Trabalho 1: Autômatos Finitos com Saída – Máquinas de Mealy e de Moore

Linguagens Formais e Autômatos – Bacharelado em Sistemas de Informação – IFES – Campus Serra

Prof. Jefferson O. Andrade

2018/1

Sumário

1 Descrição do Trabalho **1**

2 Sintaxe dos Arquivos de Entrada e Saída **2**
2.1 Sintaxe para Definição de Máquina de Mealy 2
2.2 Sintaxe para Definição de Máquina de Moore 3

3 Sobre a Entrega do Trabalho **5**

4 Sobre a Execução do Trabalho **6**

1 Descrição do Trabalho

Este trabalho de programação sobre Linguagens Formais e Autômatos propõe a implementação dos algoritmos de conversão de autômatos finitos com saída vistos em sala de aula. Mais especificamente o trabalho propõe a implementação de:

1. Conversão de Máquina de Mealy para Máquina de Moore equivalente.

2. Conversão de Máquina de Moore para Máquina de Mealy equivalente.

As definições das máquinas de Mealy e de Moore, tanto de entrada quanto de saída do programa devem ser lidas (na entrada) e gravadas (na saída) de acordo com o formato definido na Seção 2.

O programa resultante da implementação deste trabalho deve ser um programa "de linha de comando", ou seja, não deve ter interface gráfica de qualquer natureza. Será construído para ser ativado do prompt de comando do sistema operacional (Linux) de acordo com a seguinte sintaxe:

`$` *programa* `-i` ⟨*arquivo-de-entrada*⟩ `-o` ⟨*arquivo-de-saída*⟩

Caso o arquivo de entrada contenha a definição de uma máquina de Mealy, o programa automaticamente gerará a máquina de Moore equivalente e gravará a sua definição no arquivo de saída indicado; e vice-versa.

Caso não seja possível ler uma definição correta de uma máquina de Mealy ou de uma máquina de Moore do arquivo de entrada (erro de sintaxe no arquivo de entrada), deve ser gerada uma mensagem de erro, e o programa deve encerrar.

2 Sintaxe dos Arquivos de Entrada e Saída

A definição das Máquinas de Mealy e Moore será dada na forma de uma S-Expression. S-Expressions são simples de processar e de gravar, e utilizam como delimitadores apenas parênteses e espaços em branco. Para exemplos de como processar S-Expressions em diversas linguagens, consulte a página web do site RosettaCode (com exemplos de código em várias linguagens) ou da Wikipédia sobre S-Expressions.

Um lembrete importante é que a quantidade de espaços em branco e dos caracteres de fim de linha não são importantes em uma S-Expression, ou seja, o fato de haver 1 espaço em branco ou 100, para efeitos de processamento, é indiferente. Também não faz diferença o fato de a S-Expression estar escrita toda em uma linha só, ou em 30 linhas.

2.1 Sintaxe para Definição de Máquina de Mealy

A sintaxe da definição da máquina de Mealy deve seguir a definição em BNF (Backus-Naur Form) dada abaixo:[1]

⟨*maq-mealy*⟩ ::= '`(`' '`mealy`' ⟨*input-symbols*⟩ ⟨*ouput-symbols*⟩ ⟨*set-of-states*⟩ ⟨*start-state*⟩
⟨*set-of-final-states*⟩ ⟨*transition-function*⟩ '`)`'

⟨*input-symbols*⟩ ::= '`(`' '`symbols-in`' ⟨*symbol-list*⟩ '`)`'

⟨*ouput-symbols*⟩ ::= '`(`' '`symbols-out`' ⟨*symbol-list*⟩ '`)`'

⟨*symbol-list*⟩ ::= ϵ
| SYMBOL ⟨*symbol-list*⟩

⟨*set-of-states*⟩ ::= '`(`' '`states`' ⟨*state-list*⟩ '`)`'

⟨*state-list*⟩ ::= ϵ
| STATE ⟨*state-list*⟩

⟨*start-state*⟩ ::= '`(`' '`start`' STATE '`)`'

⟨*set-of-final-states*⟩ ::= '`(`' '`finals`' ⟨*state-list*⟩ '`)`'

⟨*transition-function*⟩ ::= '`(`' '`trans-fn`' ⟨*trans-list*⟩ '`)`'

⟨*trans-list*⟩ ::= ϵ
| ⟨*trans*⟩ ⟨*trans-list*⟩

[1] Note que a definição dada é um caso particular da definição de S-Expressions.

