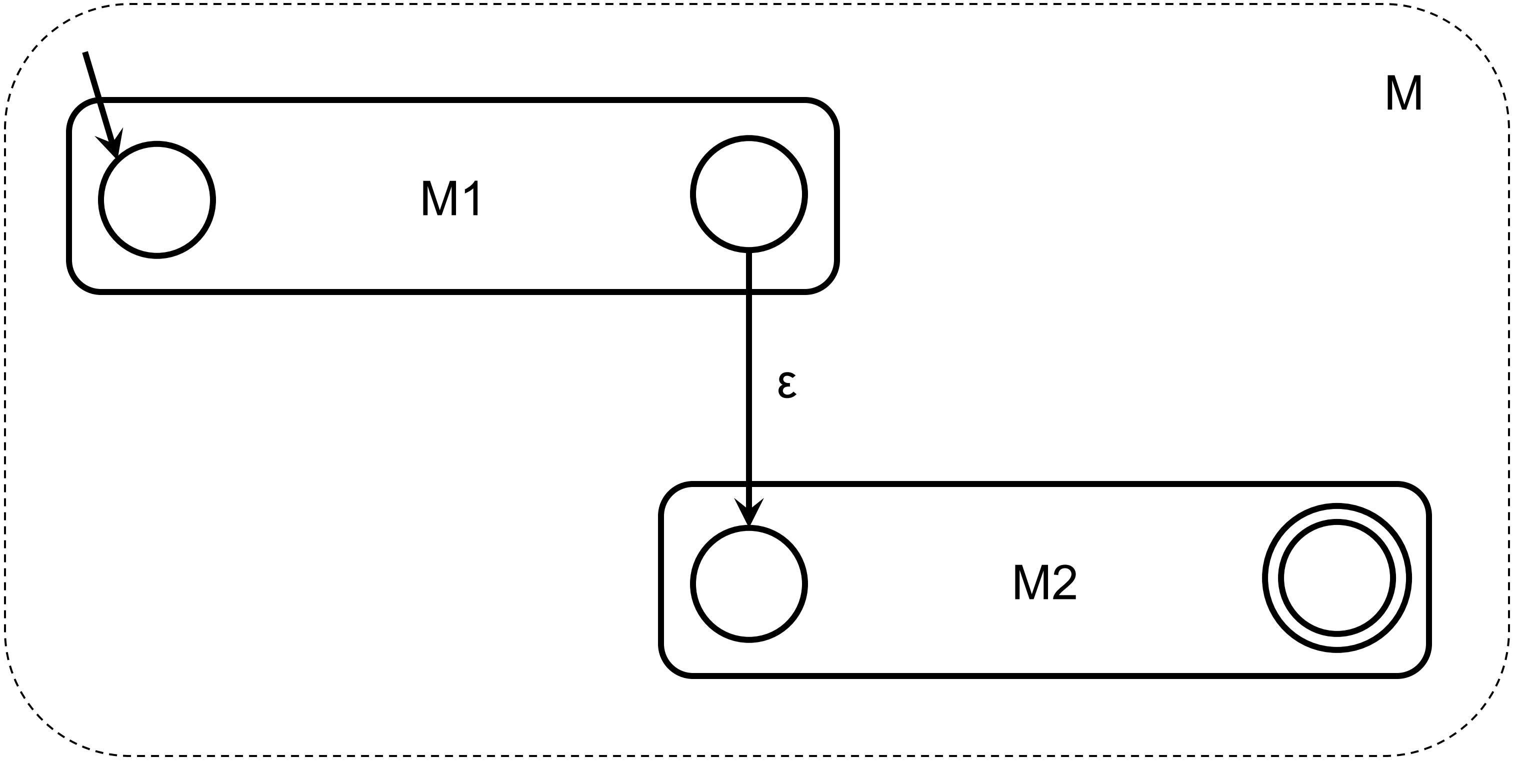
Teorema: A classe de linguagens regulares é fechada sob a operação de concatenação

- Em outras palavras, se A1 e A2 são linguagens regulares, A1.A2 é regular

Prova por Construção

- A1 é regular, existe um autômato M1
- A2 é regular, existe um autômato M2
- Construímos um autômato M que simula M1 e em seguida M2, passando de M1 para M2 quando M1 aceita, e aceitando quando M2 aceita

Operações regulares



$$L(M) = L(M1) \,.\, L(M2)$$

Operações regulares

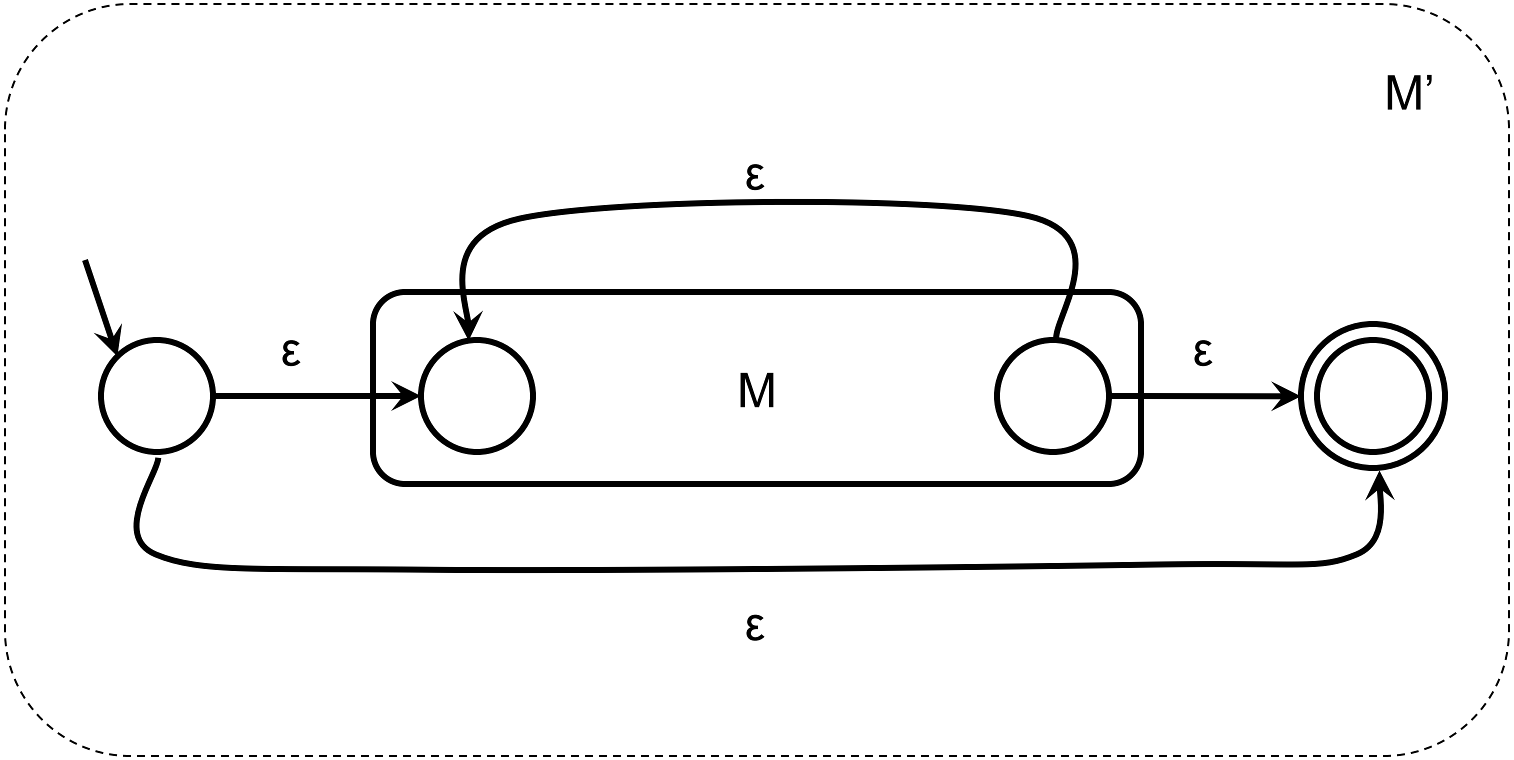
Teorema: A classe de linguagens regulares é fechada sob a operação de estrela

