# Documentação do Otimizador de autômatos

Paulo Cressoni & Lucas Biason 4A prof. Jamil Kaliu

## autootimizadorv3.pngdeletarEstadosInvalidos().pngelEInac(estado_atual).pngelEInut(estado_atual).pngeliminarEstadosInacessiveis().pngeliminarEstadosInuteis().pngeliminarNaoDeterminismo().pngeliminarTransicoesEmVazio().png

## Exemplo 01

M1=({q0,q1,q2,q3},

{a, b,c},

{(q0,a)=q0 ; (q0,b)=q1 ; (q0,c)=q3 ; (q1,a)=q3 ; (q1,b)=q1; (q1,c)=q2; (q2,a)=q3; (q2,b)=q3; (q2,c)=q2; (q3,a)=q3;

(q3,b)=q3; (q3,c)=q3 ;},

{q0},

{q1,q2})

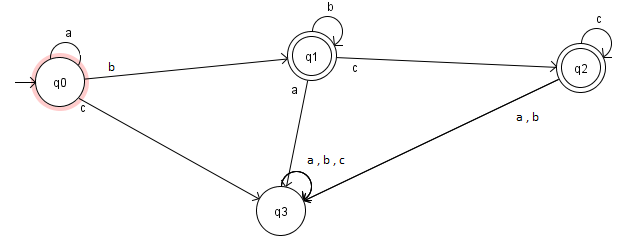


Tabela de transições:

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q0 | q1 | q3 |
| q1 (f) | q3 | q1 | q2 |
| q2 (f) | q3 | q3 | q2 |
| q3 | q3 | q3 | q3 |

- Não há transações em Vazio.

- Não há Não-Determinismo.

### Aplicação do algoritmo para eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | A | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X |  |
| q1 (f) | q3 | q1 | q2 |  |  |
| q2 (f) | q3 | q3 | q2 |  |  |
| q3 | q3 | q3 | q3 |  |  |

- q0 inicial é marcado como acessível;

- q0 referencia q0, q1, q3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1 (f) | q3 | q1 | q2 | X |  |
| q2 (f) | q3 | q3 | q2 |  |  |
| q3 | q3 | q3 | q3 | X |  |

- q1 e q3 são marcados como acessíveis;

- q0 é marcado como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1 (f) | q3 | q1 | q2 | X | X |
| q2 (f) | q3 | q3 | q2 | X |  |
| q3 | q3 | q3 | q3 | X |  |

- próximo estado acessível não considerado q1 referencia q1, q2, q3;

- q3, q1, q2 marcados como acessíveis;

- q1 é marcado como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1 (f) | q3 | q1 | q2 | X | X |
| q2 (f) | q3 | q3 | q2 | X | X |
| q3 | q3 | q3 | q3 | X |  |

- próximo estado acessível não considerado q2 referencia q2, q3;

- q3, q2 já marcados como acessíveis;

- q2 é marcado como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1 (f) | q3 | q1 | q2 | X | X |
| q2 (f) | q3 | q3 | q2 | X | X |
| q3 | q3 | q3 | q3 | X | X |

- próximo estado acessível não considerado q3 referencia q3;

- q3 já marcado como acessível;

- q3 é marcado como considerado.

As linhas não consideradas podem ser eliminadas, neste caso todas foram consideradas, portanto todos estes estados são acessíveis. Agora verificar se há estados inúteis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q0 | q1 | q3 |  |  |
| q1 (f) | q3 | q1 | q2 | X |  |
| q2 (f) | q3 | q3 | q2 | X |  |
| q3 | q3 | q3 | q3 |  |  |

- estados finais q2, q1 são marcados como úteis;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q0 | q1 | q3 | X |  |
| q1 (f) | q3 | q1 | q2 | X | X |
| q2 (f) | q3 | q3 | q2 | X | X |
| q3 | q3 | q3 | q3 |  |  |

- as únicas referências à q2 e q3 estão em q1, q2, q3;

- marcar q1 como útil;

- marcar q3 e q2 como considerados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1 (f) | q3 | q1 | q2 | X | X |
| q2 (f) | q3 | q3 | q2 | X | X |
| q3 | q3 | q3 | q3 |  |  |

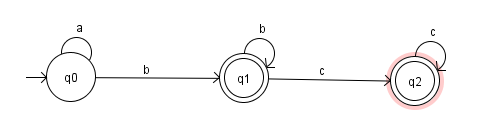
- o único útil não-considerado é q0;

- nenhum outro estado útil é considerado;

- marca q0 como considerado;

- o estado q3 pode ser eliminado.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q0 | q1 |  |
| q1 (f) |  | q1 | q2 |
| q2 (f) |  |  | q2 |



### Aplicação do algoritmo de minimização de autômatos:

Construção da tabela que relaciona os estados distintos, sendo que cada par não ordenado ocorre somente uma vez.

Marcação de estados trivialmente não equivalentes, ou seja, estados finais com não finais.

|  |  |  |
| --- | --- | --- |
|  | →q0 | q1(f) |
| q2(f) | X |  |
| q1(f) | X |  |

### Analisar os não marcados:

δ(q1, a) = não há δ.

δ(q2, a) = não há δ.

δ(q1, b) = q1.

δ(q2, b) = não há δ.

Como q1 e q2 consumindo o mesmo caractere resultam em valores diferentes, e um deles não resulta em nada, podemos assumir que não são equivalentes.

Podemos marcar {q1,q2} na tabela.

|  |  |  |
| --- | --- | --- |
|  | →q0 | q1(f) |
| q2(f) | X | X |
| q1(f) | X |  |

O autômato já está minimizado.

## Exemplo 02

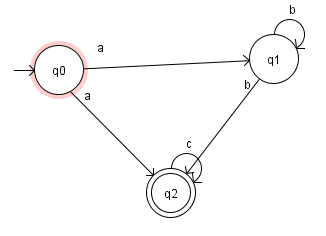
M2=({q0,q1,q2},

{a,b,c},

{(q0,a)={q1,q2} ; (q1,b)={q1,q2} ; (q2,c)=q2 ;},

{q0},

{q2})



|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | {q1, q2} |  |  |
| q1 |  | {q1, q2} |  |
| q2(f) |  |  | q2 |

Não há transições em vazio.

### Algoritmo para eliminação de não-determinismo:

Cria-se um novo estado para cada elemento da tabela de transição que contenha mais de um elemento.

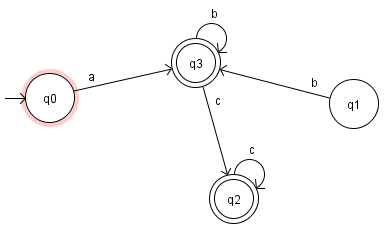
Cria-se o estado {q1,q2}, copiar as transições de q1 e q2 para o novo estado.

Como q2 é final, o novo estado, que contém q2, também é final.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | {q1, q2} |  |  |
| q1 |  | {q1, q2} |  |
| q2(f) |  |  | q2 |
| {q1, q2}(f) |  | {q1, q2} | q2 |

Vamos substitui {q1,q2} por q3.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q3 |  |  |
| q1 |  | q3 |  |
| q2(f) |  |  | q2 |
| q3(f) |  | q3 | q2 |



### Aplicação do algoritmo para eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 |  |  | X |  |
| q1 |  | q3 |  |  |  |
| q2(f) |  |  | q2 |  |  |
| q3(f) |  | q3 | q2 |  |  |

- marca q0 como acessível;

- q0 referencia q3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2(f) |  |  | q2 |  |  |
| q3(f) |  | q3 | q2 | X |  |

- marca q3 como acessível;

- marca q0 como considerado;

- q3 referencia q2 e q3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2(f) |  |  | q2 | X |  |
| q3(f) |  | q3 | q2 | X | X |

- marca q2 como acessível;

- marca q3 como considerado;

- q2 referencia a q2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2(f) |  |  | q2 | X | X |
| q3(f) |  | q3 | q2 | X | X |

- marca q2 como acessível.

- o estado q1 pode ser eliminado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 |  |  |  |  |
| q2(f) |  |  | q2 | X |  |
| q3(f) |  | q3 | q2 | X |  |

- marca-se estados finais (q2 e q3) como úteis;

- há uma referência ao estado q3 no estado q0;

- marca-se q0 como útil;

- marca-se q2 e q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 |  |  | X |  |
| q2(f) |  |  | q2 | X | X |
| q3(f) |  | q3 | q2 | X | X |

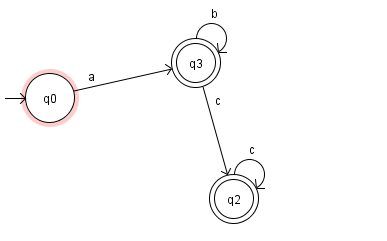
- seleciona-se q0, pois ele é o único útil não considerado;

- nenhum outro estado útil é considerado;

- marca-se q0 como considerado;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 |  |  | X | X |
| q2(f) |  |  | q2 | X | X |
| q3(f) |  | q3 | q2 | X | X |

- não há estados inúteis.



### Aplicação do algoritmo de minimização de autômatos:

Construção da tabela que relaciona os estados distintos, sendo que cada par não ordenado ocorre somente uma vez.

Marcação de estados trivialmente não equivalentes, ou seja, estados finais com não finais.

|  |  |  |
| --- | --- | --- |
|  | →q0 | q2(f) |
| q3(f) | X |  |
| q2(f) | X |  |

δ(q2,b)= não há δ

δ(q3,b)= q3

Como q3 e q2 consumindo o mesmo caractere resultam em valores diferentes, e um deles não resulta em nada (não há transição), podemos assumir que não são equivalentes.

Podemos marcar {q1,q2} na tabela.

|  |  |  |
| --- | --- | --- |
|  | →q0 | q2(f) |
| q3(f) | X | X |
| q2(f) | X |  |

O autômato já está minimizado.

## Exemplo 03

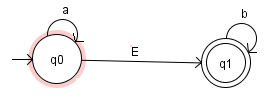
M3=({q0,q1},

{a,b},

{(q0,a)=q0 ; (q0,E)=q1 ; (q1,b)=q1 ;},

{q0},

{q1})

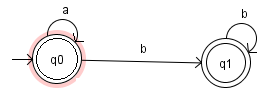


### Aplicação do algoritmo de eliminação de transições em vazio:

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | E |
| →q0 | q0 |  | q1 |
| q1(f) |  | q1 |  |

Como há transição em vazio para q1, deve-se copiar as transições de q1 para q0, e deve-se considerar q0 como final uma vez que q1 é final.

|  |  |  |
| --- | --- | --- |
| δ | a | b |
| →q0(f) | q0 | q1 |
| q1(f) |  | q1 |



- não há não-determinismo.

### Aplicação do algoritmo para eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q0 | q1 |  |  |
| q1(f) |  | q1 |  |  |

- marca q0 como acessível;

- q0 referencia q1;

- marca q1 como acessível;

- marca q0 como considerado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q0 | q1 | X | X |
| q1(f) |  | q1 | X |  |

- único estado acessível e não considerado é o q1;

- marca q1 como considerado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q0 | q1 | X | X |
| q1(f) |  | q1 | X | X |

- não há estados inacessíveis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Útil | Considerado |
| →q0(f) | q0 | q1 |  |  |
| q1(f) |  | q1 |  |  |

- marcar estados finais como úteis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Útil | Considerado |
| →q0(f) | q0 | q1 | X |  |
| q1(f) |  | q1 | X |  |

- q0 e q1 já marcados como úteis são marcados como considerados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Útil | Considerado |
| →q0(f) | q0 | q1 | X | X |
| q1(f) |  | q1 | X | X |

-não há estados inúteis.

### Aplicação do algoritmo de minimização de autômatos:

Construção da tabela que relaciona os estados distintos, sendo que cada par não ordenado ocorre somente uma vez.

Marcação de estados trivialmente não equivalentes, ou seja, estados finais com não finais.

|  |  |
| --- | --- |
|  | →q0(f) |
| q1(f) |  |

δ(q0,a)= q0

δ(q1,a)= não há δ

Como q1 e q0 consumindo o mesmo caractere resultam em valores diferentes, e um deles não resulta em nada (não há transição), podemos assumir que não são equivalentes.

Podemos marcar {q0,q1} na tabela.

|  |  |
| --- | --- |
|  | →q0(f) |
| q1(f) | X |

O autômato já está minimizado.

## Exemplo 04

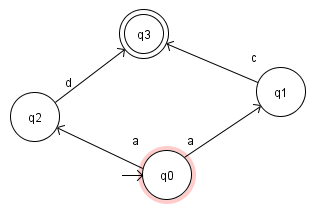
M5=({q0,q1,q2,q3},

{a,c,d},

{(q0,a)={q1,q2} ; (q1,c)=q3 ; (q2,d)=q3 ;},

{q0},

{q3})



|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | {q1,q2} |  |  |
| q1 |  | q3 |  |
| q2 |  |  | q3 |
| q3(f) |  |  |  |

### Algoritmo para eliminação de não-determinismo:

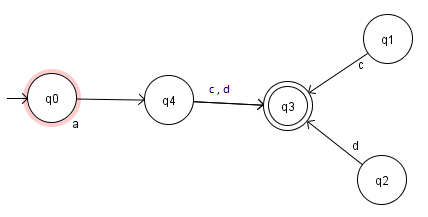
Cria-se um novo estado para cada elemento da tabela de transição que contenha mais de um elemento.

Cria-se o estado {q1,q2}, copiar as transições de q1 e q2 para o novo estado.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | {q1,q2} |  |  |
| q1 |  | q3 |  |
| q2 |  |  | q3 |
| q3(f) |  |  |  |
| {q1,q2} |  | q3 | q3 |

Vamos substitui {q1,q2} por q4.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | q4 |  |  |
| q1 |  | q3 |  |
| q2 |  |  | q3 |
| q3(f) |  |  |  |
| q4 |  | q3 | q3 |



### Aplicação do algoritmo para eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q4 |  |  |  |  |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q3 |  |  |
| q3(f) |  |  |  |  |  |
| q4 |  | q3 | q3 |  |  |

- marca q0 como acessível;

- q0 referencia q4;

- marca q4 como acessível;

- marca q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q4 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q3 |  |  |
| q3(f) |  |  |  |  |  |
| q4 |  | q3 | q3 | X |  |

- q4 referencia q3;

- marca q3 como acessível;

- marca q4 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q4 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q3 |  |  |
| q3(f) |  |  |  | X |  |
| q4 |  | q3 | q3 | X | X |

- q3 é o único acessível não considerado;

- marca q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q4 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q3 |  |  |
| q3(f) |  |  |  | X | X |
| q4 |  | q3 | q3 | X | X |

-q1 e q2 podem ser eliminados.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | q4 |  |  |
| q3(f) |  |  |  |
| q4 |  | q3 | q3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q4 |  |  |  |  |
| q3(f) |  |  |  |  |  |
| q4 |  | q3 | q3 |  |  |

- marcar estados finais como úteis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q4 |  |  |  |  |
| q3(f) |  |  |  | X |  |
| q4 |  | q3 | q3 |  |  |

- q4 faz referência à q3;

- marcar q4 como útil;

- marcar q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q4 |  |  |  |  |
| q3(f) |  |  |  | X | X |
| q4 |  | q3 | q3 | X |  |

- q0 faz referência à q4;

- marcar q0 como útil;

- marcar q4 como considerado.

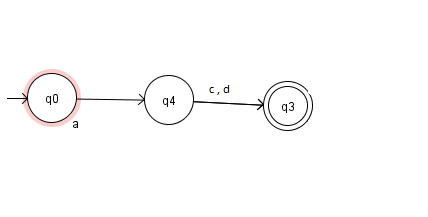
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q4 |  |  | X |  |
| q3(f) |  |  |  | X | X |
| q4 |  | q3 | q3 | X | X |

- o único útil não considerado é q0;

- marcar q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q4 |  |  | X | X |
| q3(f) |  |  |  | X | X |
| q4 |  | q3 | q3 | X | X |

- não há estados inúteis.



### Aplicação do algoritmo de minimização de autômatos:

Construção da tabela que relaciona os estados distintos, sendo que cada par não ordenado ocorre somente uma vez.

Marcação de estados trivialmente não equivalentes, ou seja, estados finais com não finais.

|  |  |  |
| --- | --- | --- |
|  | q0 | q3 |
| q4 |  |  |
| q3(f) | X |  |

δ(q0,a)= q4

δ(q4,a)= não há δ

Como q4 e q0 consumindo o mesmo caractere resultam em valores diferentes, e um deles não resulta em nada (não há transição), podemos assumir que não são equivalentes.

Podemos marcar {q0,q4} na tabela.

|  |  |  |
| --- | --- | --- |
|  | q0 | q3 |
| q4 | X |  |
| q3(f) | X |  |

δ(q3,c)= não há δ

δ(q4,c)= q3

Como q4 e q3 consumindo o mesmo caractere resultam em valores diferentes, e um deles não resulta em nada (não há transição), podemos assumir que não são equivalentes.

Podemos marcar {q3,q4} na tabela.

|  |  |  |
| --- | --- | --- |
|  | q0 | q3 |
| q4 | X | X |
| q3(f) | X |  |

O autômato já está minimizado.

## Exemplo 05

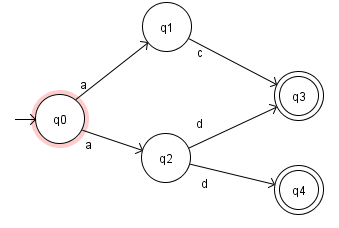
M6=({q0,q1,q2,q3,q4},

{a,c,d},

{(q0,a)={q1,q2} ; (q1,c)=q3 ; (q2,d)={q3,q4} ;},

{q0},

{q3,q4})



|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | {q1,q2} |  |  |
| q1 |  | q3 |  |
| q2 |  |  | {q3,q4} |
| q3(f) |  |  |  |
| q4(f) |  |  |  |

### Aplicando o algoritmo de eliminação de não determinismo:

Cria-se um novo estado para cada elemento da tabela de transição que contenha mais de um elemento.

