

**Matemática Discreta**

*Máquinas de Estado Finito*

Beatriz Dora Waricoda

Guilherme Henrique Gregorini Oliveira

Curso: Engenharia da Computação

Prof. Alisson C. Reinol

Data: 25/11/2018

Sumário

[Introdução 3](#_Toc531239016)

[Constituição 4](#_Toc531239017)

[Regras 4](#_Toc531239018)

[Tipos de máquinas de estado finito 5](#_Toc531239019)

[Representação gráfica 5](#_Toc531239020)

[Tabela de Estado 5](#_Toc531239021)

[Grafo de estado 6](#_Toc531239022)

[Exemplo prático – O Poste 6](#_Toc531239023)

[Implementação 7](#_Toc531239024)

[Eletrônica 7](#_Toc531239025)

[Programação 8](#_Toc531239026)

[Conclusão 9](#_Toc531239027)

[Bibliografia 10](#_Toc531239028)

# Introdução

Ao se construir soluções na área da engenharia é provável que utilizemos modelos que facilitam em seu desenvolvimento. Um desses modelos são as máquinas de estado finito, na qual se utiliza máquinas cujo armazenamento de *estados* é finito. Por exemplo, se utilizar um software como solução, a memória disponível do computador no qual esse software está sendo executado é o limite (finito) de estados possíveis de se armazenar.

A cada instante, a máquina está em um, e somente um, estado, o chamado *estado atual*.

Um relógio interno da máquina é responsável por criar o *ciclo*, em cada qual rotinas serão executas periodicamente.

Nessas rotinas é obtida uma *entrada* e a partir dela em conjunto ao *estado atual* da máquina são obtidos um novo estado e uma *saída*. Abordaremos mais sobre essas rotinas posteriormente.

Desta forma, uma sequência de ciclos requer uma sequência de entradas as quais geram uma sequência de estados e saídas.

# Constituição

Uma máquina estado finito é constituída por um **conjunto de estados finito** que ela admite, um **conjunto finito de símbolos (valores) de entrada** possíveis, um **conjunto finito de símbolos (valores) de saída** possíveis, uma **função (rotina) que gera um estado** (próximo estado) a partir de um estado (estado atual) e uma entrada. E por fim, uma **função (rotina) que gera uma saída** a partir de um estado.

Ou seja, uma máquina de estado pode ser definido por , onde:

* S : **conjunto de estados finito;**
* I : **conjunto finito de símbolos (valores) de entrada;**
* O : **conjunto finito de símbolos (valores) de saída;**
* fs : **função (rotina) que gera um estado;**
* fo : **função (rotina) que gera uma saída;**

A função determina o próximo estado a partir do estado atual e de um valor de entrada, ou seja:

A função determina a saída a partir do estado atual, ou seja:

A cada ciclo, as rotinas e são executadas. Para explicá-las considere uma máquina de ciclos no qual representa cada ciclo, onde . Ou seja, é primeiro ciclo, é segundo ciclo, assim sucessivamente até for o último ciclo. E um ciclo qualquer.

A máquina utiliza para saber qual será seu próximo estado, ou seja:

Desta forma o estado atual é atualizado a todo ciclo pela rotina .

A máquina utiliza para saber qual é saída do estado atual, ou seja:

Desta forma a cada ciclo é gerada uma saída.

# Regras

Algumas regras devem ser respeitadas para o funcionamento da máquina:

1. A máquina sempre é iniciada em um estado inicial;
2. Existem um número finito de estados que a máquina pode alcançar;
3. A cada ciclo a máquina deve estar em um, e somente um, dos seus estados;
4. As rotinas devem ser sincronizadas por ciclos, ou seja, a rotina de um ciclo t=1 não pode acontecer depois da rotina do ciclo t=2;
5. Com uma sequência de entradas determinada deve ser possível determinar a sequências das saídas;
6. A máquina pode ser capaz de produzir saídas, as quais são relacionadas diretamente ao estado atual;

# Tipos de máquinas de estado finito

Até agora foi descrito uma **máquina de Moore**, porém existe similarmente a **máquina de Mealy** na qual sua diferença é:

* A função não só depende de um estado para se obter uma saída, mas também **depende da entrada naquele ciclo**. Desta forma:

Ou

# Representação gráfica

Uma máquina de estado M pode ser representada graficamente por uma tabela e/ou grafo de estado. Para tanto, é necessário definir seus três conjuntos, , e então suas duas funções e pode ser definidas na construção do gráfico em questão.

