Aspectos Formais da Computação

Prof. Sergio D. Zorzo
Departamento de Computação – UFSCar
1º semestre / 2017
Aula 7

Propriedades das 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)

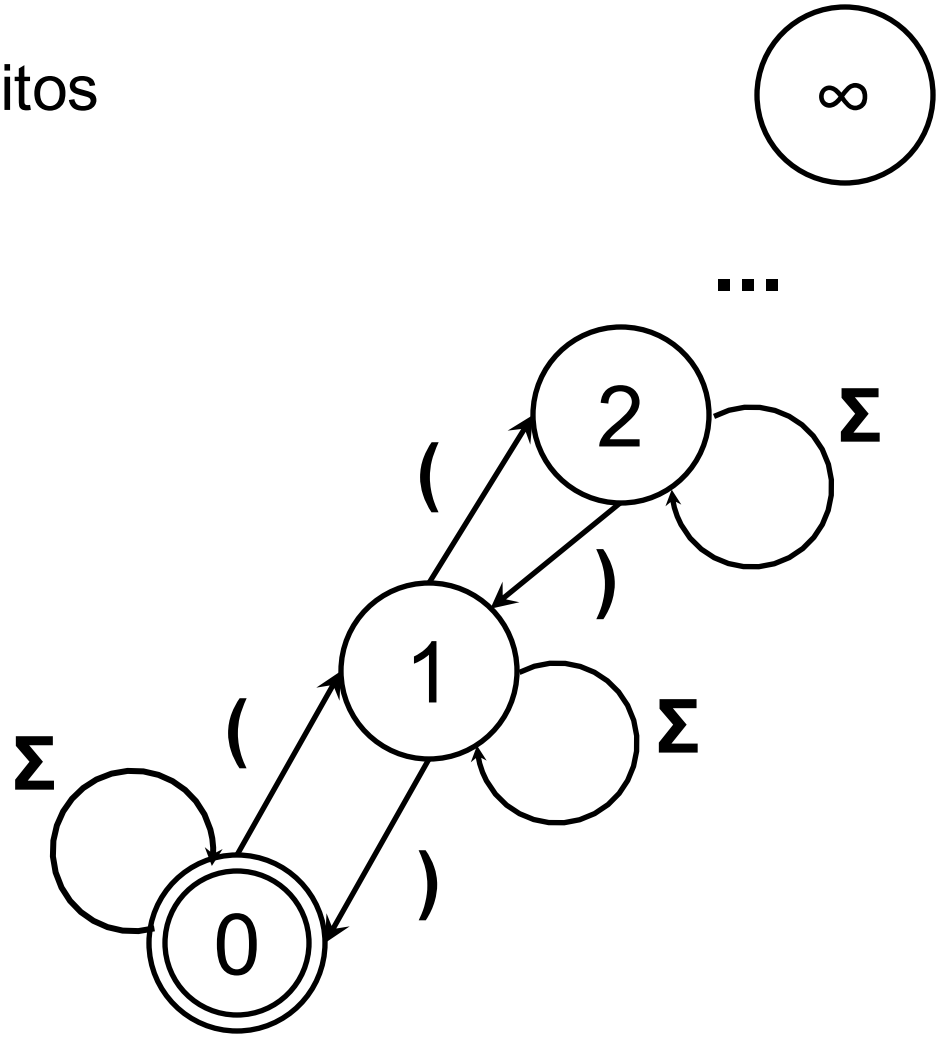
É preciso contar quantos parêntesis são abertos e quantos são fechados

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)

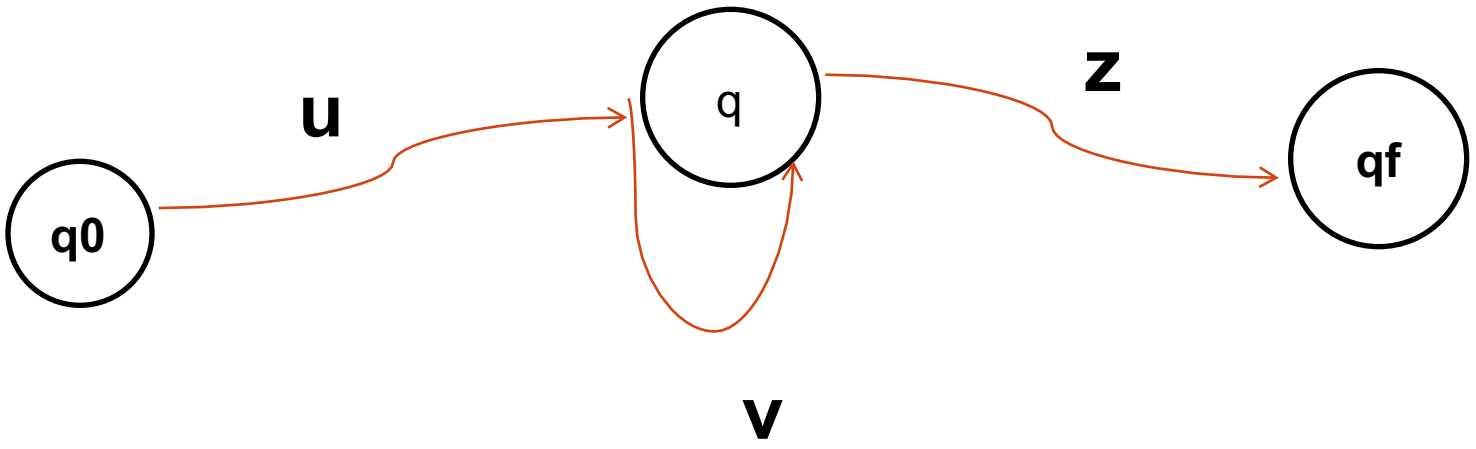
Formalmente:

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

Bombeamento para as Linguagens 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: $\{0^n1^n|n\geq 0\}$

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!

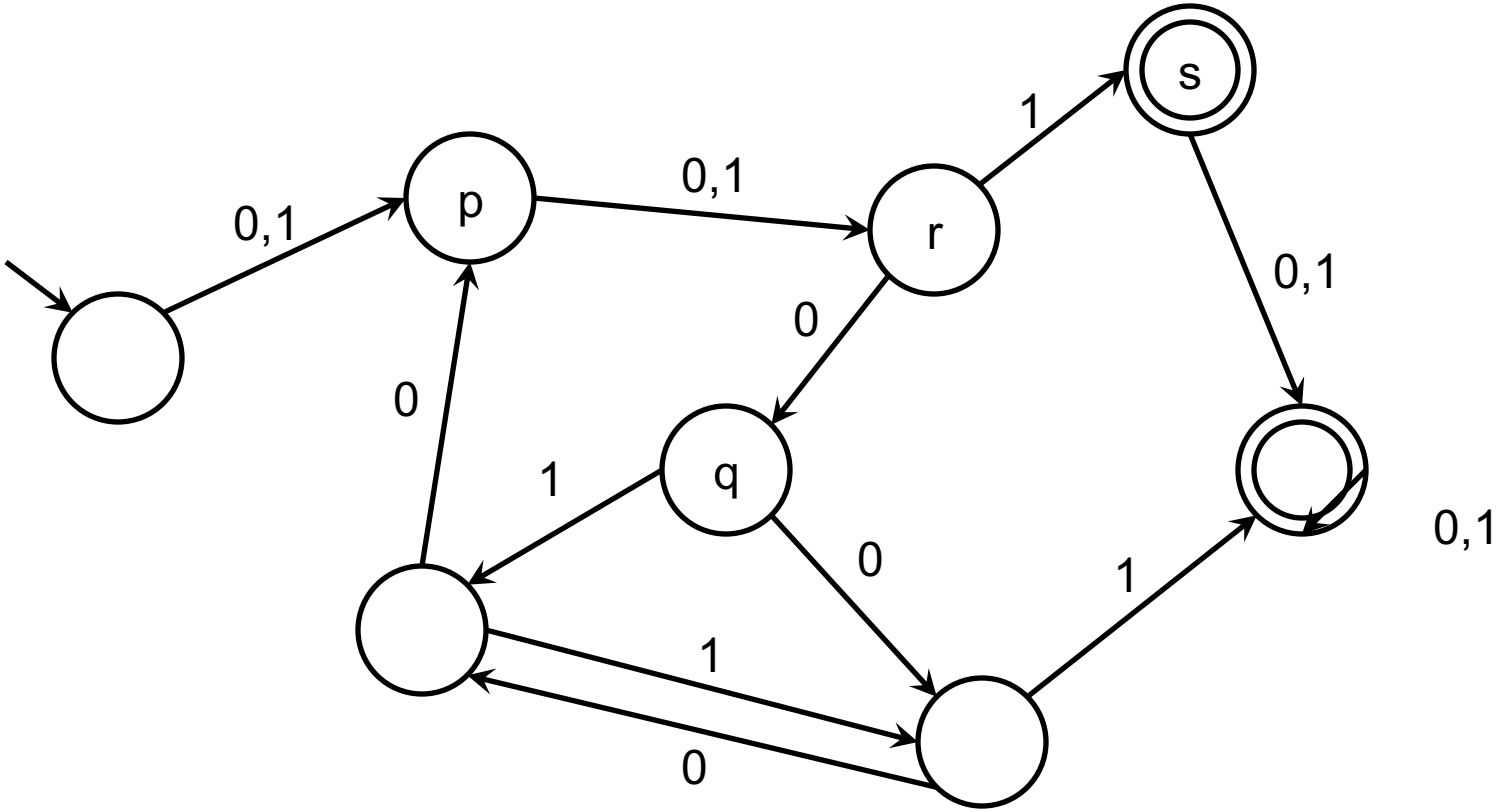
Minimização de DFAs

Ilustrando:

0,1, 010, 111 não distingue p e q

11 distingue p e q

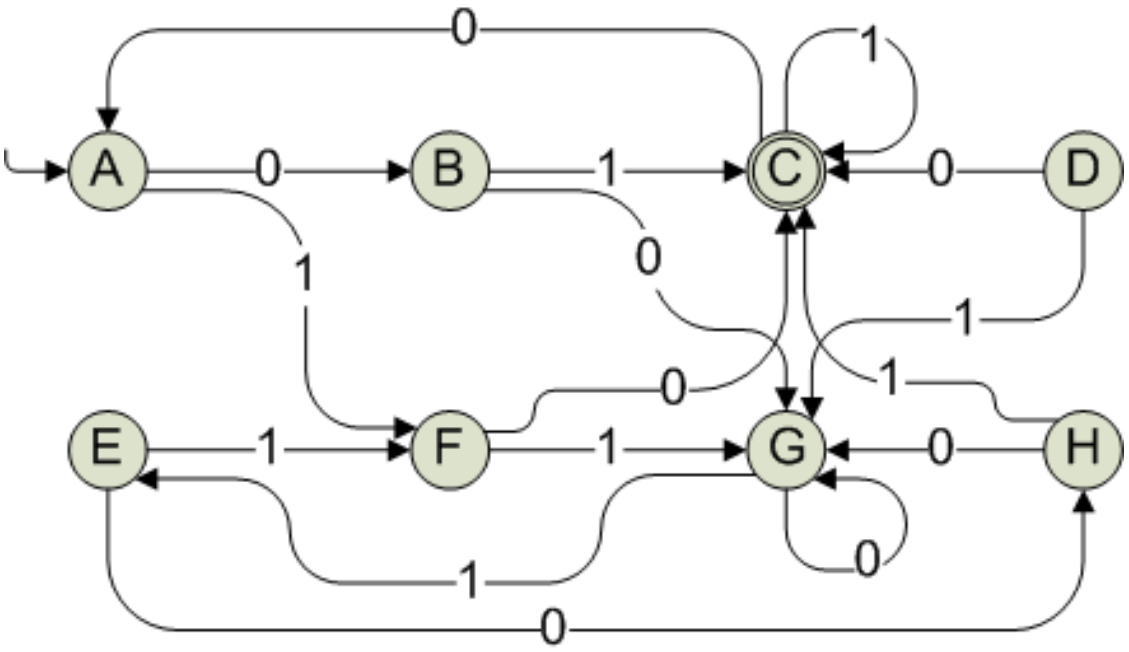
r e s são distinguíveis (ε os distingue)