- Em outras palavras, se A é uma linguagem regular, A* é regular

Prova por Construção

- A é regular, existe um autômato M
- Construímos um autômato M' que simula M, com a possibilidade de ir direto para o estado de aceitação, e a possibilidade de voltar de um estado de aceitação para o inicial

Operações regulares



$$L(M') = L(M)^*$$

Definição: Expressões Regulares

Constantes:

ε e Ø são expressões regulares

$L(\varepsilon) = \{\varepsilon\}$

$L(\emptyset) = \emptyset$

Se a é um símbolo qualquer, a é uma expressão regular

$L(a) = \{a\}$

União

Se E e F são expressões regulares, E + F é uma expressão regular

$L(E+F) = L(E) \cup L(F)$

Definição: Expressões Regulares

Concatenação

- Se E e F são expressões regulares, EF é uma expressão regular
  - L(EF) = L(E).L(F)

Estrela

- Se E é uma expressão regular, E* é uma expressão regular
  - L(E*) = (L(E))*

Parêntesis

- Se E é uma expressão regular, (E) é uma expressão regular
  - L((E)) = L(E)

Definição: Expressões Regulares

Precedência

Estrela → concatenação → união

Ex: 01*+1

Parêntesis

Mudam a precedência

Ex: (01)*+1 ou 0(1*+1)

Exemplos de expressões regulares

Alfabeto = {0,1}

- 0^*10^* = {w | w contém um único 1}
- 01 + 10 = {01,10}
- $(\varepsilon + 0)1^*$ = {w | w é uma sequência de zero ou mais 1s, começando opcionalmente com 0}
- $(0+1)^*$ = Conjunto das partes do alfabeto ou conjunto de todas as cadeias possíveis sobre o alfabeto, incluindo a cadeia vazia ($|w| \geq 0$)
- $(0+1)(0+1)^*$ = Idem ao exemplo acima, mas sem a cadeia vazia ($|w| \geq 1$)

Exercícios

Escreva expressões regulares correspondentes às seguintes linguagens:

{w | w começa com um 1 e termina com um 0}

Resp: $1(0+1)^*0$

{w | w contém pelo menos três 1s}

Resp: $0^*10^*10^*1(0+1)^*$

{w | o comprimento de w é no máximo 5}

Resp: $(0+1+\varepsilon)(0+1+\varepsilon)(0+1+\varepsilon)(0+1+\varepsilon)(0+1+\varepsilon)$

{w | toda posição ímpar de w é um 1}

Resp: $(1(0+1))^* + 1((0+1)1)^*$ ou $(1(0+1))^*(1+\varepsilon)$

Leis algébricas para expressões regulares

Leis algébricas para expressões regulares

É possível (e muitas vezes necessário) simplificar expressões regulares

Ex: $1^*0 + 1^*0(\varepsilon+0+1)^*(\varepsilon+0+1) = 1^*0(0+1)^*$

Existem algumas leis algébricas que facilitam esse processo

Leis algébricas para expressões regulares

- Associatividade e comutatividade
  - $L+M=M+L$
    - Ex: $0+1 = 1+0$
  - $(L+M)+N=L+(M+N)$
    - Ex: $(a^* + bc) + a = a^* + (bc + a)$
  - $(LM)N=L(MN)$
    - Ex: $(00(1+0))111=00((1+0)111)$

Leis algébricas para expressões regulares

Identidades (elemento neutro) e aniquiladores

$\emptyset+L=L+\emptyset=L$ ($\emptyset$ é identidade para união)

$\varepsilon L=L\varepsilon=L$ (ε é identidade para concatenação)

$\emptyset L=L\emptyset=\emptyset$ ($\emptyset$ é aniquilador para concatenação)

Exs:

$\varepsilon a(b+c)+aa\varepsilon = a(b+c)+aa$

$\emptyset(\varepsilon+1)^*(1+0(01^*10(0+1))) + 01 = \emptyset + 01 = 01$

Leis algébricas para expressões regulares

Leis distributivas

$L(M+N)=LM+LN$

$(M+N)L=ML+NL$

Exs:

$0(0+1) = 00 + 01$

$(0+1)(0+1) = (0+1)0 + (0+1)1$

$(0+1)(0+1) = 0(0+1) + 1(0+1)$

Leis algébricas para expressões regulares

Lei da idempotência

$L+L=L$

Exs:

$(0+1+\varepsilon) + (0+1+\varepsilon)=(0+1+\varepsilon)$

$(0+1+\varepsilon)+01^*0+(\varepsilon+0+1)+(\varepsilon+1+0)=(0+1+\varepsilon)+01^*0$

Leis algébricas para expressões regulares

Leis envolvendo fechamentos

- $(L^*)^* = L^*$
  - Ex: $((01)^*)^* = (01)^*$
- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$

Operadores de fechamento adicionais

- $L^+ = LL^* = L^*L$
- $L^* = L^+ + \varepsilon$
- $L? = \varepsilon + L$

Exemplos de simplificação

0+010

0**ε**+010

0**(ε+10)**

0**(10)?**

ab*b(a+ε)

ab*b**a?**

a$\mathbf{b}^{+}$a?

a+(b+c+ε)a(b+c)+ca+ba

a+**(b+c)?**a(b+c)+ca+ba

a+(b+c)?a(b+c)+**(c+b)**a

εa+(b+c)?a(b+c)+(c+b)a

εa+**(c+b)a+(b+c)?a(b+c)**

(ε+c+b)a+(b+c)?a(b+c)

(b+c)?a+(b+c)?a(b+c)

(b+c)?a**ε**+(b+c)?a(b+c)

