Simulador de Autômato Finito Determinístico

Java

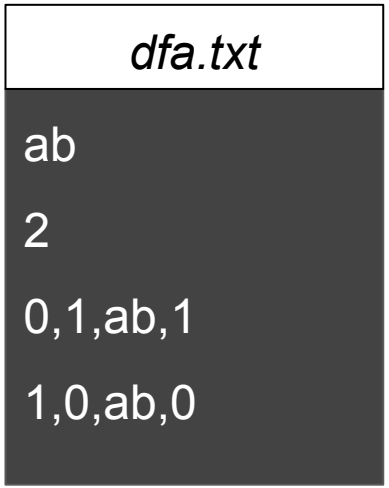
Conteúdo

Objetivo

- **Simular** autômatos finitos determinísticos (**AFDs**) a partir de uma descrição que contém:
  - Alfabeto
  - Estados
    - Funções de transição

- Suportar operações
  - União
  - Interseção
  - Complemento

Modelo de Descrição



alfabeto

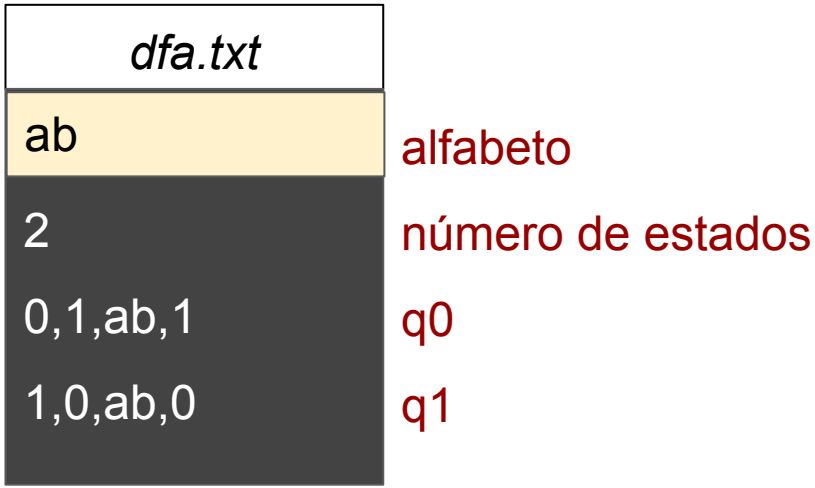
número de estados

q0

q1

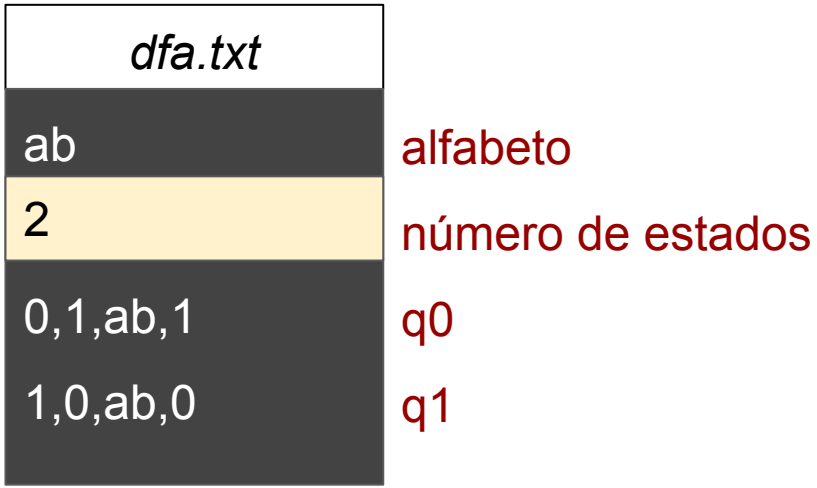
Modelo de Descrição

- Alfabeto genérico
  - ab
