SBVLIFA: Linguagens Formais e Autômatos

Aula 06: Propriedades das Linguagens Regulares

Bacharelado em Ciência da Computação
Prof. Dr. David Buzatto

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Linguagens Regulares

- Linguagens Regulares

Tipo	Classe de Linguagens	Modelo de Gramática	Modelo de Reconhecedor
0	Recursivamente enumeráveis	Irrestrita	Máquina de Turing
1	Sensíveis ao contexto	Sensível ao contexto	Máquina de Turing com fita limitada
2	Livres de contexto	Livre de contexto	Autômato de pilha
3	**Regulares**	**Linear (direita ou esquerda)**	**Autômato finito**

Propriedades das Linguagens Regulares

- Veremos nessa aula algumas propriedades importantes das linguagens regulares, a saber:
  - **Lema do Bombeamento (*Pumping Lemma*):** provar que certas linguagens não são regulares;
  - **Propriedades de Fechamento:** ferramenta para construção de autômatos complexos;
  - **Propriedades de Decisão:** equivalência entre linguagens com minimização de autômatos.

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

Teorema: (*O lema do bombeamento para linguagens regulares*) Seja L uma linguagem regular. Então, existe uma constante n, que depende de L, tal que, para toda string w em L sendo $|w| \geq n$, podemos dividir w em três strings, $w = xyz$, tais que:

1. $y \neq \varepsilon$
2. $|xy| \leq n$
3. Para todo $k \geq 0$, a string $xy^k z$ também está em L

- Isto é, sempre podemos encontrar uma string não vazia y não muito longe do início de w que pode ser "bombeada", ou seja, podemos repetir y qualquer número de vezes, ou excluí-la, o caso de $k = 0$, mantendo a string resultante em L.

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

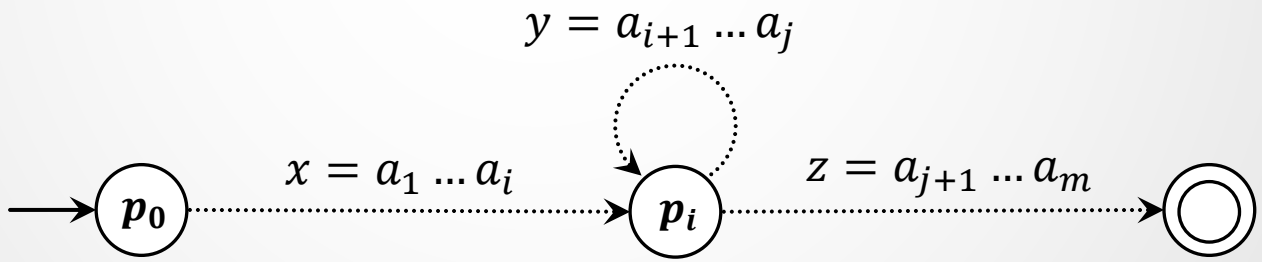
Prova: Suponha que L seja regular. Então, $L = L(A)$ para algum DFA A. Suponha que A tenha n estados. Agora, considere qualquer string w de comprimento n ou maior, digamos $w = a_1 a_2 \dots a_m$, onde $m \geq n$ e cada a_i é um símbolo de entrada. Para $i = 0, 1, \dots, n$, defina o estado p_i como $\hat{\delta}(q_0, a_1 a_2 \dots a_i)$, onde δ é a função de transição de A e q_0 é o estado inicial de A. Isto é, p_i é o estado em que A se encontra depois de ler os primeiros i símbolos de w. Note que $p_0 = q_0$. Não é possível que os $n + 1$ diferentes p_i's para $i = 0, 1, \dots, n$ sejam distintos, pois só existem n estados diferentes. Desse modo, podemos encontrar dois inteiros diferentes i e j, com $0 \leq i < j \leq n$, tais que $p_i = p_j$. Agora, podemos dividir $w = xyz$ como a seguir:

1. $x = a_1 a_2 \dots a_i$
2. $y = a_{i+1} a_{i+2} \dots a_j$
3. $z = a_{j+1} a_{j+2} \dots a_m$

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

Ou seja, x nos leva a p_i uma vez; y nos leva de p_i de volta a p_i (pois p_i também é p_j) e z é o restante de w. Os relacionamentos entre as strings e os estados são sugeridos no diagrama abaixo. Observe que x pode ser vazio, no caso em que $i = 0$. Além disso, z pode ser vazio se $j = n = m$. Entretanto, y não pode ser vazio, pois i é estritamente menor que j.



Toda string mais longa que o número de estados deve causar a repetição de um estado

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

Agora, considere o que acontece se o autômato A recebe a entrada xy^kz para qualquer $k \geq 0$. Se $k = 0$, então o autômato vai do estado inicial q_0 (que também é p_0) para p_i na entrada x. Tendo em vista que p_i também é p_j, A deve ir de p_i para o estado de aceitação, mostrado no diagrama anterior, para a entrada z. Desse modo, A aceita xz.

Se $k > 0$, então A vai de q_0 para p_i sobre a entrada x, circula de p_i para p_i k vezes para a entrada y^k, e depois vai para o estado de aceitação para a entrada z. Desse modo, para qualquer $k \geq 0$, xy^kz também é aceito por A; ou seja, xy^kz está em L. □

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

Exemplo 1: Vamos mostrar que L_{eq}, que consiste em todas as strings com um número igual de 0's e 1's (não em qualquer ordem específica), isto é $L_{eq} = \{01, 0011, 000111, 00001111, \ldots\}$, não é uma linguagem regular. Pensaremos na prova como um jogo de dois oponentes. Seremos o jogador 1 e teremos que lidar com as escolhas que o jogador 2 fizer. Suponha que n seja a constante que deve existir se L_{eq} é regular, de acordo com o lema do bombeamento; isto é o jogador 2 escolhe n. Escolheremos $w = 0^n1^n$, isto é, n 0's seguidos por n 1's, uma string que sem dúvida está em L_{eq}.

Agora, o jogador 2 desmembra nosso w em xyz. Tudo que sabemos é que $y \neq \varepsilon$ e $|xy| \leq n$, no entanto, essas informações são muito úteis, e "ganhamos" da maneira indicada a seguir.

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

Tendo em vista que $|xy| \leq n$, e xy vem na frente de w, sabemos que x e y consistem apenas em símbolos 0. O lema do bombeamento nos diz que xz está em L_{eq}, se L_{eq} é regular. Essa conclusão é o caso $k = 0$ no lema do bombeamento. Porém, xz tem n 1's, pois todos os 1's de w estão em z. Entretanto, xz também tem menos de n 0's, porque perdemos os 0's de y. Como $y \neq \varepsilon$, sabemos que não pode haver mais de que $n - 1$ 0's entre x e z. Assim, depois de supor que L_{eq} é uma linguagem regular, provados um fato conhecido como falso, de que xz está em L_{eq}. Temos uma prova por contradição de que L_{eq} não é regular. □

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

Exemplo 2: Vamos mostrar que L_{pr}, que consiste em todas as strings de 1's cujos comprimentos são primos, não é uma linguagem regular.

Primeiramente, suporemos que esta linguagem é regular. Sendo assim, há uma constante n que satisfaz as condições do lema do bombeamento. Considere algum primo $p \geq n + 2$ que deve existir, dado que existe uma infinidade de primos. Considere $w = 1^p$.

Pelo lema do bombeamento, podemos quebrar $w = xyz$, tal que $y \neq \varepsilon$ e $|xy| \leq n$. Considerando $|y| = m$, então $|xz| \leq p - m$. Agora considere a string $xy^{p-m}z$ que deve estar em L_{pr} pelo lema do bombeamento se L_{pr} for regular. Entretanto,

$$|xy^{p-m}z| = |xz| + (p-m)|y| = p - m + (p-m)m = (m+1)(p-m)$$

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

Parece que $|xy^{p-m}z|$ não é um primo, visto que possui dois fatores $(m+1)$ e $(p-m)$. Entretanto, precisamos confirmar que nenhum desses fatores é 1, visto que $(m+1)(p-m)$ poderia ser um primo. No entanto, $m+1>1$, porque $y \neq \varepsilon$ nos diz que $m \geq 1$. Também temos que $p-m>1$, visto que escolhemos $p \geq n+2$ e que $m \leq n$ pois,

$$m = |y| \leq |xy| \leq n$$

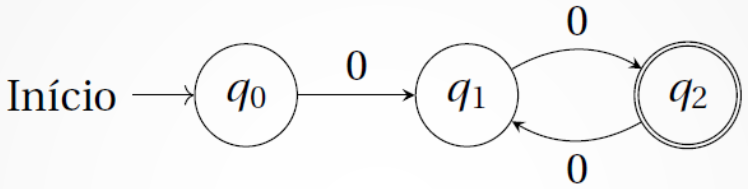
Sendo assim, $p-m \geq 2$.