⟨*trans*⟩ ::= '(' STATE STATE SYMBOL ⟨*out-string*⟩ ')'

⟨*out-string*⟩ ::= SYMBOL
| '(' ⟨*symbol-list*⟩ ')'

O terminal SYMBOL representa os símbolos dos alfabetos de entrada e de saída. Devem ser compostos por apenas um caractere Unicode. Já o terminal STATE representa os nomes dos estados do autômatos e deve ser formado por uma sequência de caracteres Unicode, iniciando com uma letra ou com o underscore (_) e seguido de zero ou mais letras, dígitos, underscores ou apóstrofes (').

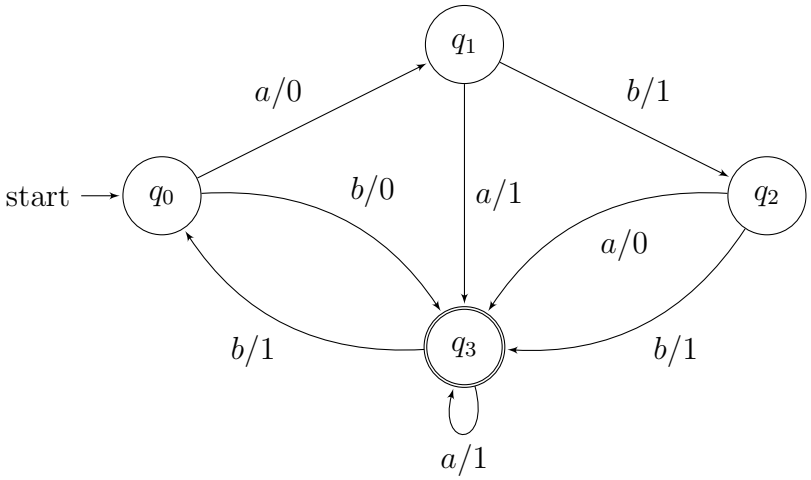

Figura 1: Exemplo de máquina de Mealy.

Seguindo a sintaxe dada acima, a definição da Máquina de Mealy mostrada na Figura 1, fica como mostrado abaixo:

```
(mealy
 (symbols-in a b)
 (symbols-out 0 1)
 (states q0 q1 q2 q3)
 (start q0)
 (finals q3)
 (trans
  (q0 q1 a 0) (q0 q3 b 0) (q1 q2 b 1) (q1 q3 a 1)
  (q2 q3 a 0) (q2 q3 b 1) (q3 q0 b 1) (q3 q3 a 1)))
```

2.2 Sintaxe para Definição de Máquina de Moore

A sintaxe da definição da máquina de Moore deve seguir a definição em BNF (Backus-Naur Form) dada abaixo:[2]

⟨*maq-moore*⟩ ::= '(' '`moore`' ⟨*input-symbols*⟩ ⟨*ouput-symbols*⟩ ⟨*set-of-states*⟩ ⟨*start-state*⟩
⟨*set-of-final-states*⟩ ⟨*transition-function*⟩ ⟨*output-function*⟩ ')'

[2]Note que a definição dada é um caso particular da definição de S-Expressions.

⟨*input-symbols*⟩ ::= '(' '`symbols-in`' ⟨*symbol-list*⟩ ')'

⟨*ouput-symbols*⟩ ::= '(' '`symbols-out`' ⟨*symbol-list*⟩ ')'

⟨*symbol-list*⟩ ::= ϵ
| SYMBOL ⟨*symbol-list*⟩

⟨*set-of-states*⟩ ::= '(' '`states`' ⟨*state-list*⟩ ')'

⟨*state-list*⟩ ::= ϵ
| STATE ⟨*state-list*⟩

⟨*start-state*⟩ ::= '(' '`start`' STATE ')'

⟨*set-of-final-states*⟩ ::= '(' '`finals`' ⟨*state-list*⟩ ')'

⟨*transition-function*⟩ ::= '(' '`trans-fn`' ⟨*trans-list*⟩ ')'

⟨*trans-list*⟩ ::= ϵ
| ⟨*trans*⟩ ⟨*trans-list*⟩

⟨*trans*⟩ ::= '(' STATE STATE SYMBOL ')'

⟨*output-function*⟩ ::= '(' '`out-fn`' ⟨*out-llist*⟩ ')'

⟨*out-list*⟩ ::= ϵ
| ⟨*out*⟩ ⟨*out-list*⟩

⟨*out*⟩ ::= '(' STATE ⟨*out-string*⟩ ')'

⟨*out-string*⟩ ::= SYMBOL
| '(' ⟨*symbol-list*⟩ ')'

Aqui, as definições para os terminais SYMBOL e STATE são as mesmas apresentadas para a gramática da máquina de Mealy.

