Propriedades de Linguagens Regulares

Propriedades de decisão

Pertinência, caráter vazio, Etc.

O Lema do Bombeamento

Tradução dos slides do Prof. Jeffrey D. Ullman (Stanford University)

Propriedades de Classes de Linguagens

- Uma *classe de linguagem* é um conjunto de linguagens.
  - Exemplo: as linguagens regulares.
- Classes de linguagens têm dois importantes tipos de propriedades:
  1. Propriedades de Decisão.
  2. Propriedades de Fechamento.

Representação de Linguagens

- Representações podem ser formal ou informal.
- Exemplo (formal): representar uma linguagem por um RE ou DFA definindo-a.
- Exemplo: (informal): uma declaração lógica sobre as strings:
  - $\{0^n1^n \mid n$ é um inteiro não-negativo$\}$
  - "Conjunto de strings consistindo de algum número de 0's seguidos pelo mesmo número de 1's."

Propriedades de Decisão

- Uma *propriedade de decisão* para uma classe de linguagens é um algoritmo para responder perguntas importantes sobre autômatos.
- Exemplo: > A linguagem L é vazia?
  > Uma string w pertence à uma linguagem descrita?
  > Dois autômatos definem a mesma linguagem?

Problemas com representaçãoes

- Vamos imaginar que a linguagem é descrita informalmente, por isso, se minha descrição é "a linguagem vazia", então sim, caso contrário, não.
- Mas se representação é um DFA (ou um RE que poderá ser convertido em um DFA).
- Podemos dizer se L(A) = ∅ para um DFA A?

Por que Propriedades de Decisão?

- Quando falamos sobre protocolos representados como DFA's, notamos que importantes propriedades de um bom protocolo são relacionadas com a linguagem de um DFA.
- Exemplo: "O protocolo termina?" = "A linguagem é finita?"
- Exemplo: "O protocolo pode falhar?" = "A linguagem é não-vazia?"

Por que Propriedades de Decisão?

- Podemos querer uma representação "mínima"para uma linguagem, ex. Um DFA com número mínimo de estados ou uma RE mais curta.
- Se não pudéssemos decidir "Estas duas linguagens são as mesmas?"
  - Ou seja, se dois DFA's definem a mesma linguagem?

Não podemos encontrar o menor.

Propriedades de Fechamento

- Uma *propriedade de fechamento* de uma classe de linguagem diz que dada uma linguagem na classe, um operador (ex., união) produz uma outra linguagem na mesma classe.
- Exemplo: as linguagens regulares são obviamente fechadas sob união, concatenação e fechamento (Kleene).
  - Use um representação RE de linguagens.

Por que Propriedades de Fechamento?