Novamente, começamos assumindo que essa linguagem é regular e derivamos uma contradição ao mostrar que alguma string que não está na linguagem era obrigada, pelo lema do bombeamento, a estar na linguagem. Desse modo, concluímos que L_{pr} não é uma linguagem regular. □

Propriedades das Linguagens Regulares

O Lema do Bombeamento para Linguagens Regulares

Exemplo 3: Provar que $L = \{0^i \mid i > 0 \text{ e } i \text{ é par}\}$ **é** regular. Será que é possível? Sabemos que L é regular, pois existe um DFA A tal que $L = L(A)$:



Vamos supor que L não é regular. Pelo lema do bombeamento, temos:

1. Escolhemos n, digamos $n = 3$;
2. Escolhemos w, digamos $w = 0000$;
3. Dividimos w em xyz, obedecendo ao lema do bombeamento:
   - $w = xyz$
   - $x = 0$, $|xy| \leq n$
   - $y = 0$, $y \neq \varepsilon$
   - $z = 00$
4. Para todo k, $w = xy^kz$ deve estar em L. Escolhemos k, digamos $k = 2$:

   $w = xy^kz = 00^200 = 00000$, ou seja,

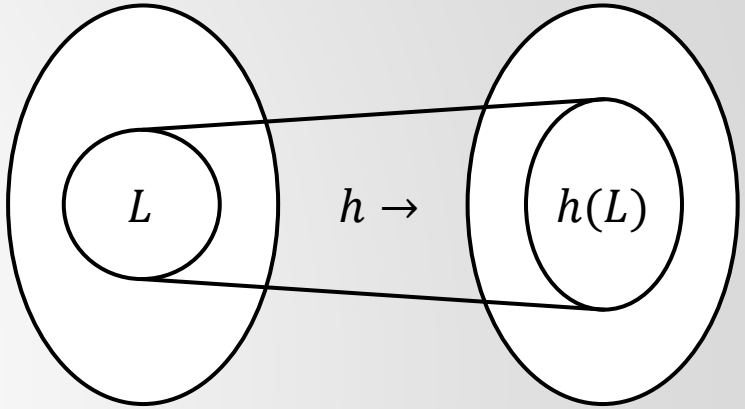
   $|w| = 5$ (ímpar), mas não podemos afirmar que L não é regular pelo lema do bombeamento, visto que L é regular, pois $L = L(A)$! **Sendo assim, não se pode aplicar o lema do bombeamento para provar que uma linguagem é regular.**

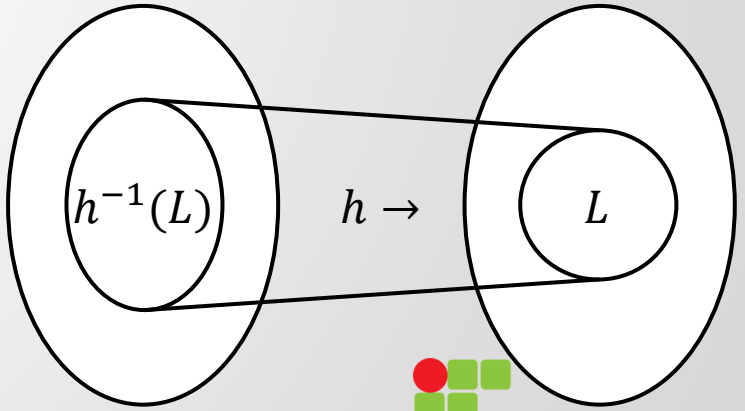
Propriedades das Linguagens Regulares
Propriedades de Fechamento das Linguagens Regulares

- Fechamento sob operações booleanas:
  - A união de duas linguagens regulares é regular;
  - A intersecção de duas linguagens regulares é regular;
  - O complemento de uma linguagem regular é regular;
  - A diferença de duas linguagens regulares é regular;
- O reverso de uma linguagem regular é regular;
  - Se $w = abc$, $w^R = cba$
  - Se $L = \{0, 10, 101, 1010\}$, $L^R = \{0, 01, 101, 0101\}$
- O fechamento (estrela) de uma linguagem regular é regular;
- A concatenação de linguagens regulares é regular;

Propriedades das Linguagens Regulares
Propriedades de Fechamento das Linguagens Regulares

- Um homomorfismo (substituição de símbolos por strings) de uma linguagem regular é regular;
  - Se $L = \{0, 1, 01, 10, 0101\}$ e h é um homomorfismo sobre Σ definido como:
    - $h(0) = ab$
    - $h(1) = \varepsilon$
  - Então $h(L) = \{\varepsilon, ab, abab\}$
- O homomorfismo inverso de uma linguagem regular é regular;
  - Se $L = \{\varepsilon, ab, abab\}$ e h é um homomorfismo sobre Σ definido como:
    - $h(0) = ab$
    - $h(1) = \varepsilon$
  - Então $h^{-1}(L) = 1^*(\varepsilon + 01^*)(\varepsilon + 01^*)$





Propriedades das Linguagens Regulares
Propriedades de Decisão das Linguagens Regulares

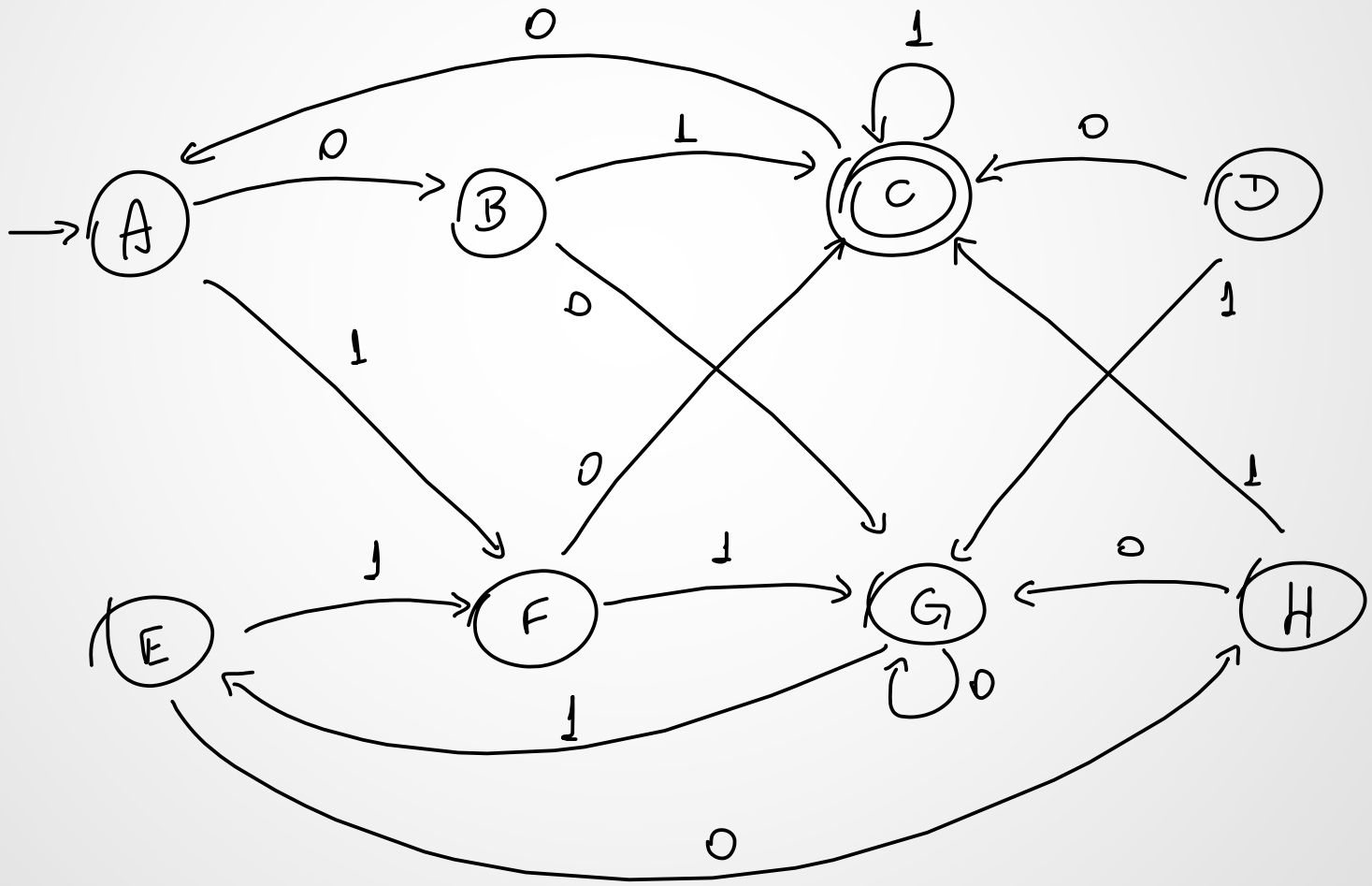
- Conversão entre representações:
  - NFA ou ε-NFA → DFA;
  - DFA → NFA ou ε-NFA;
  - DFA, NFA ou ε-NFA → RE;
  - RE → ε-NFA;
- Teste de caráter vazio de linguagens regulares;
  - Se não há caminho entre o estado inicial até algum estado de aceitação ou todos;
- Teste de pertinência em linguagens regulares;
  - Se há caminho do estado inicial até algum estado de aceitação;
- Equivalência e minimização de autômatos.

