Linguagens Formais e Autômatos (LFA)

Aula de 26/08/2013

Processadores Simbólicos
Aspectos de Arquitetura e Implementação

Tópicos

- Cadeias, símbolos e linguagens em Ruby
- Visão geral da arquitetura
- Descrição das principais classes
- Implementação de reconhecedor determinístico
- Exemplos
- Dicas sobre ambiente de execução

Representação de cadeias em Ruby

Cadeia: justaposição de quantidade finita de símbolos de um alfabeto

Representação de símbolos como string, cadeias como arrays. Exemplos para o alfabeto Σ = {a,b,c}:

Símbolo: a ⇒ "a"
Cadeia: abc ⇒ ["a","b","c"]
Cadeia vazia: ε ⇒ []

Manipulação de cadeias

Criação: método split

```
r = "abc".split(//) # ["a","b","c"]
s = "xy".split(//)  # ["x","y"]
```

Comprimento de uma cadeia

```
r.length()          # 3
[].length()         # 0
```

Concatenação de cadeias

```
r + s               # ["a","b","c","x","y"]
```

Linguagens em Ruby

Uma linguagem é um conjunto de cadeias. Portanto, a representação em Ruby pode ser feita através de um array de arrays. Exemplos:

{ ε } = `[ [] ]`

{ ab, ε, c, ccc } = `[["a","b"],[],["c"],["c","c","c"]]`

{ a, b, c } = `[["a"],["b"],["c"]]`

Restrição: representação de linguagens **finitas**

Concatenação de linguagens

```
def concatenar( linguagemX, linguagemY )
  linguagemZ = []
  linguagemX.each do |x|
     linguagemY.each do |y|
        linguagemZ << x + y
     end
  end
  return linguagemZ
end
```

Concatenação de Linguagens - Exemplos

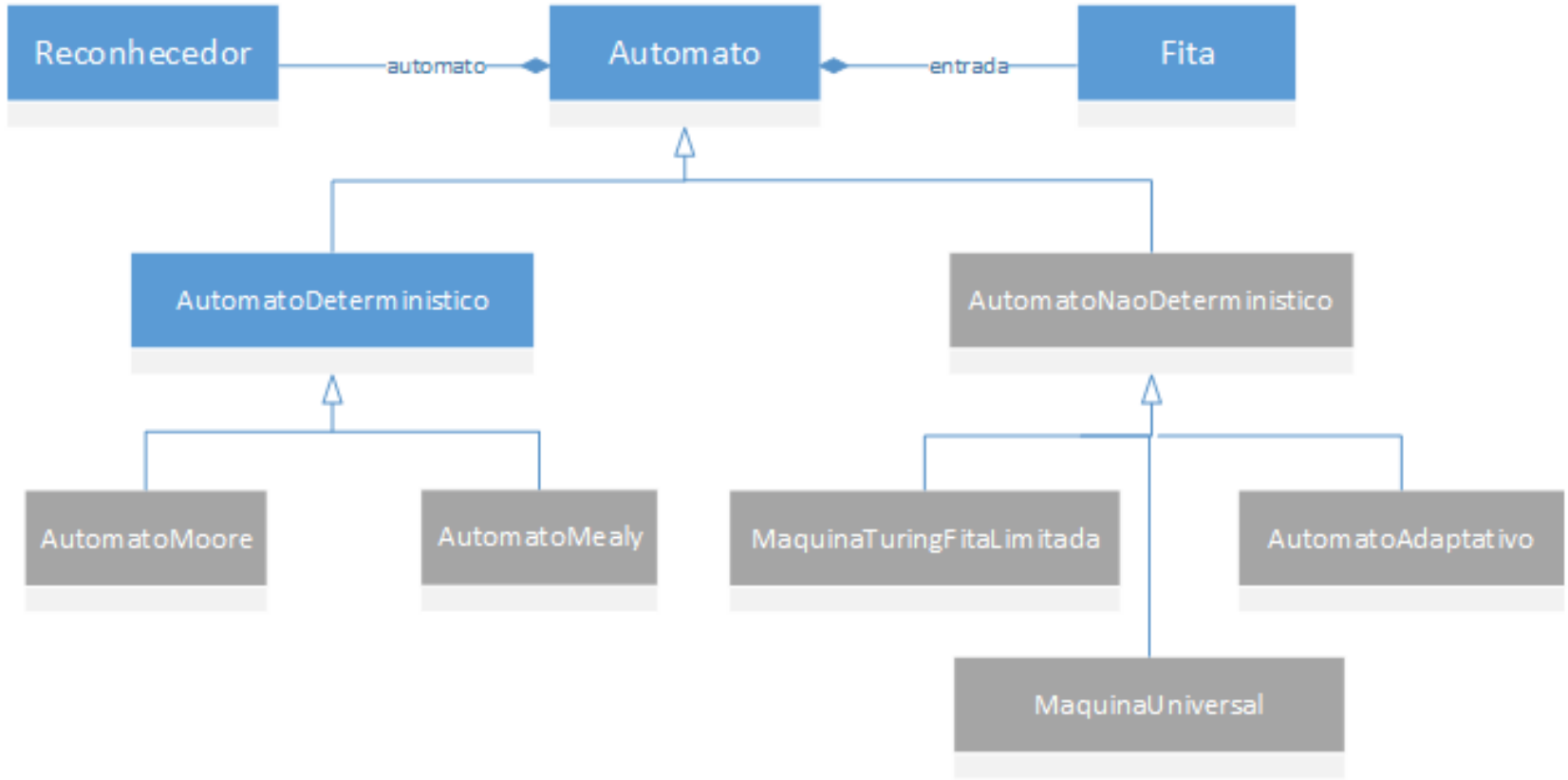
L1 = { a, b, c } ⇒ `[["a"],["b"],["c"]]`

```
concatenar( L1, L1 ) ==
[["a", "a"], ["a", "b"], ["a", "c"], ["b", "a"], ["b",
  "b"], ["b", "c"], ["c", "a"], ["c", "b"], ["c", "c"]]
```