Cria-se os estados {q1,q2} e {q3,q4}.

Copiar as transições de q1 e q2 para o novo estado {q1,q2}.

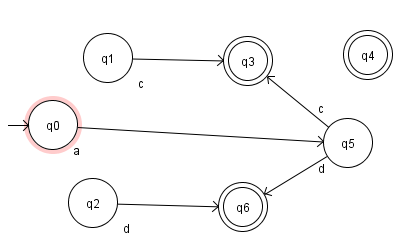
Copiar as transições de q3 e q4 para o novo estado {q3,q4}. (nenhuma)

Como q3 e q4 são finais, o novo estado, que contém q3 e q4, também é final.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | {q1,q2} |  |  |
| q1 |  | q3 |  |
| q2 |  |  | {q3,q4} |
| q3(f) |  |  |  |
| q4(f) |  |  |  |
| {q1,q2} |  | q3 | {q3,q4} |
| {q3,q4}(f) |  |  |  |

Vamos chamar {q1,q2} de q5, e {q3,q4} de q6.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | q5 |  |  |
| q1 |  | q3 |  |
| q2 |  |  | q6 |
| q3(f) |  |  |  |
| q4(f) |  |  |  |
| q5 |  | q3 | q6 |
| q6(f) |  |  |  |



### Aplicação do algoritmo para eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q5 |  |  |  |  |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q6 |  |  |
| q3(f) |  |  |  |  |  |
| q4(f) |  |  |  |  |  |
| q5 |  | q3 | q6 |  |  |
| q6(f) |  |  |  |  |  |

- marca q0 como acessível;

- q0 referencia q5;

- marca q5 como acessível;

- marca q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q5 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q6 |  |  |
| q3(f) |  |  |  |  |  |
| q4(f) |  |  |  |  |  |
| q5 |  | q3 | q6 | X |  |
| q6(f) |  |  |  |  |  |

- q5 referencia q3 e q6;

Marca q3 e q6 como acessíveis;

- marca q5 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q5 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q6 |  |  |
| q3(f) |  |  |  | X |  |
| q4(f) |  |  |  |  |  |
| q5 |  | q3 | q6 | X | X |
| q6(f) |  |  |  | X |  |

- q3 não referencia a nenhum estado;

- marca q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q5 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q6 |  |  |
| q3(f) |  |  |  | X | X |
| q4(f) |  |  |  |  |  |
| q5 |  | q3 | q6 | X | X |
| q6(f) |  |  |  | X |  |

- q6 é o único estado acessível não considerado;

- marcar q6 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Acessível | Considerado |
| →q0 | q5 |  |  | X | X |
| q1 |  | q3 |  |  |  |
| q2 |  |  | q6 |  |  |
| q3(f) |  |  |  | X | X |
| q4(f) |  |  |  |  |  |
| q5 |  | q3 | q6 | X | X |
| q6(f) |  |  |  | X | X |

- os estados q1, q2 e q4 são incessíveis e podem ser eliminados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q5 |  |  |  |  |
| q3(f) |  |  |  |  |  |
| q5 |  | q3 | q6 |  |  |
| q6(f) |  |  |  |  |  |

- marcar os estados finais como úteis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q5 |  |  |  |  |
| q3(f) |  |  |  | X |  |
| q5 |  | q3 | q6 |  |  |
| q6(f) |  |  |  | X |  |

- q5 referencia aos estados q3 e q6;

- marcar q5 como útil;

- marcar q3 e q6 como considerados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q5 |  |  |  |  |
| q3(f) |  |  |  | X | X |
| q5 |  | q3 | q6 | X |  |
| q6(f) |  |  |  | X | X |

- q0 referencia ao estado q5;

- marcar q0 como útil;

- marcar q5 como considerado.

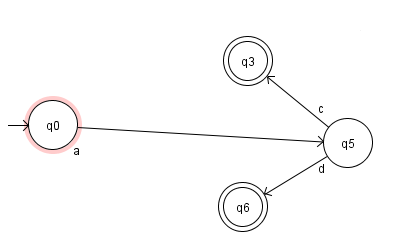
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q5 |  |  | X |  |
| q3(f) |  |  |  | X | X |
| q5 |  | q3 | q6 | X | X |
| q6(f) |  |  |  | X | X |

- o único útil não considerado é o estado q0;

- marcar q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | c | d | Útil | Considerado |
| →q0 | q5 |  |  | X | X |
| q3(f) |  |  |  | X | X |
| q5 |  | q3 | q6 | X | X |
| q6(f) |  |  |  | X | X |

- não há estados inúteis.



### Aplicação do algoritmo de minimização de autômatos:

Construção da tabela que relaciona os estados distintos, sendo que cada par não ordenado ocorre somente uma vez.

Marcação de estados trivialmente não equivalentes, ou seja, estados finais com não finais.

|  |  |  |  |
| --- | --- | --- | --- |
|  | q0 | q3(f) | q5 |
| q6(f) | X |  | X |
| q5 |  | X |  |
| q3(f) | X |  |  |

δ(q5,a)= não há δ

δ(q0,a)= q5

Como q5 e q0 consumindo o mesmo caractere resultam em valores diferentes, e um deles não resulta em nada (não há transição), podemos assumir que não são equivalentes.

Podemos marcar {q0,q5} na tabela.

|  |  |  |  |
| --- | --- | --- | --- |
|  | q0 | q3(f) | q5 |
| q6(f) | X |  | X |
| q5 | X | X |  |
| q3(f) | X |  |  |

δ(q6,a)= não há δ

δ(q3,a)= não há δ

δ(q6,c)= não há δ

δ(q3,c)= não há δ

δ(q6,d)= não há δ

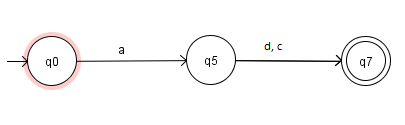
δ(q3,d)= não há δ

Os estados q3 e q6 são equivalentes, portanto podem se unir para a formação de um único estado.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | q5 |  |  |
| q5 |  | q3q6 | q3q6 |
| q3q6(f) |  |  |  |

Vamos renomear q3q6 para q7.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | c | d |
| →q0 | q5 |  |  |
| q5 |  | q7 | q7 |
| q7(f) |  |  |  |



O autômato está minimizado.

## Exemplo 06

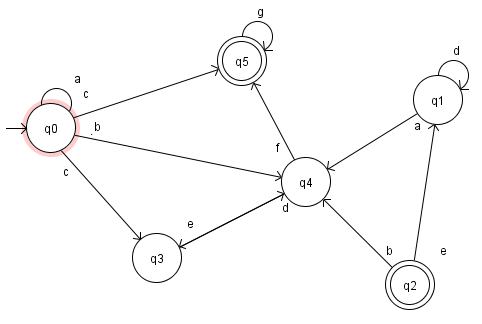
M6=({q0,q1,q2,q3,q4,q5},

{a,b,c,d,e,f,g},

{(q0,a)=q0 ; (q0,b)=q4 ; (q0,c)={q3,q5} ; (q1,d)=q1 ; (q1,a)=q4 ; (q2,b)=q4 ; (q2,e)=q1 ; (q3,e)=q4 ; (q4,d)=q3 ; (q4,f)=q5 ; (q5,g)=q5 ;},

{q0},

{q2,q5})



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g |
| →q0 | q0 | q4 | {q3,q5} |  |  |  |  |
| q1 | q4 |  |  | q1 |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |
| q3 |  |  |  |  | q4 |  |  |
| q4 |  |  |  | q3 |  | q5 |  |
| q5(f) |  |  |  |  |  |  | q5 |

### Aplicar o algoritmo de eliminação de não determinismo:

Cria-se um novo estado para cada elemento da tabela de transição que contenha mais de um elemento.

Cria-se os estados {q3,q5}.

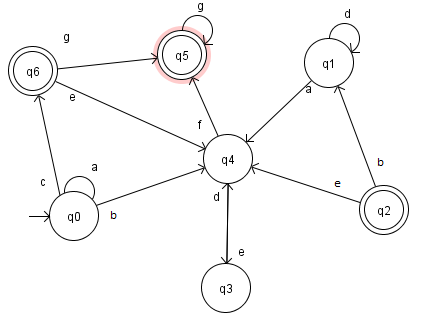
Copiar as transições de q3 e q5 para o novo estado {q3,q5}.

Como q5 é final, o novo estado, que contém q5, também é final.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g |
| →q0 | q0 | q4 | {q3,q5} |  |  |  |  |
| q1 | q4 |  |  | q1 |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |
| q3 |  |  |  |  | q4 |  |  |
| q4 |  |  |  | q3 |  | q5 |  |
| q5(f) |  |  |  |  |  |  | q5 |
| {q3,q5}(f) |  |  |  |  | q4 |  | q5 |

Trocaremos o nome do novo estado de {q3,q5} para q6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g |
| →q0 | q0 | q4 | q6 |  |  |  |  |
| q1 | q4 |  |  | q1 |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |
| q3 |  |  |  |  | q4 |  |  |
| q4 |  |  |  | q3 |  | q5 |  |
| q5(f) |  |  |  |  |  |  | q5 |
| q6(f) |  |  |  |  | q4 |  | q5 |



### Aplicação do algoritmo para eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Acessível | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  |  |  |
| q1 | q4 |  |  | q1 |  |  |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  |  |  |
| q4 |  |  |  | q3 |  | q5 |  |  |  |
| q5(f) |  |  |  |  |  |  | q5 |  |  |
| q6(f) |  |  |  |  | q4 |  | q5 |  |  |

- marca q0 como acessível;

- q0 referencia q0, q4 e q6;

- marca q4 e q6 como acessível;

- marca q0 como considerado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Acessível | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  | X | X |
| q1 | q4 |  |  | q1 |  |  |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  |  |  |
| q4 |  |  |  | q3 |  | q5 |  | X |  |
| q5(f) |  |  |  |  |  |  | q5 |  |  |
| q6(f) |  |  |  |  | q4 |  | q5 | X |  |

- q4 referencia q5 e q3;

- marca q3 e q5 como acessíveis;

- marca q4 como considerado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Acessível | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  | X | X |
| q1 | q4 |  |  | q1 |  |  |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  | X |  |
| q4 |  |  |  | q3 |  | q5 |  | X | X |
| q5(f) |  |  |  |  |  |  | q5 | X |  |
| q6(f) |  |  |  |  | q4 |  | q5 | X |  |

- q3 referencia q4(já marcado);

- marca q3 como considerado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Acessível | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  | X | X |
| q1 | q4 |  |  | q1 |  |  |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  | X | X |
| q4 |  |  |  | q3 |  | q5 |  | X | X |
| q5(f) |  |  |  |  |  |  | q5 | X |  |
| q6(f) |  |  |  |  | q4 |  | q5 | X |  |

- q5 referencia a ele próprio;

- marca q5 como considerado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Acessível | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  | X | X |
| q1 | q4 |  |  | q1 |  |  |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  | X | X |
| q4 |  |  |  | q3 |  | q5 |  | X | X |
| q5(f) |  |  |  |  |  |  | q5 | X | X |
| q6(f) |  |  |  |  | q4 |  | q5 | X |  |

- q6 referencia a q4 e q5 (já marcados);

- q6 é marcado como considerado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Acessível | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  | X | X |
| q1 | q4 |  |  | q1 |  |  |  |  |  |
| q2 (f) |  | q4 |  |  | q1 |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  | X | X |
| q4 |  |  |  | q3 |  | q5 |  | X | X |
| q5(f) |  |  |  |  |  |  | q5 | X | X |
| q6(f) |  |  |  |  | q4 |  | q5 | X | X |

Os estados q1 e q2 não são acessíveis e podem ser eliminados.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Útil | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  |  |  |
| q4 |  |  |  | q3 |  | q5 |  |  |  |
| q5(f) |  |  |  |  |  |  | q5 |  |  |
| q6(f) |  |  |  |  | q4 |  | q5 |  |  |

- marcam-se os estados finais como úteis.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Útil | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  |  |  |
| q4 |  |  |  | q3 |  | q5 |  |  |  |
| q5(f) |  |  |  |  |  |  | q5 | X |  |
| q6(f) |  |  |  |  | q4 |  | q5 | X |  |

- q4 faz referência a q5;

- marca-se q4 como útil;

- marcam-se q5 e q6 como considerados.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Útil | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  |  |  |
| q4 |  |  |  | q3 |  | q5 |  | X |  |
| q5(f) |  |  |  |  |  |  | q5 | X | X |
| q6(f) |  |  |  |  | q4 |  | q5 | X | X |

- q3 faz referência a q4;

- marca-se q3 como útil;

- marca q4 como considerado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Útil | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  |  |  |
| q3 |  |  |  |  | q4 |  |  | X |  |
| q4 |  |  |  | q3 |  | q5 |  | X | X |
| q5(f) |  |  |  |  |  |  | q5 | X | X |
| q6(f) |  |  |  |  | q4 |  | q5 | X | X |

- q0 referencia a q4 e q6;

- marca q0 como útil;

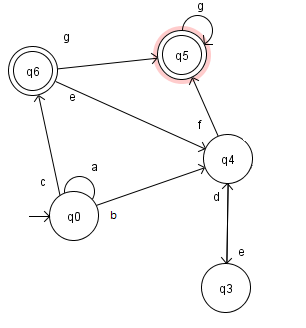
- marca q3 como considerado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Útil | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  | X |  |
| q3 |  |  |  |  | q4 |  |  | X | X |
| q4 |  |  |  | q3 |  | q5 |  | X | X |
| q5(f) |  |  |  |  |  |  | q5 | X | X |
| q6(f) |  |  |  |  | q4 |  | q5 | X | X |

- como q0 é o único útil não considerado, marca-o como considerado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | f | g | Útil | Considerado |
| →q0 | q0 | q4 | q6 |  |  |  |  | X | X |
| q3 |  |  |  |  | q4 |  |  | X | X |
| q4 |  |  |  | q3 |  | q5 |  | X | X |
| q5(f) |  |  |  |  |  |  | q5 | X | X |
| q6(f) |  |  |  |  | q4 |  | q5 | X | X |

Não há estados inúteis.



### Aplicando o algoritmo de minimização:

Construção da tabela que relaciona os estados distintos, sendo que cada par não ordenado ocorre somente uma vez.

Marcação de estados trivialmente não equivalentes, ou seja, estados finais com não finais.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | q0 | q3 | q4 | q5(f) |
| q6(f) | X | X | X |  |
| q5(f) | X | X | X |  |
| q4 |  |  |  |  |
| q3 |  |  |  |  |

δ(q0,a)= q0

δ(q4,a)= não há δ

...

q0 e q4 não são equivalentes, portanto marca-se o par.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | q0 | q3 | q4 | q5(f) |
| q6(f) | X | X | X |  |
| q5(f) | X | X | X |  |
| q4 | X |  |  |  |
| q3 |  |  |  |  |

δ(q0,a)= q0

δ(q3,a)= não há δ

...

q0 e q3 não são equivalentes, portanto marca-se o par.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | q0 | q3 | q4 | q5(f) |
| q6(f) | X | X | X |  |
| q5(f) | X | X | X |  |
| q4 | X |  |  |  |
| q3 | X |  |  |  |

δ(q4,d)= q3

δ(q3,d)= não há δ

...

q4 e q3 não são equivalentes, portanto marca-se o par.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | q0 | q3 | q4 | q5(f) |
| q6(f) | X | X | X |  |
| q5(f) | X | X | X |  |
| q4 | X | X |  |  |
| q3 | X |  |  |  |

## Exemplo 07

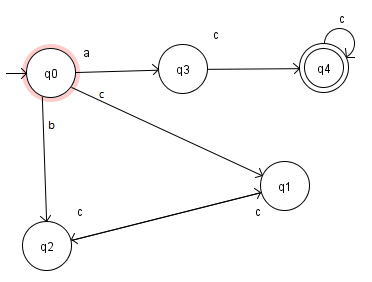
M7=({q0,q1,q2,q3,q4},

{a,b,c},

{(q0,a)=q3 ; (q0,b)=q2 ; (q0,c)=q1 ; (q1,c)=q2 ; (q2,c)=q1 ; (q3,c)=q4 ; (q4,c)=q4 ;},

{q0},

{q4})



|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q3 | q2 | q1 |
| q1 |  |  | q2 |
| q2 |  |  | q1 |
| q3 |  |  | q4 |
| q4(f) |  |  | q4 |

### Aplicando o algoritmo de eliminação de estados inacessíveis e inválidos:

- marca-se q0 como acessível, pois é o estado inicial.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 | q1 | X |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  |  | q4 |  |  |
| q4(f) |  |  | q4 |  |  |

- q0 faz referencia a q1, q2 e q3;

- marca q1, q2 e q3 como acessíveis;

- marca q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 | q1 | X | X |
| q1 |  |  | q2 | X |  |
| q2 |  |  | q1 | X |  |
| q3 |  |  | q4 | X |  |
| q4(f) |  |  | q4 |  |  |

- q3 referencia q4;

- marca q4 como acessível;

- marca q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 | q1 | X | X |
| q1 |  |  | q2 | X |  |
| q2 |  |  | q1 | X |  |
| q3 |  |  | q4 | X | X |
| q4(f) |  |  | q4 | X |  |

- todos os estados estão marcados como acessíveis;