1. Ajuda a construir representações.
2. Ajuda mostrar (descrito informalmente) que linguagens não estão em uma classe.

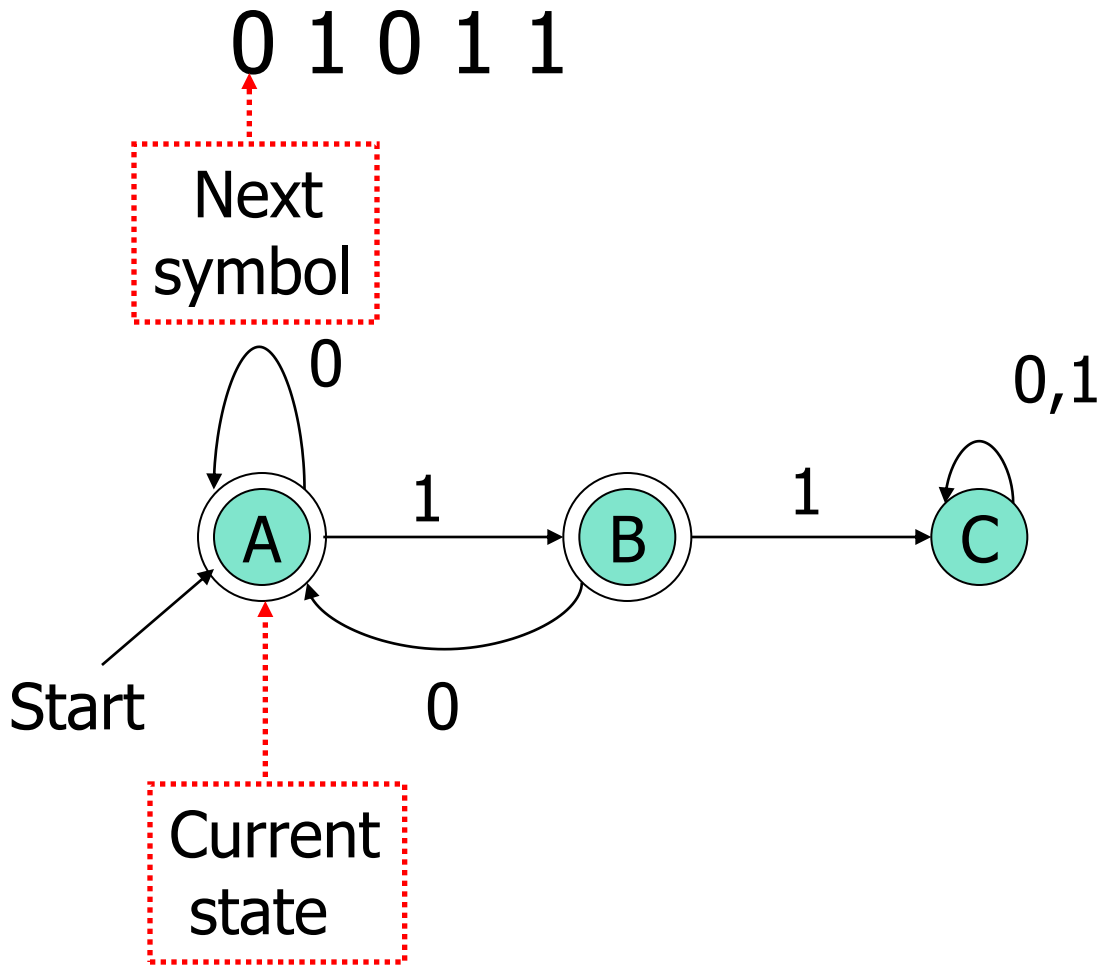
Example: Propriedade de Fechamento

- Podemos facilmente provar que $L_1 = \{0^n1^n \mid n \geq 0\}$ não é uma linguagem regular.
- L_2 = conjunto de strings com um número igual de 0's e 1's também não é regular. É mais complicado provar.
- Linguagens regulares são fechadas sob $\cap$.
- Se L_2 é regular, então $L_2 \cap L(\mathbf{0}^*\mathbf{1}^*) = L_1$ deveria ser também, mas não é.

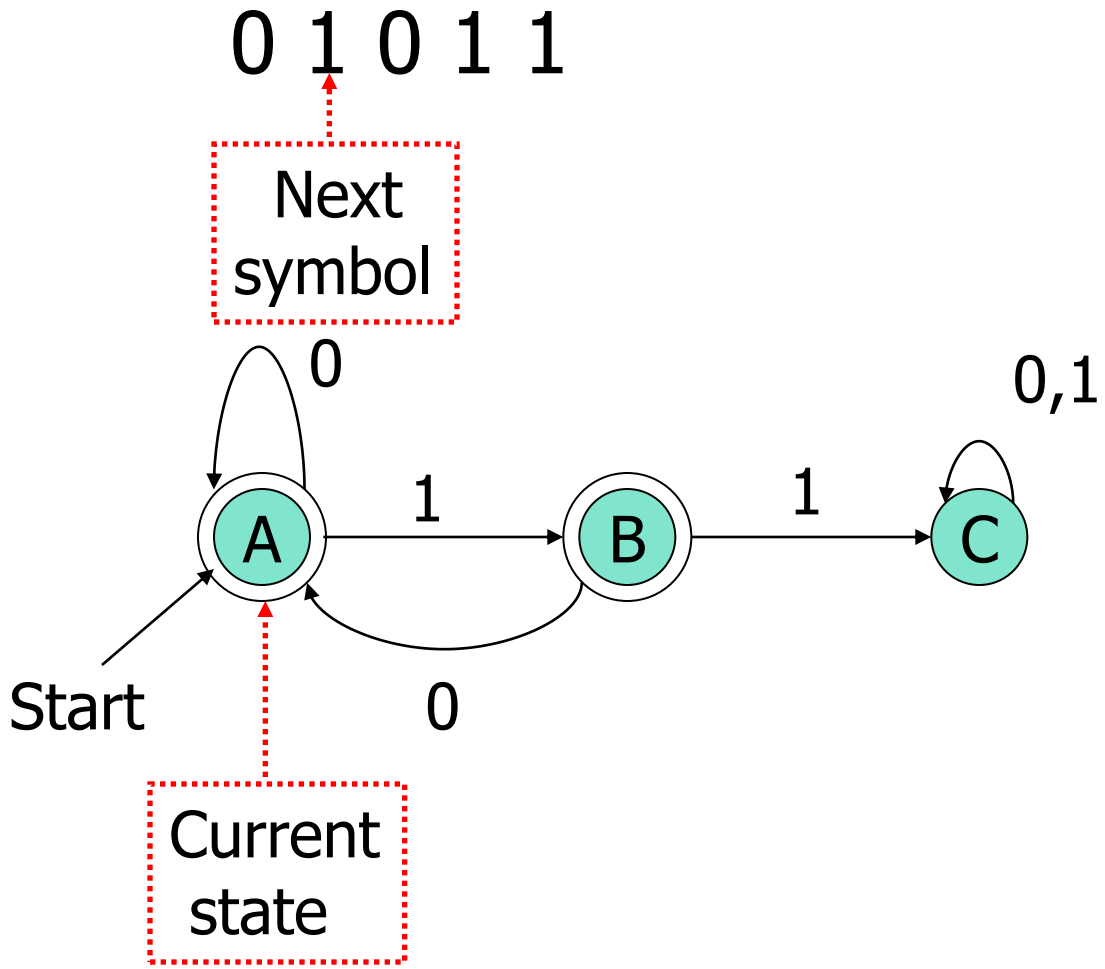
Testar a pertinência

- Nossa primeira propriedade de decisão é a questão: "a string w está na linguagem regular L?"
- Seja L representada por um DFA A.
- Simule o DFA processando o string de símbolos de entrada w.