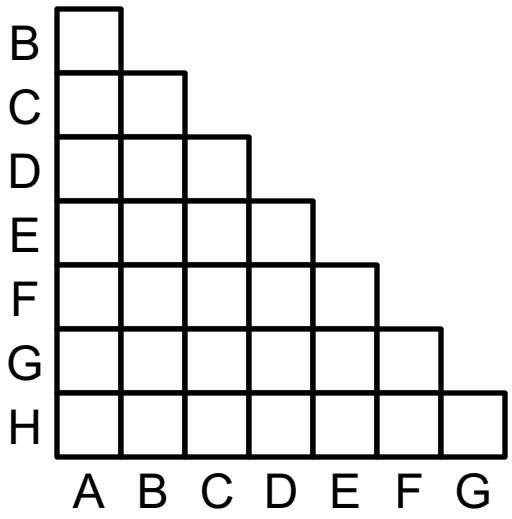


Minimização de DFAs

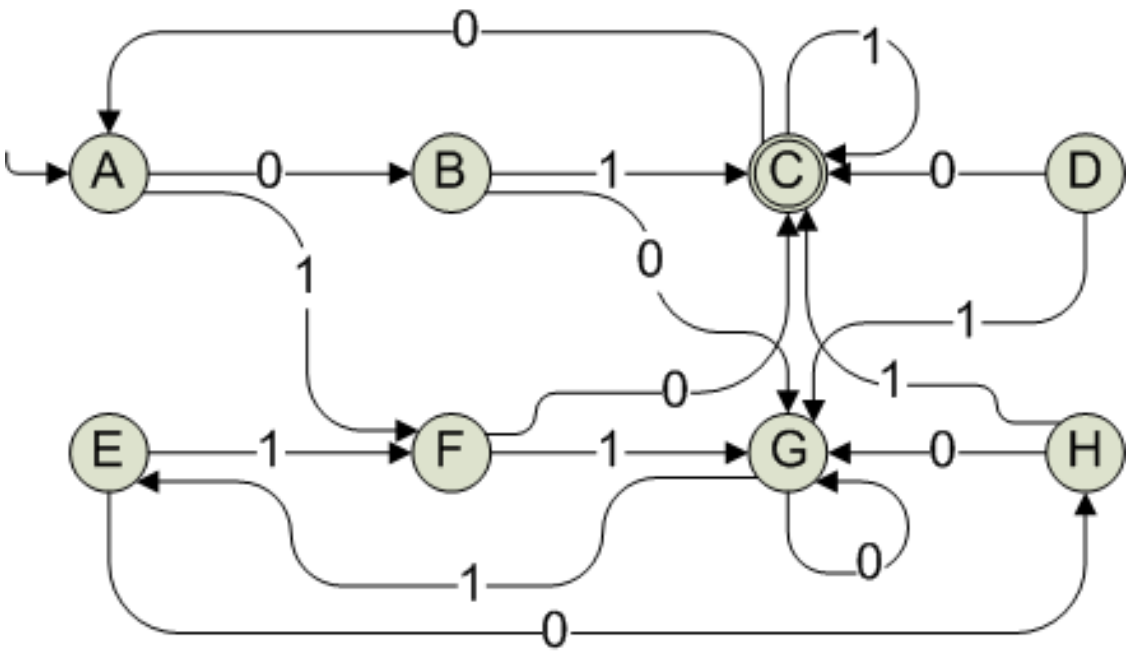
- Difícil encontrar estados equivalentes apenas "olhando" para o DFA
  - Muitas combinações, fácil se perder
- Estratégia sistemática: encontrar todos os pares de estados que sejam distinguíveis
  - Se fizermos o melhor possível
    - Qualquer par de estados que não considerarmos distinguíveis serão 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





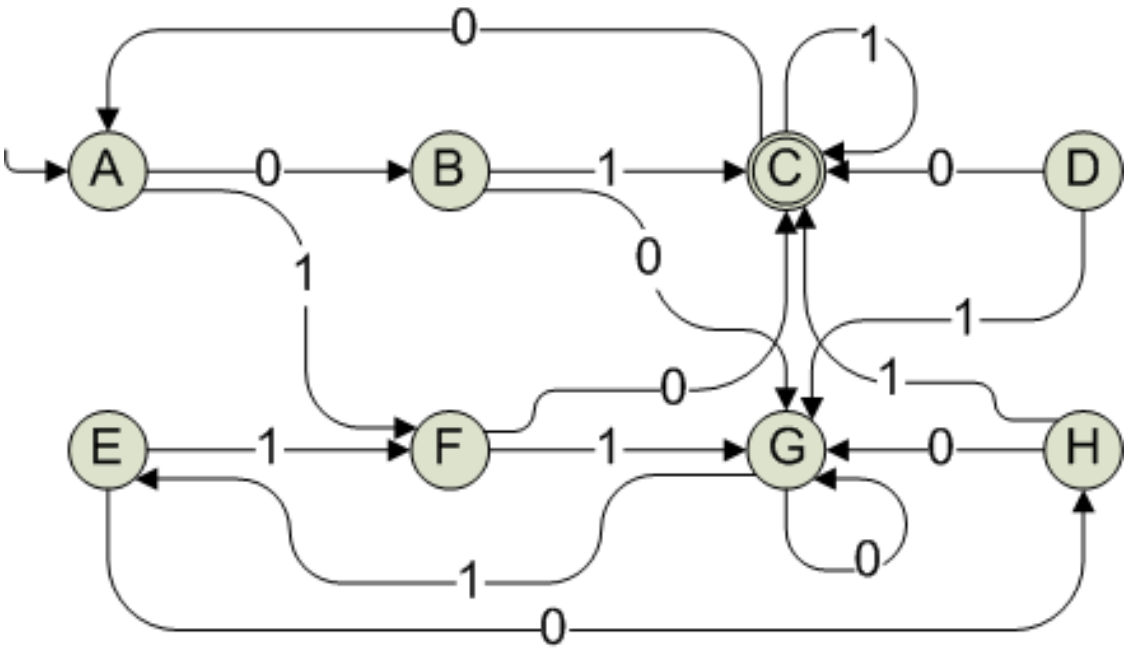
Minimização de DFAs



	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



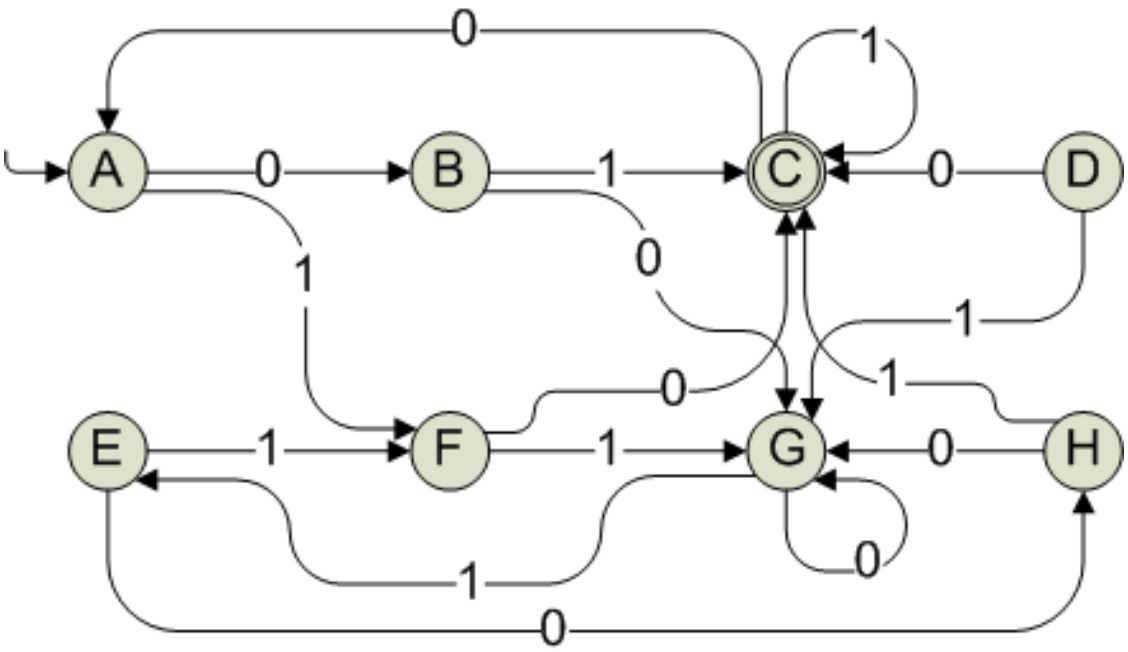
	A	B	C	D	E	F	G
B							
C	x	x					
D			x				
E			x				
F			x				
G			x				
H			x				