- marca os estados acessíveis e não considerados como considerados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 | q1 | X | X |
| q1 |  |  | q2 | X | X |
| q2 |  |  | q1 | X | X |
| q3 |  |  | q4 | X | X |
| q4(f) |  |  | q4 | X | X |

Não há estados inacessíveis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 |  |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  |  | q4 |  |  |
| q4(f) |  |  | q4 |  |  |

- marca estados finais como úteis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 |  |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  |  | q4 |  |  |
| q4(f) |  |  | q4 | X |  |

- q3 referencia q4;

- marca q3 como útil;

- marca q4 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 |  |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  |  | q4 | X |  |
| q4(f) |  |  | q4 | X | X |

- q0 referenci q3;

- marca q0 como útil;

- marca q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 | X |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  |  | q4 | X | X |
| q4(f) |  |  | q4 | X | X |

- nenhum outro estado referencia q0;

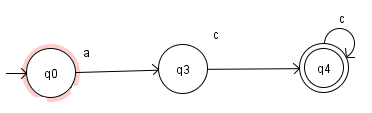
- q0 é o único estado útil não considerado;

- marca q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 | X | X |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  |  | q4 | X | X |
| q4(f) |  |  | q4 | X | X |

- os estados q1 e q2 podem ser eliminados.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q3 |  |  |
| q3 |  |  | q4 |
| q4(f) |  |  | q4 |



## Exemplo 08

M8=({q0,q1,q2,q3,q4,q5},

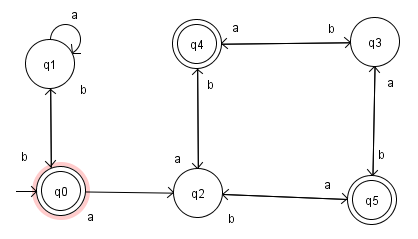
{a,b},

{(q0,a)=q2 ; (q0,b)=q1 ; (q1,a)=q1 ; (q1,b)=q0 ; (q2,a)=q4 ; (q2,b)=q5 ; (q3,a)=q5 ; (q3,b)=q4 ; (q4,a)=q3 ;

(q4,b)=q2 ; (q5,a)=q2 ; (q5,b)=q3 ;},

{q0},

{q4,q0,q5})



|  |  |  |
| --- | --- | --- |
| δ | a | b |
| →q0(f) | q2 | q1 |
| q1 | q1 | q0 |
| q2 | q4 | q5 |
| q3 | q5 | q4 |
| q4(f) | q3 | q2 |
| q5(f) | q2 | q3 |

### Aplicação do algoritmo de eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q2 | q1 |  |  |
| q1 | q1 | q0 |  |  |
| q2 | q4 | q5 |  |  |
| q3 | q5 | q4 |  |  |
| q4(f) | q3 | q2 |  |  |
| q5(f) | q2 | q3 |  |  |

- marcar estado inicial como acessível.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q2 | q1 | X |  |
| q1 | q1 | q0 |  |  |
| q2 | q4 | q5 |  |  |
| q3 | q5 | q4 |  |  |
| q4(f) | q3 | q2 |  |  |
| q5(f) | q2 | q3 |  |  |

- q0 referencia q2 e q1;

- marca q2 e q1 como acessíveis;

- marca q0 como considerado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q2 | q1 | X | X |
| q1 | q1 | q0 | X |  |
| q2 | q4 | q5 | X |  |
| q3 | q5 | q4 |  |  |
| q4(f) | q3 | q2 |  |  |
| q5(f) | q2 | q3 |  |  |

- q2 referencia q4 e q5;

- marca q4 e q5 como acessíveis;

- marca q2 como considerado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q2 | q1 | X | X |
| q1 | q1 | q0 | X |  |
| q2 | q4 | q5 | X | X |
| q3 | q5 | q4 |  |  |
| q4(f) | q3 | q2 | X |  |
| q5(f) | q2 | q3 | X |  |

- q4 referencia q3;

- marca q3 como acessível;

- marca q4 como considerado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q2 | q1 | X | X |
| q1 | q1 | q0 | X |  |
| q2 | q4 | q5 | X | X |
| q3 | q5 | q4 | X |  |
| q4(f) | q3 | q2 | X | X |
| q5(f) | q2 | q3 | X |  |

- todos os estados estão marcados como acessíveis, marcá-los como considerados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Acessível | Considerado |
| →q0(f) | q2 | q1 | X | X |
| q1 | q1 | q0 | X | X |
| q2 | q4 | q5 | X | X |
| q3 | q5 | q4 | X | X |
| q4(f) | q3 | q2 | X | X |
| q5(f) | q2 | q3 | X | X |

Não há estados inacessíveis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Útil | Considerado |
| →q0(f) | q2 | q1 |  |  |
| q1 | q1 | q0 |  |  |
| q2 | q4 | q5 |  |  |
| q3 | q5 | q4 |  |  |
| q4(f) | q3 | q2 |  |  |
| q5(f) | q2 | q3 |  |  |

- marca os estados finais como úteis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Útil | Considerado |
| →q0(f) | q2 | q1 | X |  |
| q1 | q1 | q0 |  |  |
| q2 | q4 | q5 |  |  |
| q3 | q5 | q4 |  |  |
| q4(f) | q3 | q2 | X |  |
| q5(f) | q2 | q3 | X |  |

- q5 referencia q2 e q3;

- marcar q2 e q3 como úteis;

- marcar q5 como considerado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Útil | Considerado |
| →q0(f) | q2 | q1 | X |  |
| q1 | q1 | q0 |  |  |
| q2 | q4 | q5 | X |  |
| q3 | q5 | q4 | X |  |
| q4(f) | q3 | q2 | X |  |
| q5(f) | q2 | q3 | X | X |

- q0 referencia q1;

- marca q1 como útil;

Marca q0 como considerado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Útil | Considerado |
| →q0(f) | q2 | q1 | X | X |
| q1 | q1 | q0 | X |  |
| q2 | q4 | q5 | X |  |
| q3 | q5 | q4 | X |  |
| q4(f) | q3 | q2 | X |  |
| q5(f) | q2 | q3 | X | X |

- todos estados estão marcados como úteis, então podem ser considerados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | b | Útil | Considerado |
| →q0(f) | q2 | q1 | X | X |
| q1 | q1 | q0 | X | X |
| q2 | q4 | q5 | X | X |
| q3 | q5 | q4 | X | X |
| q4(f) | q3 | q2 | X | X |
| q5(f) | q2 | q3 | X | X |

Não há estados inúteis.

### Aplicando o algoritmo de minimização:

Construção da tabela que relaciona os estados distintos, sendo que cada par não ordenado ocorre somente uma vez.

Marcação de estados trivialmente não equivalentes, ou seja, estados finais com não finais.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | q0(f) | q1 | q2 | q3 | q4(f) |
| q5(f) |  | X | X | X |  |
| q4(f) |  | X | X | X |  |
| q3 | X |  |  |  |  |
| q2 | X |  |  |  |  |
| q1 | X |  |  |  |  |

δ(q0,a)= q2

δ(q5,a)= q2

δ(q0,b)= q1

δ(q5,b)= q3 {q1,q3} não está marcado

δ(q2,a)= q4

δ(q3,a)= q5

δ(q2,b)= q5

δ(q3,b)= q4 {q4,q5} não está marcado

δ(q0,a)= q2

δ(q4,a)= q3 {q2,q3} não está marcado

δ(q0,b)= q1

δ(q4,b)= q2 {q1,q2} não está marcado

δ(q1,a)= q1

δ(q3,a)= q5

## Exemplo 09

M1=({q0,q1,q2,q3},

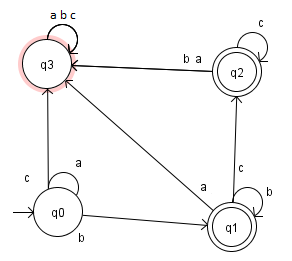
{a,b,c},

{(q0,a)=q0 ; (q0,b)=q1 ; (q0,c)=q3 ; (q1,a)=q3 ; (q1,b)=q1 ; (q1,c)=q2 ; (q2,a)=q3 ; (q2,b)=q3 ; (q2,c)=q2 ;

(q3,a)=q3 ; (q3,b)=q3 ; (q3,c)=q3 ; },

{q0},

{q2,q1})



|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q0 | q1 | q3 |
| q1(f) | q3 | q1 | q2 |
| q2(f) | q3 | q3 | q2 |
| q3 | q3 | q3 | q3 |

### Aplicação do algoritmo de eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 |  |  |
| q1(f) | q3 | q1 | q2 |  |  |
| q2(f) | q3 | q3 | q2 |  |  |
| q3 | q3 | q3 | q3 |  |  |

- marca estado inicial como acessível;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X |  |
| q1(f) | q3 | q1 | q2 |  |  |
| q2(f) | q3 | q3 | q2 |  |  |
| q3 | q3 | q3 | q3 |  |  |

- q0 referencia q1 e q3;

- marca q1 e q3 como acessíveis;

- marca q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1(f) | q3 | q1 | q2 | X |  |
| q2(f) | q3 | q3 | q2 |  |  |
| q3 | q3 | q3 | q3 | X |  |

- q1 referencia q2;

- marca q2 como acessível;

- marca q1 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1(f) | q3 | q1 | q2 | X | X |
| q2(f) | q3 | q3 | q2 | X |  |
| q3 | q3 | q3 | q3 | X |  |

- todos os estados são marcados como acessíveis, portanto podem ser marcados como considerados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1(f) | q3 | q1 | q2 | X | X |
| q2(f) | q3 | q3 | q2 | X | X |
| q3 | q3 | q3 | q3 | X | X |

Não há estados inacessíveis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q0 | q1 | q3 |  |  |
| q1(f) | q3 | q1 | q2 |  |  |
| q2(f) | q3 | q3 | q2 |  |  |
| q3 | q3 | q3 | q3 |  |  |

- marcam-se os estados finais como úteis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q0 | q1 | q3 |  |  |
| q1(f) | q3 | q1 | q2 | X |  |
| q2(f) | q3 | q3 | q2 | X |  |
| q3 | q3 | q3 | q3 |  |  |

- q0 referencia q1;

- marca q0 como útil;

- marca q1 como considerado.

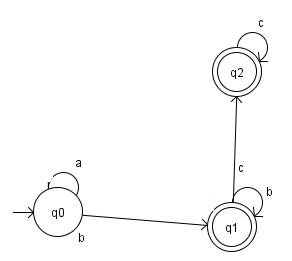
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q0 | q1 | q3 | X |  |
| q1(f) | q3 | q1 | q2 | X | X |
| q2(f) | q3 | q3 | q2 | X |  |
| q3 | q3 | q3 | q3 |  |  |

- nenhum outro estado tem a referencia de q3, portanto ele pode ser eliminado;

- q0 e q2 são marcados como considerados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q0 | q1 | q3 | X | X |
| q1(f) | q3 | q1 | q2 | X | X |
| q2(f) | q3 | q3 | q2 | X | X |
| q3 | q3 | q3 | q3 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q0 | q1 |  |
| q1(f) |  | q1 | q2 |
| q2(f) |  |  | q2 |



## Exemplo 10

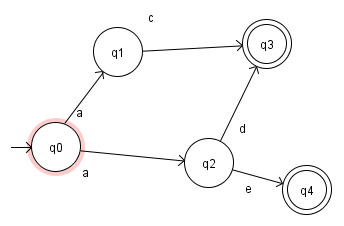
M1=({q0,q1,q2,q3,q4},

{a,c,d,e},

{(q0,a)={q1,q2} ; (q1,c)=q3 ; (q2,d)=q3 ; (q2,e)=q4 ;},

{q0},

{q3,q4})



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | c | d | e |
| →q0 | {q1,q2} |  |  |  |
| q1 |  | q3 |  |  |
| q2 |  |  | q3 | q4 |
| q3(f) |  |  |  |  |
| q4(f) |  |  |  |  |

### Aplicação do algoritmo de eliminação de não-determinismo:

Cria-se um novo estado para cada elemento da tabela de transição que contenha mais de um elemento.

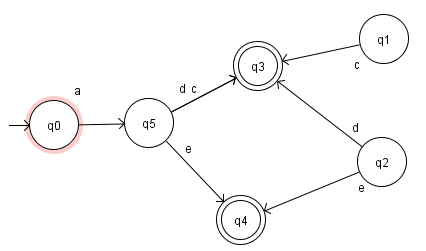
Cria-se o estado {q1,q2}.

Copiar as transições de q1 e q2 para o novo estado {q1,q2}.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | c | d | e |
| →q0 | {q1,q2} |  |  |  |
| q1 |  | q3 |  |  |
| q2 |  |  | q3 | q4 |
| q3(f) |  |  |  |  |
| q4(f) |  |  |  |  |
| {q1,q2} |  | q3 | q3 | q4 |

Vamos chamar o novo estado {q1,q2} de q5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | c | d | e |
| →q0 | q5 |  |  |  |
| q1 |  | q3 |  |  |
| q2 |  |  | q3 | q4 |
| q3(f) |  |  |  |  |
| q4(f) |  |  |  |  |
| q5 |  | q3 | q3 | q4 |



### Aplicando o algoritmo de eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  |  |  |
| q1 |  | q3 |  |  |  |  |
| q2 |  |  | q3 | q4 |  |  |
| q3(f) |  |  |  |  |  |  |
| q4(f) |  |  |  |  |  |  |
| q5 |  | q3 | q3 | q4 |  |  |

- marca-se o estado inicial como acessível.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  | X |  |
| q1 |  | q3 |  |  |  |  |
| q2 |  |  | q3 | q4 |  |  |
| q3(f) |  |  |  |  |  |  |
| q4(f) |  |  |  |  |  |  |
| q5 |  | q3 | q3 | q4 |  |  |

- q0 referencia q5;

- marca q5 como acessível;

- marca q0 como considerado.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  | X | X |
| q1 |  | q3 |  |  |  |  |
| q2 |  |  | q3 | q4 |  |  |
| q3(f) |  |  |  |  |  |  |
| q4(f) |  |  |  |  |  |  |
| q5 |  | q3 | q3 | q4 | X |  |

- q5 referencia q3 e q4;

- marca q3 e q4 como acessíveis;

- marca q5 como considerado.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  | X | X |
| q1 |  | q3 |  |  |  |  |
| q2 |  |  | q3 | q4 |  |  |
| q3(f) |  |  |  |  | X |  |
| q4(f) |  |  |  |  | X |  |
| q5 |  | q3 | q3 | q4 | X | X |

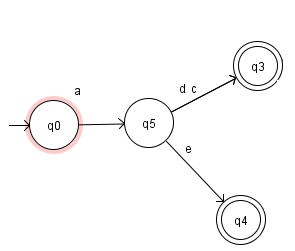
- q3 e q4 não fazem referencia aos estados q1 e q2;

- marca q3 e q4 como considerados.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  | X | X |
| q1 |  | q3 |  |  |  |  |
| q2 |  |  | q3 | q4 |  |  |
| q3(f) |  |  |  |  | X | X |
| q4(f) |  |  |  |  | X | X |
| q5 |  | q3 | q3 | q4 | X | X |

Os estados q1 e q2 podem ser eliminados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| δ | a | c | d | e |
| →q0 | q5 |  |  |  |
| q3(f) |  |  |  |  |
| q4(f) |  |  |  |  |
| q5 |  | q3 | q3 | q4 |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  |  |
| q3(f) |  |  |  |  |  |  |
| q4(f) |  |  |  |  |  |  |
| q5 |  | q3 | q3 | q4 |  |  |

- marca os estados finais como úteis.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  |  |
| q3(f) |  |  |  |  | X |  |
| q4(f) |  |  |  |  | X |  |
| q5 |  | q3 | q3 | q4 |  |  |

- q5 referencia q3 e q4;

- marca q5 como útil;

- marca q3 e q4 como considerados.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  |  |
| q3(f) |  |  |  |  | X | X |
| q4(f) |  |  |  |  | X | X |
| q5 |  | q3 | q3 | q4 | X |  |

- q0 referencia q5;

- marca q0 como útil;

Marca q5 como considerado.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  | X |  |
| q3(f) |  |  |  |  | X | X |
| q4(f) |  |  |  |  | X | X |
| q5 |  | q3 | q3 | q4 | X | X |

- q0 é o único estado útil não considerado;

- marca q0 como considerado.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| δ | a | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  | X | X |
| q3(f) |  |  |  |  | X | X |
| q4(f) |  |  |  |  | X | X |
| q5 |  | q3 | q3 | q4 | X | X |

Não há estados inúteis.

## Exemplo 11

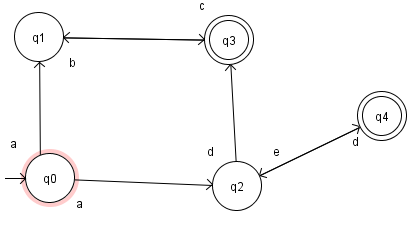
M2=({q0,q1,q2,q3,q4},

{a,b,c,d,e},

{(q0,a)={q1,q2} ; (q1,b)=q3 ; (q2,d)=q3 ; (q2,e)=q4 ; (q3,c)=q1 ; (q4,d)=q2 ;},

{q0},

{q3,q4})



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e |
| →q0 | {q1,q2} |  |  |  |  |
| q1 |  | q3 |  |  |  |
| q2 |  |  |  | q3 | q4 |
| q3(f) |  |  | q1 |  |  |
| q4(f) |  |  |  | q2 |  |

### Algoritmo para eliminação de não-determinismo:

Cria-se um novo estado para cada elemento da tabela de transição que contenha mais de um elemento.