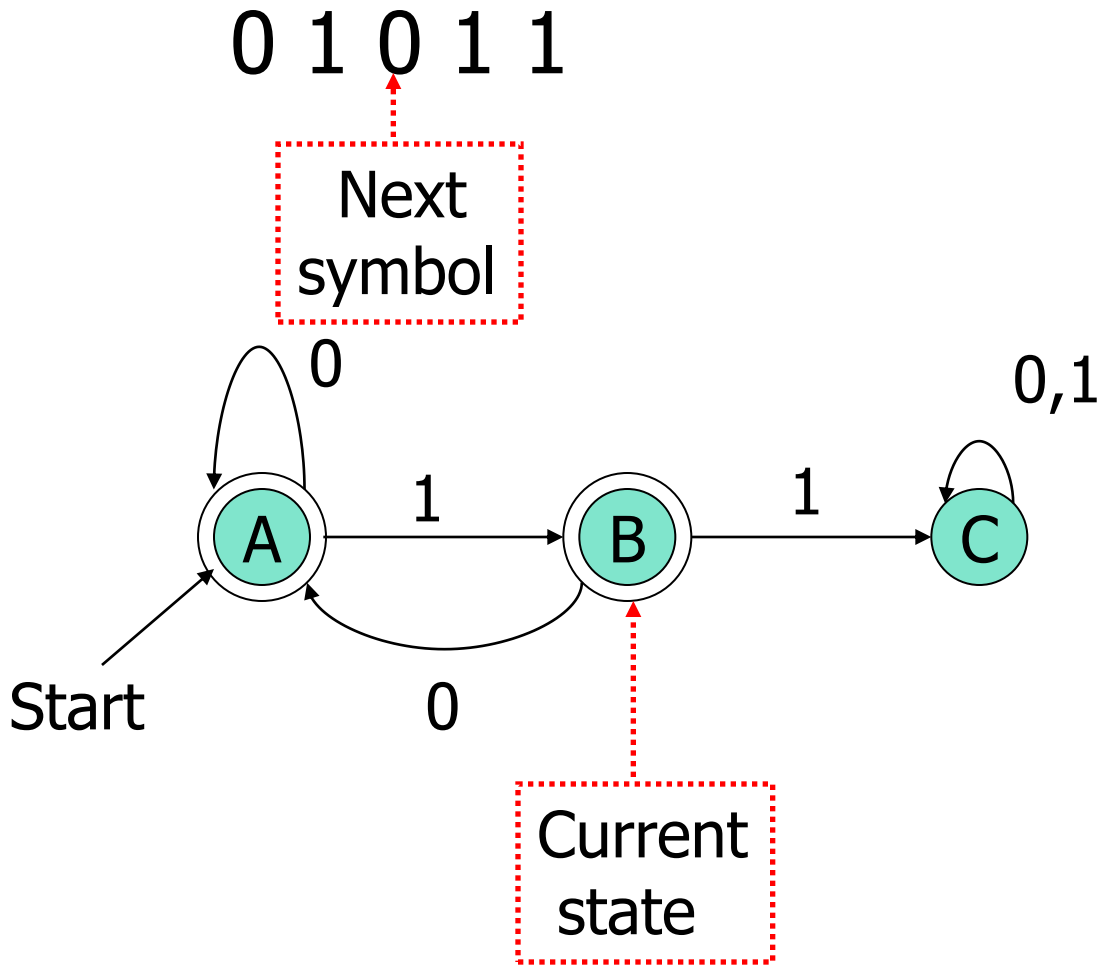
Exemplo: Testar a pertinência



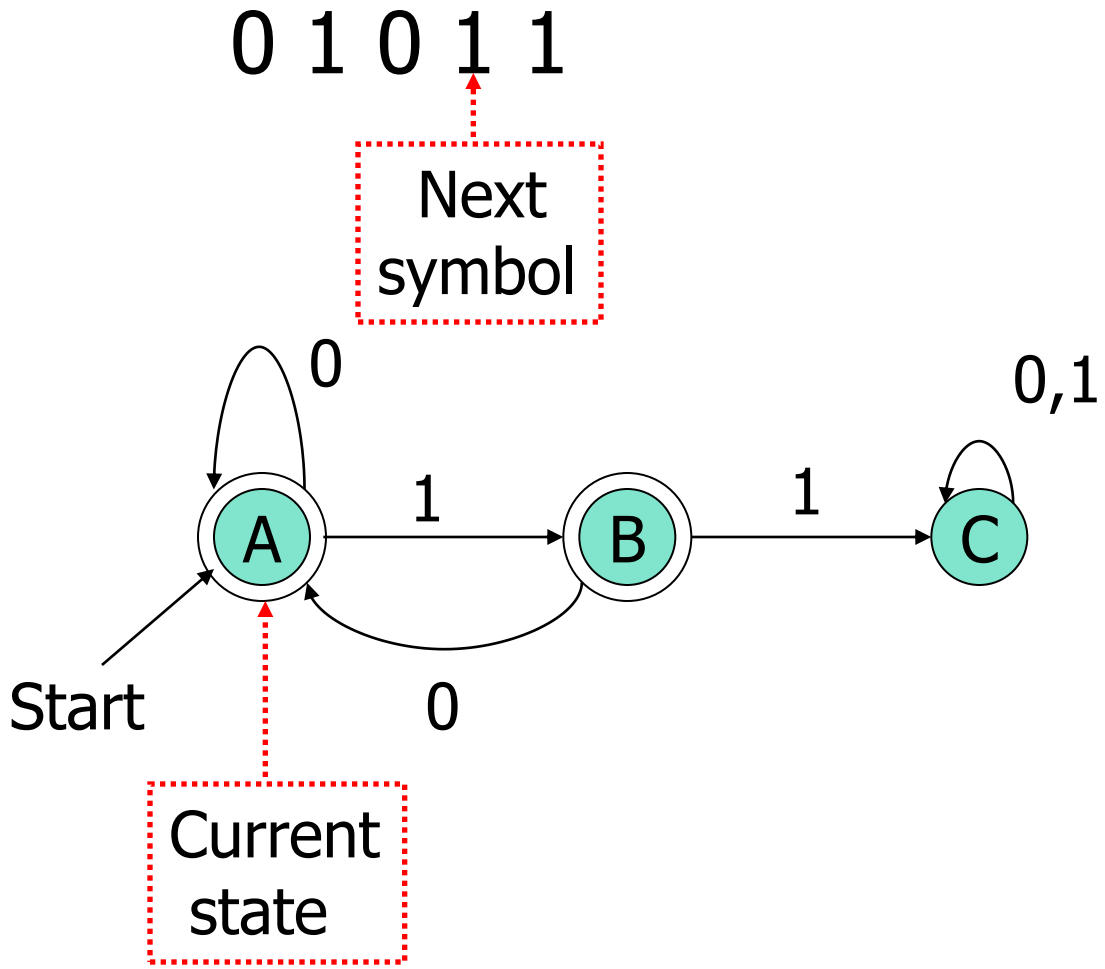
Exemplo: Testar a pertinência



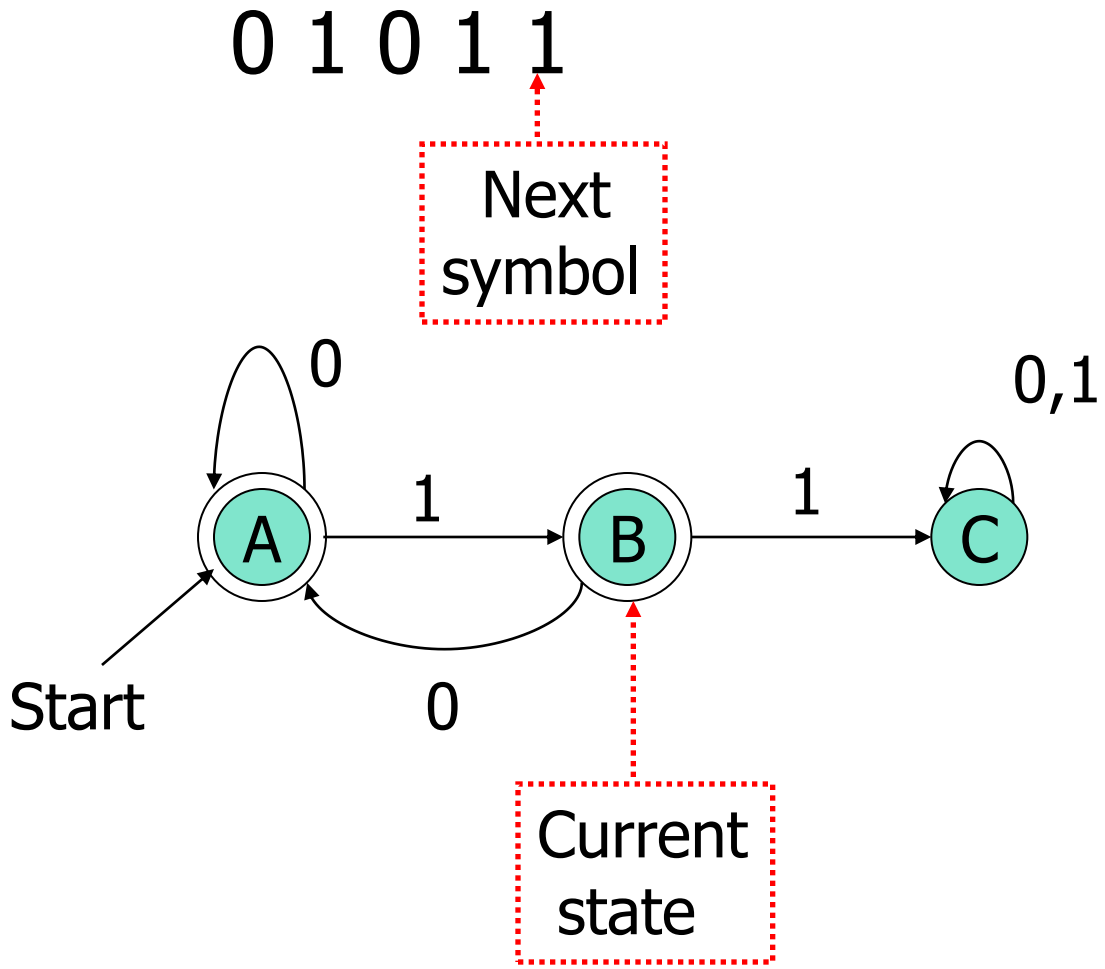
Exemplo: Testar a pertinência



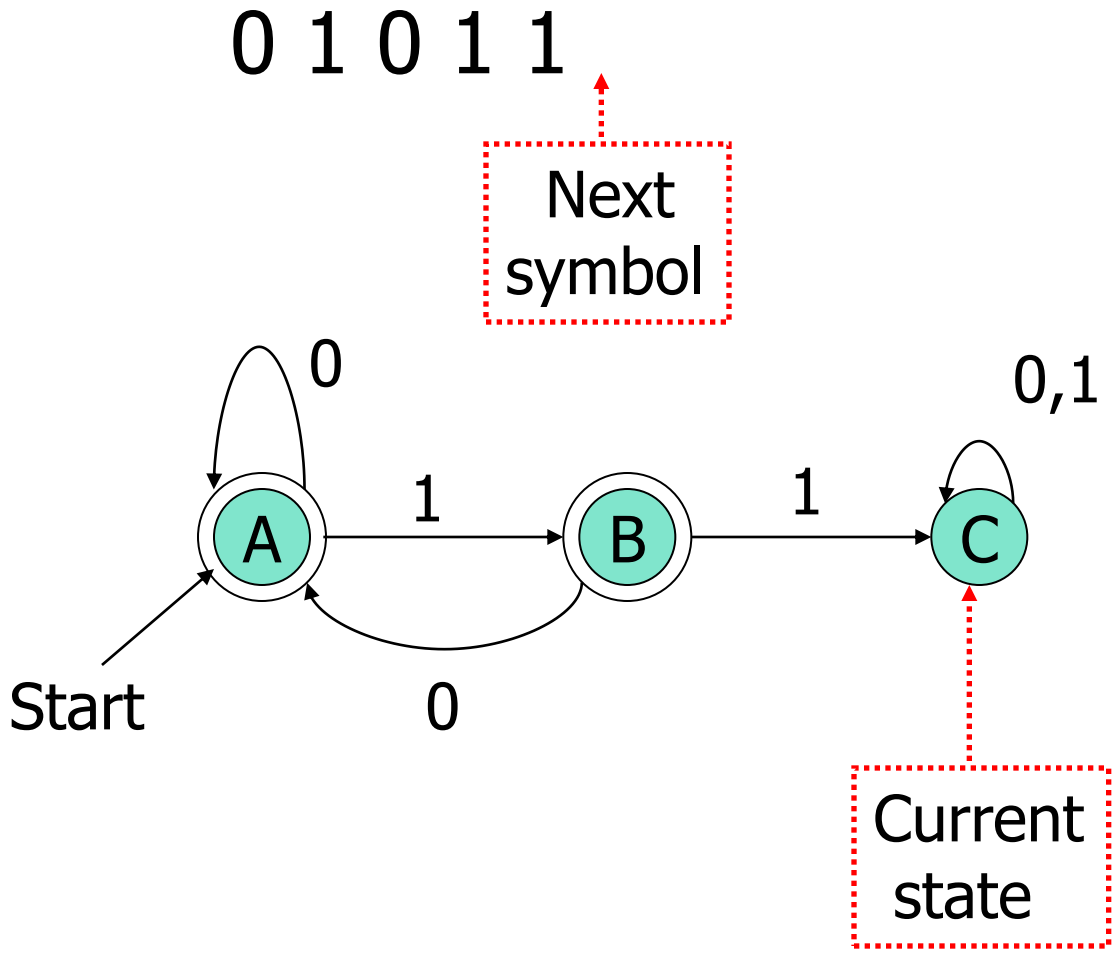
Exemplo: Testar a pertinência



Exemplo: Testar a pertinência

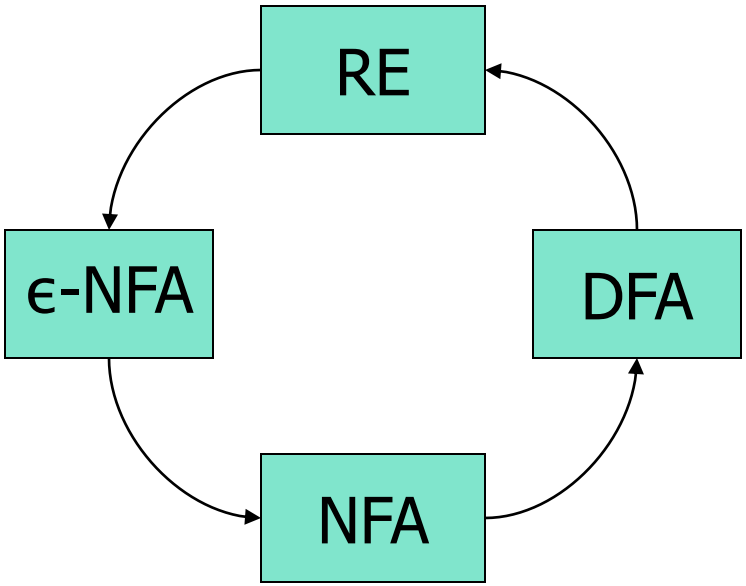


Exemplo: Testar a pertinência



O que fazer se a Linguagem Regular não for representada por um DFA?

- Existe um ciclo de conversões de uma forma para outra:



Testar o caráter vazio