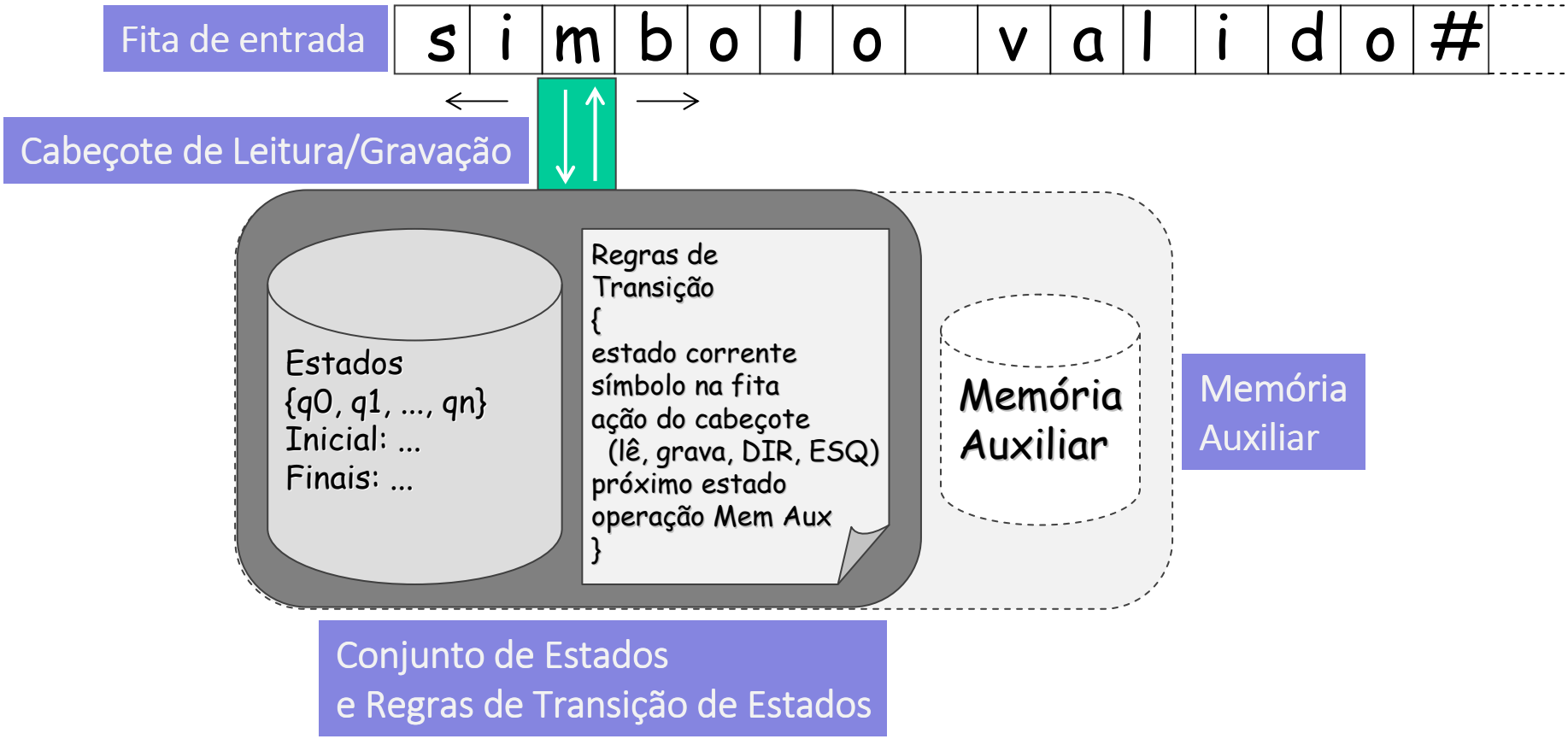
L2 = { ε } ⇒ `[ [] ]`

```
concatenar( L1, L2 ) == L1
```