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



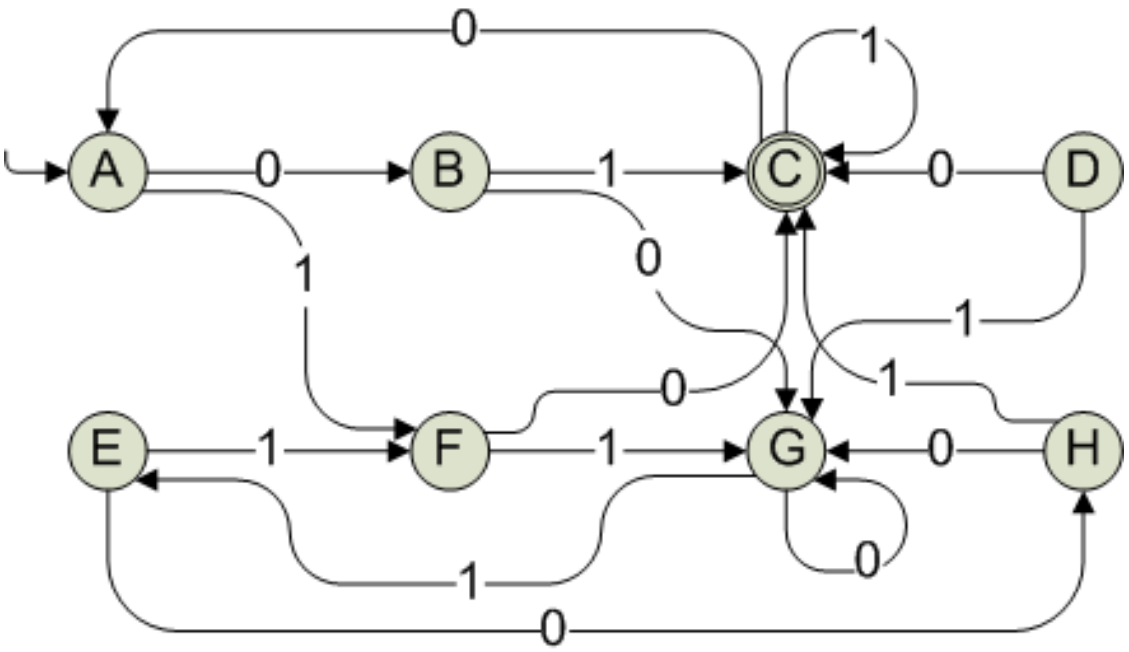
	A	B	C	D	E	F	G
B							
C	x	x					
D			x				
E			x				
F			x				
G			x				
H			x				

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



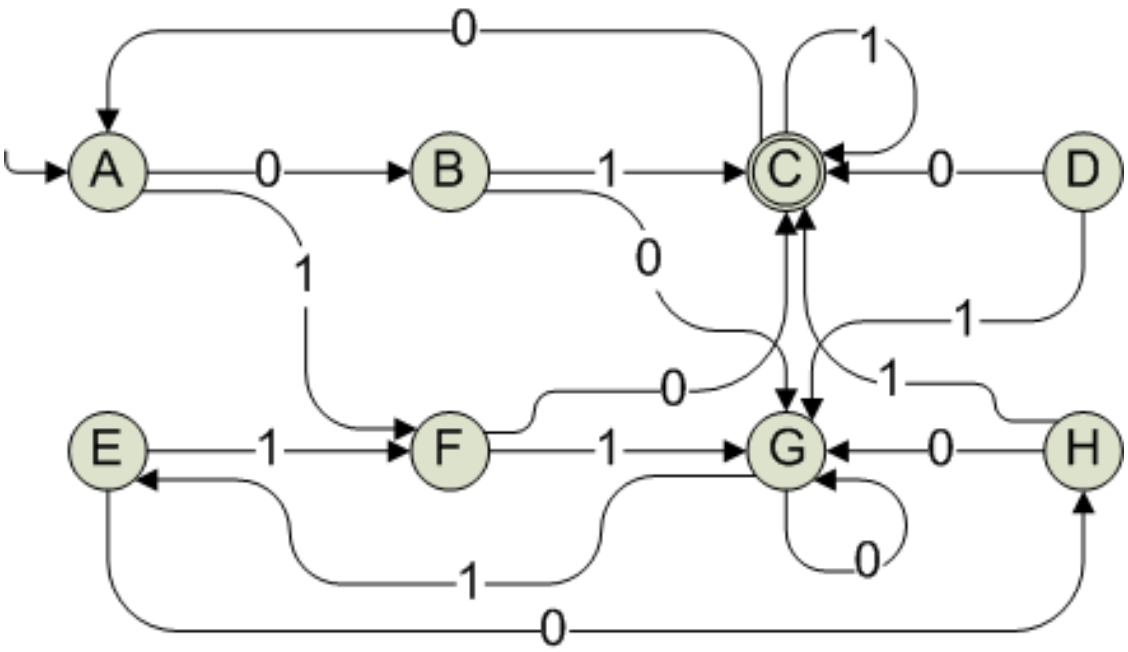
	A	B	C	D	E	F	G
B							
C	x	x					
D	x		x				
E			x				
F	x		x				
G			x				
H			x				

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



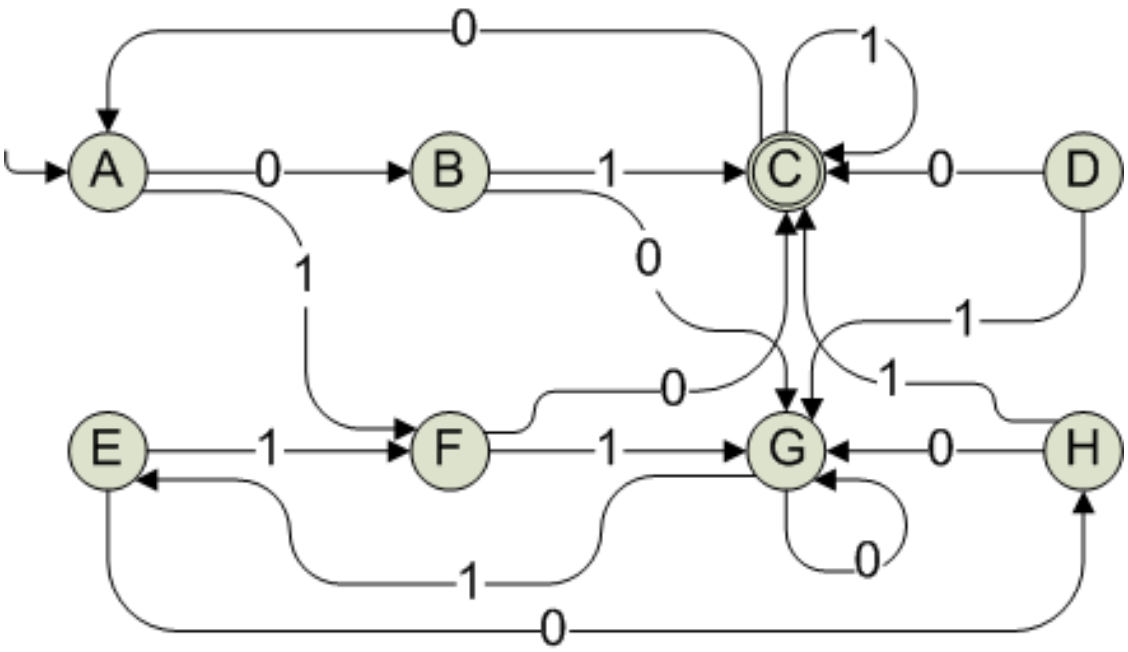
	A	B	C	D	E	F	G
B							
C	x	x					
D	x		x				
E			x				
F	x		x				
G			x				
H			x				

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



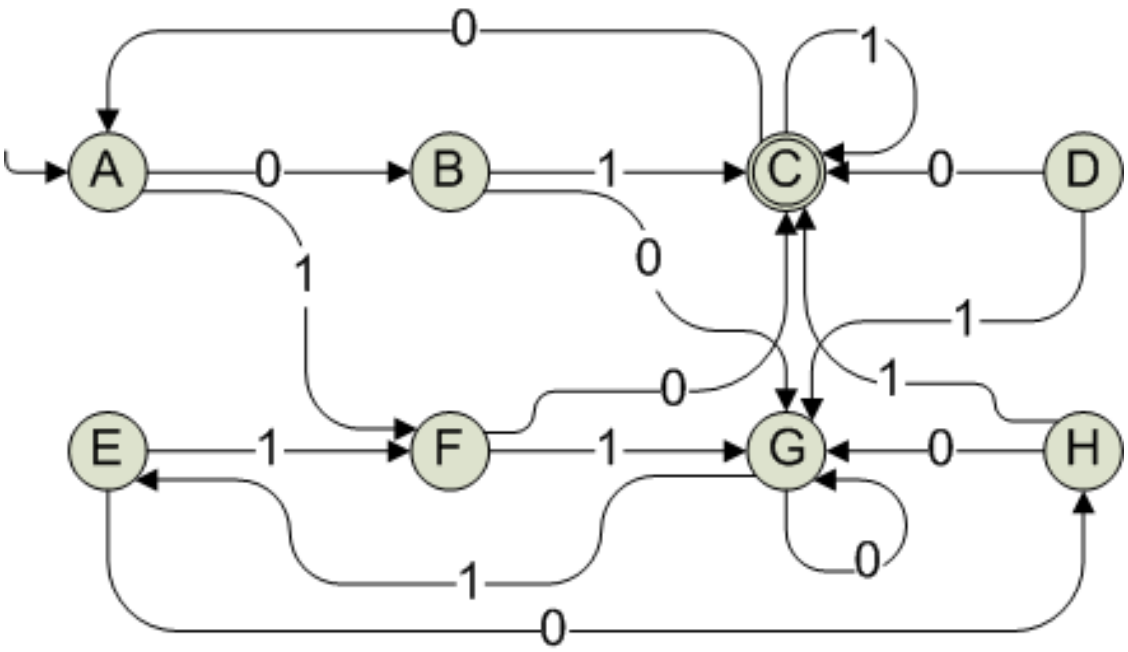
	A	B	C	D	E	F	G
B							
C	x	x					
D	x		x				
E			x				
F	x		x				
G		x	x				
H			x				x

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



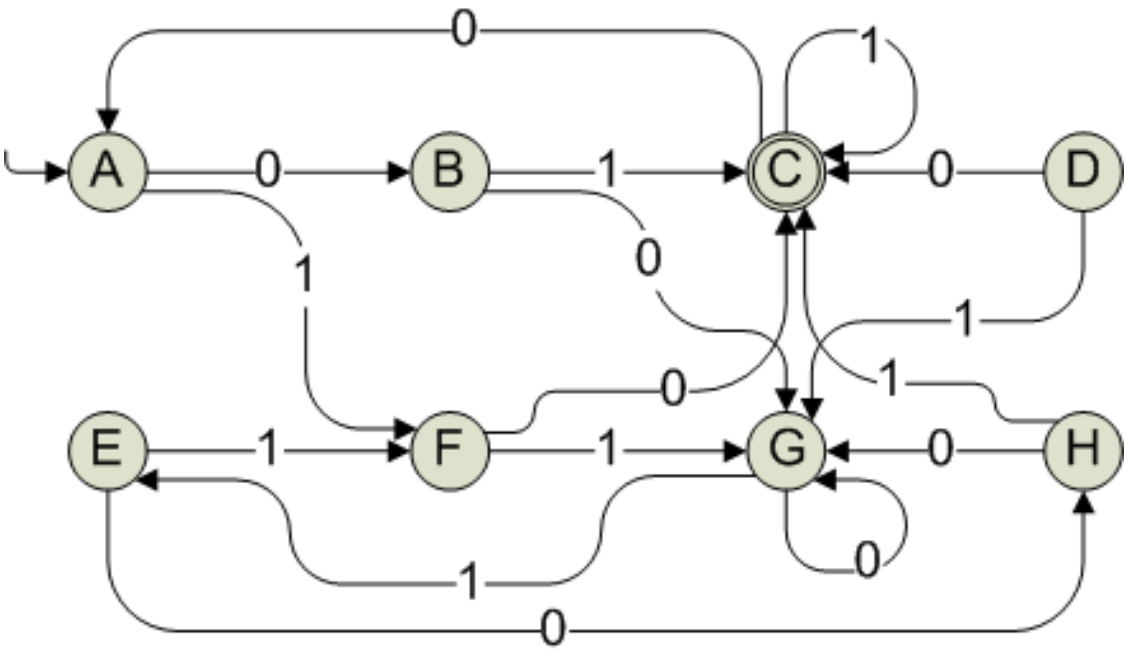
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x		x				
E		x	x				
F	x		x				
G		x	x				
H	x		x		x		x