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Equivalência de Estados:** entender quando dois estados distintos p e q podem ser substituídos por um único estado que se comporte como p e q.
- Dois estados distintos p e q são equivalentes se:
  - Para todas as strings de entrada w, $\hat{\delta}(p,w)$ é um estado de aceitação se e somente se $\hat{\delta}(q,w)$ é um estado de aceitação.
- Se dois estados não são equivalentes, então dizemos que eles são distinguíveis, isto é, o estado p é distinguível do estado q se existe pelo menos uma string w tal que um estado entre $\hat{\delta}(p,w)$ e $\hat{\delta}(q,w)$ é de aceitação e o outro é de não aceitação.

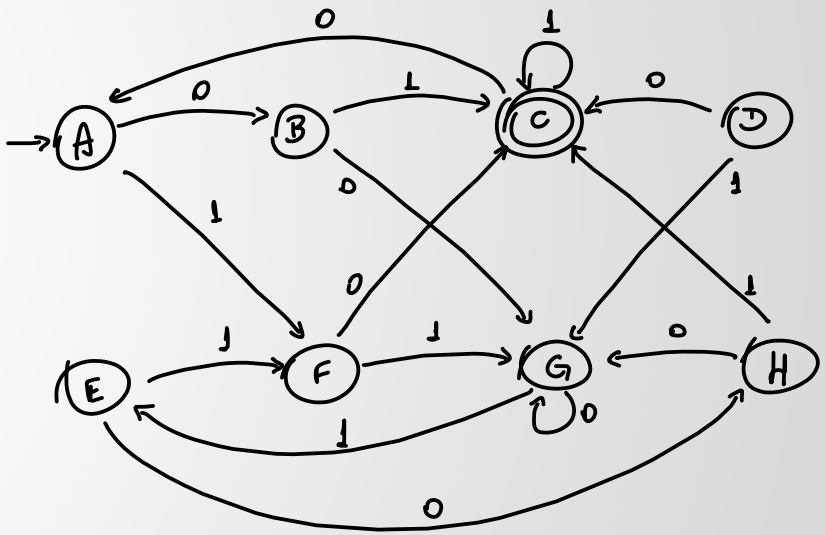
Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Exemplo:**



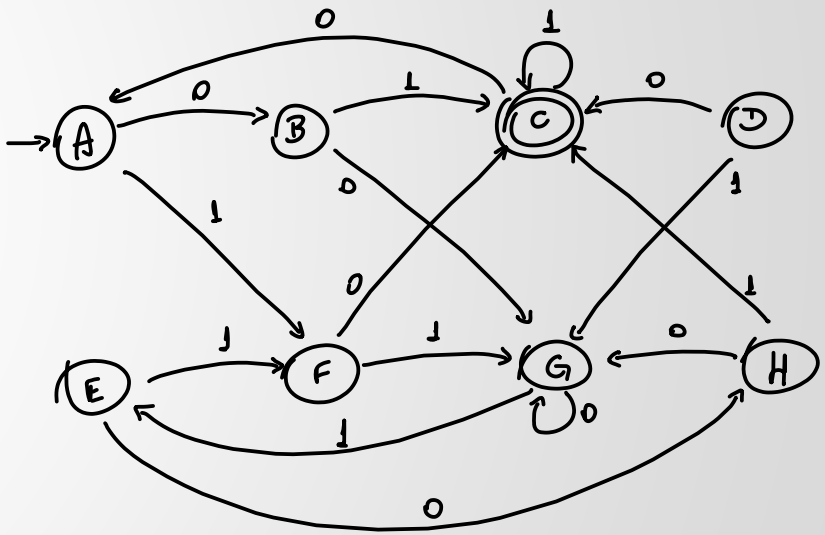
Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Exemplo:**
  - C e G:
    - Não são equivalentes pois um deles é de aceitação (C) e o outro não (G).



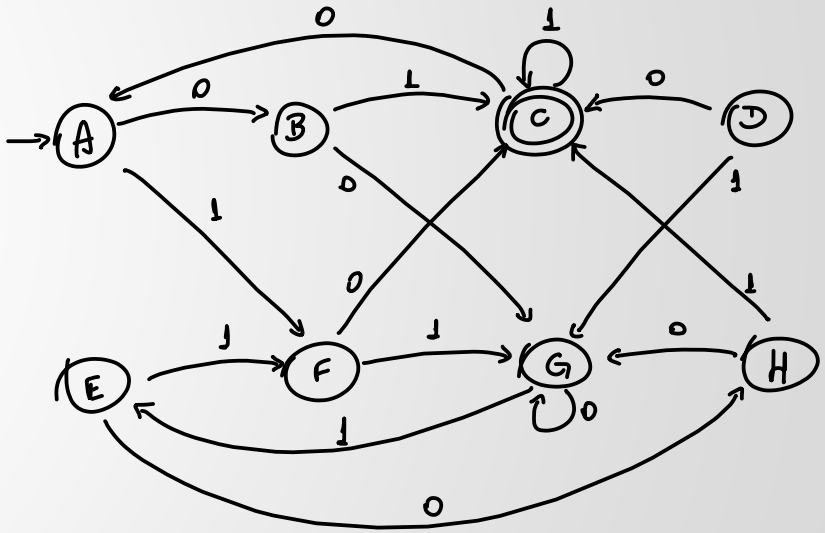
Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Exemplo:**
  - A e G:
    - A string ε não os distingue, porque ambos são estados de não-aceitação;
    - A string 0 não os distingue, porque eles vão para os estados B e G, respectivamente para a entrada 0, e ambos são estados de não-aceitação;
    - A string 1 não os distingue, porque eles vão para os estados F e E, respectivamente para a entrada 1, e ambos são estados de não-aceitação;
    - A string 01 os distingue, porque $\hat{\delta}(A, 01) = \mathrm{C}$ e $\hat{\delta}(G, 01) = \mathrm{E}$, sendo que C é de aceitação e E não é. Qualquer string de entrada que leve A e C a estados em que somente um é de aceitação é suficiente para provar que A e G não são equivalentes.



Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Exemplo:**
  - A e E:
    - A string ε não os distingue, porque ambos são estados de não-aceitação;
    - Para a entrada 1, ambos vão para o estado F. Desse modo nenhuma string de entrada que comece com 1 pode distinguir A de E pois, para qualquer string x, $\hat{\delta}(A, 1x) = \hat{\delta}(E, 1x)$;
    - Para entradas que começam com 0, eles vão para os estados B e H respectivamente. Como nenhum deles é de aceitação, a string 0 sozinha não distingue A de E. Entretanto, B e H não ajudam, pois para a entrada 1, ambos vão para C e para a entrada 0 ambos vão para G. Portanto, todas as entradas que começarem com 0 deixarão de distinguir A de E. Como não há strings que os distinguem, podemos concluir que A e E são equivalentes.



Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **O algoritmo de preenchimento de tabela:** Descoberta recursiva de pares distinguíveis em um DFA $A = (Q, \Sigma, \delta, q_0, F)$, sendo definido como:
  - **Base:** Se p é um estado de aceitação e q é de não aceitação, então o par $\{p, q\}$ é distinguível.
  - **Indução:** Sejam p e q estados tais que, para algum símbolo de entrada a, $r = \delta(p, a)$ e $s = \delta(q, a)$ formam um par de estados conhecidos por serem distinguíveis. Então, $\{p, q\}$ é um par de estados distinguíveis. A razão para essa regra fazer sentido é que deve haver alguma string w que faça distinção entre r e s, isto é, exatamente um estado entre $\hat{\delta}(r, w)$ e $\hat{\delta}(s, w)$ é de aceitação. Então, o string aw deve distinguir p de q, pois $\hat{\delta}(p, aw)$ e $\hat{\delta}(q, aw)$ constituem o mesmo par de estados que $\hat{\delta}(r, w)$ e $\hat{\delta}(s, w)$.

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Exemplo de execução do algoritmo de preenchimento de tabela:**
  - x → pares distinguíveis;
  - vazio → pares equivalentes;
  - **Executando a Base:** C é o único estado de aceitação, então marcamos com x cada par que envolve C.