Arquitetura de Implementação (livro-texto)



Modelo geral dos autômatos: Máquina de Turing



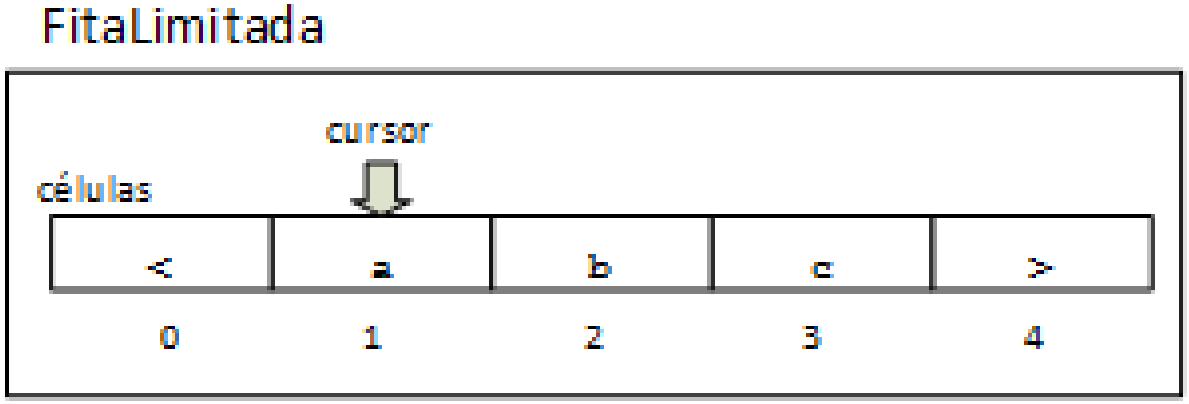
Classe Fita: base para tipos específicos

```
class Fita
    attr_accessor :celulas  #array
    attr_accessor :cursor   #inteiro

    iniciar( cadeia )
    ler()
    avancar()
    recuar()
    atingiuBOF?()
    atingiuEOF?()
    configuracao?()
    clonar()
end
```

Classe FitaLimitada

Implementação de Fita cuja cadeia encontra-se limitada por dois marcadores '<' e '>'



Ver arquivo "af/FitaLimitada.rb"

Classe Automato (af/Automato.rb)

```
class Automato
    attr_accessor :entrada
    attr_accessor :transicoes
    attr_accessor :estadosFinais
    attr_accessor :estadoCorrente
    attr_accessor :movimentacao
    attr_accessor :consulta

    initialize(estadoInicial, estadosFinais)
    instanciarEntrada()
    instanciarMovimentacao()
    adicionarTransicao( transicao )
    iniciar( cadeia )
    executar()
   (...)
end
```

Classe Reconhecedor (af/Reconhecedor.rb)

```
class Reconhecedor
    attr_accessor :automato

    initialize( estadoInicial, estadosFinais )
    instanciarAutomato( estadoInicial, estadosFinais )
    iniciar( cadeia )
    analisar()
    reconheceu?()
end
```

Classe Movimentacao (af/servico/Movimentacao.rb)

```
class Movimentacao
    initialize( automato )
    calcularOndaDeClones()
    mover( proximoEstado )
    executar()
end
```

Método `calcularOndaDeClones()` prevê a implementação de não-determinismo para os autômatos (suporte a diferentes configurações "vivas" representadas por "clones")

Serviços adicionais

- Clonagem: permite a duplicação de objetos (fitas, autômatos)
  - `clonar( original )`