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



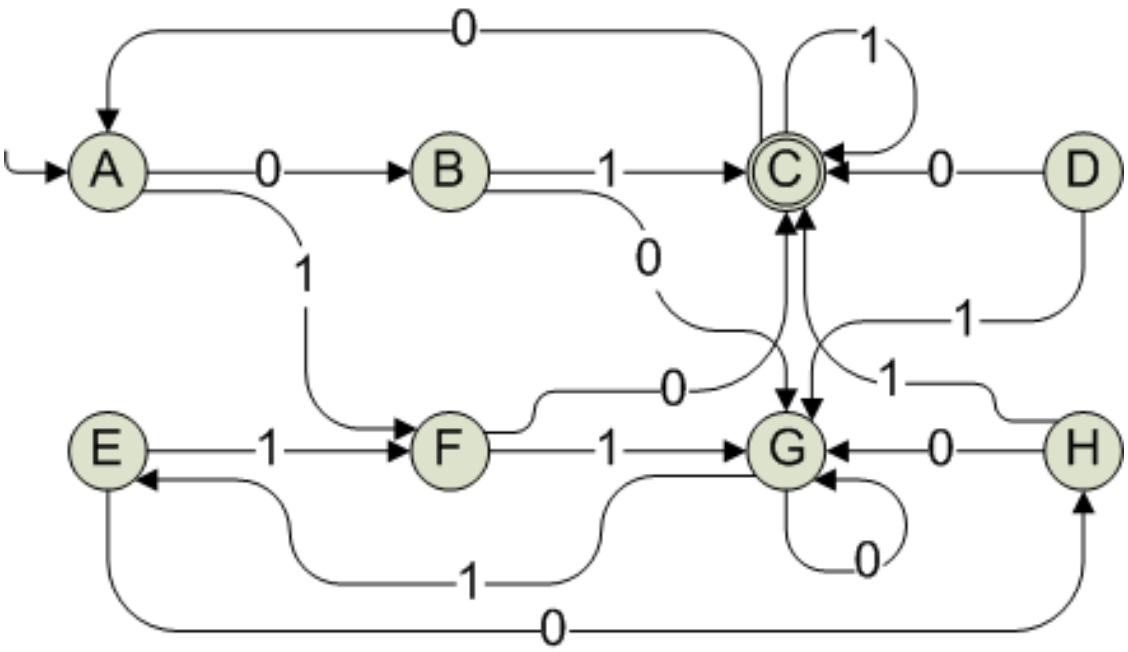
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x				
F	x	x	x				
G		x	x	x		x	
H	x		x	x	x	x	x

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



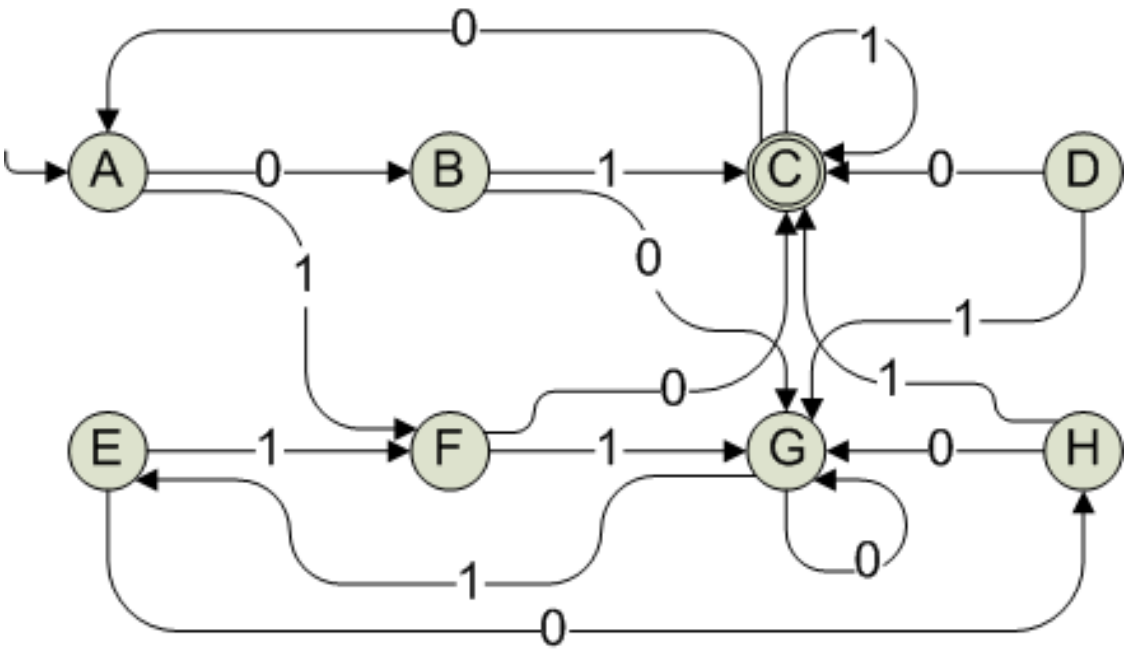
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

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



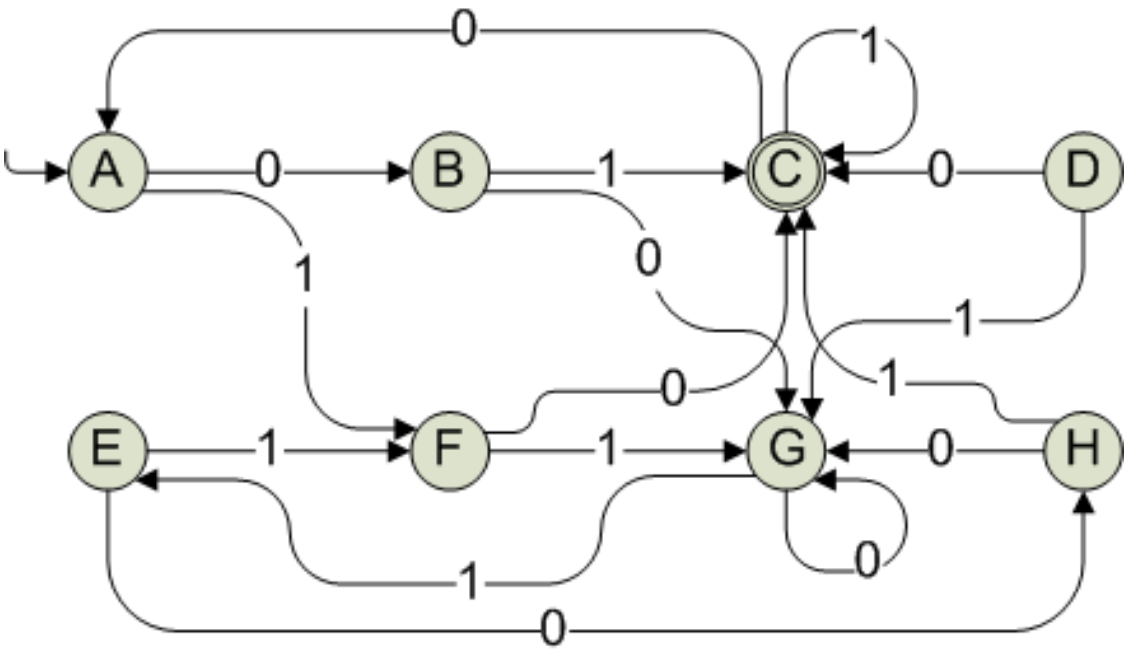
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

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



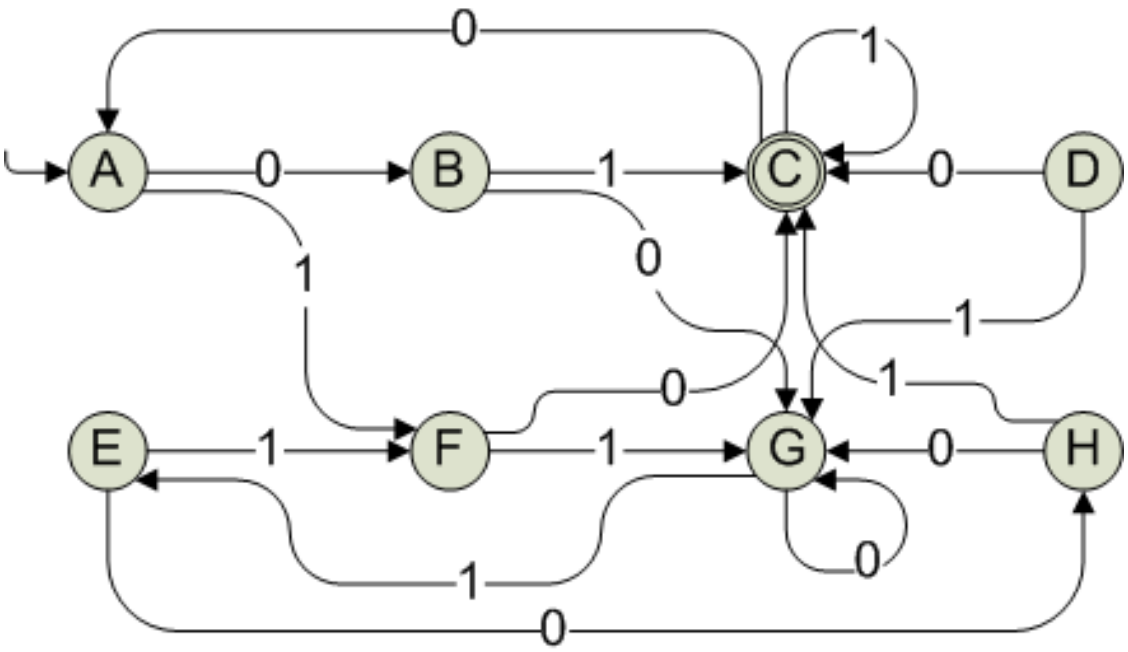
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