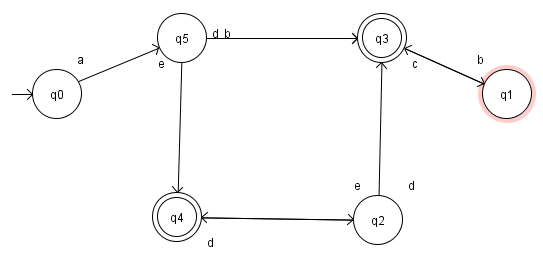
Cria-se os estados {q1,q2}.

Copiar as transições de q1 e q2 para o novo estado {q1,q2}.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e |
| →q0 | {q1,q2} |  |  |  |  |
| q1 |  | q3 |  |  |  |
| q2 |  |  |  | q3 | q4 |
| q3(f) |  |  | q1 |  |  |
| q4(f) |  |  |  | q2 |  |
| {q1,q2} |  | q3 |  | q3 | q4 |

Vamos chamar o novo estado de q5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e |
| →q0 | q5 |  |  |  |  |
| q1 |  | q3 |  |  |  |
| q2 |  |  |  | q3 | q4 |
| q3(f) |  |  | q1 |  |  |
| q4(f) |  |  |  | q2 |  |
| q5 |  | q3 |  | q3 | q4 |



### Aplicando o algoritmo de eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  |  |  |  |
| q1 |  | q3 |  |  |  |  |  |
| q2 |  |  |  | q3 | q4 |  |  |
| q3(f) |  |  | q1 |  |  |  |  |
| q4(f) |  |  |  | q2 |  |  |  |
| q5 |  | q3 |  | q3 | q4 |  |  |

- marca estado inicial como acessível.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  |  | X |  |
| q1 |  | q3 |  |  |  |  |  |
| q2 |  |  |  | q3 | q4 |  |  |
| q3(f) |  |  | q1 |  |  |  |  |
| q4(f) |  |  |  | q2 |  |  |  |
| q5 |  | q3 |  | q3 | q4 |  |  |

- q0 referencia q5;

- marca q5 como acessível;

- marca q0 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  |  | X | X |
| q1 |  | q3 |  |  |  |  |  |
| q2 |  |  |  | q3 | q4 |  |  |
| q3(f) |  |  | q1 |  |  |  |  |
| q4(f) |  |  |  | q2 |  |  |  |
| q5 |  | q3 |  | q3 | q4 | X |  |

- q5 referencia q3 e q4;

- marca q3 e q4 como acessíveis;

- marca q5 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  |  | X | X |
| q1 |  | q3 |  |  |  |  |  |
| q2 |  |  |  | q3 | q4 |  |  |
| q3(f) |  |  | q1 |  |  | X |  |
| q4(f) |  |  |  | q2 |  | X |  |
| q5 |  | q3 |  | q3 | q4 | X | X |

- q4 referencia q2;

- marca q2 como acessível;

- marca q4 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  |  | X | X |
| q1 |  | q3 |  |  |  |  |  |
| q2 |  |  |  | q3 | q4 | X |  |
| q3(f) |  |  | q1 |  |  | X |  |
| q4(f) |  |  |  | q2 |  | X | X |
| q5 |  | q3 |  | q3 | q4 | X | X |

- q3 referencia q1;

- marca q1 como acessível;

- marca q3 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  |  | X | X |
| q1 |  | q3 |  |  |  | X |  |
| q2 |  |  |  | q3 | q4 | X |  |
| q3(f) |  |  | q1 |  |  | X | X |
| q4(f) |  |  |  | q2 |  | X | X |
| q5 |  | q3 |  | q3 | q4 | X | X |

- todos os estados são acessíveis, portanto podem ser considerados.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q5 |  |  |  |  | X | X |
| q1 |  | q3 |  |  |  | X | X |
| q2 |  |  |  | q3 | q4 | X | X |
| q3(f) |  |  | q1 |  |  | X | X |
| q4(f) |  |  |  | q2 |  | X | X |
| q5 |  | q3 |  | q3 | q4 | X | X |

Não há estados inacessíveis.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  |  |  |
| q1 |  | q3 |  |  |  |  |  |
| q2 |  |  |  | q3 | q4 |  |  |
| q3(f) |  |  | q1 |  |  |  |  |
| q4(f) |  |  |  | q2 |  |  |  |
| q5 |  | q3 |  | q3 | q4 |  |  |

- marca os estados finais como úteis.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  |  |  |
| q1 |  | q3 |  |  |  |  |  |
| q2 |  |  |  | q3 | q4 |  |  |
| q3(f) |  |  | q1 |  |  | X |  |
| q4(f) |  |  |  | q2 |  | X |  |
| q5 |  | q3 |  | q3 | q4 |  |  |

- q3 referencia q1;

- marca q1 como útil;

- marca q3 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  |  |  |
| q1 |  | q3 |  |  |  | X |  |
| q2 |  |  |  | q3 | q4 |  |  |
| q3(f) |  |  | q1 |  |  | X | X |
| q4(f) |  |  |  | q2 |  | X |  |
| q5 |  | q3 |  | q3 | q4 |  |  |

- q4 referencia q2;

- marca q2 como útil;

- marca q4 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  |  |  |
| q1 |  | q3 |  |  |  | X |  |
| q2 |  |  |  | q3 | q4 | X |  |
| q3(f) |  |  | q1 |  |  | X | X |
| q4(f) |  |  |  | q2 |  | X | X |
| q5 |  | q3 |  | q3 | q4 |  |  |

- q5 referencia q3;

- marca q5 como útil;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  |  |  |
| q1 |  | q3 |  |  |  | X |  |
| q2 |  |  |  | q3 | q4 | X |  |
| q3(f) |  |  | q1 |  |  | X | X |
| q4(f) |  |  |  | q2 |  | X | X |
| q5 |  | q3 |  | q3 | q4 | X |  |

- q0 referencia q5;

- marca q0 como útil;

- marca q5 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  | X |  |
| q1 |  | q3 |  |  |  | X |  |
| q2 |  |  |  | q3 | q4 | X |  |
| q3(f) |  |  | q1 |  |  | X | X |
| q4(f) |  |  |  | q2 |  | X | X |
| q5 |  | q3 |  | q3 | q4 | X | X |

- todos os estados são marcados como úteis, portanto podem ser considerados.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q5 |  |  |  |  | X | X |
| q1 |  | q3 |  |  |  | X | X |
| q2 |  |  |  | q3 | q4 | X | X |
| q3(f) |  |  | q1 |  |  | X | X |
| q4(f) |  |  |  | q2 |  | X | X |
| q5 |  | q3 |  | q3 | q4 | X | X |

Não há estados inúteis.

## Exemplo 12

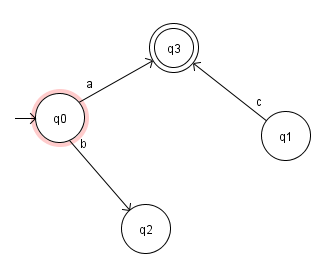
M2=({q0,q1,q2,q3},

{a,b,c},

{(q0,a)=q3 ; (q0,b)=q2 ; (q1,c)=q3 ;},

{q0},

{q3})



|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q3 | q2 |  |
| q1 |  |  | q3 |
| q2 |  |  |  |
| q3(f) |  |  |  |

### Aplicação do algoritmo de eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 |  | X |  |
| q1 |  |  | q3 |  |  |
| q2 |  |  |  |  |  |
| q3(f) |  |  |  |  |  |

- q0 referencia q3 e q2;

- marca q2 e q3 como acessíveis;

- marca q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 |  | X | X |
| q1 |  |  | q3 |  |  |
| q2 |  |  |  | X |  |
| q3(f) |  |  |  | X |  |

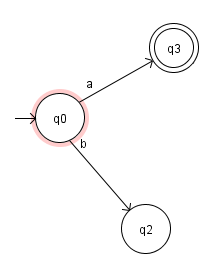
- nenhum estado faz referência a q1;

- marca q2 e q3 como considerados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 |  | X | X |
| q1 |  |  | q3 |  |  |
| q2 |  |  |  | X | X |
| q3(f) |  |  |  | X | X |

- o estado q1 pode ser eliminado.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q3 | q2 |  |
| q2 |  |  |  |
| q3(f) |  |  |  |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 |  |  |  |
| q2 |  |  |  |  |  |
| q3(f) |  |  |  | X |  |

- estados finais são marcados como úteis;

- q0 referencia q3;

-marca q0 como útil;

- marca q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 |  | X |  |
| q2 |  |  |  |  |  |
| q3(f) |  |  |  | X | X |

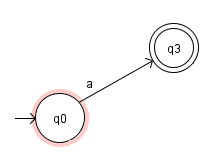
- q2 não referencia nenhum estado;

- marca q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 |  | X | X |
| q2 |  |  |  |  |  |
| q3(f) |  |  |  | X | X |

O estado q2 é inútil e pode ser elininado.

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q3 | q2 |  |
| q3(f) |  |  |  |



## Exemplo 13

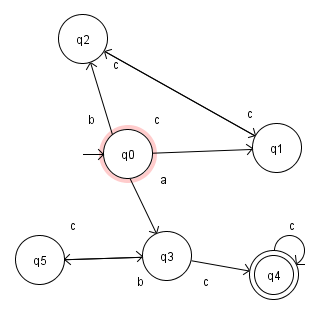
M3=({q0,q1,q2,q3,q4,q5},

{a,b,c},

{(q0,a)=q3 ; (q0,b)=q2 ; (q0,c)=q1 ; (q1,c)=q2 ; (q2,c)=q1 ; (q3,b)=q5 ; (q3,c)=q4 ; (q4,c)=q4 ; (q5,c)=q3 ;},

{q0},

{q4})



|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q3 | q2 | q1 |
| q1 |  |  | q2 |
| q2 |  |  | q1 |
| q3 |  | q5 | q4 |
| q4(f) |  |  | q4 |
| q5 |  |  | q3 |

### Aplicando o algoritmo de eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 | q1 | X |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  | q5 | q4 |  |  |
| q4(f) |  |  | q4 |  |  |
| q5 |  |  | q3 |  |  |

- q0 referencia q1, q2 e q3;

- marca q1, q2 e q3 como acessíveis;

- marca q0 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 | q1 | X | X |
| q1 |  |  | q2 | X |  |
| q2 |  |  | q1 | X |  |
| q3 |  | q5 | q4 | X |  |
| q4(f) |  |  | q4 |  |  |
| q5 |  |  | q3 |  |  |

- q3 referencia q4 e q5;

- marca q4 e q5 como acessíveis;

- marca q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 | q1 | X | X |
| q1 |  |  | q2 | X |  |
| q2 |  |  | q1 | X |  |
| q3 |  | q5 | q4 | X | X |
| q4(f) |  |  | q4 | X |  |
| q5 |  |  | q3 | X |  |

- todos os estados são acessíveis, portanto podem ser considerados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Acessível | Considerado |
| →q0 | q3 | q2 | q1 | X | X |
| q1 |  |  | q2 | X | X |
| q2 |  |  | q1 | X | X |
| q3 |  | q5 | q4 | X | X |
| q4(f) |  |  | q4 | X | X |
| q5 |  |  | q3 | X | X |

Não há estados inacessíveis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 |  |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  | q5 | q4 |  |  |
| q4(f) |  |  | q4 | X |  |
| q5 |  |  | q3 |  |  |

- q3 referencia q4;

- marca q3 como útil;

- marca q4 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 |  |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  | q5 | q4 | X |  |
| q4(f) |  |  | q4 | X | X |
| q5 |  |  | q3 |  |  |

- q0 referencia q3;

- marca q0 como útil;

- marca q3 como considerado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 | X |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  | q5 | q4 | X | X |
| q4(f) |  |  | q4 | X | X |
| q5 |  |  | q3 |  |  |

- q5 referencia q3;

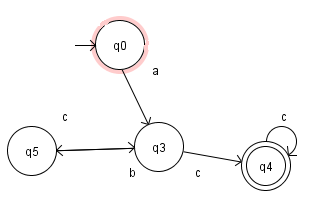
- marca q5 como útil;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 | X |  |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  | q5 | q4 | X | X |
| q4(f) |  |  | q4 | X | X |
| q5 |  |  | q3 | X |  |

- q1 e q2 não referenciam a nenhum estado útil, portanto são inúteis e podem ser eliminados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | Útil | Considerado |
| →q0 | q3 | q2 | q1 | X | X |
| q1 |  |  | q2 |  |  |
| q2 |  |  | q1 |  |  |
| q3 |  | q5 | q4 | X | X |
| q4(f) |  |  | q4 | X | X |
| q5 |  |  | q3 | X | X |

|  |  |  |  |
| --- | --- | --- | --- |
| δ | a | b | c |
| →q0 | q3 |  |  |
| q3 |  | q5 | q4 |
| q4(f) |  |  | q4 |
| q5 |  |  | q3 |



## Exemplo 14

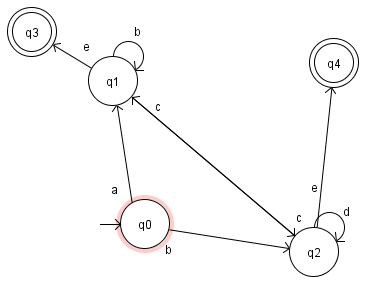
M3=({q0,q1,q2,q3,q4},

{a,b,c,d,e},

{(q0,a)=q1 ; (q0,b)=q2 ; (q1,b)=q1 ; (q1,c)=q2 ; (q1,e)=q3 ; (q2,c)=q1 ; (q2,d)=q2 ; (q2,e)=q4 ;},

{q0},

{q4,q3})



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e |
| →q0 | q1 | q2 |  |  |  |
| q1 |  | q1 | q2 |  | q3 |
| q2 |  |  | q1 | q2 | q4 |
| q3(f) |  |  |  |  |  |
| q4(f) |  |  |  |  |  |

### Aplicando o algoritmo de eliminação de estados inacessíveis e inúteis:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q1 | q2 |  |  |  | X |  |
| q1 |  | q1 | q2 |  | q3 |  |  |
| q2 |  |  | q1 | q2 | q4 |  |  |
| q3(f) |  |  |  |  |  |  |  |
| q4(f) |  |  |  |  |  |  |  |

- q0 referencia q1 e q2;

- marca q1 e q2 como acessíveis;

- marca q0 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q1 | q2 |  |  |  | X | X |
| q1 |  | q1 | q2 |  | q3 | X |  |
| q2 |  |  | q1 | q2 | q4 | X |  |
| q3(f) |  |  |  |  |  |  |  |
| q4(f) |  |  |  |  |  |  |  |

- q1 e q2 referenciam a q3 e q4;

- marca q3 e q4 como acessíveis;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q1 | q2 |  |  |  | X | X |
| q1 |  | q1 | q2 |  | q3 | X |  |
| q2 |  |  | q1 | q2 | q4 | X |  |
| q3(f) |  |  |  |  |  | X |  |
| q4(f) |  |  |  |  |  | X |  |

- todos os estados são acessíveis, portanto podem ser considerados.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Acessível | Considerado |
| →q0 | q1 | q2 |  |  |  | X | X |
| q1 |  | q1 | q2 |  | q3 | X | X |
| q2 |  |  | q1 | q2 | q4 | X | X |
| q3(f) |  |  |  |  |  | X | X |
| q4(f) |  |  |  |  |  | X | X |

- não há estados inacessíveis.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q1 | q2 |  |  |  |  |  |
| q1 |  | q1 | q2 |  | q3 |  |  |
| q2 |  |  | q1 | q2 | q4 |  |  |
| q3(f) |  |  |  |  |  | X |  |
| q4(f) |  |  |  |  |  | X |  |

- q2 referencia q4;

- marca q4 como útil;

- marca q2 como considerado.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q1 | q2 |  |  |  |  |  |
| q1 |  | q1 | q2 |  | q3 |  |  |
| q2 |  |  | q1 | q2 | q4 | X |  |
| q3(f) |  |  |  |  |  | X | X |
| q4(f) |  |  |  |  |  | X |  |

- q1 e q0 referenciam q2;

- marca q0 e q1 como úteis;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q1 | q2 |  |  |  | X |  |
| q1 |  | q1 | q2 |  | q3 | X |  |
| q2 |  |  | q1 | q2 | q4 | X | X |
| q3(f) |  |  |  |  |  | X | X |
| q4(f) |  |  |  |  |  | X |  |

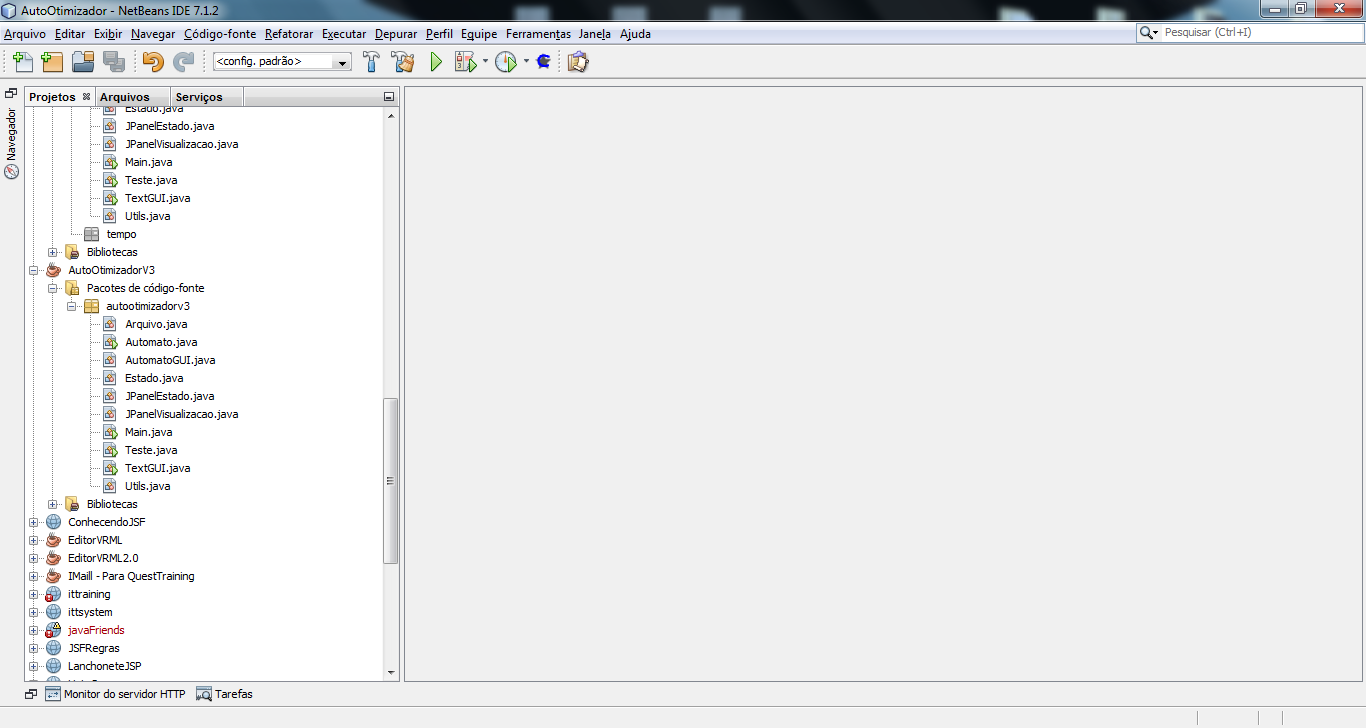
- todos os estados são úteis, portanto podem ser considerados.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| δ | a | b | c | d | e | Útil | Considerado |
| →q0 | q1 | q2 |  |  |  | X | X |
| q1 |  | q1 | q2 |  | q3 | X | X |
| q2 |  |  | q1 | q2 | q4 | X | X |
| q3(f) |  |  |  |  |  | X | X |
| q4(f) |  |  |  |  |  | X | X |

Não há estados inúteis.

