Trabalho de Linguagens Formais e Autômatos

Ariel Nogueira Kovaljski
<arielnogueirak@gmail.com>

Computer Engineering Course,
Instituto Politécnico (IPRJ) — Rio de Janeiro State University,
Rua Bonfim 25, Nova Friburgo, RJ 28625-570, Brazil

<dia-atual> de maio de 2021

Abstract

In this assignment we build and analyze the behavior and output of a Finite State Machine (FSM) and a Turing Machine (TM) for a given set of inputs.

Keywords: finite state machine, turing machine, finite automata

Resumo

Neste trabalho nós construímos e analisamos o comportamento e a saída de uma Máquina de Estado Finito (MEF) e uma Máquina de Turing (MT) para um dado conjunto de entradas.

Palavras-chave: máquina de estado finito, máquina de turing, autômatos finitos

1. Introdução

A teoria de autômatos trata do estudo de máquinas abstratas que seguem instruções pré-determinadas automaticamente.

<falar sobre níveis da hierarquia de chomsky>

A partir da hierarquia de Chomsky de gramáticas formais, é possível definir autômatos que são capazes de reconhecer linguagens formais pertencentes a distintas classes de gramática. Sendo assim, para uma linguagem formal possivelmente infinita, define-se um autômato como uma representação finita desta linguagem.

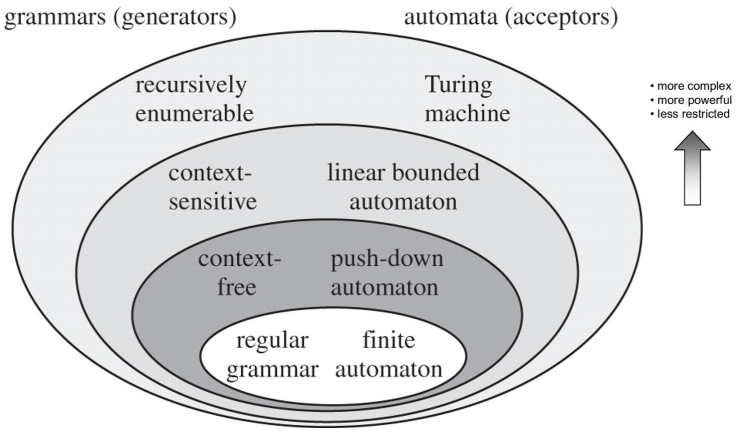

Figura 1: Hierarquia de Chomsky

Dentre os tipos de autômatos existentes, neste trabalho iremos abordar a construção e o funcionamento das *máquinas de estado finito* e das *máquinas de Turing*. Estes são responsáveis pelo reconhecimento das *gramáticas regulares* e *recusivamente enumeráveis*, respectivamente.

2. Teoria

2.1. Máquina de Estado Finito

<falar sobre gramática>

Uma máquina de estado finito (MEF) pode ser considerada como um modelo simplificado do funcionamento de um computador. Esta pode ser definida como uma quíntupla ordenada $M = (S, I, O, f_S, f_O)$ onde:

- S é o conjunto finito de estados;
- I é o conjunto finito de símbolos de entrada (alfabeto de entrada);
- O é o conjunto finito de símbolos de saída (alfabeto de saída);
- $f_S : S \times I \to S$ é uma função que retorna o próximo estado $s_{t_{k+1}} \in S$ dado o estado anterior $s_{t_k} \in S$ e um símbolo de entrada $i_{t_k} \in I$;
- $f_O : S \to O$ é a função *output*, que retorna o símbolo de saída $o_{t_k} \in O$ do estado atual $s_{t_k} \in S$.

As operações da MEF são sincronizadas por pulsos discretos de um relógio (*clock*) representados por t_k, onde $k \in \mathbb{N}_0$ representa o ciclo de *clock* atual. Partindo de um estado inicial $s_0 = s_{t_0}$, após um pulso de *clock*, a função f_S retornará um novo estado s_{t_1} dado a entrada anterior i_{t_0} e o estado anterior s_0. Generalizando para qualquer pulso de *clock*, é possível afirmar que a MEF possui um comportamento determinístico, ou seja, o novo estado sempre dependará do estado e entrada anteriores. Cada estado funciona como uma memória dos *inputs* anteriores, além de possuir um símbolo de saída, que é retornado pela função *output* f_O.