(b+c)?a**(ε+b+c)**

(b+c)?a**(b+c)?**

Autômatos finitos e expressões regulares

Autômatos finitos e expressões regulares

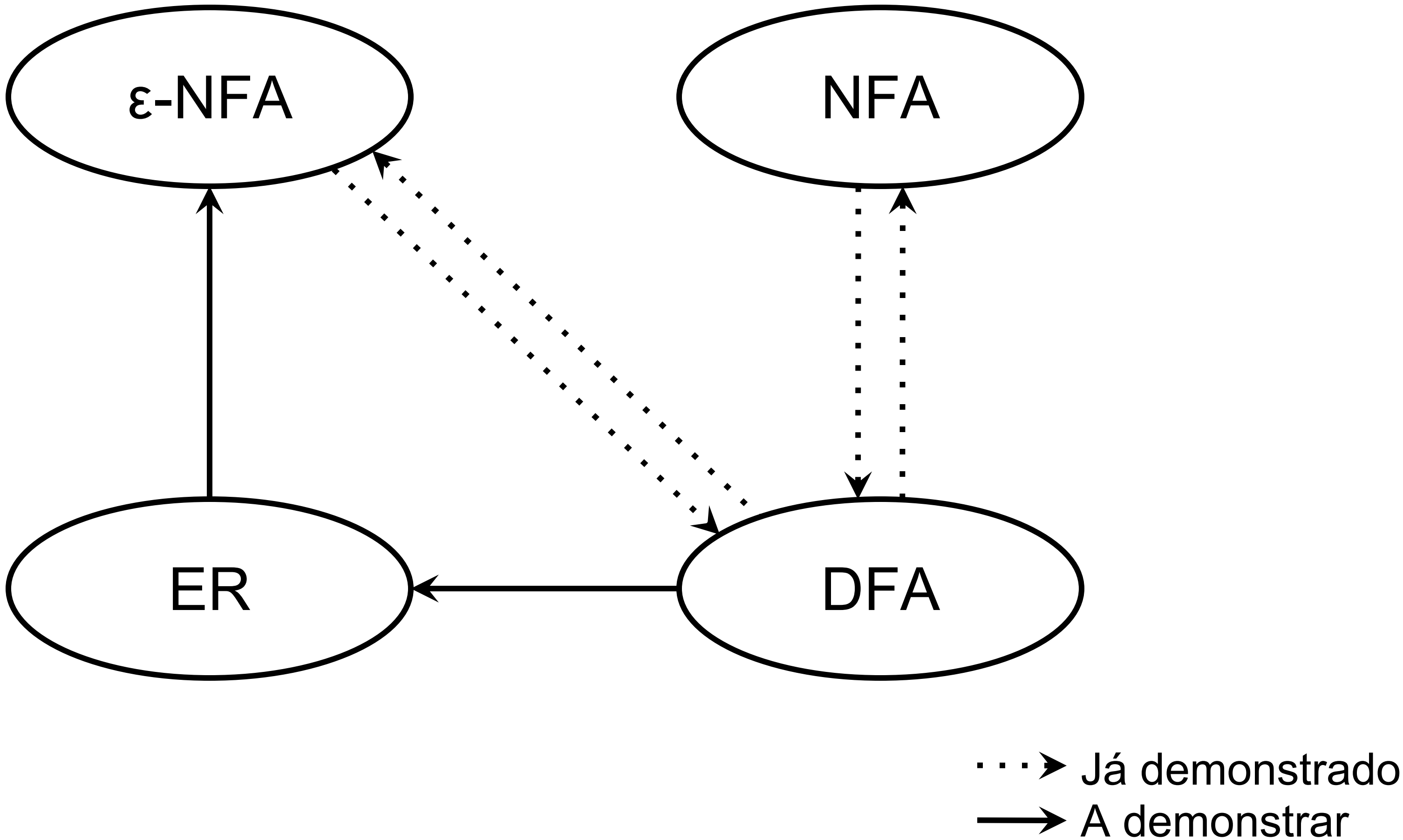
São diferentes na notação

- Mas tanto autômatos finitos como expressões regulares representam exatamente o mesmo conjunto de linguagens
  - Linguagens regulares

Ou seja:

- Toda linguagem definida por um autômato finito também é definida por uma expressão regular
- Toda linguagem definida por uma expressão regular é definida por um autômato finito

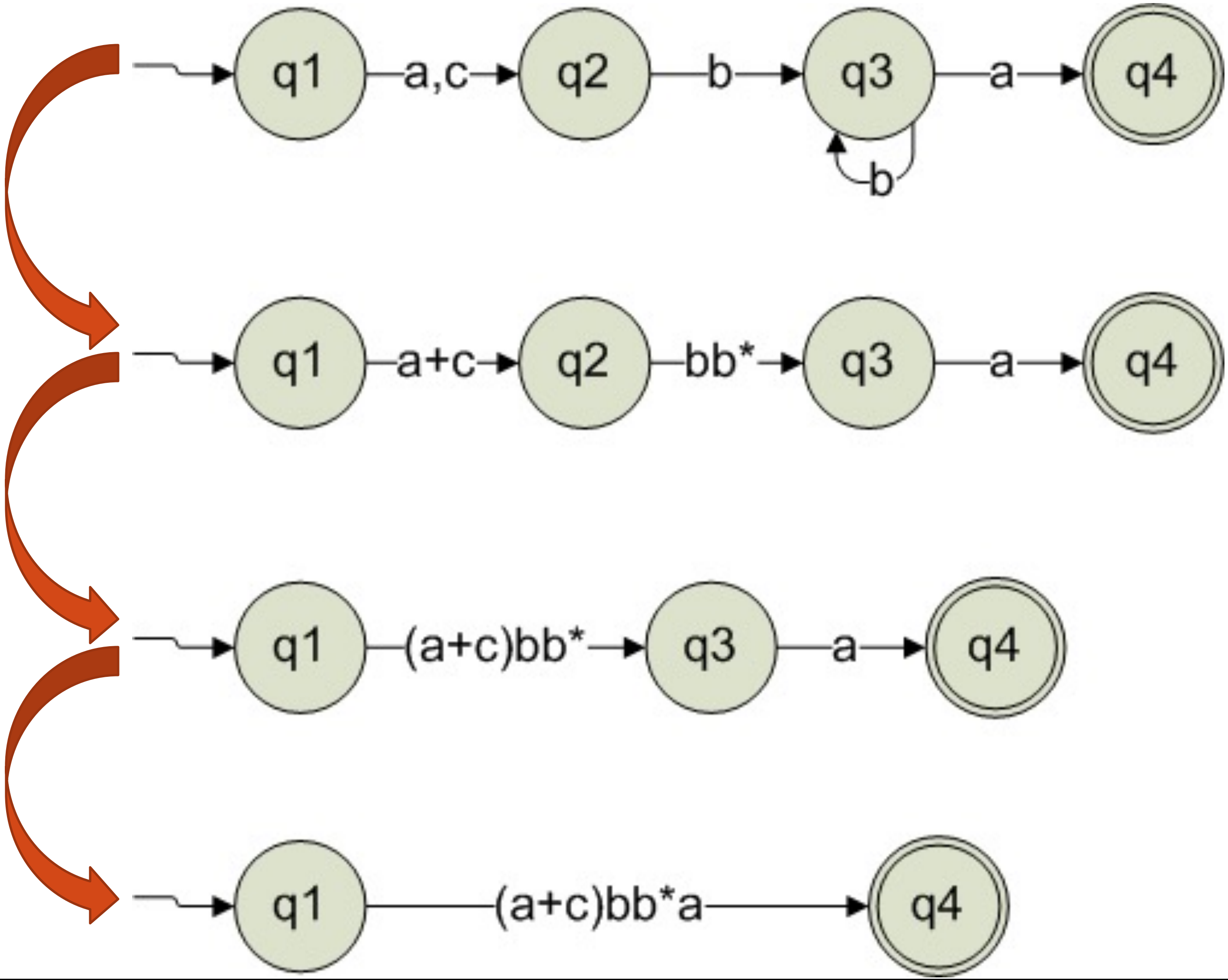
Autômatos finitos e expressões regulares