- Consulta: interface de informações sobre o estado do autômato
  - `estadoCorrente?()`
  - `configuracao?()`
  - `estaEmEstadoFinal?()`
  - `atingiuEOF?()`

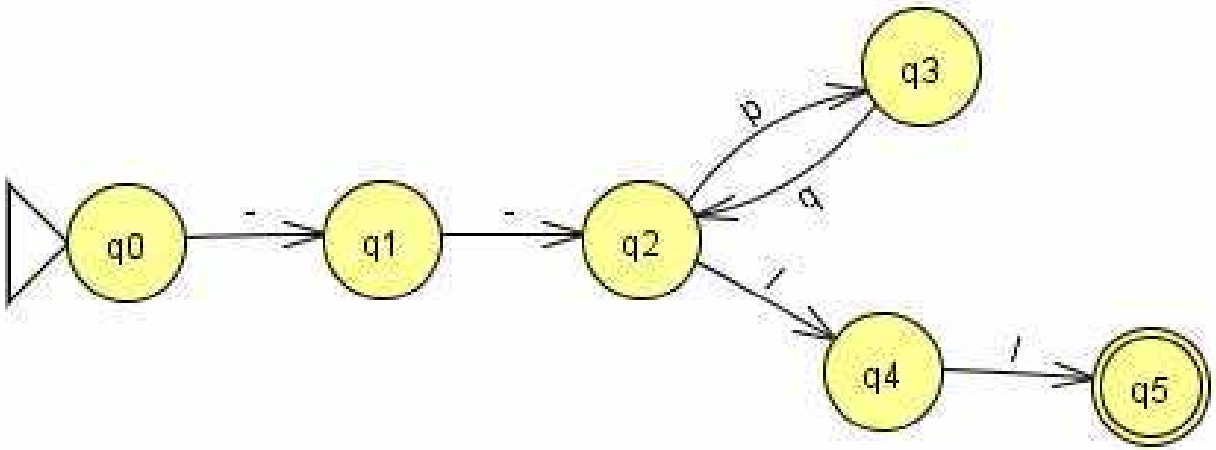
Implementação de Reconhecedor Determinístico

Ver código contido nos arquivos:

- afd/servico/MovimentacaoDeterministica.rb
  - Especialização da classe Movimentacao para AFDs
- afd/AutomatoDeterministico.rb
  - Idem, para a classe Automato
- afd/ReconhecedorDeterministico.rb
  - Idem, para a classe Reconhecedor

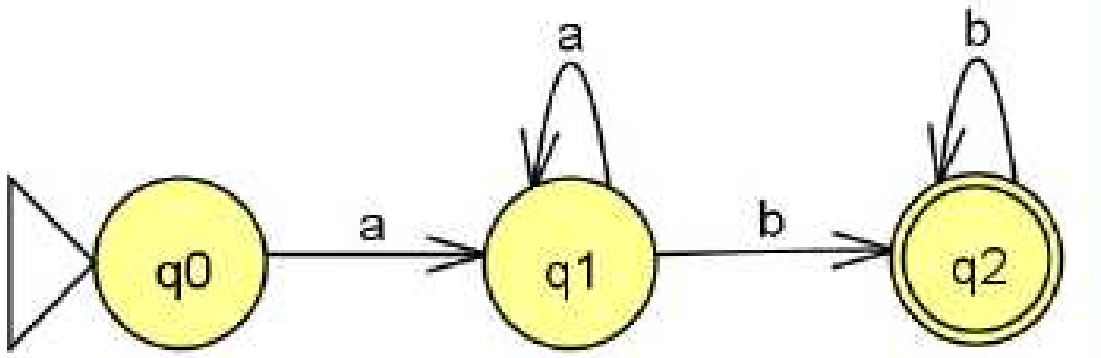
Exemplo 1 (arquivo afd/Exemplo1.rb)

Autômato "conversa de elevador"



Exemplo 2 (arquivo afd/Exemplo2.rb)

Automâto da aula 04



Dicas sobre ambiente de execução

Ruby para Windows:

- http://rubyinstaller.org/downloads/
  - Adicionar o diretório "bin" da instalação de Ruby na variável PATH

Notepad++

- http://notepad-plus-plus.org/download

Plugin NPPExec

- Instalar plugin (Plugins > PluginManager)
- Em "Plugins > NppExec > Execute", inserir o comando:
  - `ruby "$(FULL_CURRENT_PATH)"`
- Para executar script: CTRL+F6