# Código Fonte:

## Fontes:



## Classe main

package autootimizadorv3;

public class Main {

public static void main(String[] args) {

new Teste().setVisible(true);

}

}

## Classe Teste

package autootimizadorv3;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

public class Teste extends JFrame {

private JPanel contentPane;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

try {

Teste frame = new Teste();

frame.setVisible(true);

} catch (Exception e) {

}

}

});

}

private void importar() {

JFileChooser arquivo = new JFileChooser();

arquivo.showOpenDialog(null);

String path = arquivo.getSelectedFile().getAbsolutePath();

Scanner sc = new Arquivo().readFile(path);

String automato = "";

while (sc.hasNext()) {

automato += sc.next();

}

String iscompleta = "[m|M][1-9]=\\(\\{(q[0-9]\*|,)\*\\},"

+ "\\{([a-z]|,)\*\\},"

+ "\\{(\\(q[0-9]\*,([a-z]|E)\\)=\\{?(q[0-9]\*|,){1,}\\}?|\\s;\\s|;)\*\\}," + //{(q0,a) = q1 ; (q0,b) = q2 ; (q0,E) = {q2,q3} },

"\\{(q[0-9]\*)\*\\},"

+ "\\{(q[0-9]\*|,)\*\\}\\)";

if (!automato.matches(iscompleta)) {

JOptionPane.showMessageDialog(null,

"Automato de entrada não confere com o padrão.\nDeve seguir o modelo:\n "

+ "M1=(Estados,Alfabeto,Transições,Estado Inicial,Estados Finais)\n "

+ "M1=({q0,q1,q2,q3,...},\n"

+ "{a,b,...},\n"

+ "{(q0,a)=q1 ; (q0,E)={q2,q3,...} ...},\n"

+ "{q0},\n"

+ "{q2,...})\n(Tudo junto, sem quebras de linha.)",

"Erro", JOptionPane.ERROR\_MESSAGE);

return;

}

System.out.println("Automato: " + automato);

new AutomatoGUI(new Automato(automato)).setVisible(true);

}

/\*\*

\* Create the frame.

\*/

public Teste() {

setTitle("Optimizador de Automatos - Vers\u00E3o 3.1");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 601, 471);

setExtendedState(JFrame.MAXIMIZED\_BOTH);

JMenuBar menuBar = new JMenuBar();

setJMenuBar(menuBar);

JMenu mnArquivo = new JMenu("Arquivo");

menuBar.add(mnArquivo);

//JMenuItem mntmNovomodoGrfico = new JMenuItem("Novo (Modo Gr\u00E1fico)");

//mntmNovomodoGrfico.addActionListener(new ActionListener() {

// @Override

//public void actionPerformed(ActionEvent arg0) {

// new AutomatoGUI().setVisible(true);

//}

//});

//mnArquivo.add(mntmNovomodoGrfico);

JMenuItem mntmNovomodoTexto = new JMenuItem("Novo (Modo Texto)");

mntmNovomodoTexto.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

new TextGUI().setVisible(true);

}

});

mnArquivo.add(mntmNovomodoTexto);

JSeparator separator = new JSeparator();

mnArquivo.add(separator);

JMenuItem mntmImportarArquivo = new JMenuItem("Importar Arquivo");

mntmImportarArquivo.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

importar();

}

});

mnArquivo.add(mntmImportarArquivo);

contentPane = new JPanel();

contentPane.setBackground(new Color(100, 149, 237));

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

contentPane.setLayout(new BorderLayout(0, 0));

setContentPane(contentPane);

}

}

## Classe TesteGUI

package autootimizadorv3;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JTextArea;

import java.awt.Font;

public class TextGUI extends JFrame {

Automato a = new Automato();

private AutomatoGUI aa;

// Componentes Gr\_ficos

private JPanel contentPane;

private JTextField txtNome;

private JTextField txtEstados;

private JTextField txtAlfabeto;

private JTextField txtEstadosFinais;

private JTextField txtEstadoInicial;

private JTextArea txtTransicoes;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

TextGUI frame = new TextGUI();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public TextGUI() {

setTitle("Novo Automato - Modo Texto");

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

setBounds(100, 100, 524, 290);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblEstados = new JLabel("Estados:");

lblEstados.setBounds(10, 34, 96, 14);

contentPane.add(lblEstados);

JLabel lblAlfabeto = new JLabel("Alfabeto:");

lblAlfabeto.setBounds(10, 62, 96, 14);

contentPane.add(lblAlfabeto);

JLabel lblTransies = new JLabel("Transi\u00E7\u00F5es:");

lblTransies.setBounds(10, 148, 96, 14);

contentPane.add(lblTransies);

JLabel lblEstadosFinais = new JLabel("Estados Finais:");

lblEstadosFinais.setBounds(10, 92, 96, 14);

contentPane.add(lblEstadosFinais);

JLabel lblEstadosIniciais = new JLabel("Estado Inicial:");

lblEstadosIniciais.setBounds(10, 123, 96, 14);

contentPane.add(lblEstadosIniciais);

JLabel lblNomeDoAutomato = new JLabel("Nome do Automato:");

lblNomeDoAutomato.setBounds(10, 10, 129, 14);

contentPane.add(lblNomeDoAutomato);

txtNome = new JTextField();

txtNome.setBounds(141, 7, 36, 20);

contentPane.add(txtNome);

txtNome.setColumns(10);

txtEstados = new JTextField();

txtEstados.setColumns(10);

txtEstados.setBounds(116, 31, 209, 20);

contentPane.add(txtEstados);

JLabel lblSeparadosPorVirgula = new JLabel("Separados por virgula");

lblSeparadosPorVirgula.setBounds(334, 35, 164, 14);

contentPane.add(lblSeparadosPorVirgula);

txtAlfabeto = new JTextField();

txtAlfabeto.setColumns(10);

txtAlfabeto.setBounds(116, 59, 209, 20);

contentPane.add(txtAlfabeto);

JLabel label = new JLabel("Separados por virgula");

label.setBounds(334, 62, 164, 14);

contentPane.add(label);

txtEstadosFinais = new JTextField();

txtEstadosFinais.setColumns(10);

txtEstadosFinais.setBounds(116, 89, 209, 20);

contentPane.add(txtEstadosFinais);

JLabel label\_1 = new JLabel("Separados por virgula");

label\_1.setBounds(334, 92, 164, 14);

contentPane.add(label\_1);

txtEstadoInicial = new JTextField();

txtEstadoInicial.setColumns(10);

txtEstadoInicial.setBounds(116, 120, 30, 20);

contentPane.add(txtEstadoInicial);

JLabel lblModeloestadosimboloestado = new JLabel("Modelo: (estado,simbolo)=estado ; ou (estado,simbolo)={estado1,estado2,...} ;");

lblModeloestadosimboloestado.setBounds(65, 194, 399, 14);

contentPane.add(lblModeloestadosimboloestado);

JButton btnCriarAutomato = new JButton("Criar Automato");

btnCriarAutomato.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (!validar()) {

return;

}

capturarDados();

if (aa != null) {

if (aa.isVisible()) {

aa.dispose();

}

}

aa = new AutomatoGUI(a);

aa.setVisible(true);

}

});

btnCriarAutomato.setBounds(369, 225, 129, 23);

contentPane.add(btnCriarAutomato);

txtTransicoes = new JTextArea();

txtTransicoes.setFont(new Font("Arial", Font.PLAIN, 11));

txtTransicoes.setColumns(1);

txtTransicoes.setRows(5);

txtTransicoes.setBounds(10, 163, 488, 20);

contentPane.add(txtTransicoes);

//teste inicial

txtNome.setText("M2");

txtEstados.setText("q0,q1,q2,q3,q4");

txtEstadosFinais.setText("q2,q3");

txtAlfabeto.setText("a,b,c");

txtEstadoInicial.setText("q0");

txtTransicoes.setText("(q0,a)=q1 ; (q0,b)=q2 ; (q0,E)={q2,q3} ; (q2,c)={q3,q1} ; (q3,a)=q4;");

}

private boolean validar() {

if ("".equals(txtNome.getText())) {

JOptionPane.showMessageDialog(null, "Entre com o Nome do Automato:", "Aten\_\_o!", JOptionPane.ERROR\_MESSAGE);

return false;

}

if ("".equals(txtEstados.getText())) {

JOptionPane.showMessageDialog(null, "Entre com os Estados do Automato:", "Aten\_\_o!", JOptionPane.ERROR\_MESSAGE);

return false;

}

if ("".equals(txtAlfabeto.getText())) {

JOptionPane.showMessageDialog(null, "Entre com o Alfabeto do Automato:", "Aten\_\_o!", JOptionPane.ERROR\_MESSAGE);

return false;

}

if ("".equals(txtEstadosFinais.getText())) {

JOptionPane.showMessageDialog(null, "Entre com os Estados Finais do Automato:", "Aten\_\_o!", JOptionPane.ERROR\_MESSAGE);

return false;

}

if ("".equals(txtEstadoInicial.getText())) {

JOptionPane.showMessageDialog(null, "Entre com o Estado inicial do Automato:", "Aten\_\_o!", JOptionPane.ERROR\_MESSAGE);

return false;

}

if ("".equals(txtTransicoes.getText())) {

JOptionPane.showMessageDialog(null, "Entre com as transi\_\_es do Automato:", "Aten\_\_o!", JOptionPane.ERROR\_MESSAGE);

return false;

}

return true;

}

private void capturarDados() {

//cria\_\_o do nome

a.setNome(txtNome.getText());

//cria\_\_o do alfabeto

char[] alfabeto = txtAlfabeto.getText().replaceAll(",", "").toCharArray();

a.setAlfabeto(alfabeto);

//cria\_\_o dos estados

String[] estados = txtEstados.getText().split(",");

for (String s : estados) {

a.addEstado(s);

}

//setando estados finais

estados = txtEstadosFinais.getText().split(",");

for (String s : estados) {

a.turnFinal(s);

}

//setando estado inicial

Matcher m = Pattern.compile("q[0-9]").matcher(txtEstadoInicial.getText());

while (m.find()) {

a.setInicial(m.group());

}

//setando transi\_\_es

String transicoes = txtTransicoes.getText();

String patt = "\\(q[0-9],([a-z]|E)\\)=\\{?(q[0-9]|,){1,}\\}?"; //captura: (q0,b) = q2 ou (q0,E) = {q2,q3}

m = Pattern.compile(patt).matcher(transicoes);

String estadoOrigem = "", simbolo = "", transicao = "";

String[] estadosDestinos = estados;

Matcher subm;

while (m.find()) {

transicao = m.group(); //(qu,a)={qv,qn,qm,...}

subm = Pattern.compile("\\(q[0-9],").matcher(transicao); //captura o estado origem

while (subm.find()) {

estadoOrigem = subm.group().replaceAll("\\(|,", "");

}

subm = Pattern.compile("([a-z]|E)\\)").matcher(transicao); //captura o estado origem

while (subm.find()) {

simbolo = subm.group().replaceAll("\\)", "");

}

subm = Pattern.compile("=\\{?(q[0-9]|,){1,}\\}?").matcher(transicao); //captura o estado origem

while (subm.find()) {

estadosDestinos = subm.group().replaceAll("\\)|=|\\{|\\}", "").split(",");

}

a.newTransicao(estadoOrigem, simbolo, estadosDestinos);

}

System.out.println(a);

}

}

## Classe AutomatoGUI

package autootimizadorv3;

import java.awt.Color;

import java.awt.Scrollbar;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import javax.swing.border.LineBorder;

import javax.swing.border.TitledBorder;

import javax.swing.table.DefaultTableModel;

public final class AutomatoGUI extends JFrame {

public Automato a;

private Estado atual = new Estado("");

boolean criarInicial; // flag para verificar se \_ possivel criar um inicial, ou se j\_ existe um

//componentes gr\_ficos

private JPanel contentPane;

private JTable tblTrans;

private JComboBox cbAlfabeto;

private JCheckBox isfinal;

private JComboBox cbEstadosDestinos;

private JCheckBox cbisFinal;

private JButton btnExcluir;

private JButton btnNovo;

private JTable tblEstados;

private JPanel panel\_3;

private JButton btnNova;

private JButton btnRemover;

private JPanel panel\_4;

private JLabel lblModeloDoAutomato;

private JTextArea textArea;

private JPanelVisualizacao painelView;

private JLabel lblNewLabel;

public AutomatoGUI() {

this.a = new Automato();

initComponents();

refresh();

//caso o automato n\_o seja criado antes dessa tela, o variavel criarInicial d\_ permissoa a cria\_\_o de um estado inicial

criarInicial = true;

}

public AutomatoGUI(Automato a) {

this.a = a;

this.setExtendedState(JFrame.MAXIMIZED\_BOTH);

initComponents();

refresh();

//caso o automatoj\_ venha pr\_ estabelecido, n\_o permite a cria\_\_o de um estdo inicial

criarInicial = false;

refreshEstados();

}

protected void refreshEstados() {

//adiciona os estados na visualiza\_\_o

painelView.removeAll();

for (Estado e : a.getEstados()) {

//um objeto painelestado \_ criado, ele \_ um circulo que representa um estado, ao final, todos eles s\_o colocados em um painelVisualiza\_\_o

JPanelEstado p = new JPanelEstado(e, this); //cria a visualiza\_\_o gr\_fica

//colocam o visualizador no painel

painelView.add(p);

painelView.repaint();

}

}

//atualiza os dados gerias do automato

protected void refresh() {

//modelo de combobox para os estados existentes

DefaultComboBoxModel<Estado> modelcb = new DefaultComboBoxModel<>();

//modelo para a tabela de estados

Object[][] o = new Object[a.getEstados().size()][2];

int i = 0; //variavel para indicar a posi\_\_o da linha na matriz

for (Estado e : a.getEstados()) {

modelcb.addElement(e);

// se for incinicial aparece ->, se for final aparece (F), se for ambos, (F)->

o[i][0] = (e.isFinal() ? "(f)" : "") + "" + (e.equals(a.getInicial()) ? "->" : "");

o[i][1] = e.getNome();

i++;

}

cbEstadosDestinos.setModel(modelcb);

tblEstados.setModel(new DefaultTableModel(o, new String[]{"", ""}));

//modelo para combox de simbolos do alfabeto

DefaultComboBoxModel modelsim = new DefaultComboBoxModel();

modelsim.addElement('E');

for (char c : a.getAlfabeto()) {

modelsim.addElement(c);

}

cbAlfabeto.setModel(modelsim);

//atualiza o modelo texto do automato

textArea.setText(a.toString());

//atualiza o modelo gr\_fico do automato

painelView.repaint();

}

//atualiza os dados das transi\_\_es do estado selecionado

protected void refreshTrans() {

if (atual == null) {

return; //n\_o executa se o estado atual for nulo

}

Object[][] o = new Object[atual.transicoes.size()][2];

int i = 0;

for (char c : atual.getTransicoes().keySet()) {

o[i][0] = c;

o[i][1] = atual.getTransicoes().get(c);

i++;

}

tblTrans.setModel(new DefaultTableModel(o, new String[]{"Simbolo", "Estado(s) Destino(s)"}));

//atualiza o modelo texto do automato

textArea.setText(a.toString());

//atualiza o modelo gr\_fico do automato

painelView.repaint();

