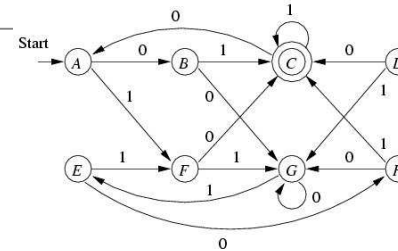


Equivalência e Minimização de Autômatos

■ Seja um AFD $A = (Q, \Sigma, \delta, q_0, F)$ e $\{p, q\} \subseteq Q$. Defina-se:

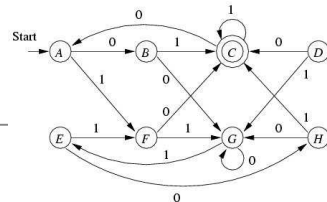
- $p \equiv q \Leftrightarrow \forall w \in \Sigma^*, \delta(p, w) \in F \text{ sse } \delta(q, w) \in F$;
 - Se $p \equiv q$ diz-se que p e q são equivalentes;
 - Se p não equivale a q dizemos que p e q são distintos;
- Em outras palavras, p e q são distintos sse existe uma palavra w tal que: $\delta(p, w) \in F$ e $\delta(q, w) \notin F$, ou vice-versa.

Exemplo



- $\delta(C, \epsilon) \in F$ e $\delta(G, \epsilon) \notin F \Rightarrow C$ não equivale à G
- $\delta(A, 01) = C \in F$ e $\delta(G, 01) = E \notin F \Rightarrow A$ não equivale à G

Exemplo (cont.)



■ Analisando os estados A e E:

- $\delta(A, \epsilon) \notin F$ e $\delta(E, \epsilon) \notin F$
- $\delta(A, 1) = F = \delta(E, 1) = F$
- $\delta(A, 1x) = \delta(E, 1x)$
- $\delta(A, 00) = G = \delta(E, 00)$
- $\delta(A, 01) = C = \delta(E, 01)$

Conclusão: $A \equiv E$

Computando Estados Distintos (Estados Não-Equivalentes)

- Construir uma tabela relacionando os estados distintos, onde cada par de estados ocorre somente uma vez.
- Algoritmo indutivo (Table Filling Algorithm – TF-algo):
 - Base: Se $p \in F$ e $q \notin F$, então p não equivale a q (o par (p,q) deve ser marcado);
 - Indução: Supõe-se $\exists a \in \Sigma$, onde $\delta(p, a) = r$ e $\delta(q, a) = s$,
 - Se $r = s$, então p é equivalente a q e o par (p, q) não deve ser marcado;
 - Se $r \neq s$ e o par (r, s) não está marcado, então (p, q) é incluído em uma lista a partir de (r, s) para posterior análise;
 - Se $r \neq s$ e o par (r, s) está marcado, então (p, q) não é equivalente e deve ser marcado. Caso (p, q) encabece uma lista de pares, então marcar todos os pares da lista (e, recursivamente, se algum par da lista encabeça outra lista).

No exemplo:

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

Teorema: Se p e q não são marcados na tabela pelo algoritmo anterior, então $p \equiv q$.

Prova: Suponha que (p, q) seja um par ruim tal que:

1. Existe uma palavra w onde: $\delta(p, w) \in F$ e $\delta(q, w) \notin F$, ou vice-versa, i. e., p e q não são equivalentes e,
2. A tabela não distingue p de q ;

Seja $w = a_1 a_2 \dots a_n$ a menor palavra que identifica o par (p, q) .

Agora $w \neq \epsilon$ desde que, de outra maneira, estaria na base distinguindo p de q . Portanto $n \geq 1$.

Prova (cont.)

Considere os estados $r = \delta(p, a_1)$ e $s = \delta(q, a_1)$. Agora, (r, s) não pode ser um par ruim desde que (r, s) deve ser identificado por uma palavra menor do que w . Portanto, TF-algo deve descobrir que r e s não são equivalentes.

Mas então, TF-algo distingue p de q na parte indutiva. Logo, não existem pares ruins e o teorema é verdadeiro.

Equivalência de Linguagens Regulares

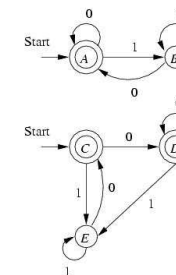
Sejam L e M linguagens regulares (cada uma expressa de alguma forma). Para testar se $L = M$ deve-se:

1. Converter ambas para AFD's;
2. Imaginar o AFD que é a união dos estados dos dois AFD's (tecnicamente existem dois estados iniciais, mas o estado inicial é irrelevante já que estamos testando a equivalência; então qualquer um pode ser considerado como o inicial);
3. Se o TF-algo os dois estados iniciais são não-equivalentes, então $L \neq M$. Caso contrário, $L = M$.