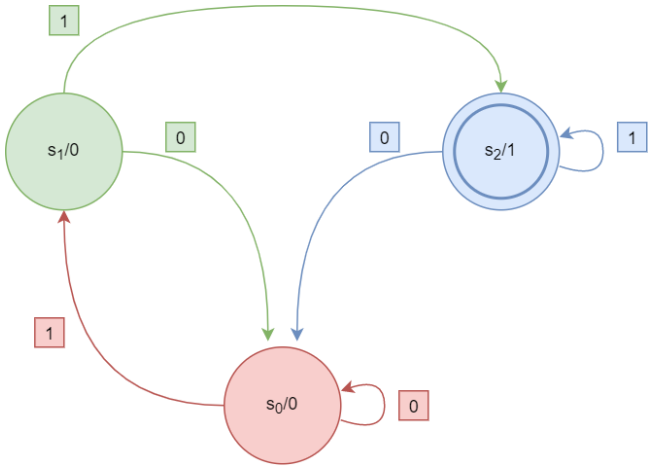

Figura 2: Exemplo de máquina de estado finito

Tomando como exemplo a MEF acima, começamos no estado s_0 e temos como saída o símbolo **0**. Aqui, há duas possibilidades: enquanto a entrada for **0**, permanecemos neste estado; caso a entrada seja **1**, seguimos para o estado s_1. Ao seguirmos para o estado s_1 temos como saída o símbolo **0**. Novamente, há duas possíveis entradas: caso a entrada seja **0**, retornamos ao estado s_0; caso a entrada seja **1**, seguimos para o estado s_2. Por fim, ao seguirmos para o estado s_2 temos como saída o símbolo **1**. As alternativas são: caso a entrada seja **0**, retornarmos ao estado s_0; enquanto a entrada seja **1**, permanecemos neste estado.

Esta forma um tanto verbosa de se descrever uma MEF pode ser colocada em uma tabela, com o estado atual, possíveis entradas e saídas como colunas da mesma.

Estado Atual	Próximo Estado		Saída
	0	**1**	
s_0	s_0	s_1	**0**
s_1	s_0	s_2	**0**
s_2	s_0	s_2	**1**

Nota-se que dentre todas as possíveis entradas, esta MEF só aceitará, isto é, terminará no estado final s_2, para *strings* de entrada que terminem com dois ou mais **1** (**11**, **111**, **1111**, ...).

2.2. Máquina de Turing

Máquinas de estado finito são capazes de processar apenas gramáticas do tipo 3: "gramática regular". Sendo assim é necessário o uso de outros tipos de autômatos para o processamento de gramáticas do tipo 2, 1 e 0: "livre de contexto", "sensível ao contexto" e "recursivamente enumerável", respectivamente.

Segundo a hierarquia de Chomsky, cada nível de gramática é um superconjunto da gramática anterior. Sendo assim, a gramática mais geral, a "recursivamente enumerável" com o seu respectivo autômato, a máquina de Turing, é

capaz de representar e processar linguagens formais de qualquer outro nível de gramática.

A máquina de Turing (MT) é o modelo mais geral do funcionamento de um computador. Considera-se uma fita de comprimento infinito que armazena os dados de entrada da MT. Cada dado ocupa uma célula da fita. Anexado a esta fita, há um cabeçote, que pode ler e escrever dados da fita sob a posição em que se encontra. Este pode mover-se para a esquerda e para direita apenas uma célula por vez. O cabeçote serve como dispositivo de entrada/saída da MT.

Para um conjunto finito de estados S e para um conjunto finito de símbolos da fita (alfabeto da fita) I, define-se uma MT como uma quíntupla ordenada $T = (s, i, i', s', d)$ onde:

- $s \in S$ é o estado da MT;
- $i \in I$ é um símbolo de entrada;
- $i' \in I$ é um símbolo de saída;
- $s' \in S$ é o novo estado da MT;
- $d \in \{L, R\}$ é direção de movimento do cabeçote.

Para cada dado i lido pela MT, dado o estado atual s, resultará em uma saída i', um novo estado s' e uma direção de movimento do cabeçote d. Nota-se que exceto pelo componente d, a MT é indêntica a MEF, mas devido a fita com capacidade de memória infinita e a possibilidade de ler e escrever e reler os dados da própria fita, a MT é capaz de processar gramáticas que seriam impossíveis em uma MEF.

3. Desenvolvimento

4. Conclusão

Aqui está a conclusão.

5. Referências Bibliográficas

Figura 1: Fitch, Tecumseh. 2014. *"Toward a computational framework for cognitive biology: Unifying approaches from cognitive neuroscience and comparative cognition"*