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



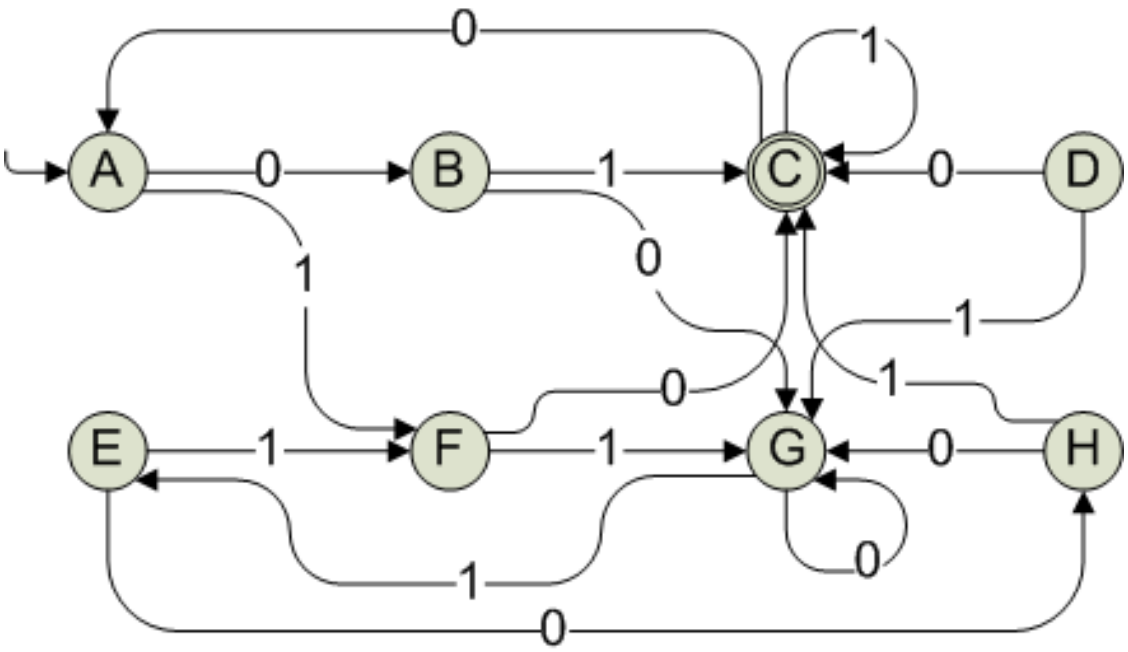
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

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



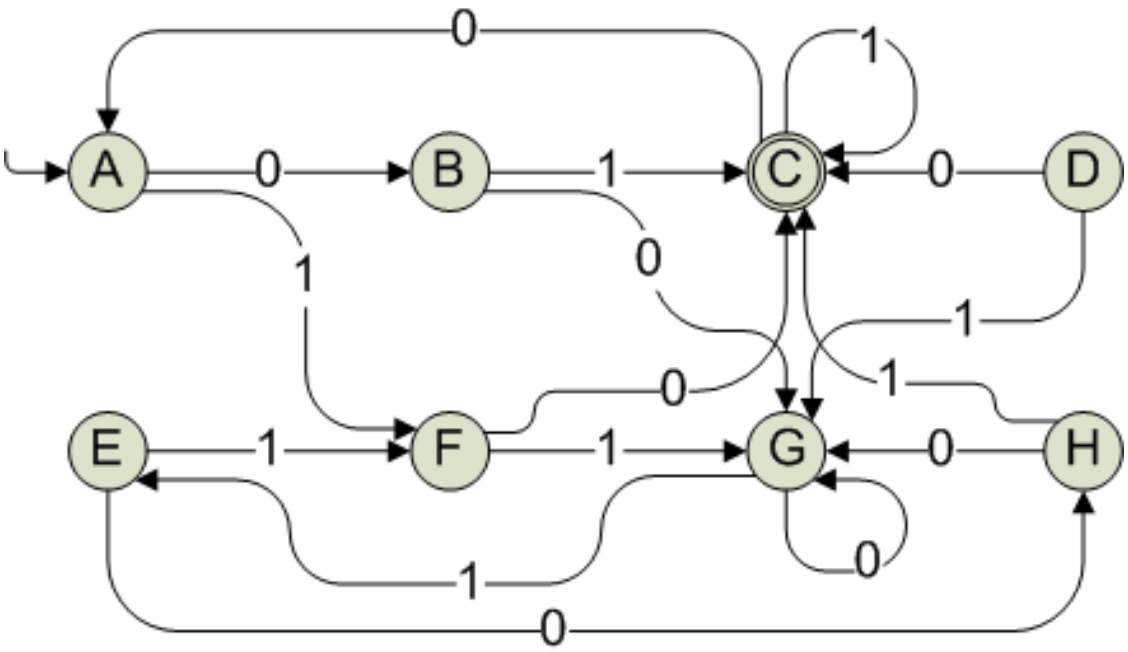
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

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



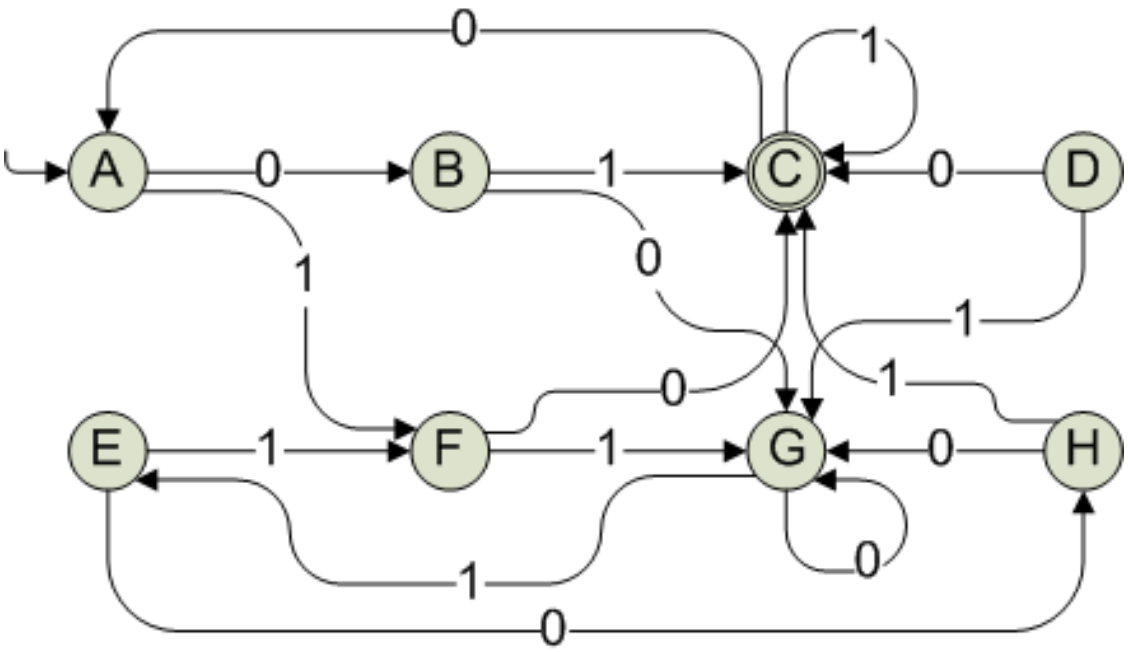
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

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



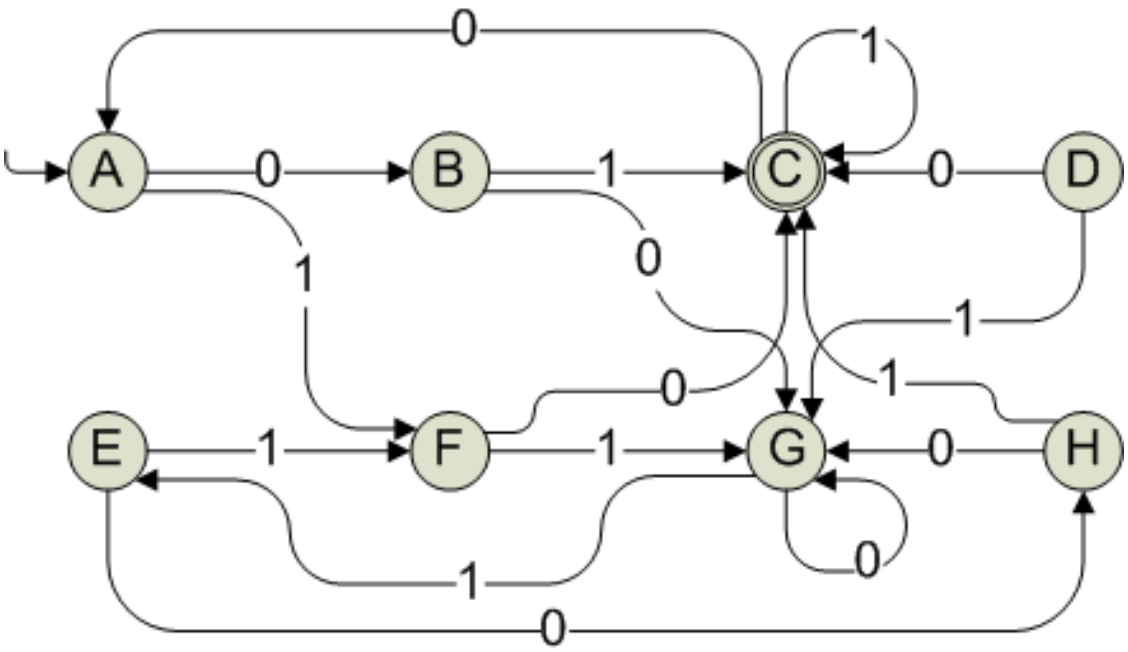
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

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



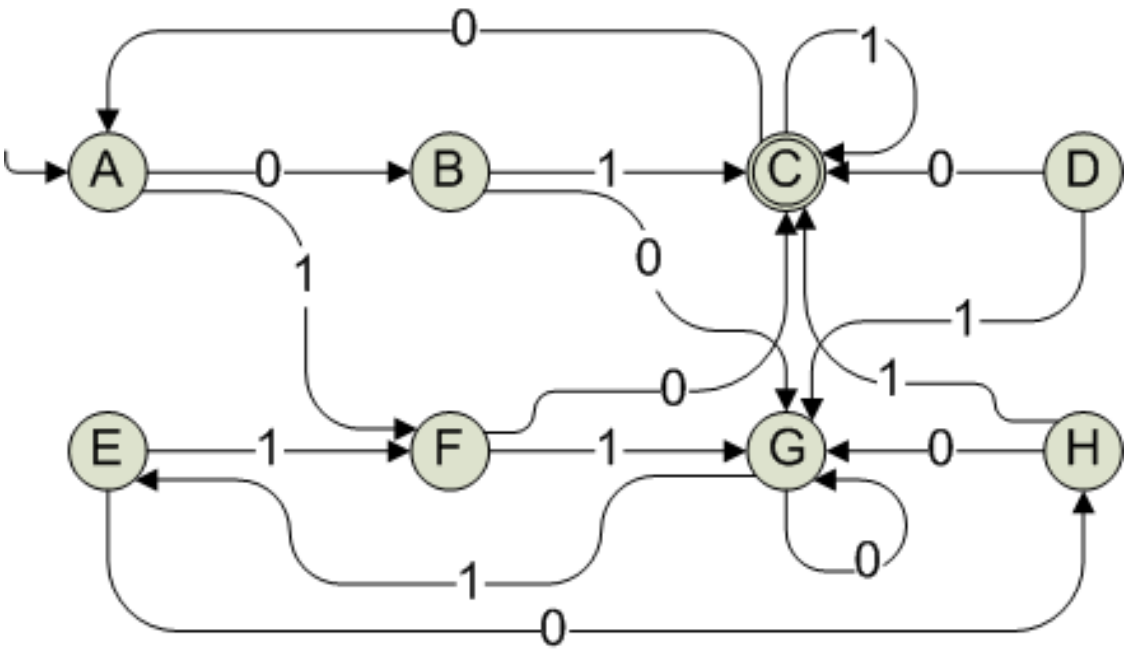
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x