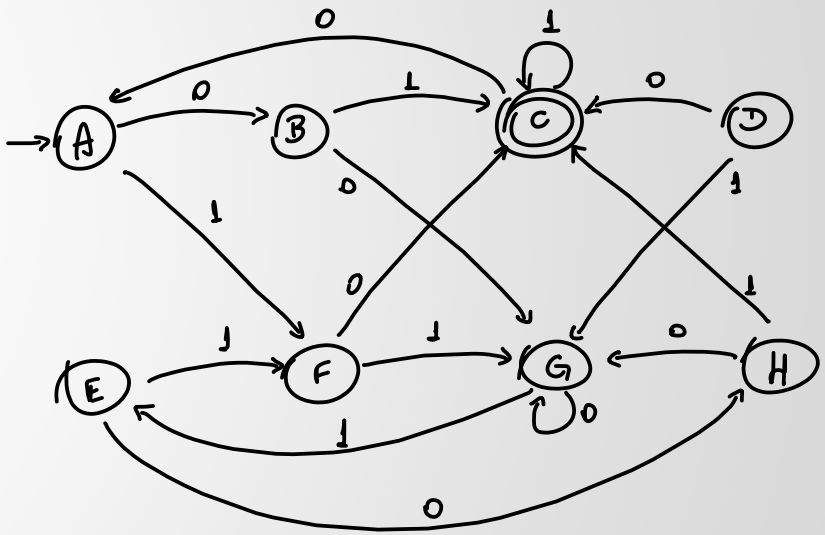
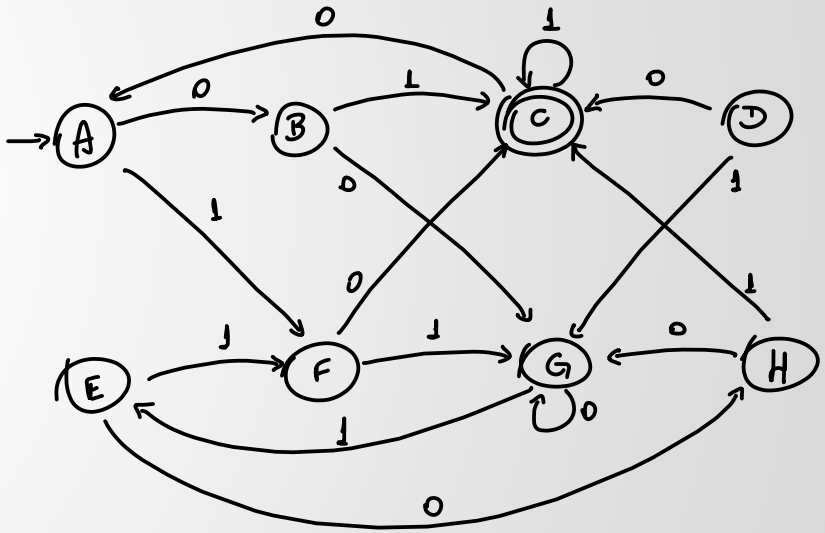


Tabela de distinções

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

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

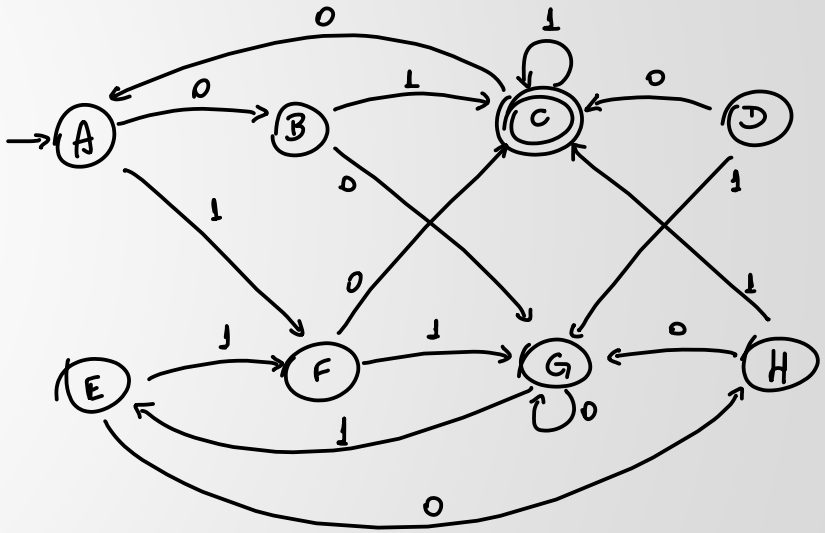
- **Exemplo de execução do algoritmo de preenchimento de tabela:**
  - x → pares distinguíveis;
  - vazio → pares equivalentes;
  - **Executando a Base:** C é o único estado de aceitação, então marcamos com x cada par que envolve C.



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

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

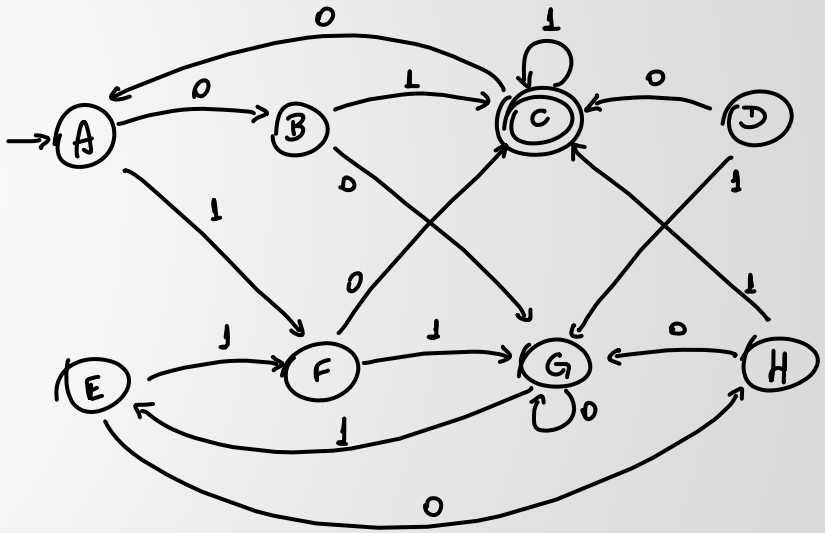
- **Exemplo de execução do algoritmo de preenchimento de tabela:**
  - x → pares distinguíveis;
  - vazio → pares equivalentes;
  - **Executando a Indução:** Para cada par restante, avaliamos suas transições e aumentamos a string testada um símbolo por vez, se necessário, até completar a tabela.



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

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

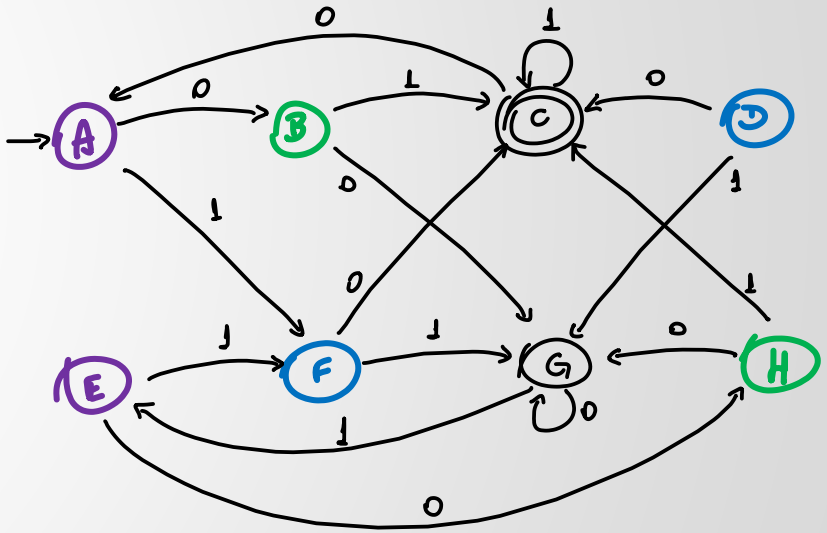
- **Exemplo de execução do algoritmo de preenchimento de tabela:**
  - x → pares distinguíveis;
  - vazio → pares equivalentes;
  - **Executando a Indução:** Para cada par restante, avaliamos suas transições e aumentamos a string testada um símbolo por vez, se necessário, até completar a tabela.



	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

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Exemplo de execução do algoritmo de preenchimento de tabela:**
  - x → pares distinguíveis;
  - vazio → pares equivalentes;
  - **Executando a Indução:** Para cada par restante, avaliamos suas transições e aumentamos a string testada um símbolo por vez, se necessário, até completar a tabela.