Conversão DFA→ER

- Teorema: Se L = L(A) para algum DFA A, então existe uma expressão regular R tal que L = L(R)
- Conversão é surpreendentemente complicada
  - Método 1: n^3 expressões, com 4^n símbolos (pior caso)
  - Método 2: eliminação de estados
    - Mais simples, porém também trabalhosa
    - Envolve uma notação mista: autômatos + ERs
      - Autômato finito não-determinístico generalizado
      - Transições são expressões regulares

Autômatos + ERs



Conversão DFA→ER

Método da eliminação de estados

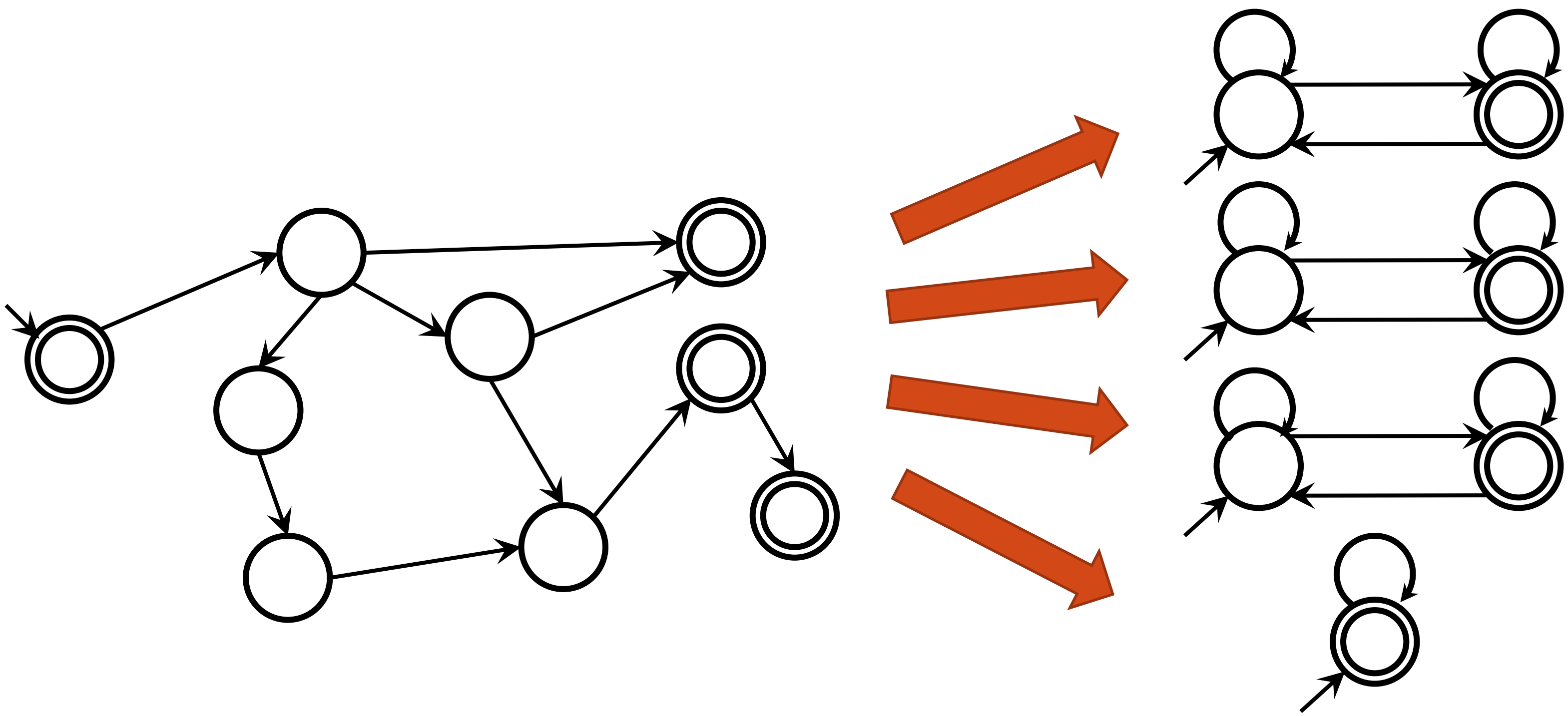
Eliminamos todos os estados, um por um

- Ao eliminar um estado s, todos os caminhos que passam por s não mais existem no autômato

Substituiremos símbolos por ER nas transições, para representar as transições eliminadas

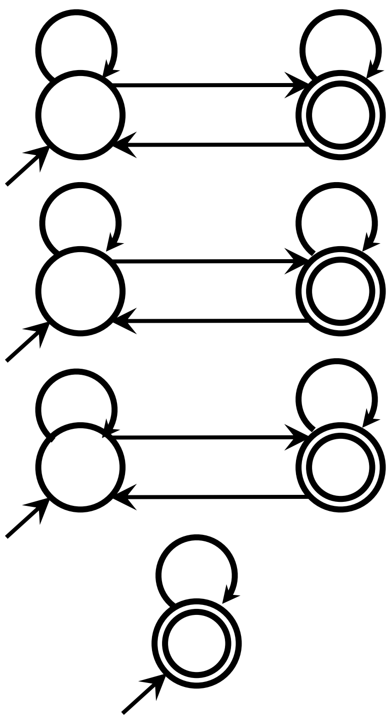
Conversão DFA→ER

- Para cada estado de aceitação q, elimine todos os estados, com exceção de q e q0 (estado inicial)
  - Resultado = um autômato para cada estado de aceitação