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



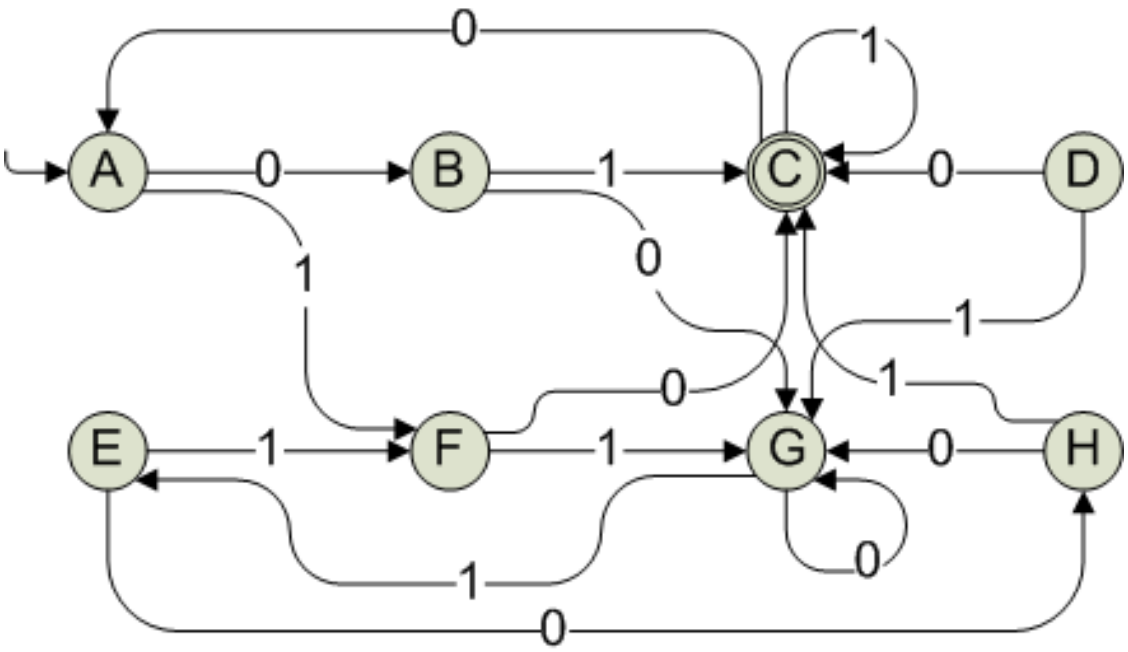
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

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



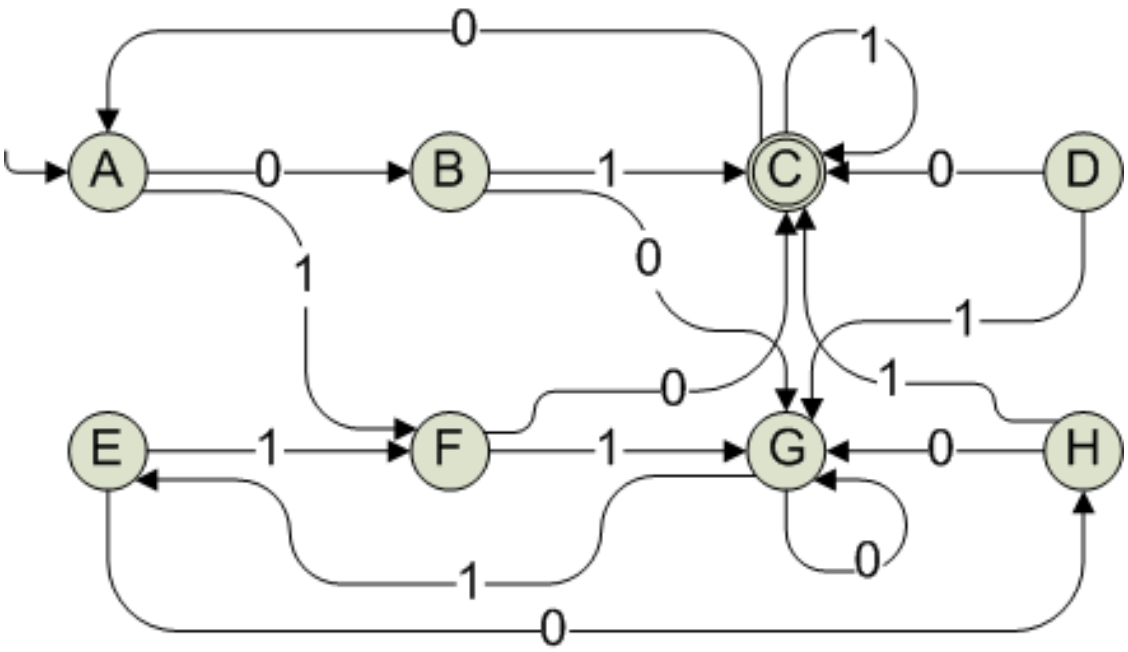
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

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



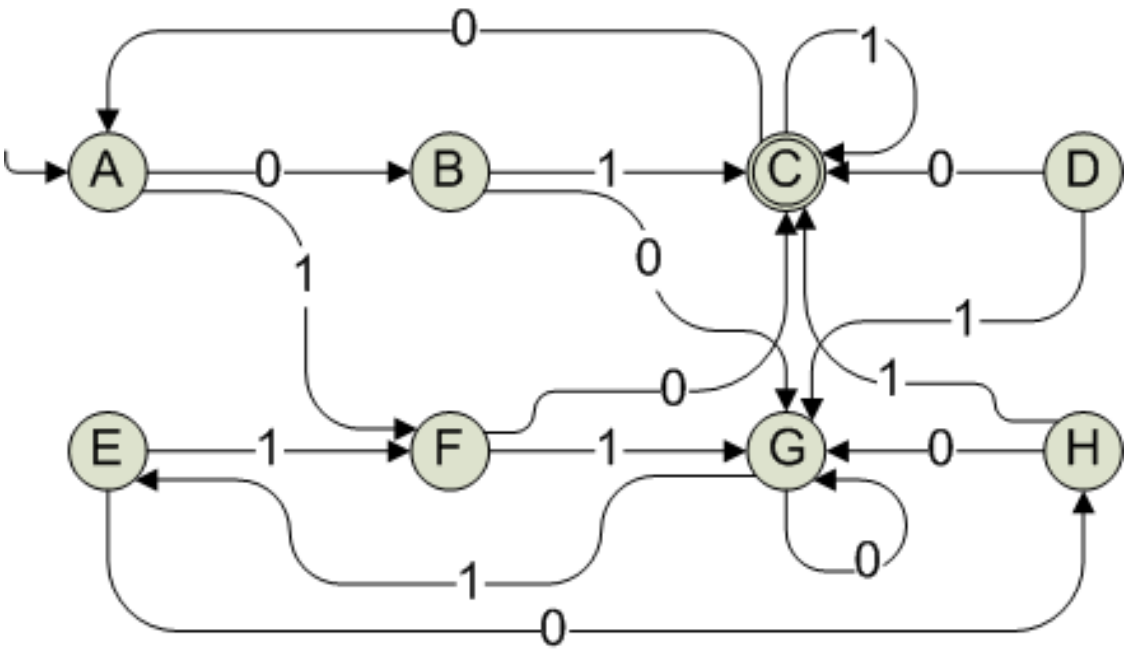
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

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



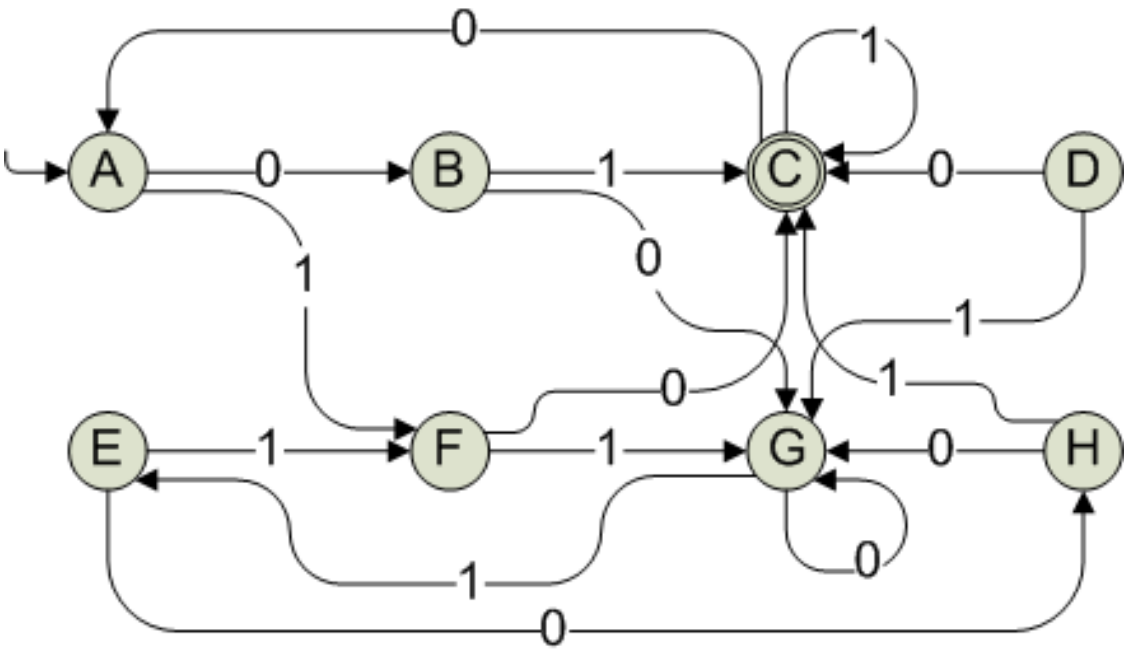
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