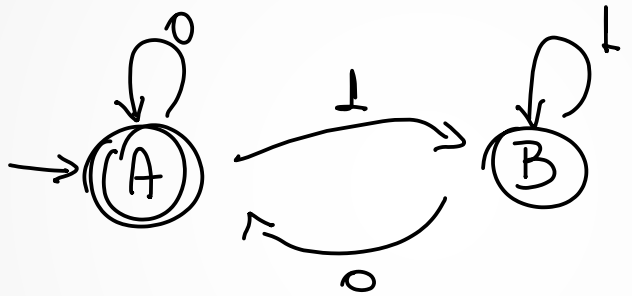
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

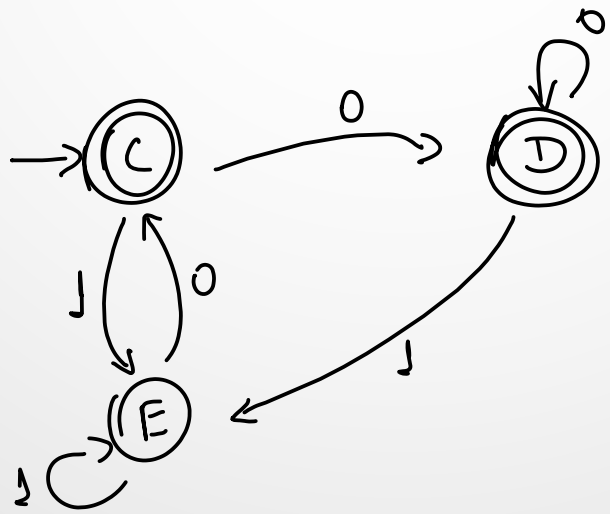
Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Equivalência de Linguagens Regulares:** verificar se duas representações de linguagens regulares representam a mesma linguagem.
- Dadas duas linguagens regulares, L e M, primeiramente convertemos suas representações para DFA's, se necessário, visto que uma, ou ambas, já podem estar representadas na forma de um DFA.
- Agora imaginamos um DFA composto pela união dos DFA's obtidos anteriormente. Tecnicamente, este DFA tem dois estados iniciais, mas para o propósito do teste de equivalência de estados isso é irrelevante, sendo assim, tomamos um dos estados iniciais como o estado inicial do autômato da união.
- Testamos assim se o estados iniciais dos dois DFA's originais são equivalentes, usando o algoritmo de preenchimento de tabelas. Se forem equivalentes, isso implica que $L = M$, caso contrário, $L \neq \mathrm{M}$.

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

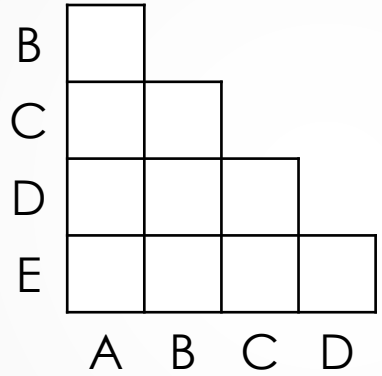
- **Exemplo:** Dados os dois DFA's abaixo, considere-os como apenas um DFA. Esses DFA's aceitam a string vazia e todas as strings que terminam em 0, ou seja, algo como $\varepsilon + (0+1)^*0$.

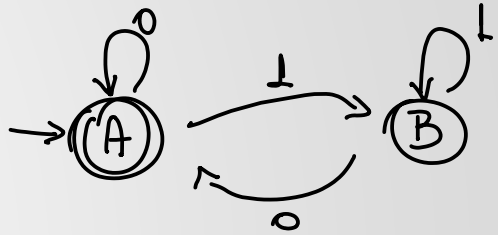


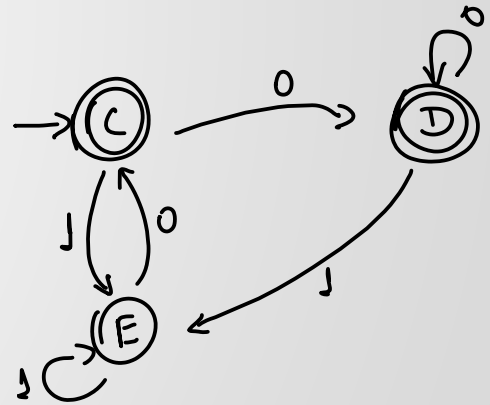


Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- Começamos marcando com **x** todos os pares de estados em que exatamente um dos estados é de aceitação.



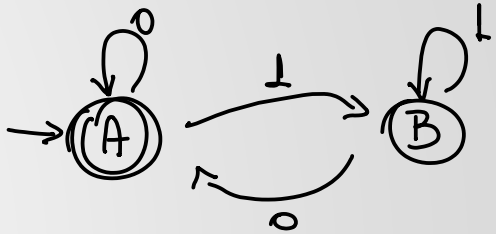


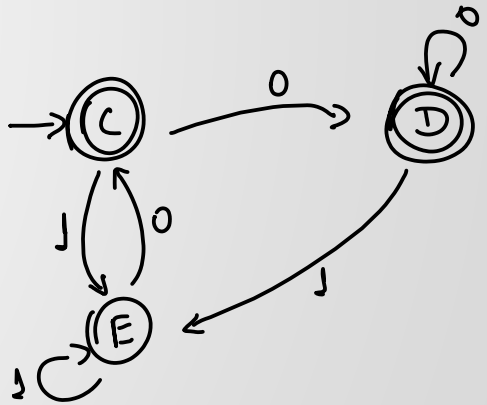


Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- Começamos marcando com x todos os pares de estados em que exatamente um dos estados é de aceitação.

	A	B	C	D
B	x			
C		x		
D		x		
E	x		x	x

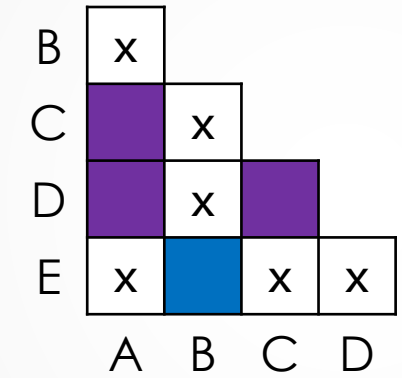


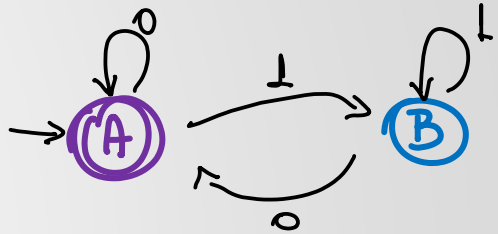


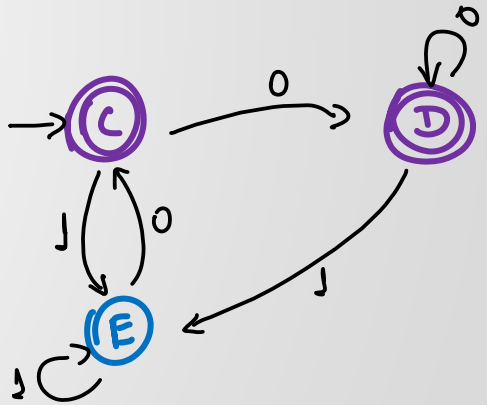
- Nesse caso, não há mais o que fazer, visto que os quatro pares restantes, $\{A, C\}$, $\{A, D\}$, $\{C, D\}$ e $\{B, E\}$ são equivalentes. Como os estados A e C são equivalentes e eles são os estados iniciais dos dois autômatos, podemos concluir que esses DFA's aceitam a mesma linguagem.

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- Começamos marcando com x todos os pares de estados em que exatamente um dos estados é de aceitação.







Transitividade!
Teorema 4.23, página 172
(HOPCROFT et al., 2003)