}

//chamado no evento do bot\_o novo

private void novoEstado() {

Estado novo = new Estado(isfinal.isSelected(), ("q" + a.getEstados().size()));

a.getEstados().add(novo);

if (criarInicial) { //seta como inicial se for inicial

a.setInicial(novo.getNome());

criarInicial = false; //n\_o deixa mais criar estados iniciais

}

JPanelEstado p = new JPanelEstado(novo, this); //cria a visualiza\_\_o gr\_fica

//colocam o visualizador no painel

painelView.add(p);

painelView.repaint();

refresh();

}

public void selecionaEstadoAtual(Estado e) {

if (e == null) {

return; //n\_o executa se o novo estado for nulo

}

atual = e; //atualiza o estado atual

lblNewLabel.setText("Detalhes de : " + atual.getNome()); //atualiza o seu nome

cbisFinal.setSelected(atual.isFinal()); //atualiza se \_ inicial

refreshTrans();

}

public Estado getEstadoAtual() {

return atual;

}

private void turnFinal() {

if (atual == null) {

return; //n\_o executa se o estado atual for nulo

}

atual.setFinal(cbisFinal.isSelected()); //torna final o estado atual

refresh();

}

private void excluirEstado() {

if (a.getInicial().equals(atual)) {

return; //n\_o deixa excluir se for inicial

}

a.deletarEstado(atual.getNome()); //deleta estado atual

refreshEstados();

refresh();

refreshTrans();

}

private void novaTransicao() {

if (atual == null) {

return; //n\_o executa se o estado atual for nulo

}

Character c = (Character) cbAlfabeto.getSelectedItem(); //pega o simbolo a ser usado

String estadoDestino = ((Estado) cbEstadosDestinos.getSelectedItem()).getNome(); //pega o nome do estado a ser usado

if(a.newTransicao(atual.getNome(), c + "", estadoDestino)){ //cria a nova transi\_\_o

refresh();

refreshTrans();

}

}

private void removerTransicao() {

if (atual == null) {

return; //n\_o executa se o estado atual for nulo

}

if (tblTrans.getSelectedRow() == -1) {

return; //se n\_o houve uma sele\_\_o efetiva, n\_o executa

} // deleta a transi\_\_o

atual.getTransicoes().remove((Character) atual.getTransicoes().keySet().toArray()[tblTrans.getSelectedRow()]);

// pega o simbolo-chave do array de chaves conforme o indice da linha selecionada da tabela

refreshTrans();

}

private void minimizarAutomato() {

a.minimizar(); //minimiza

//atualiza a tela

atual = a.getInicial();

refreshEstados();

refresh();

refreshTrans();

}

public void initComponents() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

setBounds(100, 100, 902, 597);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

panel\_4 = new JPanel();

panel\_4.setBorder(new TitledBorder(new LineBorder(new Color(0, 0, 0)), "Geral", TitledBorder.LEFT, TitledBorder.TOP, null, null));

panel\_4.setBounds(10, 11, 426, 236);

contentPane.add(panel\_4);

panel\_4.setLayout(null);

lblModeloDoAutomato = new JLabel("Defini\u00E7\u00E3o do Automato:");

lblModeloDoAutomato.setBounds(10, 47, 125, 14);

panel\_4.add(lblModeloDoAutomato);

JScrollPane scrollPane\_2 = new JScrollPane();

scrollPane\_2.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_ALWAYS);

scrollPane\_2.setBounds(10, 72, 406, 124);

panel\_4.add(scrollPane\_2);

textArea = new JTextArea();

textArea.setColumns(100);

scrollPane\_2.setColumnHeaderView(textArea);

textArea.setAutoscrolls(true);

textArea.setRows(9);

JButton btnMinimizar = new JButton("Minimizar");

btnMinimizar.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

minimizarAutomato();

}

});

btnMinimizar.setBounds(10, 207, 104, 23);

panel\_4.add(btnMinimizar);

JPanel panel = new JPanel();

panel.setBounds(10, 15, 250, 25);

panel\_4.add(panel);

panel.setBorder(null);

panel.setLayout(null);

JLabel lblNome = new JLabel("Criar estado:");

lblNome.setBounds(10, 5, 82, 14);

panel.add(lblNome);

lblNome.setHorizontalAlignment(SwingConstants.CENTER);

isfinal = new JCheckBox("Final");

isfinal.setBounds(98, 1, 56, 23);

panel.add(isfinal);

btnNovo = new JButton("Novo");

btnNovo.setBounds(160, 1, 73, 23);

panel.add(btnNovo);

btnNovo.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

novoEstado();

}

});

JPanel panel\_5 = new JPanel();

panel\_5.setBorder(new TitledBorder(new LineBorder(new Color(0, 0, 0)), "Visualiza\u00E7\u00E3o", TitledBorder.LEADING, TitledBorder.TOP, null, null));

panel\_5.setBounds(10, 258, 1310, 390);

contentPane.add(panel\_5);

panel\_5.setLayout(null);

painelView = new JPanelVisualizacao(this);

painelView.setBounds(10, 22, 1290, 357);

panel\_5.add(painelView);

JPanel panel\_2 = new JPanel();

panel\_2.setBounds(558, 11, 314, 251);

contentPane.add(panel\_2);

panel\_2.setLayout(null);

lblNewLabel = new JLabel("Detalhes de ");

lblNewLabel.setBounds(10, 11, 152, 14);

panel\_2.add(lblNewLabel);

cbisFinal = new JCheckBox("Final");

cbisFinal.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent arg0) {

turnFinal();

}

});

cbisFinal.setBounds(168, 7, 60, 23);

panel\_2.add(cbisFinal);

btnExcluir = new JButton("Excluir");

btnExcluir.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

excluirEstado();

}

});

btnExcluir.setBounds(231, 7, 73, 23);

panel\_2.add(btnExcluir);

panel\_3 = new JPanel();

panel\_3.setBorder(new TitledBorder(new LineBorder(new Color(0, 0, 0)), "Transi\u00E7\u00F5es", TitledBorder.LEADING, TitledBorder.TOP, null, null));

panel\_3.setBounds(10, 36, 294, 204);

panel\_2.add(panel\_3);

panel\_3.setLayout(null);

JLabel lblSimbolo = new JLabel("Simbolo:");

lblSimbolo.setBounds(10, 29, 96, 14);

panel\_3.add(lblSimbolo);

cbAlfabeto = new JComboBox();

cbAlfabeto.setBounds(122, 26, 49, 20);

panel\_3.add(cbAlfabeto);

JLabel lblEstadosDestinos = new JLabel("Estados Destinos:");

lblEstadosDestinos.setBounds(10, 57, 96, 14);

panel\_3.add(lblEstadosDestinos);

cbEstadosDestinos = new JComboBox();

cbEstadosDestinos.setBounds(122, 54, 49, 20);

panel\_3.add(cbEstadosDestinos);

JScrollPane scrollPane\_1 = new JScrollPane();

scrollPane\_1.setBounds(10, 82, 274, 111);

panel\_3.add(scrollPane\_1);

tblTrans = new JTable();

scrollPane\_1.setViewportView(tblTrans);

btnNova = new JButton("Nova");

btnNova.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

novaTransicao();

}

});

btnNova.setBounds(181, 25, 89, 23);

panel\_3.add(btnNova);

btnRemover = new JButton("Remover");

btnRemover.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

removerTransicao();

}

});

btnRemover.setBounds(181, 53, 89, 23);

panel\_3.add(btnRemover);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new TitledBorder(new LineBorder(new Color(0, 0, 0)), "Estados", TitledBorder.LEADING, TitledBorder.TOP, null, null));

panel\_1.setBounds(446, 11, 102, 236);

contentPane.add(panel\_1);

panel\_1.setLayout(null);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(10, 26, 82, 199);

panel\_1.add(scrollPane);

tblEstados = new JTable();

scrollPane.setViewportView(tblEstados);

tblEstados.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent arg0) {

selecionaEstadoAtual(a.getEstados().get(tblEstados.getSelectedRow()));

}

});

tblEstados.setAutoscrolls(true);

tblEstados.add(new Scrollbar());

}

}

## Classe Arquivo

package autootimizadorv3;

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Formatter;

import java.util.Scanner;

/\*\*

\*

\* @author Lucas

\*/

public class Arquivo {

private Formatter output;

private Scanner input;

public void openFile(String url){

try{

output = new Formatter(url);

}catch ( SecurityException securityException ){

System.out.println("Você não tem permissão de escrita em "+url);

System.exit( 1 );

}catch ( FileNotFoundException filesNotFoundException ){

System.out.println( "Arquivo não encontrado.");

System.exit( 1 );

}

}

public Scanner readFile(String url){

try{

return input = new Scanner( new File(url) );

}catch ( SecurityException securityException ){

System.out.println("Você não tem permissão de leitura em "+url);

System.exit( 1 );

}catch ( FileNotFoundException filesNotFoundException ){

System.out.println( "Arquivo não encontrado.");

System.exit( 1 );

}

return null;

}

public void closeFile() {

if (input != null) input.close();

if (output != null) output.close();

}

public Formatter getOutput() {

return output;

}

public void setOutput(Formatter output) {

this.output = output;

}

}

## Classe JPanelEstado

package autootimizadorv3;

import java.awt.\*;

import java.awt.geom.Ellipse2D;

import java.awt.geom.Line2D;

//representa o estado

public class JPanelEstado extends javax.swing.JPanel {

private boolean isInicial; //representa se o estado \_ inicial

private Estado estado; //o proprio estado

private int larguraFlecha; //variavel auxiliar para a largura flecha que muda de acordo com o estado ser incial ou n\_o.

private boolean clicado; // flag para indicar se o estado est\_ selecionado

private AutomatoGUI janelaPrincipal;

public JPanelEstado(Estado estado, AutomatoGUI janelaPrincipal) {

this.isInicial = janelaPrincipal.a.getInicial().equals(estado);

this.estado = estado;

this.janelaPrincipal = janelaPrincipal;

larguraFlecha = isInicial ? 20 : 0; //se o estado for inicial, a largura será 20, senão 0

setSize(new Dimension(50 + larguraFlecha, 50)); //a tamanho do painel \_ dado pelo tamanho do circulo do estado mais o da flecha

setBackground(new Color(0, 0, 0, 0));

initComponents();

}

private void initComponents() {

addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

formMousePressed(evt);

}

public void mouseReleased(java.awt.event.MouseEvent evt) {

formMouseReleased(evt);

}

});

addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {

public void mouseDragged(java.awt.event.MouseEvent evt) {

formMouseDragged(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);

this.setLayout(layout);

layout.setHorizontalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addGap(0, 400, Short.MAX\_VALUE));

layout.setVerticalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addGap(0, 300, Short.MAX\_VALUE));

}

//evento que move o estado ao ser arrastado pelo painel

private void formMouseDragged(java.awt.event.MouseEvent evt) {

if (getParent() != null) {

//seta a localiza\_\_o do estado enquanto \_ arrastado

int x = (int) getParent().getMousePosition().getX() - getWidth() / 2;

int y = (int) getParent().getMousePosition().getY() - getHeight() / 2;

//limite direito (x mais do que o painel)

if (getX() + getWidth() > getParent().getWidth()) {

x = getParent().getWidth() - getWidth() - 10;

}

//limite esquerdo x negativo

if (getX() < 0) {

x = 0;

}

//limite inferior (y mais do que o painel)

if (getY() + getHeight() > getParent().getHeight()) {

y = getParent().getHeight() - getHeight() - 2;

}

//limite superior y negativo

if (getY() < 0) {

y = 0;

}

estado.setXCentral(x);

estado.setYCentral(y);

setLocation(x, y);

// coordenadas para desenho das transi\_\_es enquanto ele \_ arrastado

if (isInicial) {

estado.setXCentral(x + (larguraFlecha / 2) + (getWidth() / 2));

estado.setYCentral(y + (getHeight() / 2));

} else {

estado.setXCentral(x + (getWidth() / 2));

estado.setYCentral(y + (getHeight() / 2));

}

// repinta o container pai, fazendo com as transi\_\_es sejam desenhadas

getParent().repaint();

}

}

//evento que seleciona o estado clicado

private void formMousePressed(java.awt.event.MouseEvent evt) {

janelaPrincipal.selecionaEstadoAtual(estado); //faz com que o estado clicado seja o atual na janela principal

clicado = true; //mostra que ele esta sendo clicado

repaint();

}

//evento que o estado \_ solto

private void formMouseReleased(java.awt.event.MouseEvent evt) {

clicado = false;

repaint();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

//se estiver clicado, a cor do fundo do automato se altera

g2d.setPaint(clicado ? new Color(220, 245, 255) : Color.WHITE);

//desenha o fundo

g2d.fill(new Ellipse2D.Double(larguraFlecha, 0,

this.getWidth() - larguraFlecha, getHeight()));

//se estiver clicado, a cor da borda do automato se altera

g2d.setPaint(clicado ? new Color(0, 102, 153) : Color.BLACK);

//desenha a borda

g2d.draw(new Ellipse2D.Double(larguraFlecha, 0,

this.getWidth() - (larguraFlecha + 1), getHeight() - 1));

setLocation(estado.getXCentral() - getWidth() / 2,

estado.getYCentral() - getHeight() / 2);

//se for incial, haver\_ um desenho, o da seta que represneta o estado incial

if (isInicial) {

g2d.draw(new Line2D.Double(0, getHeight() / 2,

larguraFlecha, getHeight() / 2));

g2d.draw(new Line2D.Double(larguraFlecha - 5, (getHeight() / 2) - 5,

larguraFlecha, getHeight() / 2));

g2d.draw(new Line2D.Double(larguraFlecha - 5, (getHeight() / 2) + 5,

larguraFlecha, getHeight() / 2));

}

//se for final, haverá um outro circulo concentrico representando o estado final

if (estado.isFinal()) {

g2d.draw(new Ellipse2D.Double(larguraFlecha + 5, 5,

this.getWidth() - (larguraFlecha + 11), getHeight() - 11));

}

//por ultimo o nome do estado

FontMetrics fm = g2d.getFontMetrics();

g2d.drawString(estado.getNome(),

(larguraFlecha + getWidth() - fm.stringWidth(estado.getNome()))/ 2,

(getHeight() / 2) + (fm.getHeight() / 3));

}

}

## Classe JPanelVisualizacao

package autootimizadorv3;

import java.awt.\*;

import java.awt.geom.Arc2D;

import java.awt.geom.Ellipse2D;

import java.awt.geom.Line2D;

import java.awt.geom.Rectangle2D;

//painel de visualiza\_\_o que guarda os estados EstOrigem suas transi\_\_o graficamente

public class JPanelVisualizacao extends javax.swing.JPanel {

private AutomatoGUI janelaPrincipal; //referencia a janela principal

public JPanelVisualizacao(AutomatoGUI janelaPrincipal) {

initComponents();

setBackground(Color.WHITE);

this.janelaPrincipal = janelaPrincipal;

}

private void initComponents() {

addMouseListener(new java.awt.event.MouseAdapter() {

@Override

public void mousePressed(java.awt.event.MouseEvent evt) {

formMousePressed(evt);

}

});

setLayout(null);

}

//evento quando o painel for clicado

private void formMousePressed(java.awt.event.MouseEvent evt) {

janelaPrincipal.selecionaEstadoAtual(null); //nenhum estado é selecionado

repaint();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g; //objeto grafico que desenha o painel

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON); // liga o antialiasing

g2d.setPaint(Color.BLACK); //cor preta como padrão para as linhas

FontMetrics fm = g2d.getFontMetrics(); //objeto usado para os caracteres escritos

g2d.draw(new Rectangle2D.Double(0, 0, getWidth() - 1, getHeight() - 1)); //desenha o fundo do painel

//desenha a seleção, quando um estado for clicado ele fica com um efeito atrás

if (janelaPrincipal.getEstadoAtual() != null) { //apenas é selecionado se não for nulo

g2d.setPaint(new Color(255, 204, 204)); //seleciona a cor da seleção

//desenha a elipse que da o efeito ao estado selecionado

double decr = janelaPrincipal.a.isInicial(janelaPrincipal.getEstadoAtual())? 19 : 29;

g2d.fill(new Ellipse2D.Double(

janelaPrincipal.getEstadoAtual().getXCentral() - decr,

janelaPrincipal.getEstadoAtual().getYCentral() - 29,

58, 58));

}

g2d.setPaint(Color.BLACK); //retorna a cor original

//Desenha as transições entre estados para cada estado do automato

int flagSimbol=0;

//flag simbol é usada para não permitir um caracter em cima de outro no desenho.

for (Estado EstOrigem : janelaPrincipal.a.getEstados()) {

//Dois tipos de transições possiveis:

//Entre dois estados diferentes : ele desenha uma reta

//Para si proprio: Ele desenha um arco

flagSimbol=0;

for (Character c : EstOrigem.transicoes.keySet()) {

for (Estado EstDestino : EstOrigem.transicoes.get(c)) {

// se estado origem EstOrigem EstDestino s\_o diferentes, tra\_a a reta

if (!EstDestino.equals(EstOrigem)) {

// -- desenha a linha de ligação (xO,yO)-(xD,yD)

g2d.draw(new Line2D.Double(EstOrigem.getXCentral(), EstOrigem.getYCentral(),

EstDestino.getXCentral(), EstDestino.getYCentral()));

// -- desenha o símbolo da ligação

//versão atual

//simbolo é posicionado de acordo com o mapa de posições - vide documentação

float xs = EstOrigem.getXCentral() , ys = EstOrigem.getYCentral();

double raiox = 30, raioy = 30;

xs = (float) ((EstOrigem.getXCentral()<EstDestino.getXCentral()) ? xs+raiox : xs-raiox);

ys = (float) ((EstOrigem.getYCentral()<EstDestino.getYCentral()) ? ys+raioy : ys-raioy);

if(EstOrigem.getYCentral()==EstDestino.getYCentral()) ys = EstOrigem.getYCentral() ;

if(EstOrigem.getXCentral()==EstDestino.getXCentral()) xs = EstOrigem.getXCentral() ;

g2d.drawString(c + "",xs+flagSimbol,ys);