- Dada uma linguagem regular, a linguagem não contêm qualquer string.
- Suponha que a representação é um DFA.
- Construa o diagrama de transição (grafo).
- Calcule o conjunto de estados alcançáveis a partir do estado inical.
- Se qualquer estado final é alcançável, então não, senão sim.

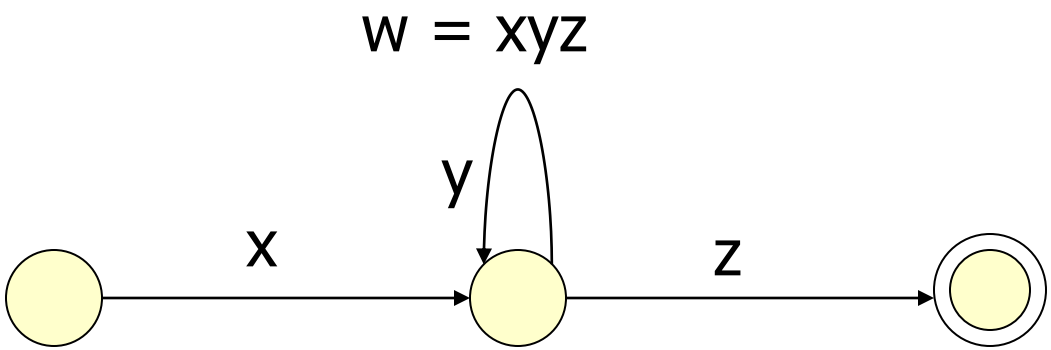
O Problema da Infinitude

- Uma dada linguagem regular é infinita?
- Comece com um DFA para a linguagem.
- Ideia: se o DFA tem *n* estados, e a linguagem contem algum string de comprimento *n* ou maior, então a linguagem é infinita.
- Caso contrário, a linguagem é finita.
  - Limitado para strings de comprimento < *n.*

Prova da Ideia

- Se um DFA com n-estados aceita uma string w de comprimento *n* ou maior, então deve haver um estado que aparece duas vezes no caminho rotulado por w do estado inicial até o estado final.
- Uma vez que existem pelo menos n +1 estados ao longo do caminho de w.

Prova – (2)



Então xy^iz está na linguagem para todo $i \geq 0$.

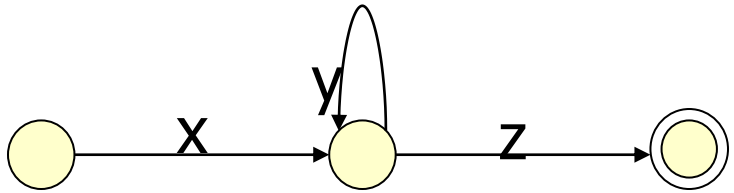
Como y não é ϵ, vemos um número infinito de strings em L.