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



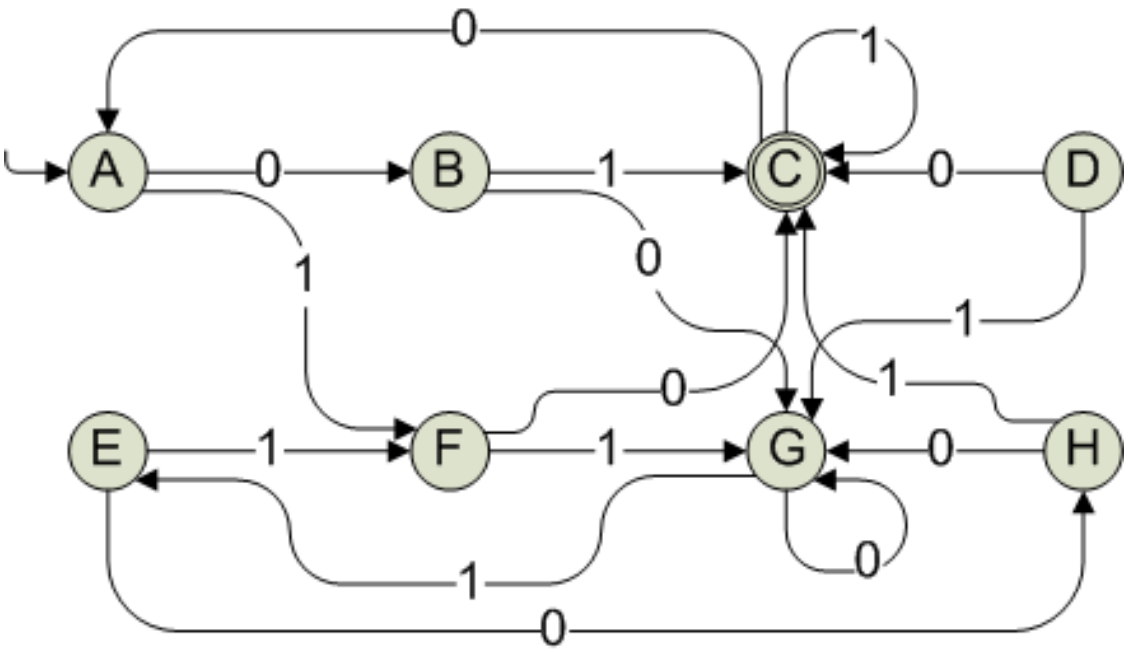
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

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



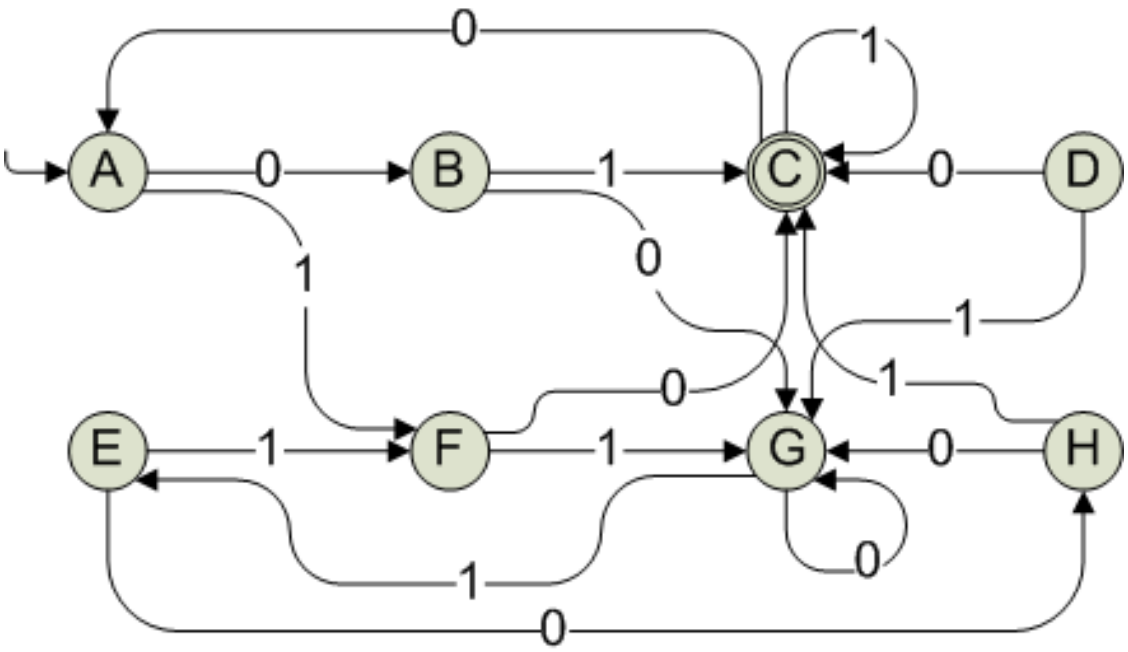
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

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



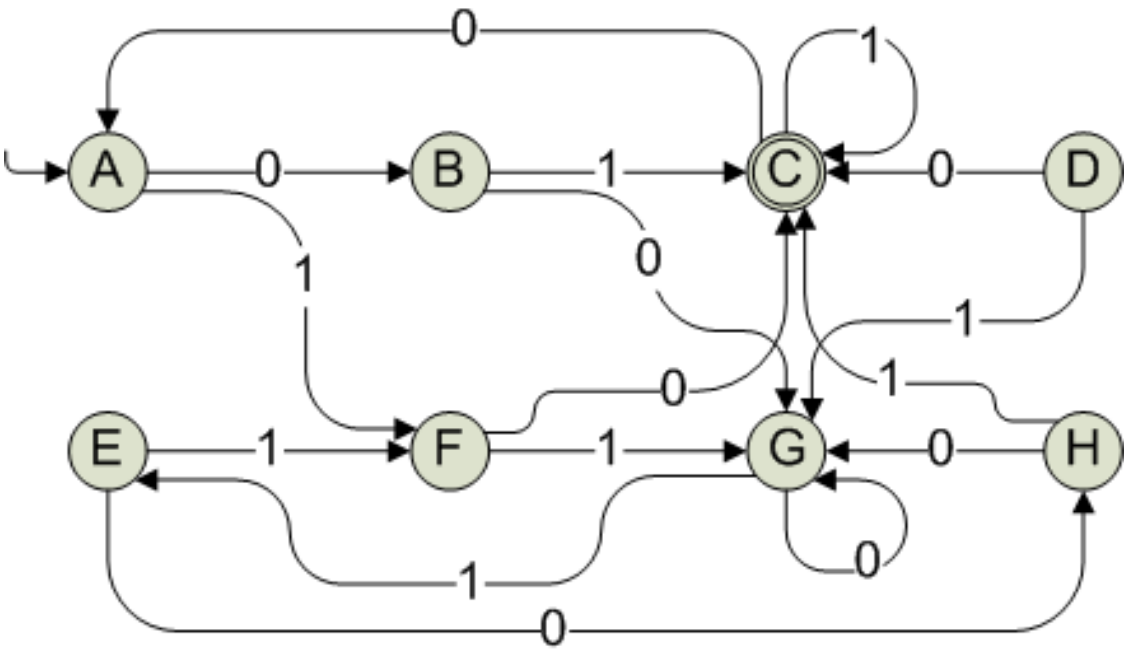
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x		x	
H	x		x	x	x	x	x

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



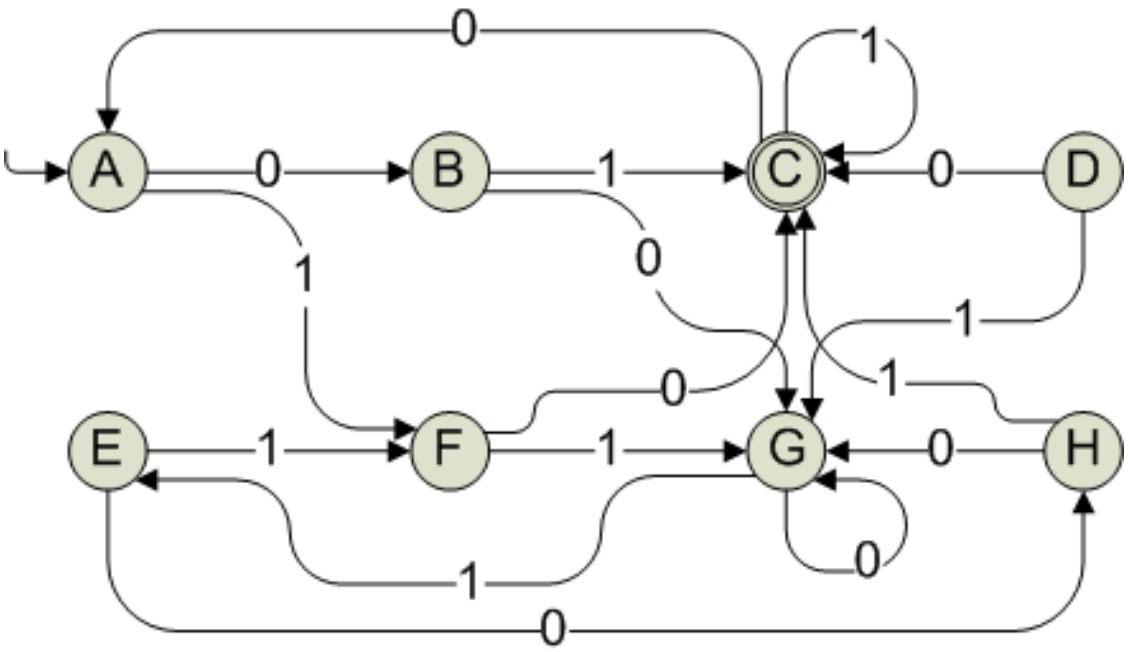
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

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



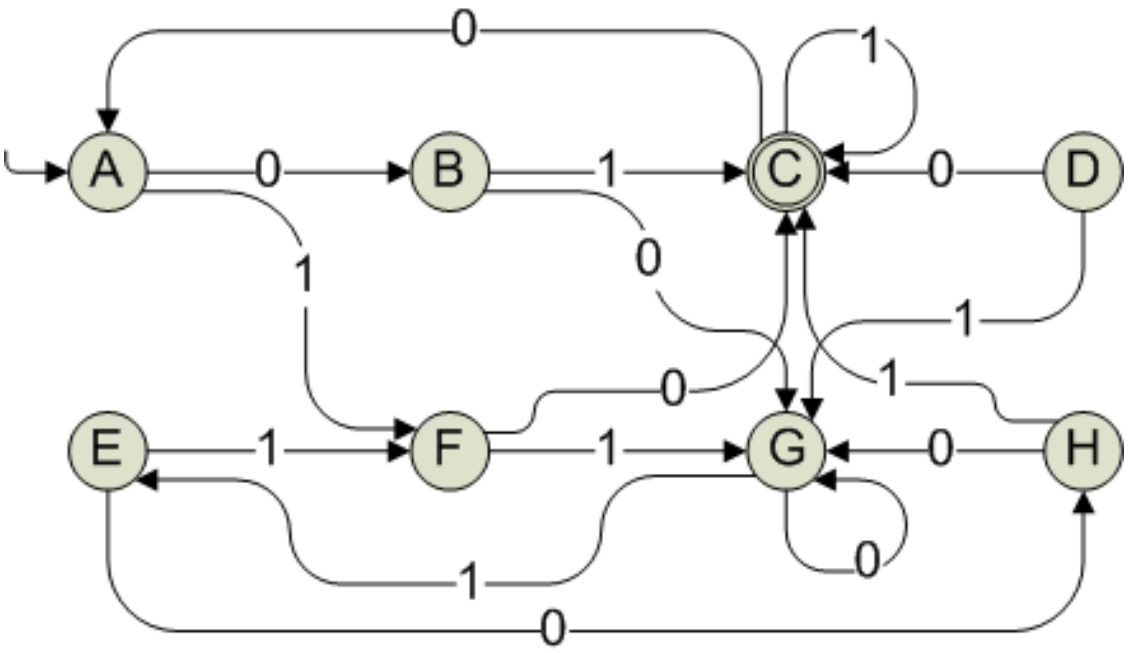
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