  - 01
  - abcdefghijklmnopqrstuvwxyz



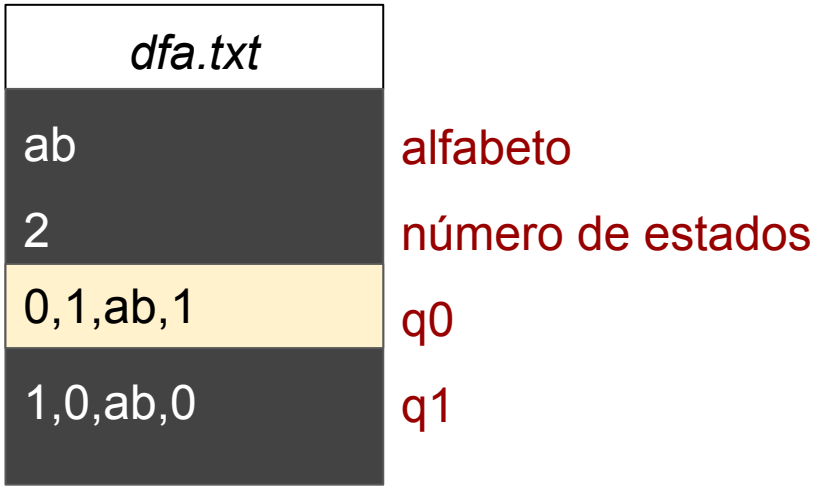
Modelo de Descrição

- Número de estados
  - Todos os estados devem ser descritos abaixo



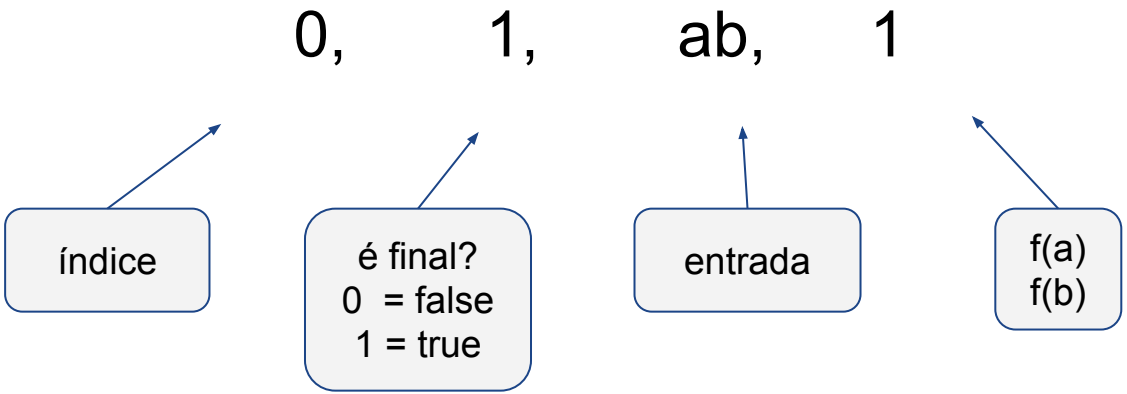
Modelo de Descrição

- Descrição de um estado



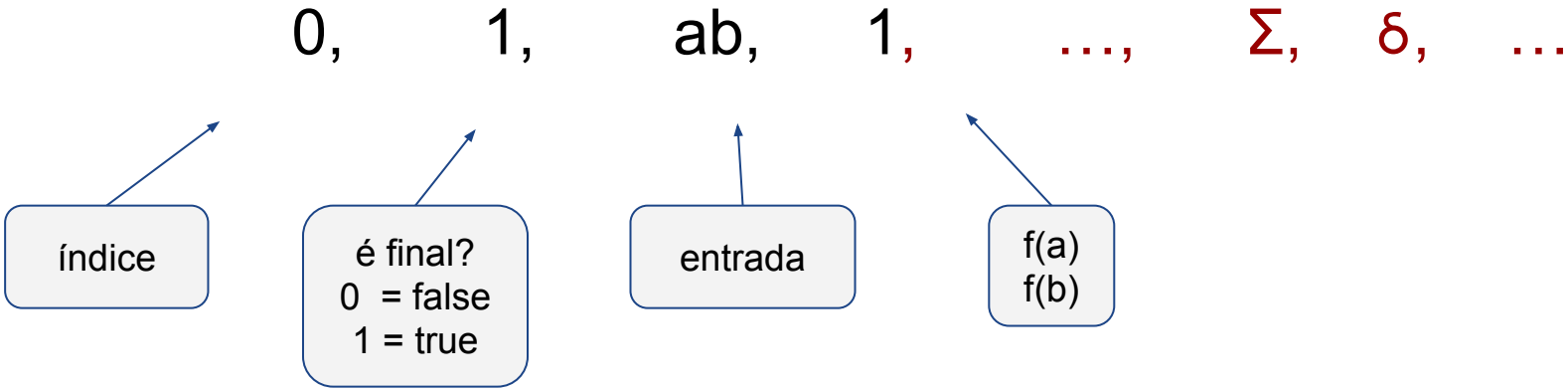
Modelo de Descrição

- Descrição de um estado



Modelo de Descrição

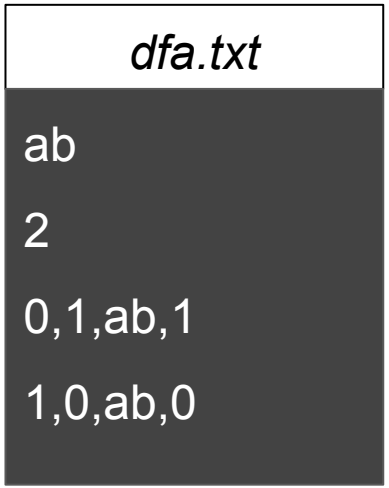
- Descrição de um estado



Modelo de Descrição

L(dfa.txt) =

{ w | w possui **comprimento par** }.