Seguindo a sintaxe dada acima, a máquina de Moore ilustrada na Figura 2 pode ser definida como mostrado abaixo:

```
(moore
 (symbols-in a b)
 (symbols-out 0 1)
 (states q0 q0' q1 q2 q3 q3')
 (start q0)
 (finals q3 q3')
 (trans
  (q0 q1 a) (q0 q3 b) (q1 q3' a) (q1 q2 b)
  (q2 q0' a) (q2 q3' b) (q3 q3' a) (q3 q0' b)
  (q3' q3' a) (q3' q0' b))
 (out-fn
  (q0 ()) (q0' 1) (q1 0)
  (q2 1) (q3 0) (q3' 1)))
```

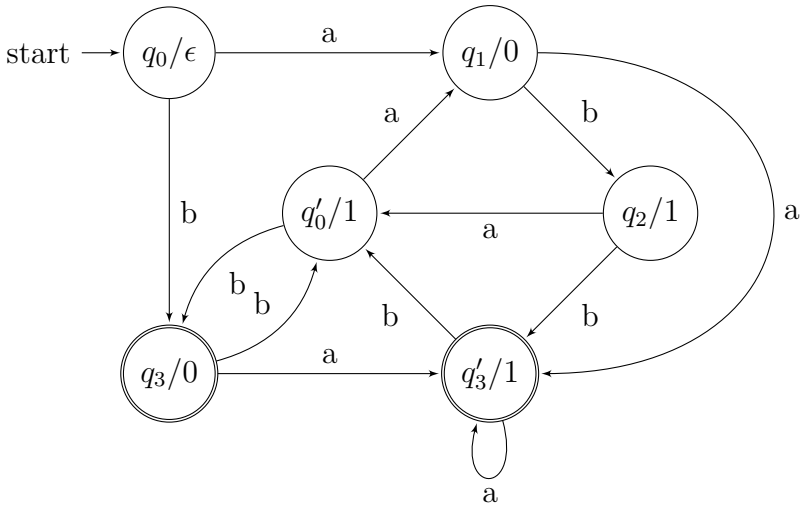

Figura 2: Exemplo de máquina de Moore.

3 Sobre a Entrega do Trabalho

O trabalho deve ser entregue, através do sistema Moodle institucional do Ifes, até o dia **14 de maio de 2018**, as 23:55h.

Devem ser entregues:

- Todo o código fonte desenvolvido para o trabalho;
- Ao menos 3 exemplos de máquina de Mealy para teste;
- Ao menos 3 exemplos de máquina de Moore para teste;
- Um arquivo `Readme.md`, em formato Markdown, contendo:
  - Nome dos autores;
  - Uma descrição breve da estrutura do código fonte;
  - Procedimento para compilação (se for o caso) do trabalho;
  - Nome e modo de uso do programa desenvolvido;
  - Outras informações que os autores julgarem apropriadas para o entendimento do trabalho realizado.
- Um arquivo de configuração de *build* para algum sistema de automação de *build* (ou automação de compilação), como por exemplo: *make*, *ant*, *maven*, *sbt*; ou *NAnt* e *xbuild* para Mono.

  Note que alguns ambientes de desenvolvimento, tais como Eclipse e NetBeans, já geral automaticamente os arquivos de automação de compilação.

O trabalho deve ser entregue **obrigatoriamente** em um arquivo no formato ZIP contendo todos os itens descritos acima.

4 Sobre a Execução do Trabalho

- O trabalho deve ser desenvolvido para o sistema operacional Linux. O trabalho pode ser desenvolvido em outra plataforma, mas o professor irá testar o trabalho em Linux, portanto certifique-se de que o trabalho poderá ser executado em Linux sem problemas.
- O trabalho pode se desenvolvido **individualmente ou em duplas**.
- O trabalho pode ser desenvolvido em qualquer linguagem que esteja disponível para o sistema operacional Linux
- Em todos os casos, o aluno deve se certificar de que o código gerado seja compatível/portável para o sistema operacional Linux e que as instruções de compilação seja claras, precisas e completas o suficiente para serem seguidas pelo professor.
- No caso específico de linguagens como C# e F# (desenvolvidas originalmente para a plataforma .Net), o aluno deve se certificar de que o código gerado seja compatível com o sistema Mono.