Infinitude – Continuação

- Ainda não temos um algoritmo.
- Existe um número infinito de strings de comprimento > n, e não podemos testar todas.
- Segunda ideia: se existe um string de comprimento $\geq$ n (= número de estados) em L, então existe um string de comprimento entre n e 2n-1.

Prova da 2nd Ideia

- Lembre:



- Podemos escolher y para ser o primeiro ciclo no caminho.
- Assim $|xy| \leq n$; em particular, $1 \leq |y| \leq n$.
- Portanto, se w tem comprimento 2n ou mais, existe um string mais curto em L que é ainda de comprimento pelo menos n.
- Manter encurtando para alcançar [n, 2n-1].

Algoritmo de Infinitude

- Teste a pertinência de todos strings de comprimento entre n e 2n-1.
  - Se alguma for aceita, então L é infinita, senão finita.
- Um algoritmo terrível.
- Melhor: encontrar ciclos entre o estado inicial e um estado final.

Encontrando Ciclos

1. Elimine os estados que não podem ser alcançadas a partir do estado inicial.
2. Elimine os estados que não atingem um estado final.
3. Testar se o grafo de transição restante tem algum ciclo.

O Lema do Bombeamento

- Provou-se (quase acidentalmente) uma afirmação que é muito útil para mostrar que certas linguagens não são regulares.
- Chamado *lema do bombeamento para linguagens regulares*.

Teorema do Lema do bombeamento

Para toda linguagem regular L

Número de estados do DFA para L

Existe um inteiro n, tal que

Para todo string w em L de comprimento $\geq$ n

Podemos dividir w = xyz tal que:

1. $|xy| \leq n$.
2. $|y| > 0$.
3. Para todo $i \geq 0$, xy^iz está em L.

Exemplo: Lema do bombeamento

- Nós afirmamos quw $\{0^k1^k \mid k \geq 1\}$ não é uma linguagem regular.
- Vamos supor que fosse. Então haveria uma constante n satisfazendo às condições do lema.
- Seja $w = 0^n1^n$. Podemos escrever $w = xyz$, onde x e y consiste de 0's, e $y \neq \epsilon$.
- Mas então xz estaria em L, e esta string tem menos 0's que 1's.

Lema do bombeamento como um jogo de competição

- Para todas linguagens regulares L existe *n* tal que, para todo w em L com $|w| \geq n$ existe xyz igual a w tal que $y \neq \varepsilon$, $|xy| \leq n$ e para todo $i \geq 0$, xy^iz também está em L.
  - O jogador 1 escolhe a linguagem
  - O jogador 2 escolhe n, mas não o revela
  - O jogador 1 escolhe w ($|w| \geq n$)
  - O jogador 2 divide w em x, y e z (mas não revela)
  - O jogador 1 ganha escolhendo i, tal que xy^iz não está em L

Testar a equivalência de estados

- Quando dois estados distintos *p* e *q* podem ser substituídos por um único estado que se comporte como *p* e *q*?

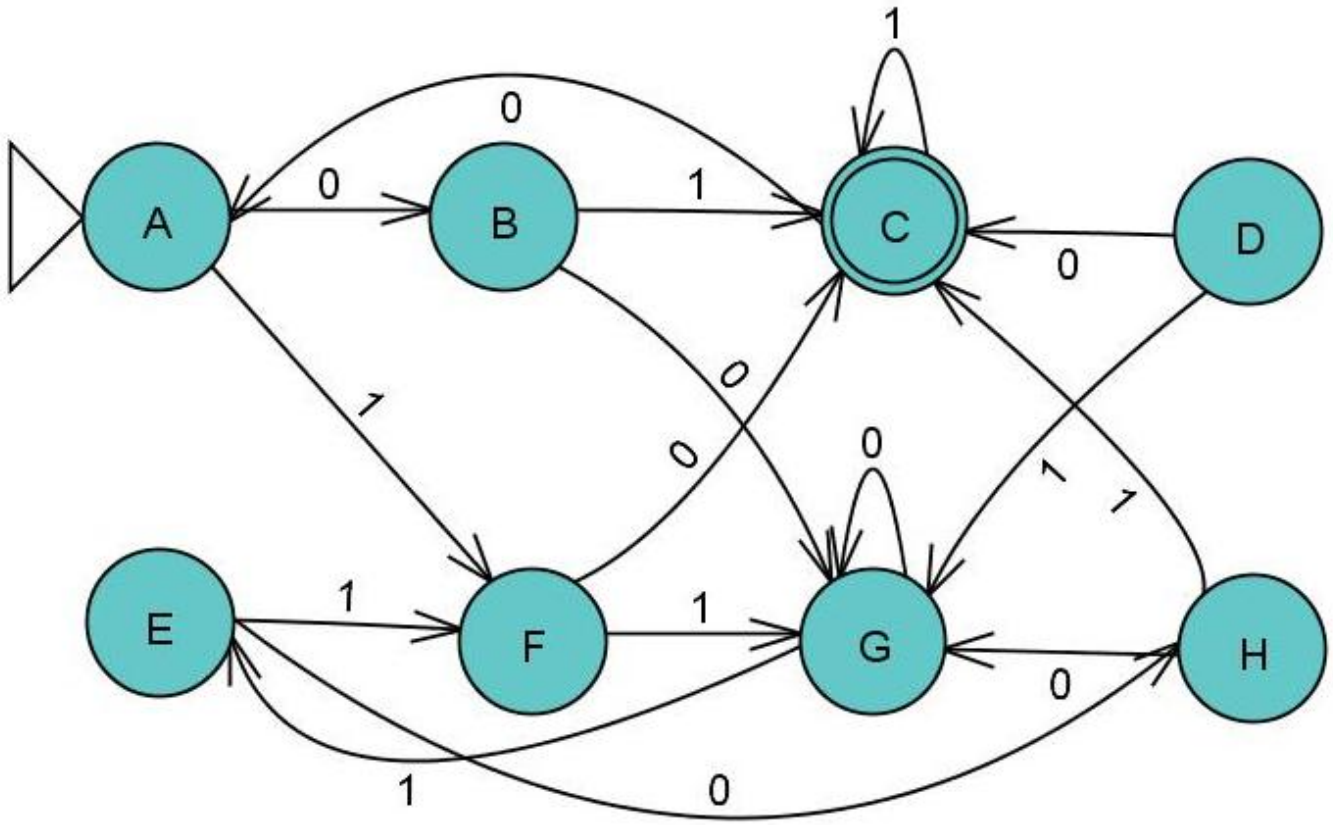
  p e *q* são equivalentes se:

  - Para todos os strings de entrada w, δ(p,w) é um estado de aceitação se e somente se δ(q,w) é um estado de aceitação.

Testar a equivalência de estados

- Informal: Não exigimos que $\delta(p,w)$ e $\delta(q,w)$ sejam o mesmo estado, apenas que ambos sejam de aceitação ou ambos sejam de não-aceitação.
- Se dois estados não são equivalentes, então são *distinguíveis*.
- p é distinguível de q se existe pelo menos um string w tal que $\delta(p,w)$ ou $\delta(q,w)$ é de aceitação, e o outro é de não-aceitação.

Testar a equivalência de estados



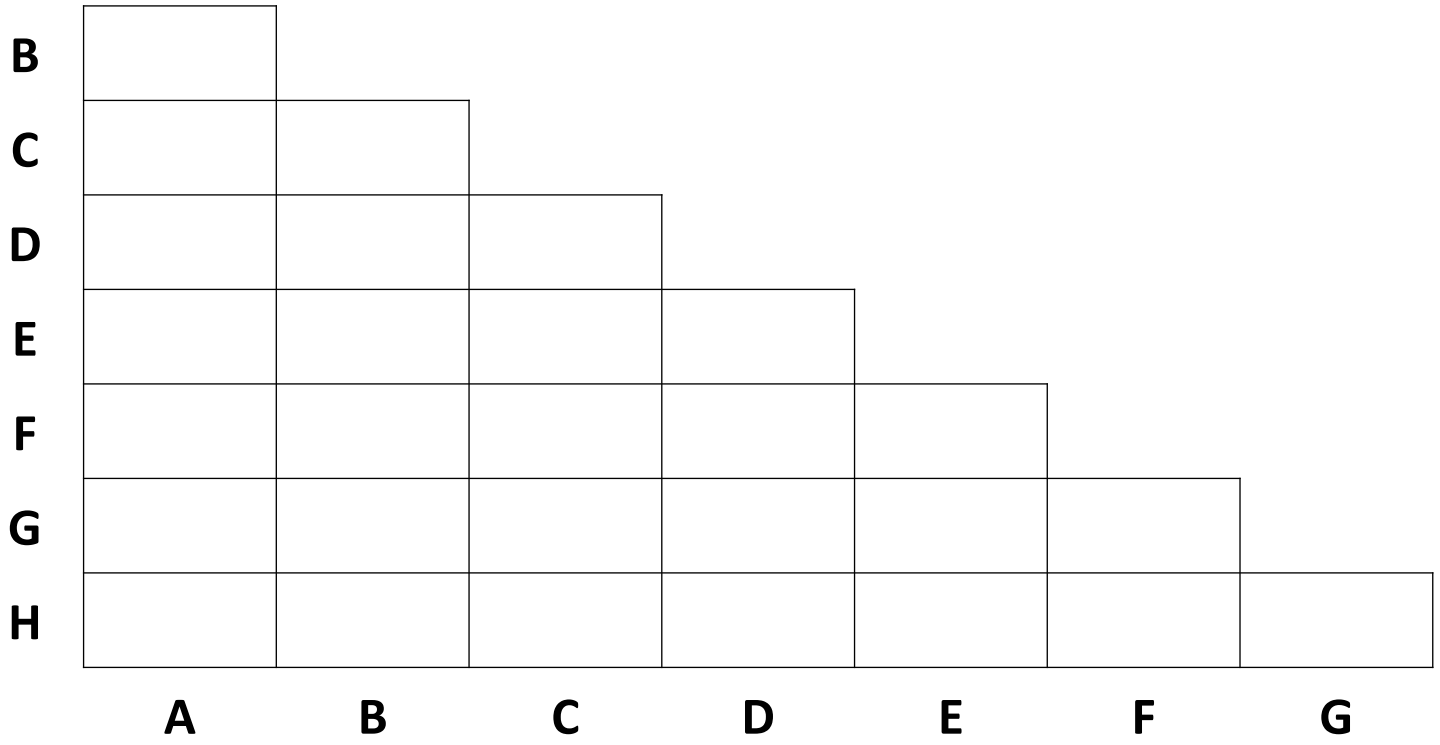
Testar a equivalência de estados

- Para encontrar estados que sejam equivalentes, dedicaremos o melhor de nosso esforço a encontrar pares de estados que sejam distinguíveis.
  - Qualquer par de estados que não sejam distinguíveis serão equivalentes
  - **Algoritmo de preenchimento de tabela**

    Descoberta recursiva de pares distinguíveis em um DFA

Algoritmo de preenchimento de tabela

Tabela de não equivalência de estados



Algoritmo de preenchimento de tabela

Tabela de não equivalência de estados

	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

Testar a equivalência de linguagens regulares

- Suponha que cada uma das linguagens L e M seja representada de algum modo.
- Converta cada representação em um DFA.
- Imagine um DFA cujos estados sejam a **união** dos estados dos DFA's de L e M.
- Teste se os estados iniciais dos dois DFA's originais são equivalentes, usando o algoritmo de preenchimento de tabela.
  - Se eles forem equivalentes, então **L = M** e,
  - Em caso contrário, então **L ≠ M**.