Modelo de Descrição

L(dfa.txt) =

{ w | w possui **pelo menos dois b's** }.

Mais de um destino na transição de um estado.

dfa.txt	
ab	alfabeto
3	número de estados
0,0,a,0,b,1	q0
1,0,a,1,b,2	q1
2,1,a,2,b,2	q2

Construção de Autômato

```
int[] transitions;
```

q0

	a	b
i	0	1
v	0	1

q1

	a	b
i	0	1
v	1	2

q2

	a	b
i	0	1
v	2	2

dfa.txt

```
ab
3
0,0,a,0,b,1
1,0,a,1,b,2
2,1,a,2,b,2
```

alfabeto

número de estados

q0

q1

q2

Diagrama de Classe

State
- alphabet : String - acceptance : boolean - transitions : int[] - index : int
~ State(index : int, F : boolean, alphabet : String) ~ isAcceptance() : boolean ~ setTransition(from : int, to : int) ~ getTransition(from: int) : int

Diagrama de Classe

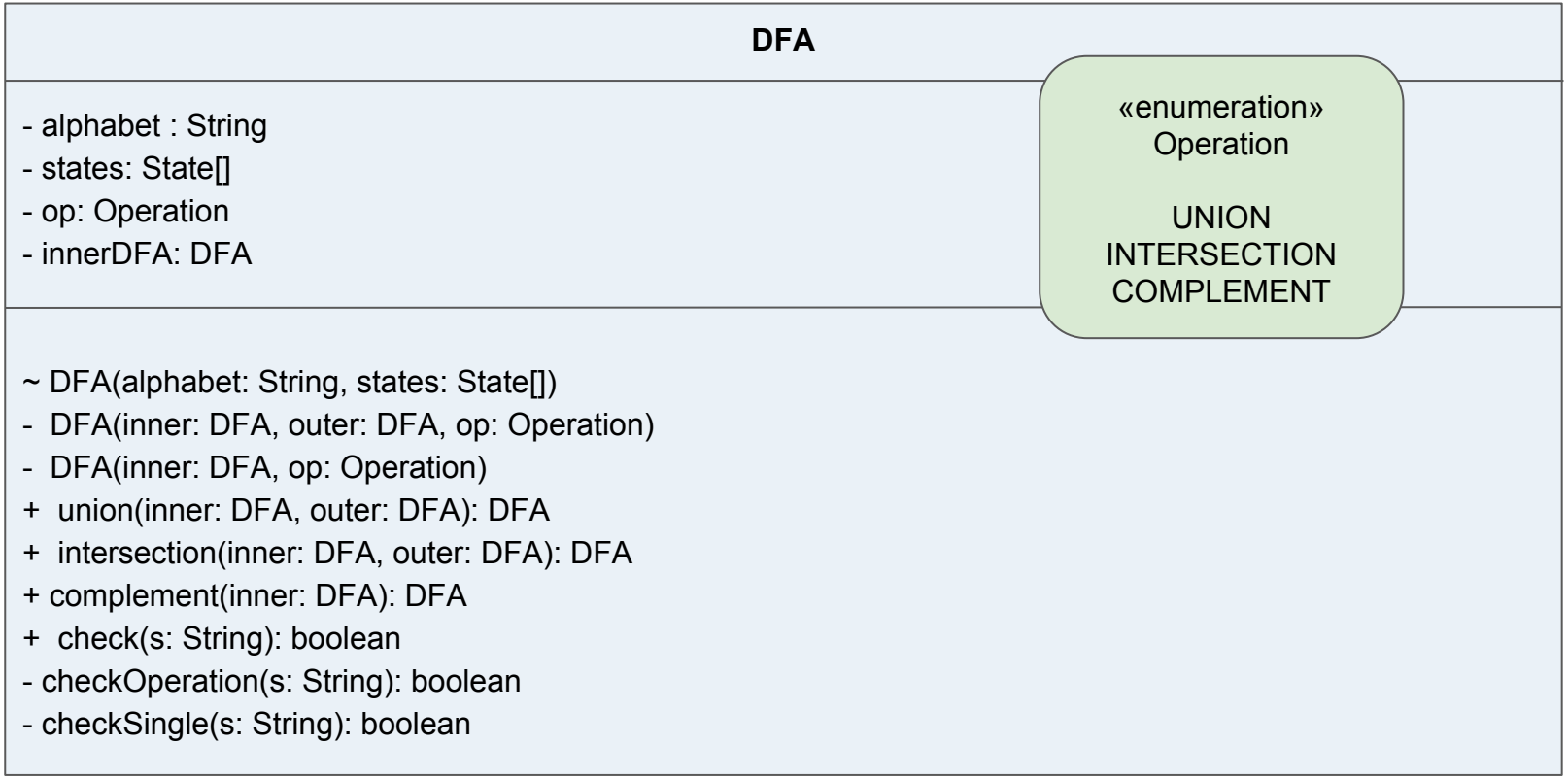
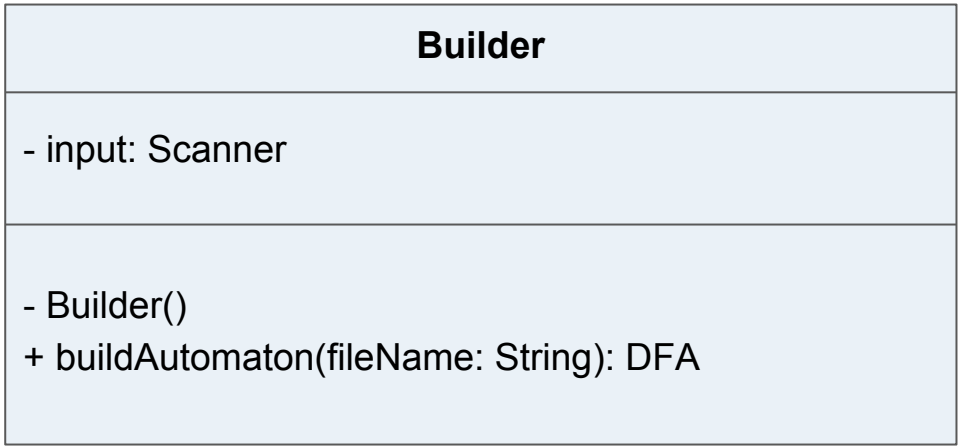
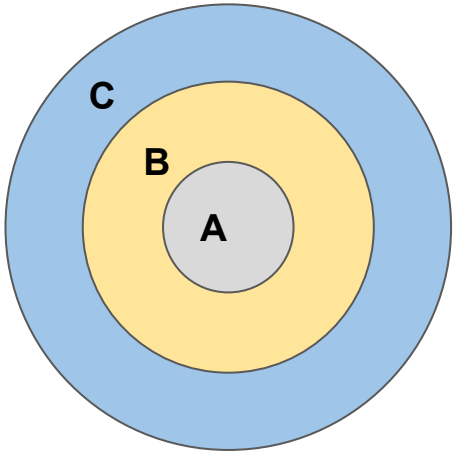