Conversão DFA→ER

- Cada autômato terá uma ER equivalente



01*(1+0)*11

0

0+1*

11

1

1(0+ε)10

Conversão DFA→ER

- Basta fazer a união de todas as expressões

$01^*(1+0)^*11$

0

$0+1^*$

11

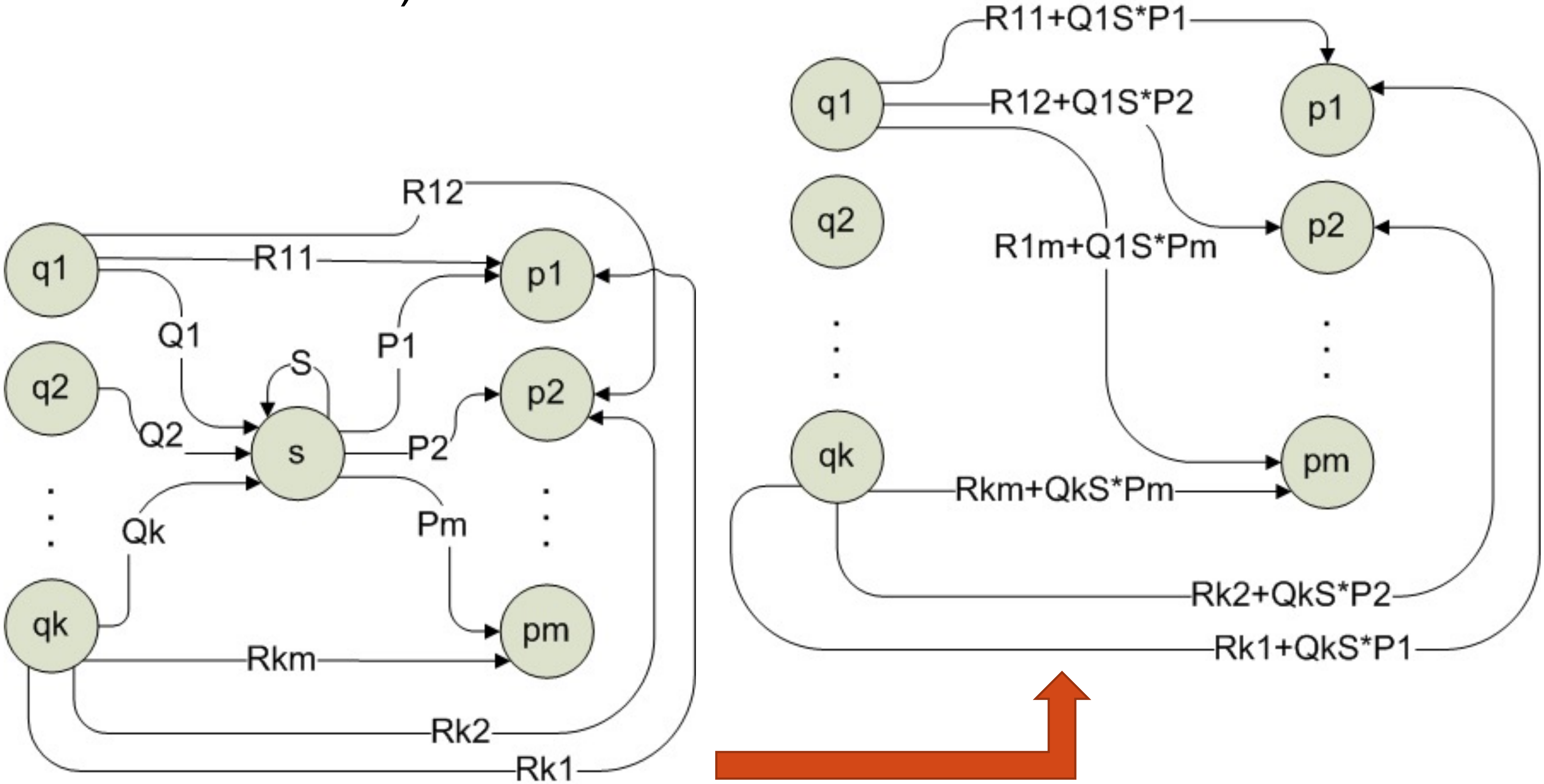
1

$1(0+\varepsilon)10$

$01^*(1+0)^*110 + 0 + 1^* + 111 + 1(0+\varepsilon)10$

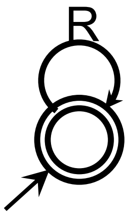
Conversão DFA→ER

- Eliminando um estado s (caso não haja um determinado arco, considerar que existe um arco com rótulo Ø)

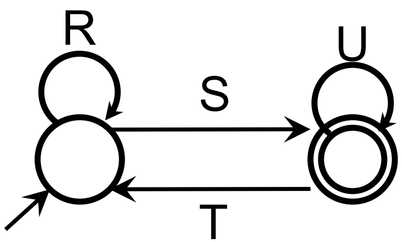


Conversão DFA→ER

- Repetir esse procedimento para todos os estados
- No final, existem duas possibilidades:
  - q0=q
    - Resta um único estado, com uma transição R
    - R é a expressão regular equivalente



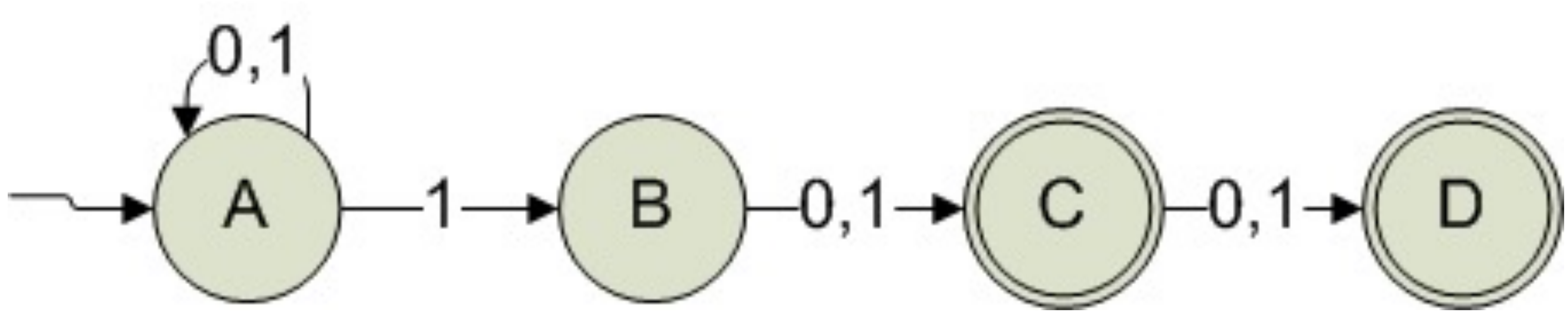
  - q0≠q
    - Restam dois estados, no seguinte formato genérico:



A expressão regular final é (R+SU*T)*SU*

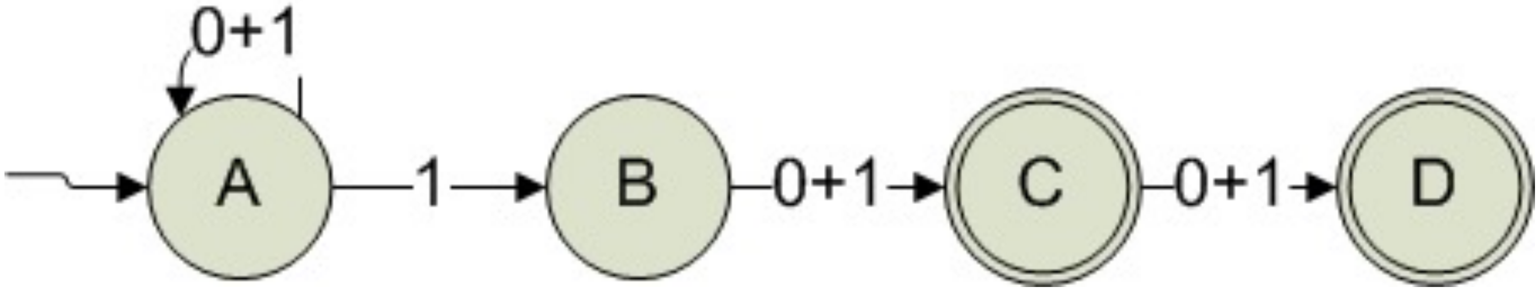
Conversão DFA→ER

- Exemplo: cadeias com símbolo 1 a duas ou três posições a partir do final



Conversão DFA→ER

Primeiro passo, substituir as transições rotuladas 0,1 por 0 + 1



Conversão DFA→ER

Eliminando B (assim dá para reaproveitar em outras reduções)