Para exemplificação a seguir, considere os três conjuntos da máquina de estado M:

## Tabela de Estado

|  |  |  |  |
| --- | --- | --- | --- |
| Estado atual | Próximo estado | | Saída d |
| Entrada atual | |
| 0 | 1 b |
| NOITE a | NOITE | DIA c | 1 |
| DIA e | NOITE | DIA | 0 f |

Tabela 1 - Tabela de estado

É importante perceber na tabela de estado acima que as e foram definidas. Pois, como visto anteriormente define qual será o próximo estado partindo de um estado atual e uma entrada atual, por exemplo, no estado atual *NOITE a* quando a entrada atual é *1* b, o próximo estado será *DIA* c. Na última coluna (*Saída* d) é possível observar , pois esta define qual será a saída de cada estado, por exemplo, se o estado atual for *DIA* e, então a saída será *0* f.

## Grafo de estado

Um grafo direcionado cujo vértices representa cada estado, arestas definem os rótulos de seus vértices definem é chamada de **grafo de estado**. Veja o grafo de estado abaixo para M:

1 d

1

0

DIA b

NOITE a

0 c

1

0

No grafo de estado acima, é possível perceber a relação através das arestas direcionadas cuja início tem um valor de entrada. Por exemplo, no estado *NOITE* a há uma aresta cujo início tem valor *1* d com fim conectado ao vértice *DIA* b, isso significa que se o valor de entrada for *1* e a entrada atual for *NOITE* para máquina M, então o próximo estado será *DIA.* Também é possível observar a , por exemplo o estado *DIA* b tem com saída *0* c.

# Exemplo prático – O Poste

Um exemplo prático pode ser elaborado em cima de um problema simples, o poste. Um poste de luz, desses que tem na rua, apaga durante a noite e acende durante o dia. Abstraindo dessa problemática para uma máquina de estados finitos, temos dois conjuntos:

* , que representa os **estados possíveis do poste**
* , que representa o nível lógico que a luz do poste pode assumir (0 para luz apagada e 1 para a luz acessa), ou seja, **conjunto de saída**

Nesse exemplo usaremos um *LDR (Light Dependent Resistor)* como sensor da luz solar. A saída desse sensor é analógica com resolução de 10bits, ou seja, conseguimos fazer uma leitura desse sensor com um conjunto de números inteiros de 0 a 1024. Quanto **maior luminosidade o LDR recebe, menor é o valor lido**. Obviamente, quanto mais escuro, maior será o valor lido. Daí temos um novo conjunto para nossa máquina de estado finito:

* , que representa o **conjunto de entrada**

Para concluir a nossa máquina temos que definir suas e . Como o valor de entrada vai de 0 a 1024, temos q determina uma ***constante de sensibilidade*** que servirá como indicador de luminosidade, ou seja, se o **valor de entrada for menor que ela**, então quer dizer que **está de dia**, do mesmo modo ao contrário, se **o valor de entrada for maior que ela, então estamos de noite**, daí temos a . Pensando na , como quando está de **noite**, o poste deve **acender**, então traduzindo para máquinas de estado finito, temos que **a saída do estado NOITE é 1**, o mesmo vale ao contrário, **quando o estado é DIA, a saída é 0 para a luz apagar**.

Para melhor compreensão, veja a tabela de estado abaixo:

|  |  |  |  |
| --- | --- | --- | --- |
| Estado atual | Próximo estado | | Saída |
| Entrada atual | |
|  |  |
| NOITE | NOITE | DIA | 1 |
| DIA | NOITE | DIA | 0 |

Tabela 2 - Tabela de estado do poste

## Implementação

Para implementar nossa máquina, utilizaremos um *Arduino UNO* que é uma plataforma de prototipagem eletrônica de hardware, pois tem fácil implementação e boa didática para esse exemplo.