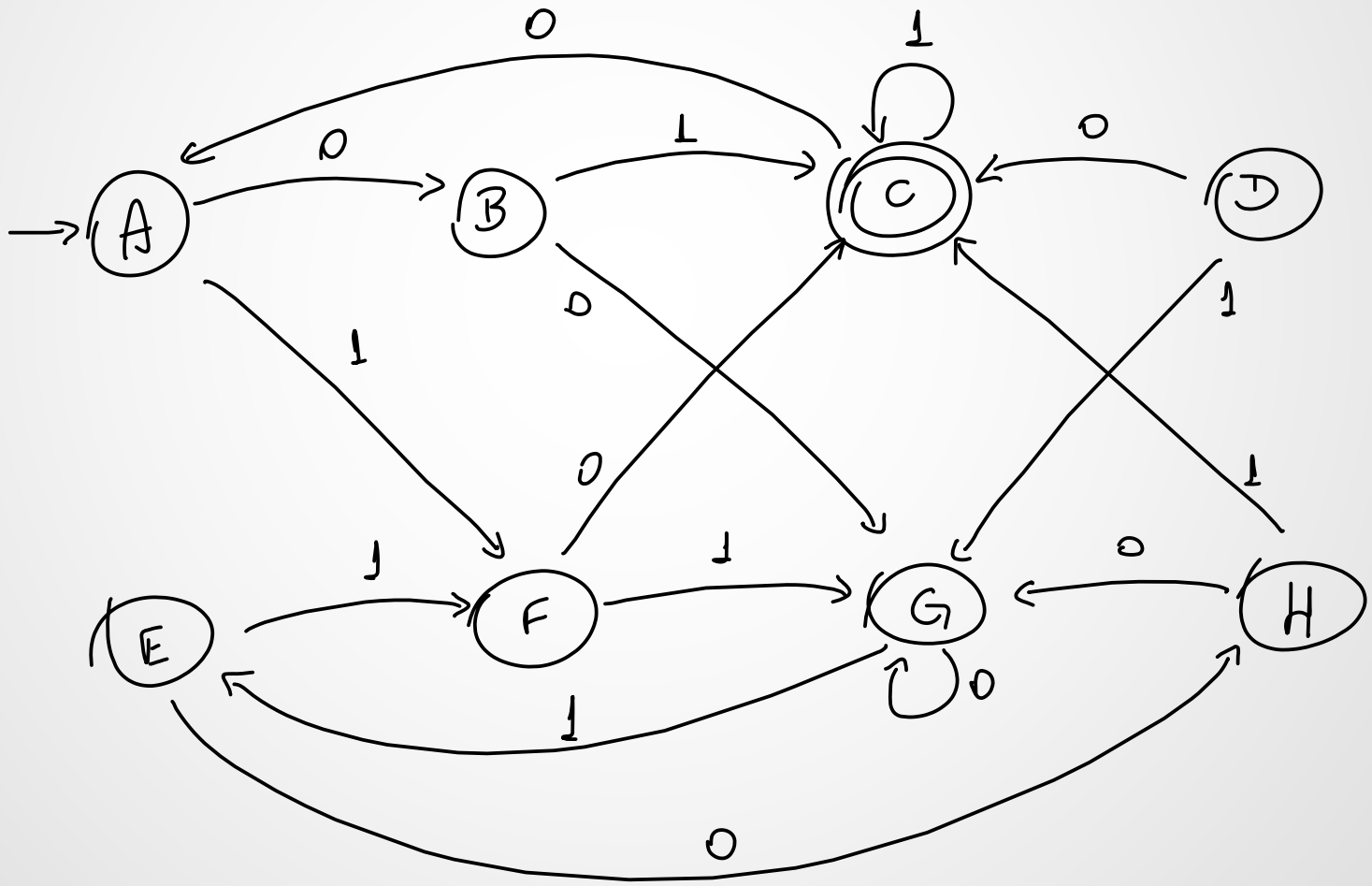
- Nesse caso, não há mais o que fazer, visto que os quatro pares restantes, $\{A, C\}$, $\{A, D\}$, $\{C, D\}$ e $\{B, E\}$ são equivalentes. Como os estados A e C são equivalentes e eles são os estados iniciais dos dois autômatos, podemos concluir que esses DFA's aceitam a mesma linguagem.

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- Algoritmo para minimização do DFA $A = (Q, \Sigma, \delta, q_0, F)$:
  1. Use o algoritmo de preenchimento de tabela para descobrir todos os pares de estados equivalentes;
  2. Particione o conjunto de Q estados em blocos de estados mutuamente equivalentes, ou seja, para cada estado q do DFA, crie um bloco consistindo em q e em todos os estados equivalentes a q. Todos os elementos de um bloco são equivalentes e nenhum par de estados escolhidos de diferentes blocos é equivalente;
  3. Construa o DFA B que é o autômato equivalente a A com número mínimo de estados. Tomamos γ como a função de transição de B. Suponha que S seja um conjunto de estados equivalentes de A e que a seja um símbolo de entrada. Então deve haver um bloco T de estados tais que, para todos os estados q em S, $\delta(q, a)$ é um elemento do bloco T. Caso contrário, o símbolo de entrada a tomará dois estados p e q de S como estados em diferentes blocos e esses estados serão distinguíveis. Isso nos leva a concluir que p e q não são equivalentes e que eles não pertenciam ambos a S. Como consequência, podemos fazer $\gamma(S, a) = T$. Além disso:
     a) O estado inicial de B é o bloco que contém o estado inicial de A.
     b) O conjunto de estados de aceitação de B é o conjunto de blocos que contém estados de aceitação de A. Pelo o que já foi visto, pode-se notar também que se um bloco contém um estado de aceitação, obrigatoriamente todos os estados desse bloco devem ser de aceitação.

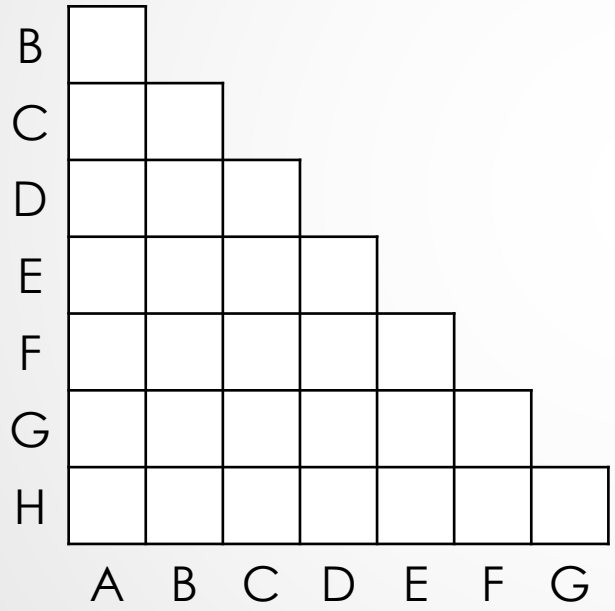
Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

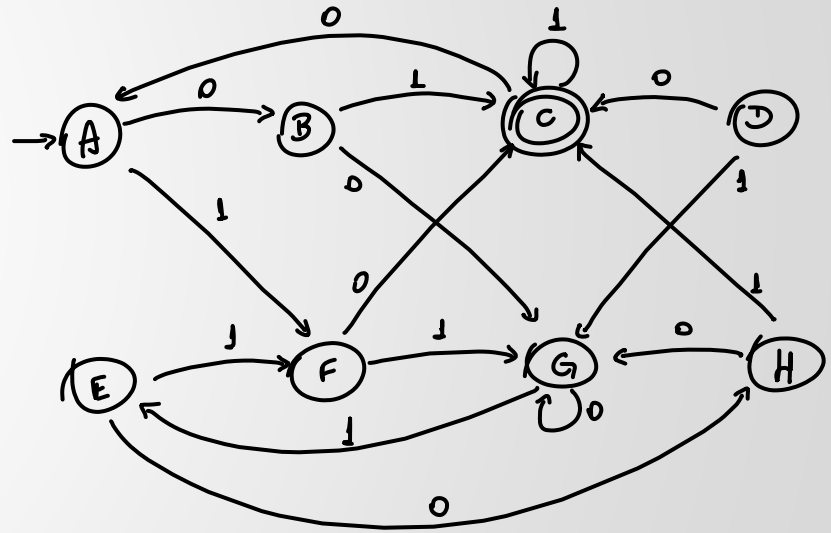
- **Exemplo:** Minimizar o DFA a seguir:



Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Passo 1:**

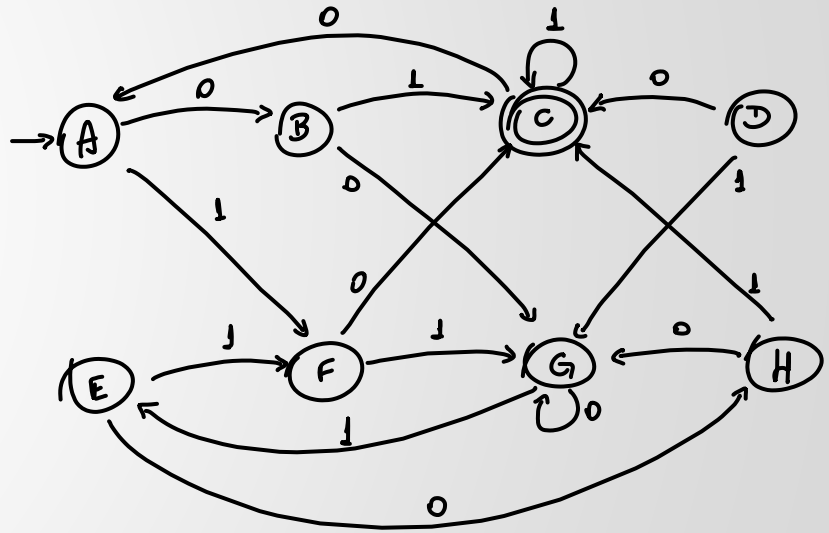




Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

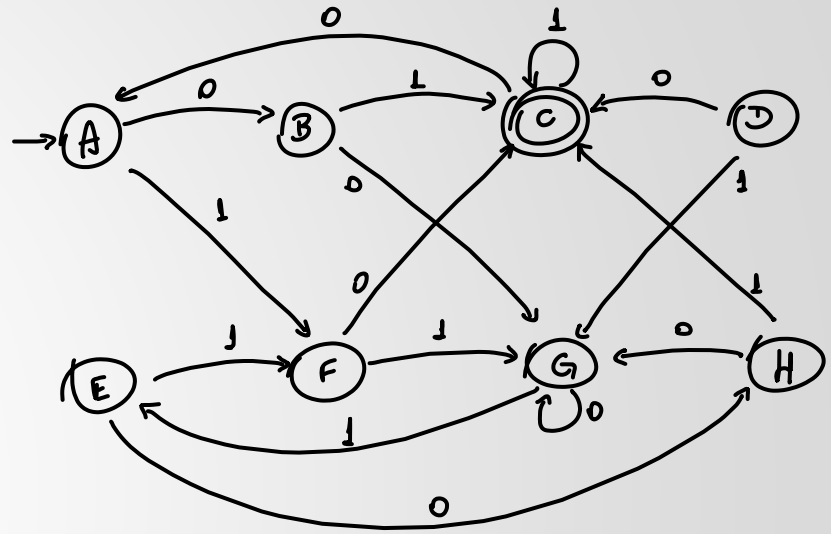
- **Passo 1:**

	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



Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

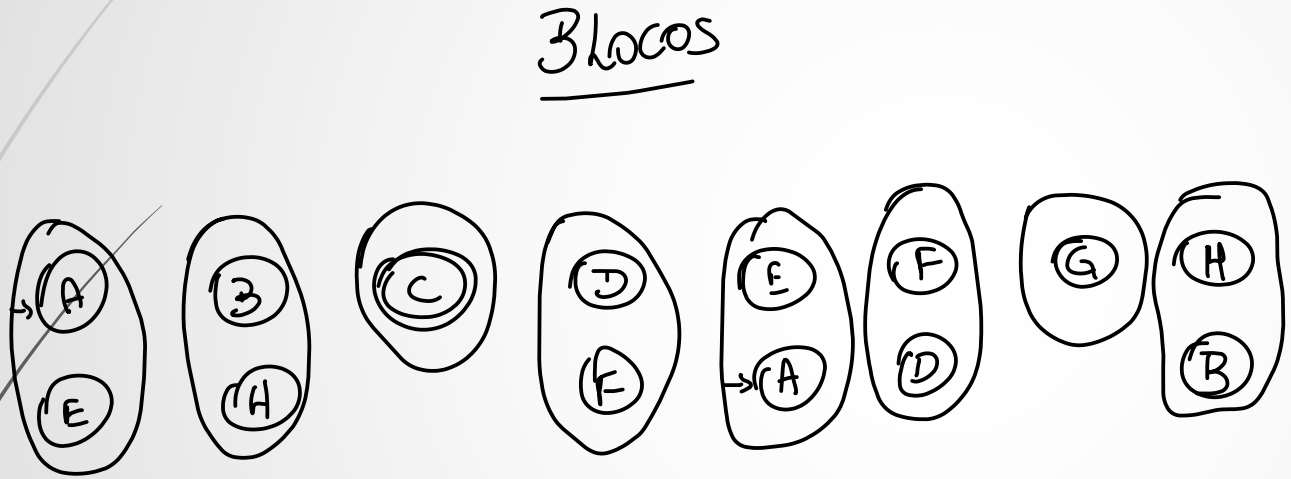
- **Passo 2:**

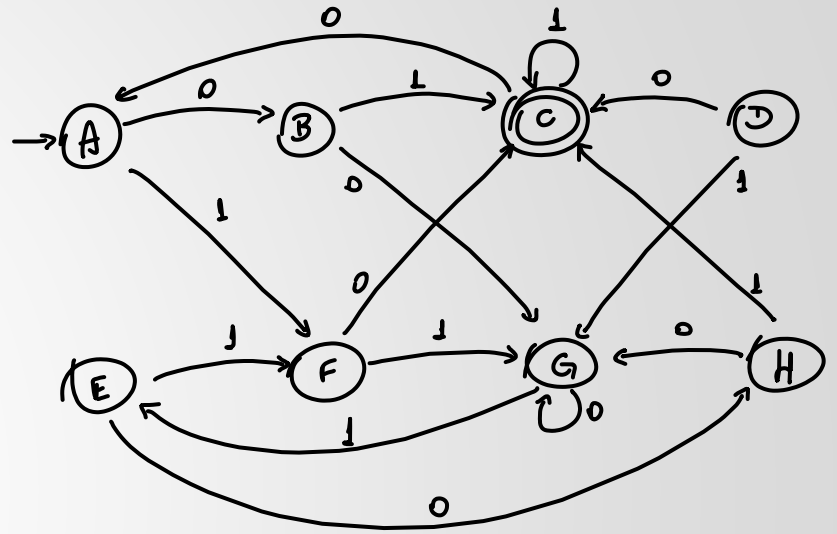


	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Passo 2:**

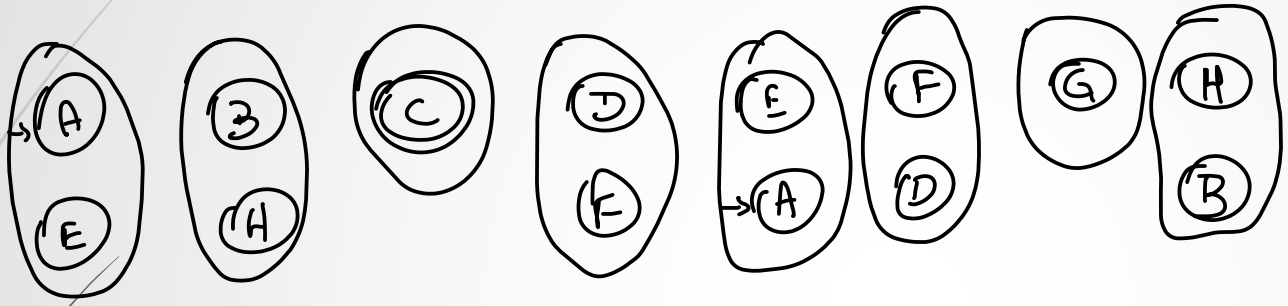


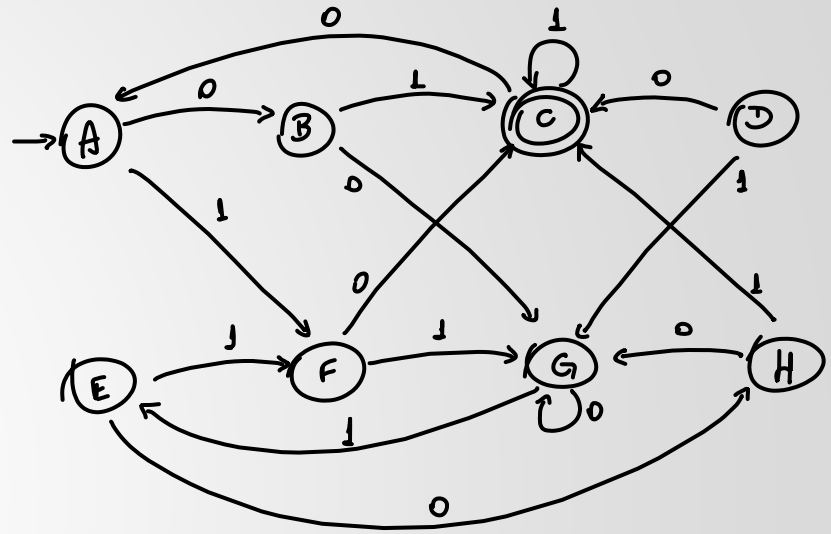


	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Passo 3:**

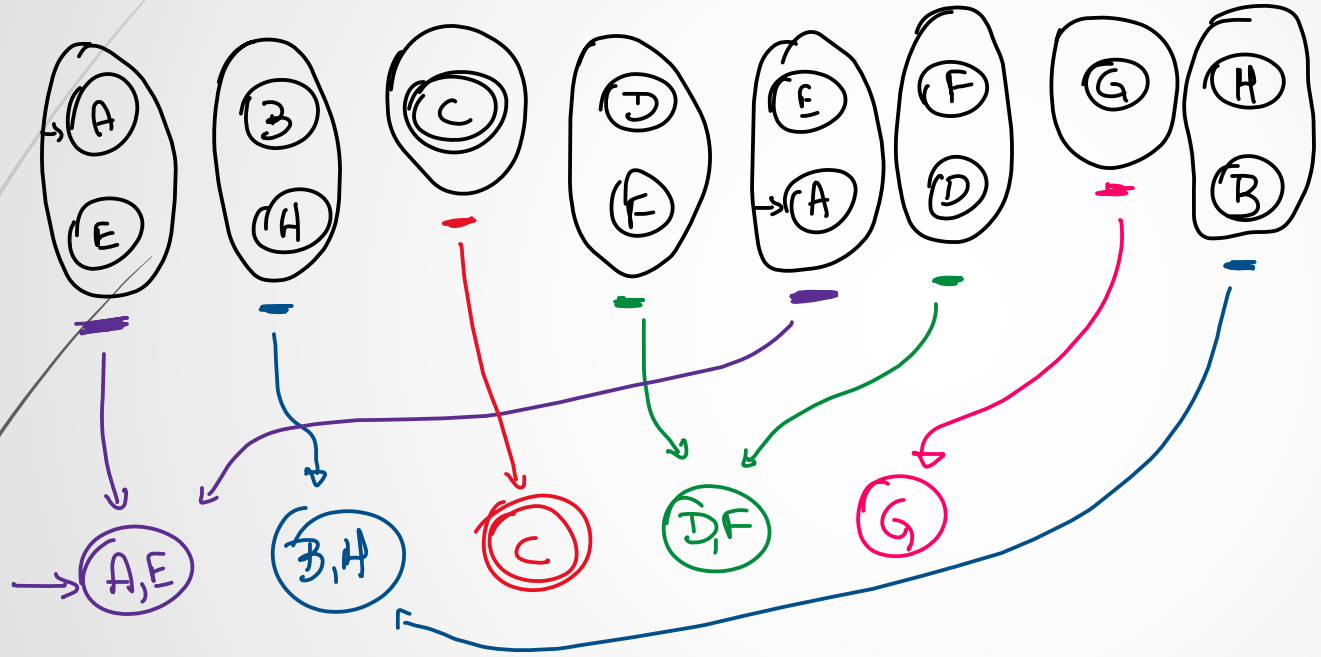


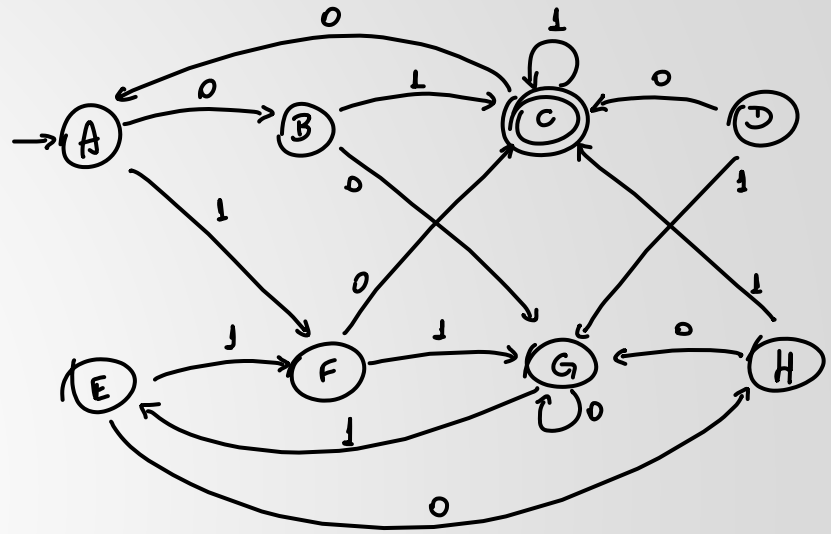


	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Passo 3:**

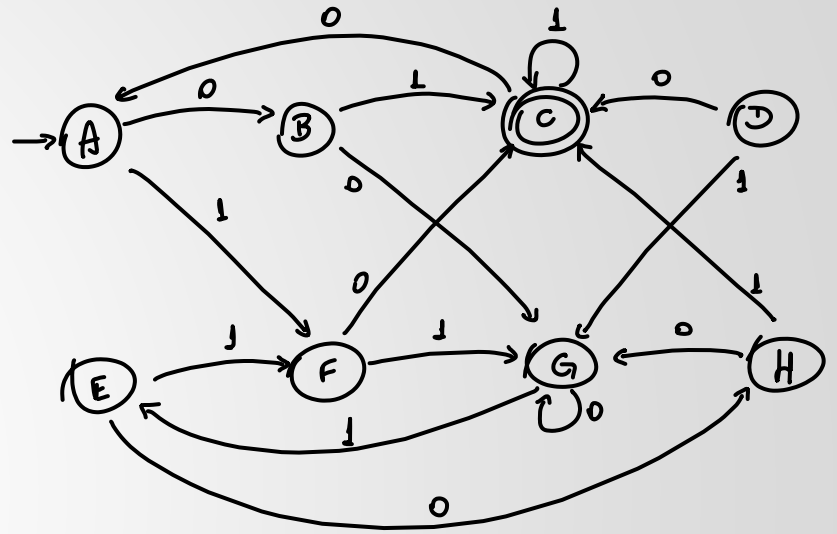




	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

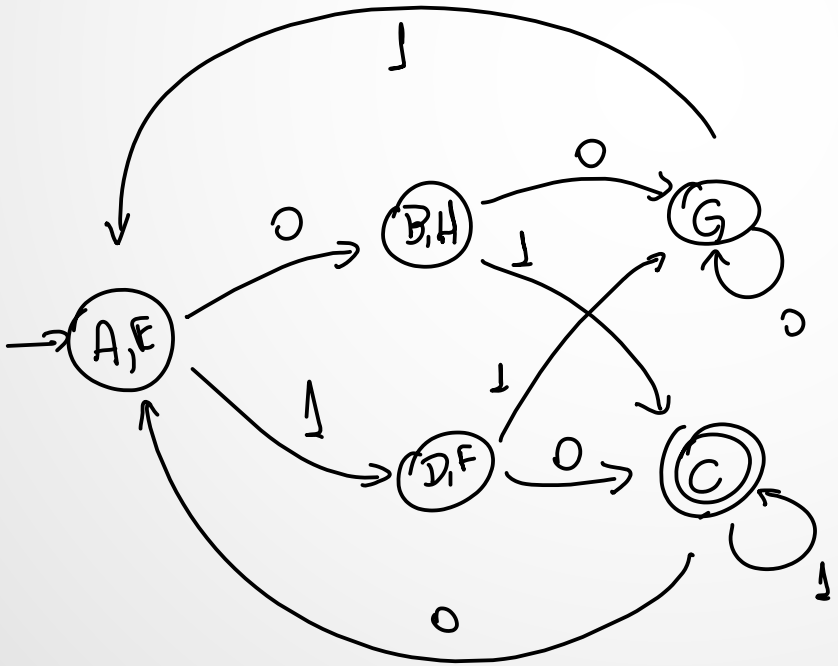
- **Passo 3:**

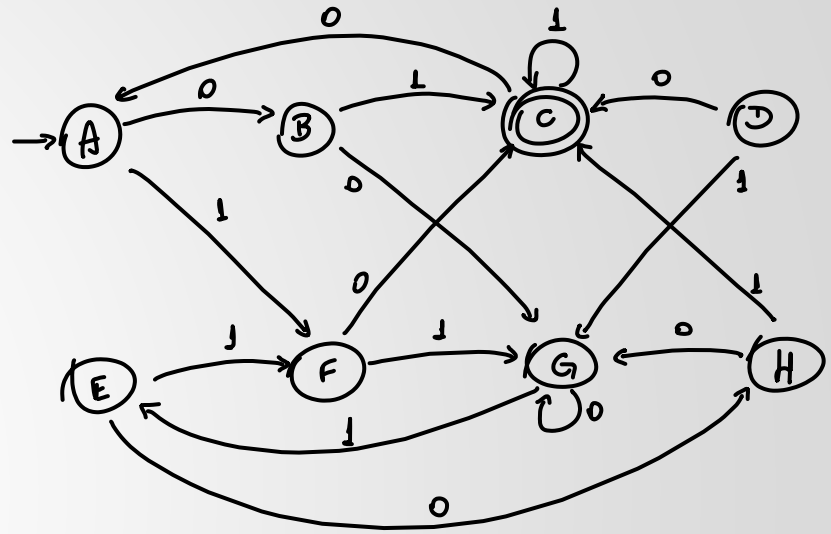


B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G

Propriedades das Linguagens Regulares
Equivalência e Minimização de Autômatos

- **Passo 3:**





	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

Propriedades das Linguagens Regulares