Q1=1

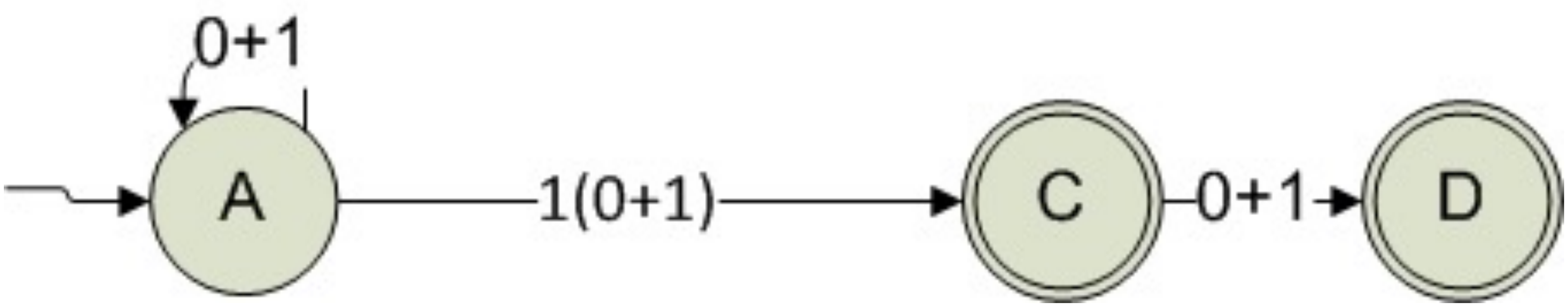
P1=0+1

R11=Ø

S=Ø

Arco de A para C = R11+Q1S*P1 = Ø+1Ø*(0+1)

Simplificando: 1(0+1)



Conversão DFA→ER

Eliminando C

$Q1=1(0+1)$

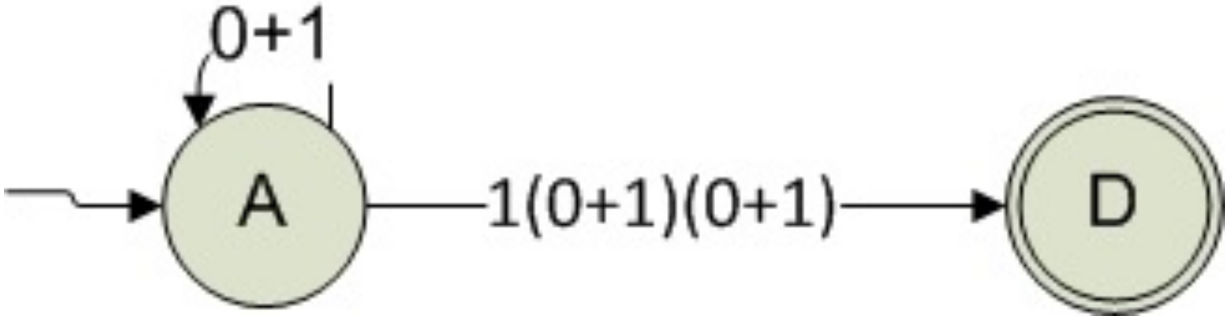
$P1=0+1$

$R11=\emptyset$

$S=\emptyset$

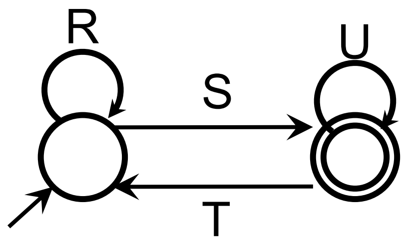
Arco de A para D = $R11+Q1S^*P1 = \emptyset+1(0+1)\emptyset^*(0+1)$

Simplificando: $1(0+1)(0+1)$



Conversão DFA→ER

Restou um autômato de 2 estados



- $R=0+1$
- $S=1(0+1)(0+1)$
- $U=\varnothing$
- $T=\varnothing$

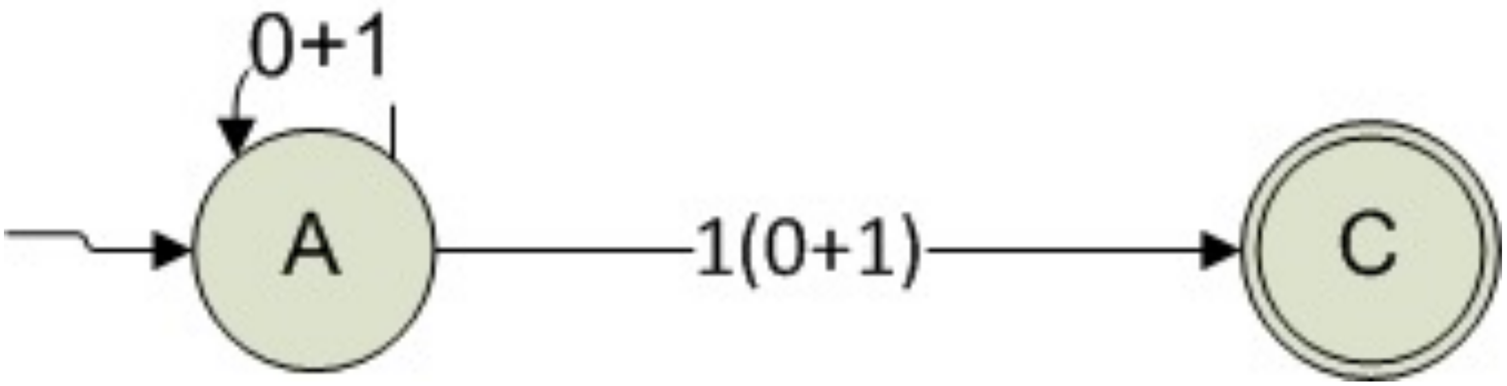
Fórmula: $(R+SU^*T)^*SU^*$

- Resultado: $(0+1+1(0+1)(0+1)\varnothing^*\varnothing)^*1(0+1)(0+1)\varnothing^*$
- Simplificando: $(0+1)^*1(0+1)(0+1)$

Conversão DFA→ER

Eliminando D agora

- Não há sucessor, portanto não haverá mudanças de arcos
- Da mesma forma, restou um autômato de 2 estados
  - Expressão regular resultante: (0+1)*1(0+1)



Conversão DFA→ER

- Haviam dois estados de aceitação
  - Foram obtidos dois autômatos
    - Duas expressões regulares equivalentes
      - A expressão regular final é a união dessas duas:
      - (0+1)*1(0+1)+(0+1)*1(0+1)(0+1)
- Simplificando
  - (0+1)*1(0+1)(0+1)?