### Eletrônica

Listagem de componentes:

* 1 - Arduino Uno
* 1 - LDR de 20KΩ
* 1 - Resistor de 1KΩ
* Pedaços de fio de cobre (jumpers)

Sendo que cada componente está conectado da seguinte forma:

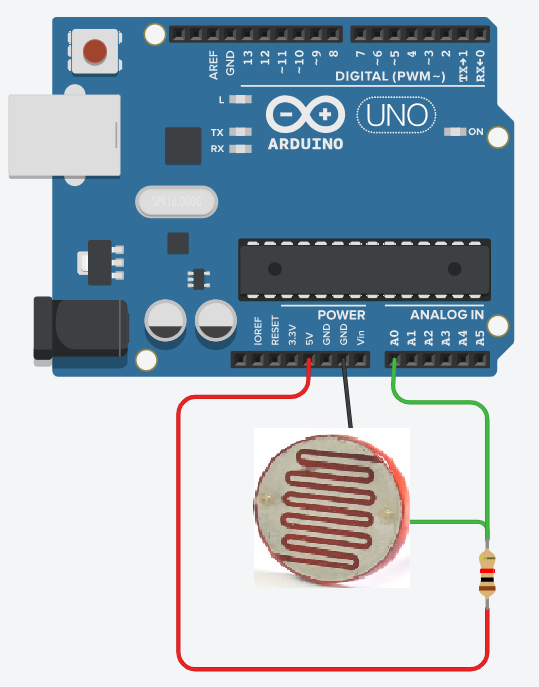


Figura 1 - Esquematização eletrônica na plataforma do Arduino

### Programação

O seguinte código na linguagem **C** utilizando a **IDE do Arduino** foi compilado no Arduino.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | typedef enum { NOITE,DIA } S;  S estadoAtual;  int entradaAtual;  int saida;  int cSensibilidade = 750;  void setup() {  // Inicia serial à 9600bps  Serial.begin(9600);  //Inicia estado atual como NOITE (S0)  estadoAtual = NOITE;  pinMode(LED\_BUILTIN, OUTPUT);  }  void loop() {  //Recebe entradaAtual da leitura analogica do sensor LDR  entradaAtual = analogRead(A0);  //atualizamos o novo estado com fs  estadoAtual = fs(estadoAtual, entradaAtual);  //pegamos a saida com fo  saida = fo(estadoAtual);  //enviamos a saida na serial  Serial.print(saida);  digitalWrite(LED\_BUILTIN, saida);    delay(10); // delay pra estabilidade  }  S fs(S estado, int entrada) {  if(entrada<cSensibilidade){  return DIA;  } else {  return NOITE;  }  }  int fo(S estado){  if(estado==NOITE) {  return 1;  }  if(estado==DIA) {  return 0;  }  } |

# Conclusão

Máquinas de estado finito operam de forma síncrona e determinística com capacidade de memória limitada. Porém esse modelo consegue resolver muitos dos problemas na engenharia, pois ele processa entradas junto com o estado e produz uma saída, além de que tendo uma máquina especificada com entradas é possível prever suas saídas, o que auxilia na construção de soluções como visto no exemplo do poste.

# Bibliografia

Bonato, V. **Elementos de Lógica Digital II**. Disponível em:  [http://wiki.icmc.usp.br/images/7/7d/Aula\_5\_-\_StateMachine.pdf/](https://fenix.tecnico.ulisboa.pt/downloadFile/3779571788876/) . Acesso: 26 de nov. 2018.

GERSTING, J.L. **Fundamentos Matemáticos para a Ciência da Computação**. 3 ed. LTC, 2010.

**Máquinas de Estado**. Disponível em: <https://fenix.tecnico.ulisboa.pt/downloadFile/3779571788876/> . Acesso: 25 de nov. 2018.

Vieira, N.J. **Introdução aos fundamentos da computação**, Editora Thomson. Cap. 2. Disponível em: <https://homepages.dcc.ufmg.br/~nvieira/cursos/tl/a17s2/livro/cap2.pdf> . Acesso: 25 de nov. 2018.