//--Desenho das setas das retas

// gera a hipotenusa

double h = Utils.gerarHipotenusa(EstOrigem.getXCentral(), EstOrigem.getYCentral(),

EstDestino.getXCentral(), EstDestino.getYCentral());

// gera o grau relativo entre os estados, para rotacionar a ponta da flecha

double gr = Utils.obtemGrauRelativoJava(EstOrigem.getXCentral(), EstOrigem.getYCentral(),

EstDestino.getXCentral(), EstDestino.getYCentral());

// calcula o x e y do inicio da flecha

// sendo que h deve ser subtraido do raio do estado que no caso

// é 25, pois a flecha deve ser desenhada na borda do estado

double x = (h - 25) \* Math.cos(Math.toRadians(gr));

double y = (h - 25) \* Math.sin(Math.toRadians(gr));

// rotaciona a ponta da flecha

Graphics2D g2df = (Graphics2D) g2d.create(); // cria um novo Graphics a partir do original

g2df.translate(x + EstOrigem.getXCentral(), y + EstOrigem.getYCentral()); // faz a transla\_\_o para a coordenada que deve ser a origem

g2df.rotate(Math.toRadians(gr)); // rotaciona usando o grau relativo

// desenha a flecha

g2df.draw(new Line2D.Double(0, 0, -5, -5));

g2df.draw(new Line2D.Double(0, 0, -5, 5));

// libera o graphics, n\_o sendo necess\_rio voltar a transla\_\_o nem a rota\_\_o

g2df.dispose();

} else {

// caso contr\_rio, desenha arco

g2d.draw(new Ellipse2D.Double(EstOrigem.getXCentral(), EstOrigem.getYCentral() - 40, 30, 30));

// desenha a flecha

g2d.draw(new Line2D.Double(EstOrigem.getXCentral() + 21, EstOrigem.getYCentral() - 11, EstOrigem.getXCentral() + 30, EstOrigem.getYCentral() - 11));

g2d.draw(new Line2D.Double(EstOrigem.getXCentral() + 22, EstOrigem.getYCentral() - 11, EstOrigem.getXCentral() + 21, EstOrigem.getYCentral() - 20));

// desenha o s\_mbolo da liga\_\_o

g2d.drawString(c + "", EstOrigem.getXCentral() + 15 +flagSimbol, EstOrigem.getYCentral() - 45);

}

flagSimbol+=6;

}

}

}

}

}

## Classe Utils

package autootimizadorv3;

public class Utils {

public static double gerarHipotenusa(double x1, double y1, double x2, double y2 ) {

double x = Math.abs( x1 - x2 );

double y = Math.abs( y1 - y2 );

return Math.sqrt( Math.pow( x, 2 ) + Math.pow( y, 2 ) );

}

// para sistema carteziano normal

private static int detectarQuadrante(double x1, double y1, double x2, double y2 ) {

if ( x2 >= x1 && y2 >= y1 ) return 1;

if ( x2 < x1 && y2 > y1 ) return 2;

if ( x2 <= x1 && y2 <= y1 ) return 3;

return 4;

}

private static int gerarIncrementoAngulo(double x1, double y1,double x2, double y2 ) {

int q = detectarQuadrante( x1, y1, x2, y2 );

switch(q){

case 1: return 0;

case 2: return 90;

case 3: return 180;

default: return 270;

}

}

//inutil, ocorre um erro - vide explicação na documentação.

public static double obtemGrauRelativo(double x1, double y1,double x2, double y2 ) {

double x = Math.abs(x1-x2);

double y = Math.abs(y1-y2);

return gerarIncrementoAngulo( x1, y1, x2, y2 ) + Math.toDegrees( Math.atan2( y, x ) );

}

// adaptado, erro concertado aplicando uma adptação nos 2º e 4º quadrantes

public static double obtemGrauRelativoJava(double x1, double y1, double x2, double y2 ) {

double x = Math.abs(x1-x2);

double y = Math.abs(y1-y2);

double ang = Math.toDegrees( Math.atan2( y, x ) );

int incr = gerarIncrementoAngulo( x1, y1, x2, y2 );

if ( incr == 90 || incr == 270 ) ang = 90 - ang;

return incr + ang;

}

}

## Classe Estado

package autootimizadorv3;

import java.util.ArrayList;

import java.util.HashMap;

public class Estado {

private boolean isFinal;

private String nome;

private boolean consideracao; //usado para os algoritmos de elimina\_\_o de estados inuteis e inacessiveis

// q: (a => qo,q1,...),(b => qo,q1,...),(c => qo,q1,...)...

HashMap<Character,ArrayList<Estado>> transicoes = new HashMap<>();

private int xCentral,yCentral;

public Estado(String nome) {

super();

this.nome = nome;

isFinal = false;

consideracao = false;

}

public Estado(boolean isFinal, String nome) {

super();

this.nome = nome;

this.isFinal = isFinal;

consideracao = false;

}

public int getIndex(){

return Integer.parseInt(nome.replace("q",""));

}

@Override

public String toString(){

return nome;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((nome == null) ? 0 : nome.hashCode());

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (getClass() != obj.getClass())

return false;

Estado other = (Estado) obj;

if (nome == null) {

if (other.nome != null)

return false;

} else if (!nome.equals(other.nome))

return false;

return true;

}

public boolean isFinal() {

return isFinal;

}

public void setFinal(boolean isFinal) {

this.isFinal = isFinal;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public boolean isConsideracao() {

return consideracao;

}

public void setConsideracao(boolean consideracao) {

this.consideracao = consideracao;

}

public HashMap<Character, ArrayList<Estado>> getTransicoes() {

return transicoes;

}

public void setTransicoes(HashMap<Character, ArrayList<Estado>> transicoes) {

this.transicoes = transicoes;

}

public int getXCentral() {

return xCentral;

}

public void setXCentral(int xCentral) {

this.xCentral = xCentral;

}

public int getYCentral() {

return yCentral;

}

public void setYCentral(int yCentral) {

this.yCentral = yCentral;

}

}

## Classe Automato

package autootimizadorv3;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintStream;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JOptionPane;

public final class Automato {

private String nome;

private Estado inicial;

private char[] alfabeto;

private ArrayList<Estado> estados = new ArrayList<>();

public Automato() {

}

public Automato(String automato) {

Matcher m;

//\*\* estudo do alfabeto em modo texto \*\*//

m = Pattern.compile("\\{([a-z]|,)\*\\}").matcher(automato); //captura {a,b,c...}

String simbolos = m.find() ? m.group().replaceAll("\\{|\\}|,", "") : "E";

setAlfabeto(simbolos.toCharArray());

//\*\* estudo dos estados em modo texto \*\*//

m = Pattern.compile("[m|M][1-9]=").matcher(automato); //captura M1=

nome = m.find() ? m.group().replaceAll("\\{|\\}|,|=", "") : "M";

m = Pattern.compile("[m|M][1-9]=\\(\\{(q[0-9]\*|,)\*\\}").matcher(automato); //captura M1={q0,q1,q2,q3,...}

m = Pattern.compile("q[0-9]\*").matcher(m.find() ? m.group() : ""); //captura os estados

while (m.find()) {

addEstado(m.group());

}

//\*\* estudo das transições em modo texto \*\*//

//captura: (q0,b) = q2 ou (q0,E) = {q2,q3}

m = Pattern.compile("\\(q[0-9]\*,([a-z]|E)\\)=\\{?(q[0-9]\*|,){1,}\\}?").matcher(automato);

String estadoOrigem = "", simbolo = "", transicao;

String[] estadosDestinos = {""};

Matcher subm;

while (m.find()) {

transicao = m.group(); //(qu,a)={qv,qn,qm,...}

subm = Pattern.compile("\\(q[0-9]\*,").matcher(transicao); //captura o estado origem

while (subm.find()) {

estadoOrigem = subm.group().replaceAll("\\(|,", "");

}

subm = Pattern.compile("([a-z]|E)\\)").matcher(transicao); //captura o estado origem

while (subm.find()) {

simbolo = subm.group().replaceAll("\\)", "");

}

subm = Pattern.compile("=\\{?(q[0-9]\*|,){1,}\\}?").matcher(transicao); //captura o estado origem

while (subm.find()) {

estadosDestinos = subm.group().replaceAll("\\)|=|\\{|\\}", "").split(",");

}

newTransicao(estadoOrigem, simbolo, estadosDestinos);

}

//\*\* estudo dos estados finais em modo texto \*\*//

m = Pattern.compile("\\{(q[0-9]\*|,)\*\\}\\)").matcher(automato); //captura {q2,q3})

m = Pattern.compile("q[0-9]\*").matcher(m.find() ? m.group() : ""); //captura os estados finais

while (m.find()) {

turnFinal(m.group());

}

//\*\* estudo do estado inicial em modo texto \*\*//

m = Pattern.compile(",\\{q[0-9]\*\\},").matcher(automato); //captura {q0})

while (m.find()) {

setInicial(m.group().replaceAll("\\{|\\}|,", ""));

}

}

public void addEstado(String nome) {

Estado e = new Estado(nome);

if (estados.contains(e)) {

return; //verifica se o estado j\_ existe

}

estados.add(e); //adiciona novo estado

}

public boolean isInicial(Estado e) {

return inicial.equals(e);

}

public void turnFinal(String nome) {

estados.get(estados.indexOf(new Estado(nome))).setFinal(true);

}

public void setInicial(String nome) {

inicial = estados.get(estados.indexOf(new Estado(nome)));

}

public boolean newTransicao(String estadoOrigem, String simbolo, String... estadosDestinos) {

//verifica se existem os parametros

if ("".equals(estadoOrigem) || "".equals(simbolo) || estadosDestinos.length < 1) {

return false;

}

//verifica se o simbolo é vazio(E) ou pertençe ao alfabeto, caso contrario não adiciona a transição

boolean pertence = true;

char simbol = simbolo.charAt(0);

for (char a : alfabeto) {

if (simbol == a || simbol == 'E') {

pertence = true;

break;

}

}

if (!pertence) {

return false;

}

//captura do estado de origem

Estado e = estados.get(estados.indexOf(new Estado(estadoOrigem)));

if (e == null) {

return false;

}

//Caso já exista a transição com o simbolo,

//o(s) estado(s) e(são) adicionado(s) a lista

Estado ad;

int i;

if (e.transicoes.containsKey(simbol)) {

for (String est : estadosDestinos) {

i = estados.indexOf(new Estado(est));

if (i == -1) {

continue;

}

ad = estados.get(i);

if (e.transicoes.get(simbol).contains(ad)) {

continue;

}

e.transicoes.get(simbol).add(ad);

}

} else {

//caso não exista, uma lista de estados é criada e inserida a nova transição

ArrayList<Estado> listadestinos = new ArrayList<>();

for (String est : estadosDestinos) {

i = estados.indexOf(new Estado(est));

if (i == -1) {

continue;

}

ad = estados.get(i);

listadestinos.add(ad);

}

if (listadestinos.isEmpty()) {

return false;

}

e.transicoes.put(simbol, listadestinos);

}

return true;

}

public void eliminarTransicoesEmVazio() {

//percorre cada estado do automato

for (Estado q : estados) {

//se contem uma trasi\_\_o em vazio (caracter 'E')

if (q.transicoes.containsKey('E')) {

//capturo a lista de estados destinos dessa transi\_\_o

ArrayList<Estado> listadestinos = q.transicoes.get('E');

//para cada estado destino dessa transi\_\_o, eu vou copiar as transi\_\_es

for (Estado destino : listadestinos) {

for (char c : destino.transicoes.keySet()) //para cada estado da transi\_\_o em quest\_o,

//crio uma transi\_\_o nova para o estado q , mantenho o simbolo c, com o novo estado

{

for (Estado est : destino.transicoes.get(c)) {

newTransicao(q.getNome(), c + "", est.getNome());

}

}

}

q.transicoes.remove('E'); //ao final removo a transi\_\_o em vazio

}

}

}

private void eliminarNaoDeterminismo() {

ArrayList<Estado> transNovo = new ArrayList<>();

Estado Qnovo, q;

//percorre cada estado do automato

//OBS.: N\_o foi utilizado a itera\_\_o foreach devido a altera\_\_o da cole\_\_o

//que cria outro estado e gere um erro de concorrencia

for (int i = 0; i < estados.size(); i++) {

q = estados.get(i);

//para todas as transi\_\_es, capturo a lista de estados

for (char chave : q.transicoes.keySet()) {

transNovo = q.transicoes.get(chave);

//se o tamanho da lista de transi\_\_es for maior que 1 existe n\_o-determinismo

if (transNovo.size() > 1) {

Qnovo = new Estado("q" + estados.size()); //\_ criado um novo estado

estados.add(Qnovo); //esse estado pe adicionado a lista

//para cada estado da transi\_\_o em quest\_o,

//para cada estado destino, copiam-se suas transi\_\_es para o novo estado.

for (Estado destino : transNovo) {

if (destino.isFinal()) {

Qnovo.setFinal(true); //basta um dos estados serem finais para o novo estado tambem ser.

}

for (char c : destino.transicoes.keySet()) {

for (Estado est : destino.transicoes.get(c)) {

newTransicao(Qnovo.getNome(), c + "", est.getNome());

}

}

}

//ao final troco a referencia a todos os estados da lista para o novo estado.

transNovo.clear();

transNovo.add(Qnovo);

}

}

}

}

private void eliminarEstadosInacessiveis() {

elEInac(inicial); //fun\_\_o que percorre recursivamente o 'grafo' inciando pelo estad inicial

deletarEstadosInvalidos(); //ap\_s a identifica\_\_o dos estados inacessiveis: considerado=false

}

private void elEInac(Estado atual) {

if (atual.isConsideracao()) {

return;

}

atual.setConsideracao(true); //todo estado a ser analizado é acessivel, pois teve referencia de outro que chamou a função

System.out.println(" - " + atual + " é Acessivel");

ArrayList<Estado> lista;

//para cada transição, captura a lista de estados "que na teoria e pr\_tica ter\_ apenas um estado" (AFD)

for (char c : atual.transicoes.keySet()) {

lista = atual.transicoes.get(c);

for (Estado e : lista) {

elEInac(e); //cada estado dessa transição (um apenas, devido ao algoritmo anterior) será marcado e analizado

}

}

}

private void eliminarEstadosInuteis() {

for (Estado e : estados) //com quase o mesmo principio do anterior, a itera\_\_o varre e procura apenas os estados finais

{

if (e.isFinal()) {

elEInut(e); //esses n\_o considerados uteis, e um itera\_\_o \_ feita para se decobrir os estados que levam a eles

}

}

deletarEstadosInvalidos(); //ap\_s a identifica\_\_o dos estados inuteis, eles s\_o deletados

}

private void elEInut(Estado atual) {

if (atual.isConsideracao()) {

return;

}

atual.setConsideracao(true); //todo estado a ser analizado é acessivel, pois teve referencia de outro que chamou a função

System.out.println(" - " + atual + " é útil");

for (Estado e : estados) { //percorre os estados em busca de referencias ao estado atual.

if (e.equals(atual) || e.isConsideracao()) {

continue; //despreza se for o atual ou já considerado

}

for (char c : e.transicoes.keySet()) { //para cada transição

// verifica se a lista de cada transição contem o estado, senão tiver, passa para a proxima

if (e.transicoes.get(c).contains(atual)) {

//senao for inicial..continua a procura recursivamente, até chegar no estado inicial.

//demonstrando que n\_o somente \_ util, como acessivel.

if (!inicial.equals(e)) {

elEInut(e);

}

e.setConsideracao(true); //uma vez que se mostrou util, \_ marcado.

break; //a itera\_\_o para, posi n\_o h\_ mais necessidade de percorrer as transi\_\_es, uma vez que o estado j\_ \_ util.

}

}

}

atual.setConsideracao(true); //por ultimo, o estado atual \_ marcado como util.

}

private void deletarEstadosInvalidos() {

//percorre os estado, o uso de um contador ao inves da itera\_\_o for each, se baseia na mesma razao do algotitmo de elimina\_\_o

//de n\_o determinismo

for (int i = 0; i < estados.size(); i++) {

Estado e = estados.get(i); //captura o estado atual

if (e.isConsideracao()) {

continue; //se for considerado util ou acessivel, ignora as instrutl\_es e vai ao proximo;

}

for (Estado ee : estados) //inicia-se a busca, em cada estado por referencias/transsi\_\_es que o contenha

{

for (char c : ee.transicoes.keySet()) //cas encontre alguma transição a esse estado a ser deletado, ela \_ removida

{

if (ee.transicoes.get(c).contains(e)) {

ee.transicoes.get(c).remove(e);

}

}

}

estados.remove(i); //somente apos remover as referencias do estado, \_ possivel removelo com seguran\_a.

}

try{

//apos as remo\_\_es, uma nova itera\_\_o \_ feita ara remover transi\_\_es vazias (que ficaram se referencia a nenhum estado)

for (int i = 0; i < estados.size(); i++) {

Estado e = estados.get(i); //captura o estado atual

e.setConsideracao(false); //os estados n\_o desmarcados para execu\_\_o de um proximo algoritmo.

//como n\_o ser\_ acrescentado nenhuma tarsni\_\_o,o estado dessa pode ser guardado

//para controlar a itera\_\_o

Set<Character> cs = e.transicoes.keySet();

for (char c : cs) //para cada transi\_\_o, se esta estiver vazia, \_ removida.