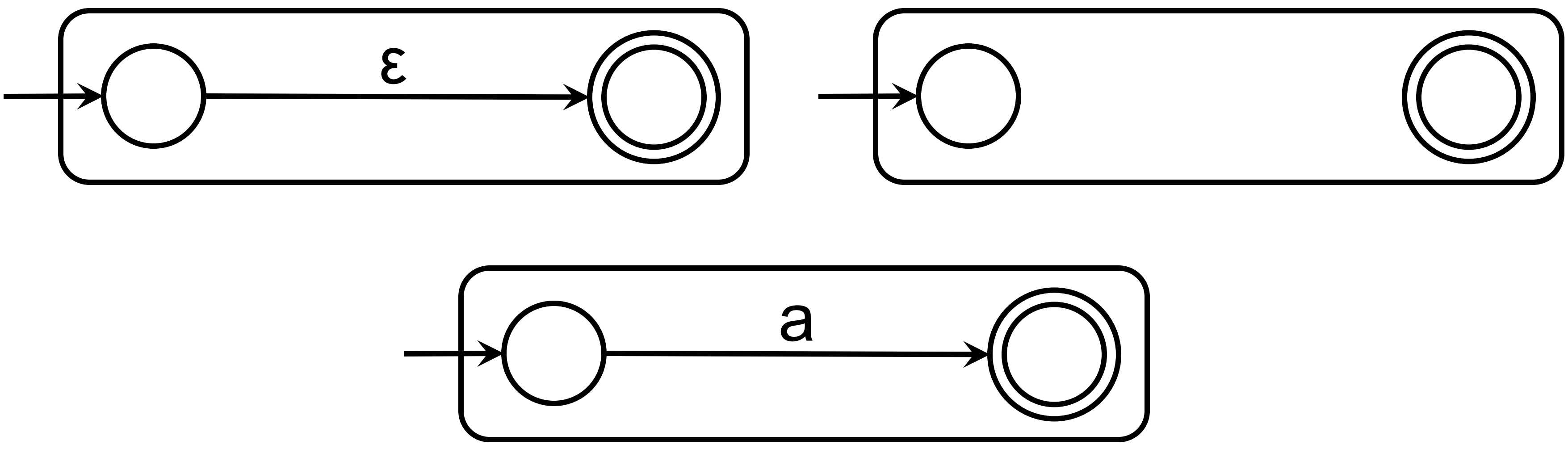
Conversão ER→ε-NFA

Teorema: Toda linguagem definida por uma expressão regular também é definida por um autômato finito

Prova por construção: ER→ε-NFA

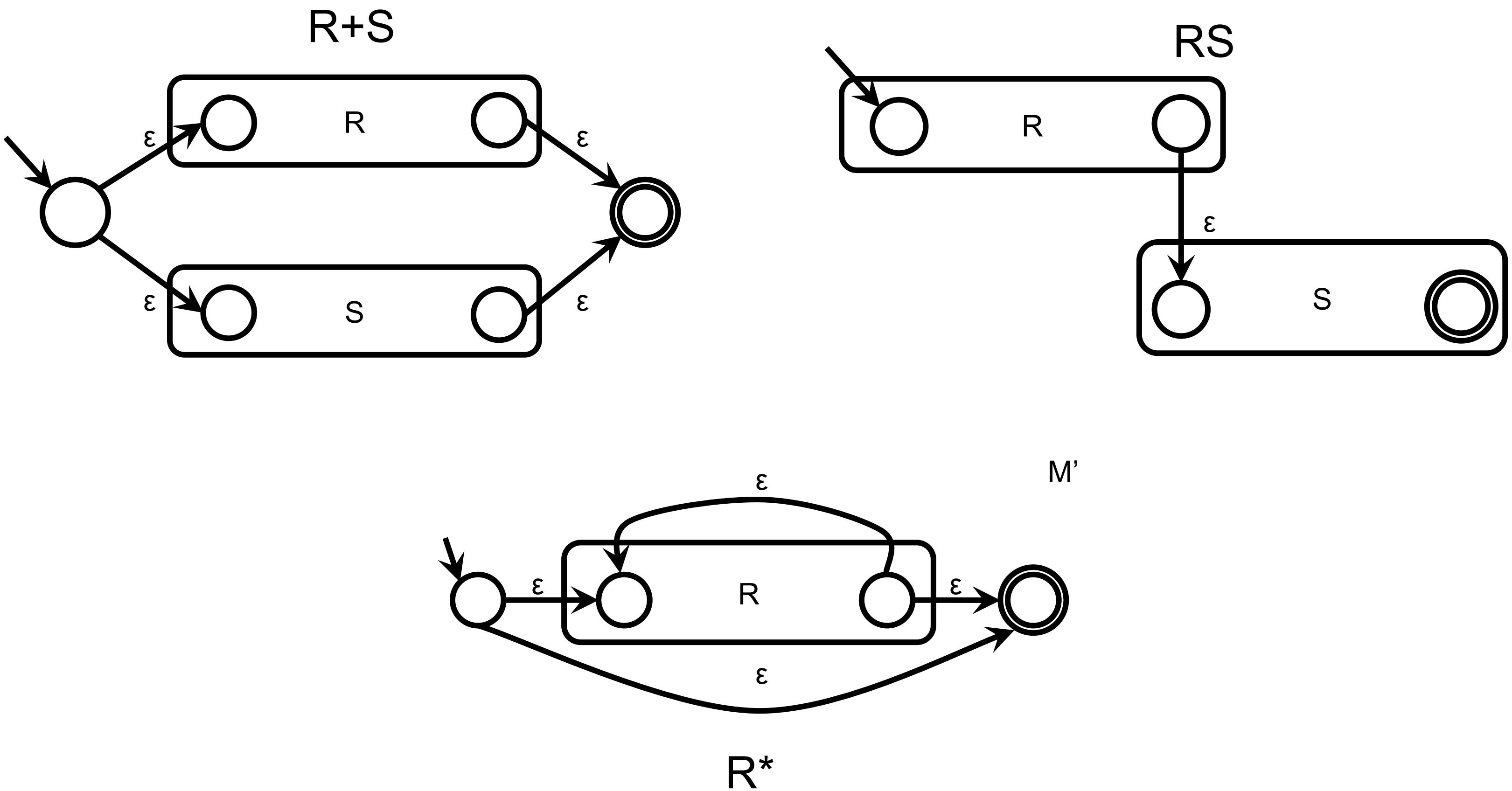
Base + indução

Base: ε, Ø e a (um símbolo qualquer)



Conversão ER→ε-NFA

Indução: Os mesmos autômatos das provas sobre o fechamento das operações regulares