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



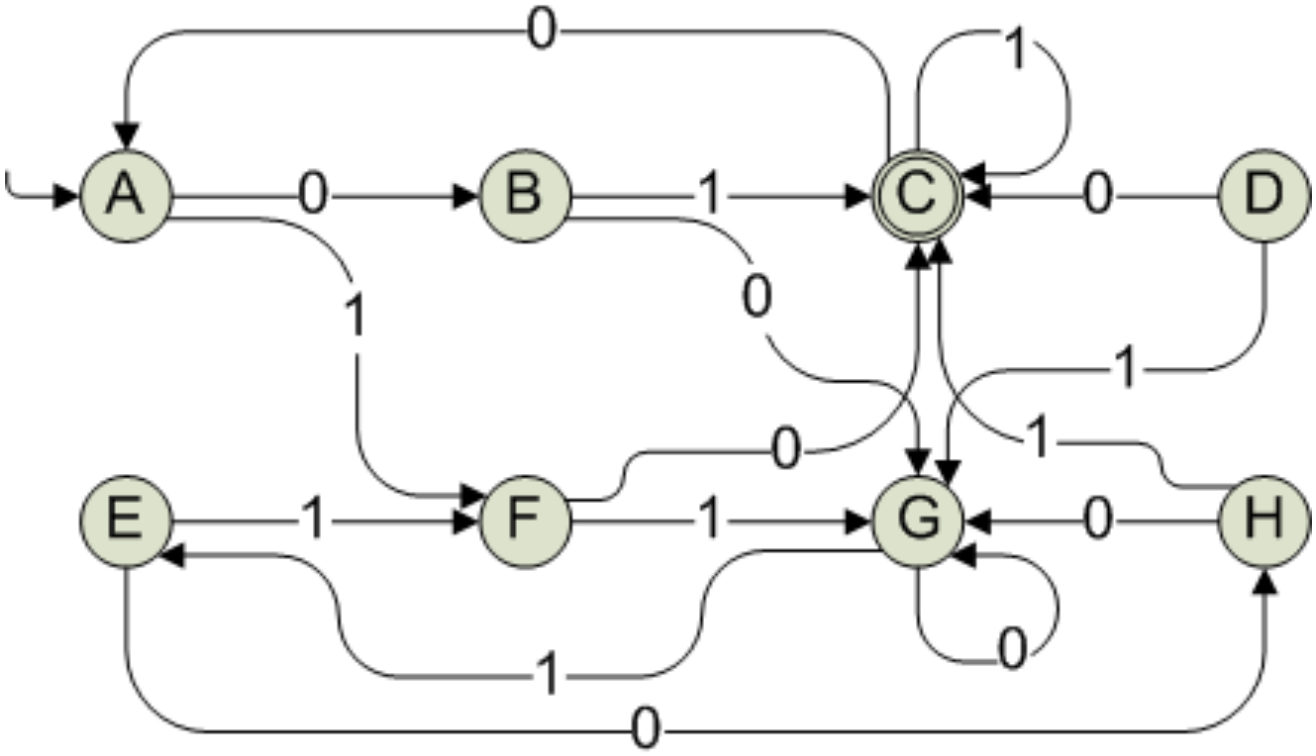
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

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

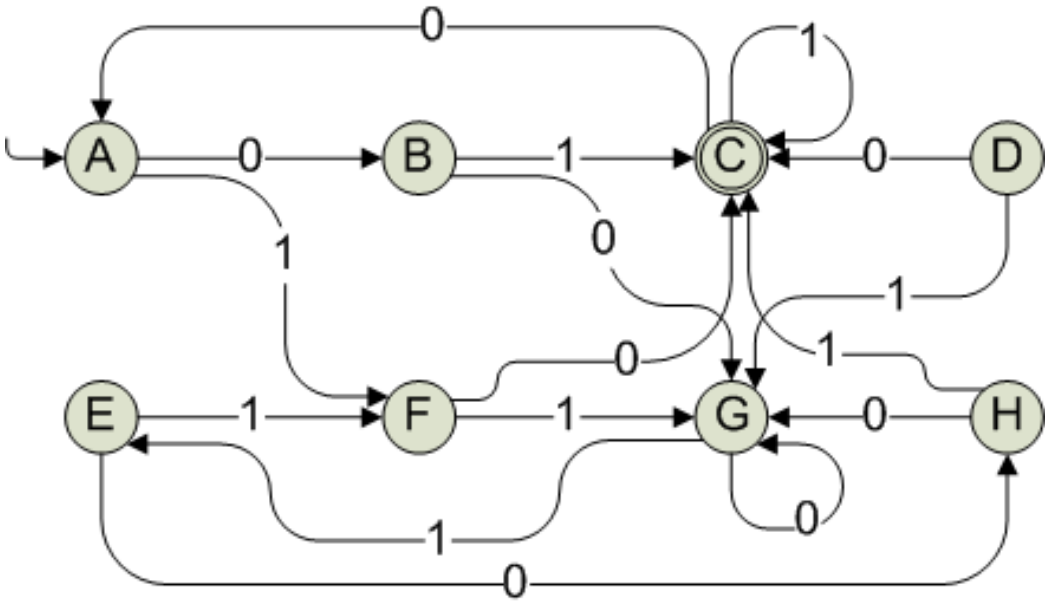
Estados inalcançáveis



Estados alcançáveis devem ter um caminho a partir do estado inicial

Neste exemplo, o estado D é inalcançável

Particionamento em grupos de estados equivalentes



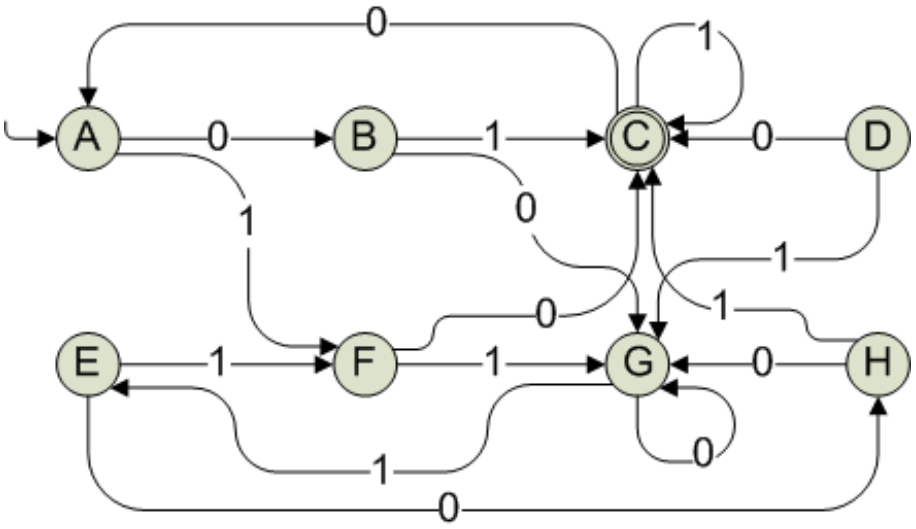
Neste exemplo, são pares equivalentes: (A,E), (B,H) e (D,F)
- Partição: {A,E},{B,H},{D,F},{C},{G}

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}

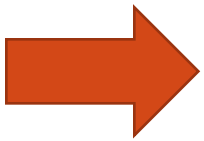
Agora, basta substituir os valores das células por grupos que representam estados válidos (a primeira coluna da tabela)

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

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

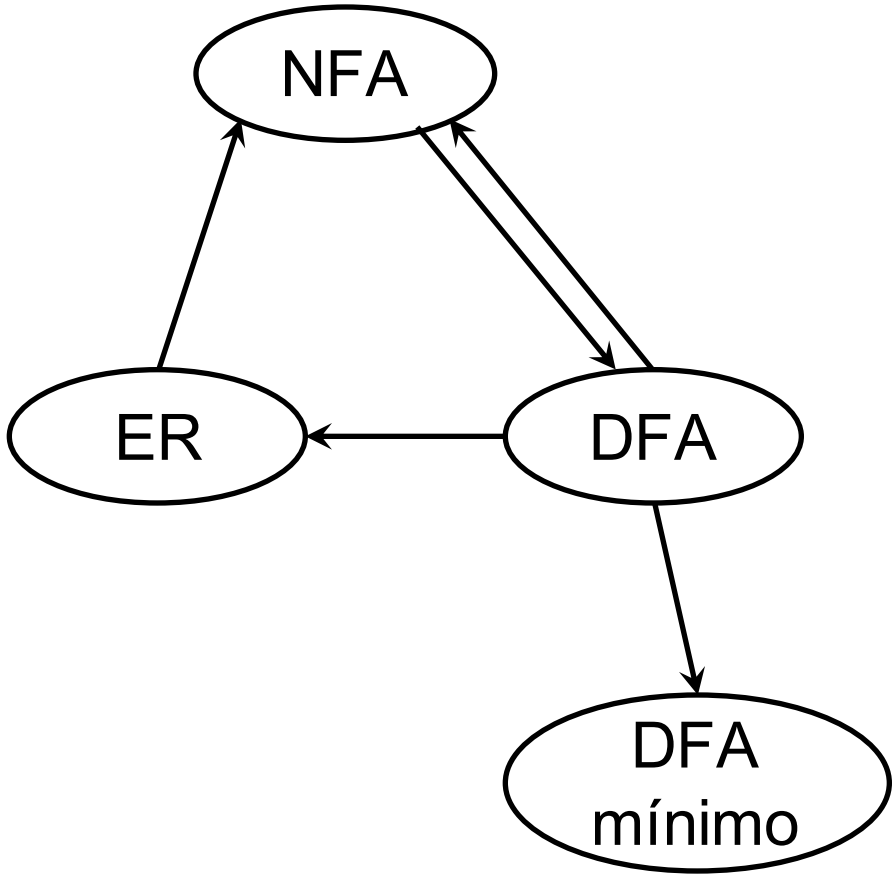


Para concluir, renomeie os estados para ficar mais legível

	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



Fim