Diagrama de Classe



Composição Recursiva



$(A \cup B) \cap C$

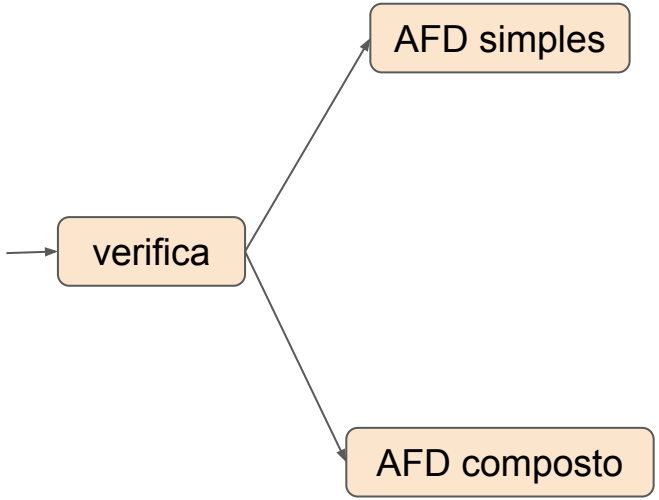
$A \cup B$

$A \cap B$

$\overline{A}$

```
class DFA {

      ...

      DFA innerDFA;

      ...

}
```

Verificação de String




```
public boolean check(String s) {
      if (op == null && innerDFA == null) {
            return checkSingle(s);
      }
      else {
            return checkOperation(s);
      }
}
```

Verificação de String

- AFD Simples

```
current = 0; // initial state
for(… string.length …) {
      from = alphabet.indexOf(c);
      current = states[current].getTransition(from); // f(from) -> to
}
if(states[current].isF()){
      return true;
}
return false;
```

Verificação de String

- AFD Composto

Ver código...

Bibliografia

- SIPSER, Introduction to the Theory of Computation, 3rd Edition - 2013 Cengage Learning

Execução