Minimização de DFA's

- Para cada DFA podemos encontrar um DFA equivalente que tem tão poucos estados quanto qualquer DFA que aceita a mesma linguagem.
  - Esse DFA de número mínimo de estados é **único** para a linguagem.

Minimização de DFA's

- Algoritmo:
  - Elimine qualquer estado que não possa ser acessado a partir do estado inicial.
  - Particione os estados restantes em blocos, de forma que todos os estados no mesmo bloco sejam equivalentes, e que nenhum par de estados de blocos diferentes seja eequivalentes.

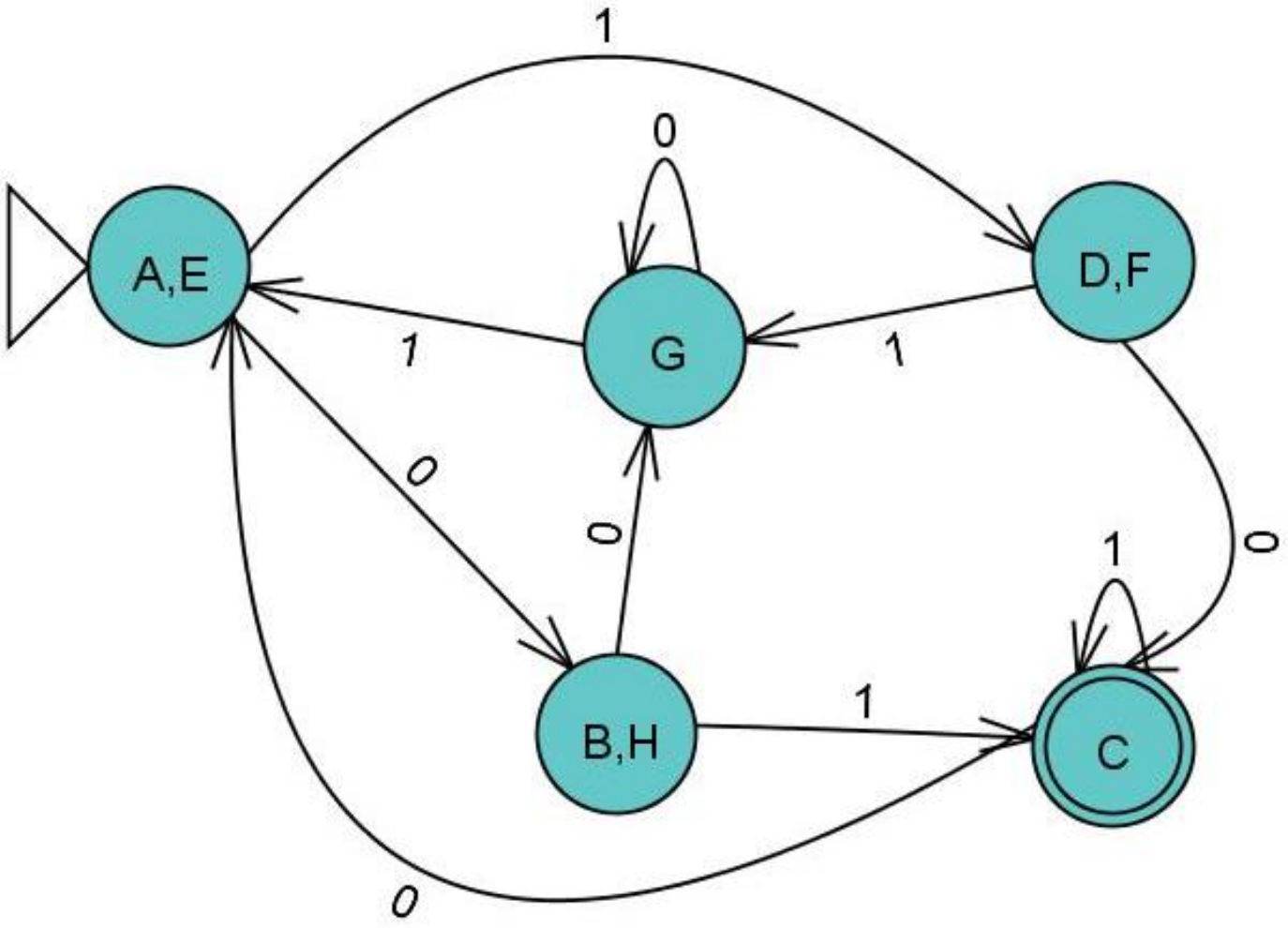
Minimização de DFA's

- A equivalência de estados é transitiva
  - Se *p* e *q* são equivalentes, e *q* e *r* são equivalentes, então *p* e *r* também são equivalentes.

Minimização de DFA's

- Se criarmos para cada estado *q* de um DFA um bloco consistindo em *q* e em todos os estados equivalentes a q, então os diferentes blocos de estados formarão uma **partição** do conjunto de estados.
  - Cada estado está exatamente em um bloco
  - Todos os elementos de um bloco são equivalentes
  - Nenhum par de estados escolhidos de diferentes blocos é equivalente

Minimização de DFA's



Exercícios

- Desenhe a tabela de distinções para esse autômato.
- Construa o DFA com o número mínimo de estados equivalente.

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

Exercícios

- Desenhe a tabela de distinções para esse autômato.
- Construa o DFA com o número mínimo de estados equivalente.

	0	1
→ 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