Exemplo

Ambos AFD's aceitam $L(\epsilon + (0 + 1)^*0)$.

Portanto, os dois AFD's são equivalentes.



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

Minimização de AFD's

É possível minimizar um AFD unificando os estados equivalentes a partir de TF-algo. Basta substituir cada estado p pela p/\equiv

Ex.1(slide 2) O AFD é formado pelas seguintes classes de equivalência $\{\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}\}$

Ex.2(slide 8) A união dos AFD's tem as seguintes classes de equivalência $\{\{A, C, D\}, \{B, E\}\}$

Obs.: Para que p/\equiv seja uma classe de equivalência, a relação \equiv deve ser uma relação de equivalência (reflexiva, simétrica, transitiva).

Teorema

(Transitividade) Se $p \equiv q$ e $q \equiv r$, então $p \equiv r$.

Prova: Suponha que p não equivale a r . Então existe uma palavra w tal que $\delta(p, w) \in F$ e $\delta(r, w) \notin F$, ou vice-versa.

Agora existem duas possibilidades: $\delta(q, w)$ é final ou não.

Caso 1: Se $\delta(q, w)$ é final. Então q não equivale a r . Contradição!

Caso 2: Se $\delta(q, w)$ não é final. Então q não equivale a p . Contradição!

A prova do vice-versa é similar. Portanto, deve-se ter $p \equiv r$.

Minimização

Para minimizar um AFD $A = (Q, \Sigma, \delta, q_0, F)$, constrói-se um AFD $B = (Q/\equiv, \Sigma, \gamma, q_0/\equiv, F/\equiv)$, onde

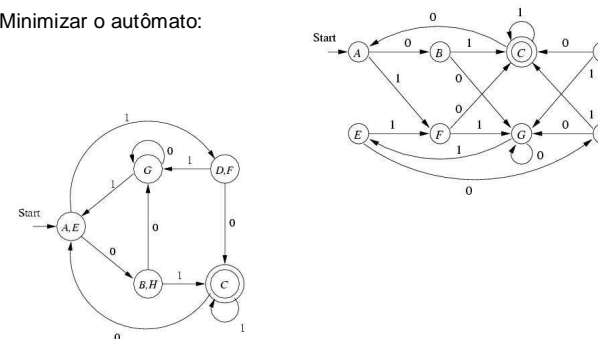
$$\gamma(p/\equiv, a) = \delta(p, a)/\equiv.$$

Para que B esteja bem definido devemos mostrar que: Se $p \equiv q$ então $\delta(p, a) \equiv \delta(q, a)$.

Se $\delta(p, a)$ não equivale a $\delta(q, a)$, então o TF-algo concluiria que p não equivale a q , então B estão bem definidos. Note também que F/\equiv contém apenas estados de aceitação de A .

Exemplo

Minimizar o autômato:

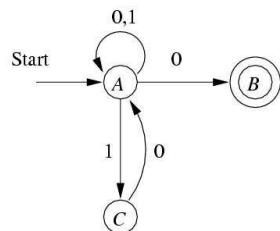


AFN?

- Não é possível aplicar a Minimização para AFN!

Exemplo:

Para minimizar o AFN ao lado basta remover o estado C, embora A não equivale a C.



Unicidade do AFD minimizado

Seja B um AFD minimizado obtido pelo TF-algo a partir de A.

Sabe-se que $L(A) = L(B)$.

Suponha que exista um AFD C equivalente com menos estados do que B.

Aplicando o TF-algo a união de B com C, desde que $L(B) = L(C)$ teremos que $q_0B \equiv q_0C$. Também $\delta(q_0B, a) \equiv \delta(q_0C, a)$, para qualquer a. Assim:

Para cada estado p em B existe pelo menos um estado q em C, tal que $p \equiv q$.

Prova: Como não existem estados inacessíveis então $p = \delta(q_0B, a_1a_2..ak)$, para alguma palavra $a_1a_2..ak$. Agora $q = \delta(q_0C, a_1a_2..ak)$, e $p \equiv q$.

Desde que C tem menos estados que B, devem existir dois estado r e s de B tal que $r \equiv t \equiv s$, para algum estado t de C. Mas então $r \equiv s$, o que é uma contradição, desde que B foi construído pelo TF-algo.