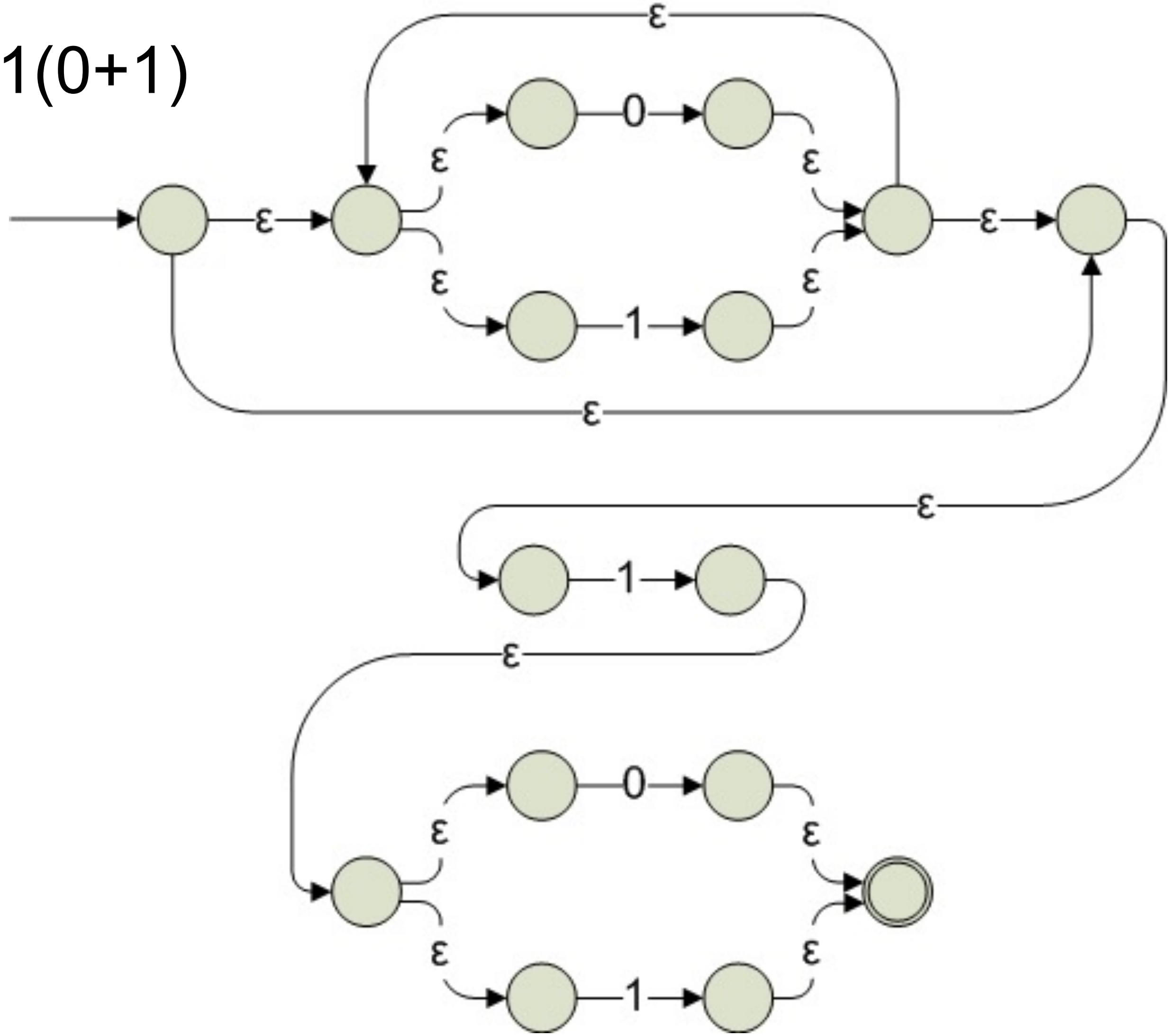
Conversão ER→ε-NFA

Características do ε-NFA:

- Possui exatamente um estado de aceitação
- Nenhum arco chega no estado inicial
- Nenhum arco sai do estado de aceitação

Conversão ER→ε-NFA

- Ex: (0+1)*1(0+1)

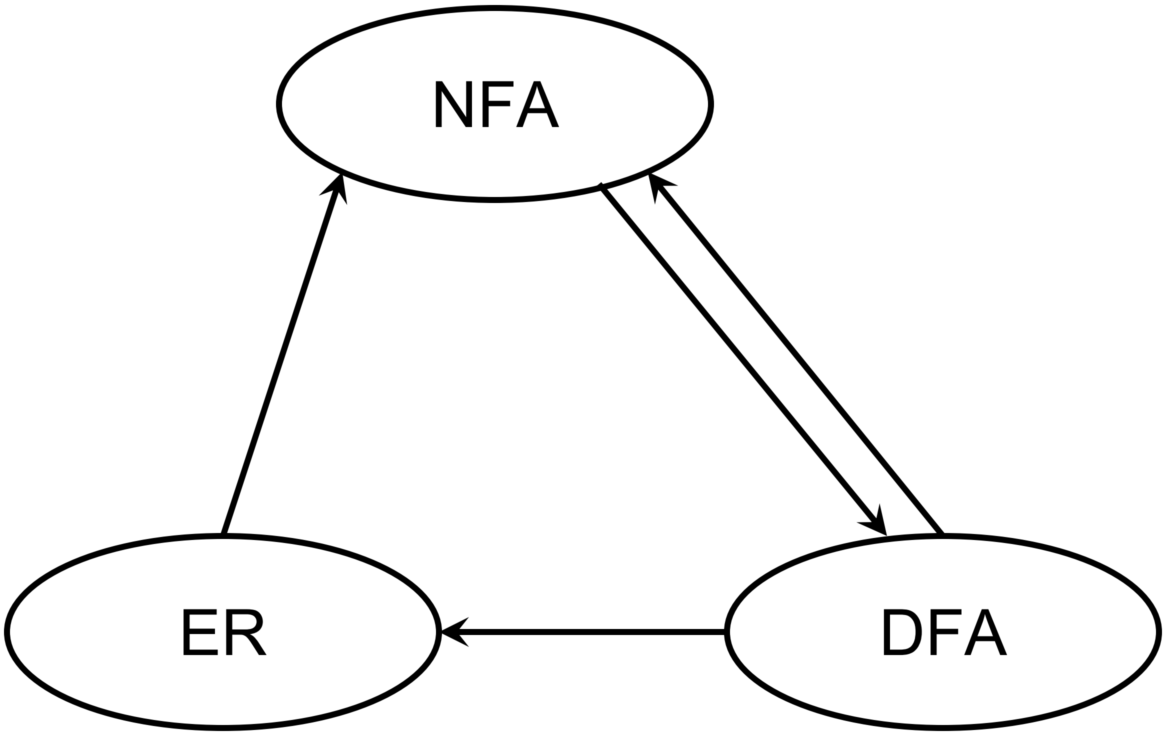


Questões sobre linguagens regulares

Questões sobre linguagens regulares

- Linguagens regulares existem sob muitas formas
  - DFA
  - NFA
  - ε-NFA
  - Expressões regulares
- Cada uma tem suas características
  - DFA = rápida execução
  - NFA (e ε-NFA) = mais fácil projeto, porém execução mais lenta
  - ER = boa legibilidade e projeto fácil
- É possível passar de uma para outra

Conversão entre representações



Fim