Exercícios Escritos

Exercício e6.1: Aplique o lema do bombeamento para provar que as seguintes linguagens não são regulares:

a) $L = \{a^i b^i c \mid i \geq 1\}$

b) $L = \{ba^i b^{2i} \mid i \geq 1\}$

c) $L = \{a^i b^{i+2} \mid i \geq 1\}$

d) $L = \{a^i b^i c^i \mid i \geq 1\}$

e) $L = \{wcw^R cw \mid w \in \{a, b\}^*\}$

f) $L = \{a^{i^3} b^{i^2} c^i \mid i \geq 1\}$

Propriedades das Linguagens Regulares
Exercícios Escritos

Exercício e6.2: Minimize o DFA descrito pela sua tabela de transições abaixo.

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
$* D$	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Propriedades das Linguagens Regulares
Exercícios Escritos

Exercício e6.3: Minimize o DFA descrito pela sua tabela de transições abaixo.

	0	1
$\rightarrow A$	B	E
B	C	F
$* C$	D	H
D	E	H
E	F	I
$* F$	G	B
G	H	B
H	I	C
$* I$	A	E

Bibliografia

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. **Introdução à Teoria de Autômatos, Linguagens e Computação**. 2. ed. Rio de Janeiro: Elsevier, 2002. 560 p.

RAMOS, M. V. M.; JOSÉ NETO, J.; VEGA, I. S. **Linguagens Formais: Teoria, Modelagem e Implementação**. Porto Alegre: Bookman, 2009. 656 p.

SIPSER, M. **Introdução à Teoria da Computação**. 2. ed. São Paulo: Cengage Learning, 2017. 459 p.