{

if (e.transicoes.get(c).isEmpty()) {

e.transicoes.remove(c);

}

}

}

}catch(Exception e){

}

}

public void deletarEstado(String nome) {

for (int i = 0; i < estados.size(); i++) {

Estado e = estados.get(i); //captura o estado atual

if (!e.getNome().equals(nome)) {

continue; //se n\_o for o estado ignora tudo

}

for (Estado ee : estados) //inicia-se a busca, em cada estado por referencias/transsi\_\_es que o contenha

{

for (char c : ee.transicoes.keySet()) //cas encontre alguma transi\_\_o a esse estado a ser deletado, ela \_ removida

{

if (ee.transicoes.get(c).contains(e)) {

ee.transicoes.get(c).remove(e);

}

}

}

estados.remove(i); //somente apos remover as referencias do estado, \_ possivel removelo com seguran\_a.

}

//apos as remo\_\_es, uma nova itera\_\_o \_ feita ara remover transi\_\_es vazias (que ficaram se referencia a nenhum estado)

for (Estado e : estados) {

Set<Character> cs = e.transicoes.keySet();

for (char c : cs) //para cada transi\_\_o, se esta estiver vazia, \_ removida.

{

if (e.transicoes.get(c).isEmpty()) {

e.transicoes.remove(c);

}

}

}

}

protected class Classe {

protected boolean marcada;

protected List<ParEq> pares = new ArrayList<>();

protected void addPar(Estado q1, Estado q2) {

if(marcada)return;

ParEq pe = new ParEq(q1, q2);

ParEq peInv = new ParEq(q2, q1);

if (pares.contains(pe) || pares.contains(peInv)) return;

pares.add(pe);

}

protected void marcar(){

marcada=true;

pares.clear();

}

}

protected class ParEq {

protected Estado q1;

protected Estado q2;

protected ParEq(Estado q1, Estado q2) {

this.q1 = q1;

this.q2 = q2;

}

@Override

public String toString() {

return "{" + q1 + "," + q2 + '}';

}

@Override

public boolean equals(Object obj) {

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

final ParEq other = (ParEq) obj;

if (!Objects.equals(this.q1, other.q1)) {

return false;

}

if (!Objects.equals(this.q2, other.q2)) {

return false;

}

return true;

}

@Override

public int hashCode() {

int hash = 7;

hash = 97 \* hash + Objects.hashCode(this.q1);

hash = 97 \* hash + Objects.hashCode(this.q2);

return hash;

}

}

int tam;

private Classe[][] classes;

private void minimizando() {

int tam = estados.size() - 1;

classes = new Classe[tam][tam];

//marcando os pares trivialmente não equivalentes

for (int i = 0; i < tam; i++) {

for (int y = 0; y < tam; y++) { //para cada coluna (relação de pares)

classes[i][y] = new Classe();

if ((i + 1) <= y) {

classes[i][y].marcada = true; //ajuste da diagonal principal

} //caso ambos sejam finais ou não finais, são potencialemnte equivalentes, e não serão marcados

if ((estados.get(i + 1).isFinal() && estados.get(y).isFinal()) //se ambos forem finais

|| (!estados.get(i + 1).isFinal() && !estados.get(y).isFinal())) //se ambos forem não finais

{

continue; //pula para não marcar os estados

} //se um for final e o outro não, não são potencialmente equivalentes e serão marcados.

classes[i][y].marcada = true;

}

}

System.out.println("---- Iniciando o algoritmo de Minimização ----");

System.out.println("-- 1ª fase: Classes de equivalencia --");

mostraClasses();

System.out.println("-- 2ª fase: Análise dos pares --");

for (int i = 0; i < tam; i++) { //para cada coluna (relação de pares)

for (int y = 0; y < tam; y++) {

if (classes[y][i].marcada) {

continue; //se for uma relação marcada, pula

}

Estado qu = estados.get(i); //captura o estado qu

Estado qv = estados.get(y + 1); //captura o estado qv

System.out.println("-- a) par {" + qu + "," + qv + "}");

//para cada transição de qv, vejo se qu tem uma transição com o mesmo simbolo

for (char a : qv.transicoes.keySet()) {

//se houve uma transição para esse estado

if (qu.transicoes.keySet().contains(a)) {

if(qu.transicoes.get(a).isEmpty()) continue;

Estado pu = qu.transicoes.get(a).get(0); //captura o estado destino pu

if(qv.transicoes.get(a).isEmpty()) continue;

Estado pv = qv.transicoes.get(a).get(0); //captura o estado destino pv

System.out.println("-- G(" + qu + "," + a + ")=" + pu + " e G(" + qv + "," + a + ")=" + pv + ";");

//Se pu=pv, então

if (pu.equals(pv)) {

System.out.println("-- " + pu + "=" + pv + ":");

//qu é equivalente a qv para o simbolo a e não deve ser marcado

System.out.println("-- " + qu + " é equivalente a " + qv + " para " + a + " e não deve ser marcado;");

continue;

}

//se pu!= pv, então

System.out.println("-- " + pu + "!=" + pv + ":");

boolean marcado;

if (pv.getIndex() < pu.getIndex()) {

marcado = classes[estados.indexOf(pu) - 1][estados.indexOf(pv)].marcada;

} else {

marcado = classes[estados.indexOf(pv) - 1][estados.indexOf(pu)].marcada;

}

if (!marcado) {

//se pu!=pv e o par {pu,pv) não está marcado

System.out.println("-- {" + pu + "," + pv + "} não está marcado,");

//{qu,qy} é incluido em uma lista a partir de {pu,pv} para posterior analise

if (pv.getIndex() > pu.getIndex()) {

classes[estados.indexOf(pv) - 1][estados.indexOf(pu)].addPar(qv, qu);

} else {

classes[estados.indexOf(pu) - 1][estados.indexOf(pv)].addPar(qu, qv);

}

System.out.println("-- {" + qu + "," + qv + "} é incluido em uma lista a partir de"

+ " " + pu + "," + pv + "} para posterior analise.");

continue;

} else {

//se pu!=pv e o par {pu,pv) está marcado

System.out.println("-- {" + pu + "," + pv + "} está marcado,");

//{qu,qy} não é equivalente e deve ser marcado

System.out.println("-- {" + qu + "," + qv + "} deve ser marcado.");

marcarEncabecado(qu, qv);

marcarEncabecado(qv, qu);

continue;

}

}

}

mostraClasses();

}

}

System.out.println("-- 3ª fase: Fusão dos equivalentes --");

// identificando os estados equivalentes

ArrayList<Estado> pararemover = new ArrayList<>();

for (int i = 0; i < tam; i++) {

for (int y = 0; y < tam; y++) {

//se estiver maracda, ignora.

if(classes[i][y].marcada) continue;

Estado qu = estados.get(i+1); //captura o estado qu

Estado qv = estados.get(y); //captura o estado qv

// criando um novo estado que o substitui

Estado Qnovo = new Estado("q0"+qu.getIndex()+qv.getIndex()); //q012, q023...

Qnovo.setFinal(qu.isFinal());

if(qu.equals(inicial) | qv.equals(inicial)) inicial=Qnovo;

estados.add(Qnovo);

System.out.println("-- Fundindo o par {" + qu + "," + qv + "} no estado "+Qnovo);

// copiando as transições a esse novo estado

// não é precisa ver as transições de qv, pois esses são equivalentes.

for(char c : qu.transicoes.keySet()){

if(qu.transicoes.get(c).isEmpty()) continue;

System.out.println("newTransicao("+Qnovo.getNome()+","+c+", "+qu.transicoes.get(c).get(0).getNome()+")");

newTransicao(Qnovo.getNome(), c+"", qu.transicoes.get(c).get(0).getNome());

}

// procurando referencias a esses estados e trocando para o novo

for(Estado e :estados){

//se for um dos dois ignora

if(e.equals(qu) | e.equals(qv)) continue;

//se não, para cada transição, verificamos se existe o estado.

for(char c : e.transicoes.keySet()){

// cada transição contem qu ou qv?

if (e.transicoes.get(c).contains(qu) | e.transicoes.get(c).contains(qv)){

//se houver, limpa e troca a referencia

e.transicoes.get(c).clear();

e.transicoes.get(c).add(Qnovo);

}

}

}

pararemover.add(qu);

pararemover.add(qv);

}

}

estados.removeAll(pararemover);

}

private void marcarEncabecado(Estado qu, Estado qv) {

// se {qu,qv} encabeçam a lista de pares, então maracmos todos os pares da lista e recursivamente

// se algum par da lista encabeça outra lista

System.out.println("-- {" + qu + "," + qv + "} encabeça a lista: " + classes[estados.indexOf(qu) - 1][estados.indexOf(qv)].pares);

if (!classes[estados.indexOf(qu) - 1][estados.indexOf(qv)].pares.isEmpty()) {

System.out.println("-- os pares devem ser marcado.");

for (ParEq p : classes[estados.indexOf(qu) - 1][estados.indexOf(qv)].pares) {

marcarEncabecado(p.q1, p.q2);

}

}

classes[estados.indexOf(qu) - 1][estados.indexOf(qv)].marcar();

}

private void mostraClasses() {

System.out.println("-- -------------- --");

tam = classes.length;

System.out.print("\t");

for (int y = 0; y < tam; y++) {

System.out.print(estados.get(y) + "\t");

}

System.out.println("");

for (int i = 0; i < tam; i++) {

System.out.print(estados.get(i + 1) + "\t");

for (int y = 0; y < tam; y++) {

System.out.print(

(classes[i][y].marcada? "[X]" : "[ ]") +"\t");

}

System.out.println("");

}

for (int i = 0; i < tam; i++) {

for (int y = 0; y < tam; y++) {

if(classes[i][y].pares.isEmpty()) continue;

System.out.println("\*"+estados.get(y)+","+estados.get(i + 1)+"-->"+ classes[i][y].pares + "");

}

}

System.out.println("-- -------------- --");

}

public void minimizar() {

PrintStream ps = null;

PrintStream atual = System.out;

try {

//cria o arquivo de log

File arquivo = new File("logMinimizacao.txt");

ps = new PrintStream(arquivo); //cria um stream de saida

System.setOut(ps); //troca a saida padrão do sistema.

gerarRelatorio("----Automato original----");

System.out.println("-- ----------------------- --");

eliminarTransicoesEmVazio();

gerarRelatorio("----Eliminada as transições em vazio----");

System.out.println("-- ----------------------- --");

eliminarNaoDeterminismo();

gerarRelatorio("----Eliminados os Não determinismos----");

System.out.println("-- ----------------------- --");

eliminarEstadosInacessiveis();

gerarRelatorio("----Eliminados estados inacessiveis----");

System.out.println("-- ----------------------- --");

eliminarEstadosInuteis();

gerarRelatorio("----Eliminados estados inuteis----");

System.out.println("-- ----------------------- --");

minimizando();

gerarRelatorio("----Minimizando ----");

System.out.println("-- ----------------------- --");

eliminarEstadosInacessiveis();

eliminarEstadosInacessiveis();

gerarRelatorio("----Eliminados estados inacessiveis----");

System.out.println("-- ----------------------- --");

System.out.println("-- ----------------------- --");

eliminarEstadosInuteis();

gerarRelatorio("----Eliminados estados inuteis----");

System.out.println("-- ----------------------- --");

gerarRelatorio("----Automato Minimizado:----");

System.out.println("-- ----------------------- --");

} catch (FileNotFoundException ex) {

} finally {

ps.close();

}

System.setOut(atual); //devolve a saida padrão

}

@Override

public String toString() {

String toReturn = nome + " = (Q, \u03a3, \u03b4, " + inicial + ", F);";

toReturn += "\nQ=[ ";

for (Estado e : estados) {

toReturn += e + ",";

}

toReturn = toReturn.substring(0, toReturn.length() - 1) + "]";

toReturn += "\n\u03a3=[ ";

for (char c : alfabeto) {

toReturn += c + ",";

}

toReturn = toReturn.substring(0, toReturn.length() - 1) + "]";

toReturn += "\n\u03b4 = { ";

for (Estado e : estados) {

for (char c : e.transicoes.keySet()) {

toReturn += "(" + e + "," + c + ") = {";

for (Estado ee : e.transicoes.get(c)) {

toReturn += ee + ",";

}

toReturn = toReturn.substring(0, toReturn.length() - 1) + "} ; ";

}

}

toReturn += " }\nF= [ ";

for (Estado ee : estados) {

if (ee.isFinal()) {

toReturn += ee + ",";

}

}

toReturn = toReturn.substring(0, toReturn.length() - 1) + "]";

return toReturn;

}

public void gerarRelatorio(String oper) {

System.out.println(oper);

System.out.println(toString());

}

public static void main(String[] args) {

Automato a = new Automato();

String automato = "M1=({q0,q1,q2,q3,q4},"

+ "{a,b,c},"

+ "{(q0,a)=q1 ; (q0,b)=q2 ; (q0,E)={q2,q3} ; (q2,c)={q3,q1} ; (q3,a)=q4;},"

+ "{q0},"

+ "{q2,q3})";

String iscompleta = "[m|M][1-9]=\\(\\{(q[0-9]|,)\*\\},"

+ "\\{([a-z]|,)\*\\},"

+ "\\{(\\(q[0-9],([a-z]|E)\\)=\\{?(q[0-9]|,){1,}\\}?|\\s;\\s|;)\*\\}," + //{(q0,a) = q1 ; (q0,b) = q2 ; (q0,E) = {q2,q3} },

"\\{(q[0-9])\*\\}," + //{q0},

"\\{(q[0-9]|,)\*\\}\\)"; //{q2,q3})

if (automato.matches(iscompleta)); else {

System.out.println("Automato de entrada n\_o confere com o padr\_o:\n " + automato + " \nDeve seguir o modelo:\n "

+ "M1=(Estados,Alfabeto,Transi\_\_es,Estado Inicial,Estados Finais)\n "

+ "M1=({q0,q1,q2,q3,...},"

+ "{a,b,...},"

+ "{(q0,a)=q1 ; (q0,E)={q2,q3,...} ...},"

+ "{q0},"

+ "{q2,...})");

return;

}

System.out.println("Automato: " + automato);

Matcher m;

//\*\* estudo do alfabeto em modo texto \*\*//

m = Pattern.compile("\\{([a-z]|,)\*\\}").matcher(automato); //captura {a,b,c...}

String simbolos = m.find() ? m.group().replaceAll("\\{|\\}|,", "") : "E";

a.alfabeto = simbolos.toCharArray();

//\*\* estudo dos estados em modo texto \*\*//

m = Pattern.compile("[m|M][1-9]=\\(\\{(q[0-9]|,)\*\\}").matcher(automato); //captura M1={q0,q1,q2,q3,...}

m = Pattern.compile("q[0-9]").matcher(m.find() ? m.group() : ""); //captura os estados

while (m.find()) {

a.addEstado(m.group());

}

//\*\* estudo das transi\_\_es em modo texto \*\*//

String patt = "\\(q[0-9],([a-z]|E)\\)=\\{?(q[0-9]|,){1,}\\}?"; //captura: (q0,b) = q2 ou (q0,E) = {q2,q3}

m = Pattern.compile(patt).matcher(automato);

String estadoOrigem = "", simbolo = "", transicao;

String[] estadosDestinos = args;

Matcher subm;

while (m.find()) {

transicao = m.group(); //(qu,a)={qv,qn,qm,...}

subm = Pattern.compile("\\(q[0-9],").matcher(transicao); //captura o estado origem

while (subm.find()) {

estadoOrigem = subm.group().replaceAll("\\(|,", "");

}

subm = Pattern.compile("([a-z]|E)\\)").matcher(transicao); //captura o estado origem

while (subm.find()) {

simbolo = subm.group().replaceAll("\\)", "");

}

subm = Pattern.compile("=\\{?(q[0-9]|,){1,}\\}?").matcher(transicao); //captura o estado origem

while (subm.find()) {

estadosDestinos = subm.group().replaceAll("\\)|=|\\{|\\}", "").split(",");

}

a.newTransicao(estadoOrigem, simbolo, estadosDestinos);

}

//\*\* estudo dos estados finais em modo texto \*\*//

m = Pattern.compile("\\{(q[0-9]|,)\*\\}\\)").matcher(automato); //captura {q2,q3})

m = Pattern.compile("q[0-9]").matcher(m.find() ? m.group() : ""); //captura os estados finais

while (m.find()) {

a.turnFinal(m.group());

}

//\*\* estudo do estado inicial em modo texto \*\*//

m = Pattern.compile(",\\{q[0-9]\\},").matcher(automato); //captura {q0})

while (m.find()) {

a.setInicial(m.group().replaceAll("\\{|\\}|,", ""));

}

a.minimizar();

}

public Estado getInicial() {

return inicial;

}

public void setInicial(Estado inicial) {

this.inicial = inicial;

}

public char[] getAlfabeto() {

return alfabeto;

}

public void setAlfabeto(char[] alfabeto) {

this.alfabeto = alfabeto;

}

public ArrayList<Estado> getEstados() {

return estados;

}

public void setEstados(ArrayList<Estado> estados) {

this.estados = estados;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

}

# Estrutura de dados do projeto

## Estrutura do Automato

Transicoes : HashMap<Character,ArrayList<Estado>>

Autômato

Nome: String

Inicial : Estado

Alfabeto : char[]

Estados :ArrayList<Estado>

Estado

Nome: String

isFinal : boolean

Consideracao; boolean

## Estrutura da minimização

Classe interna: Classe

Marcado: boolean

i

pares :ArrayList<parEq>

Classe interna: parEq

q2: Estado

Q1: Estado

Qo q1 q2 q3 q4 q5

Qo q1 q2 q3 q4 q5

o 1 2 3 4

o 1 2 3 4

i

qu

qv

y