

FUNDAMENTOS TEÓRICOS DA COMPUTAÇÃO

-- AUTÔMATO FINITO DETERMINÍSTICO --

Conceitos Básicos

Autômatos

Autômato é usado para descrever uma “**máquina**” que funciona de modo **automático**

A palavra autômato vem da palavra grega αὐτόματα que significa “**comportamento automático**”, isto é, sem influência externa, ou que age por vontade própria

Os autômatos **existem** a muitos e muitos anos ...

Autômatos



Autômatos escritores

Autômatos



Autômato que toca piano

Autômatos



O autômato desenhista

Autômatos



Relógio Automático

Modelo Computacional

Já falamos que as **consequências** do isomorfismo entre sistemas formais e computadores digitais são **importantes** para a teoria da computação

“**Prova**” corresponde a “**computação**”. Portanto, os **limites demonstráveis da provabilidade** se traduzirão em limites demonstráveis da **computabilidade** e vice-versa

Qualquer coisa que possa ser **simulada** por um **computador** pode ser **simulada** por um **sistema formal** e vice-versa

Portanto, a tentativa de escrever **sistemas formais** para qualquer aspecto da **realidade** (por exemplo, geometria plana, teoria dos conjuntos, lógica de predicados, física de partículas, projeção de sombras...) é análogo à tentativa de escrever **programas de computador** que simulam determinados aspectos dessa mesma **realidade**

Modelo Computacional

Um **autômato finito** é definido como sendo um **modelo matemático** de uma **máquina de estados finitos**

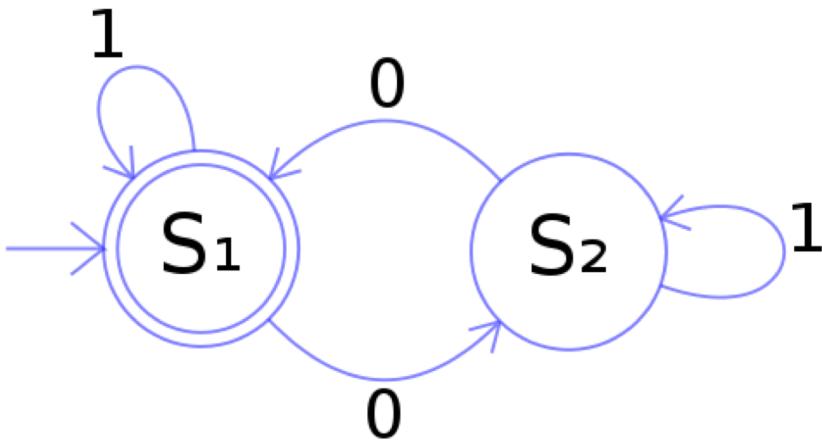
Os computadores são máquinas **complicadas** para se construir uma teoria matemática para representá-los completamente

O autômato tem limites de representatividade, mas pode-se utilizá-lo como um **computador idealizado** ou **modelo computacional**

Dependendo do **modelo computacional** pode-se modelar **diferentes** elementos do computador

O modelo mais **simples** é denominado **máquina de estados finitos** ou **autômato finito determinístico**

Autômato Finito Determinístico



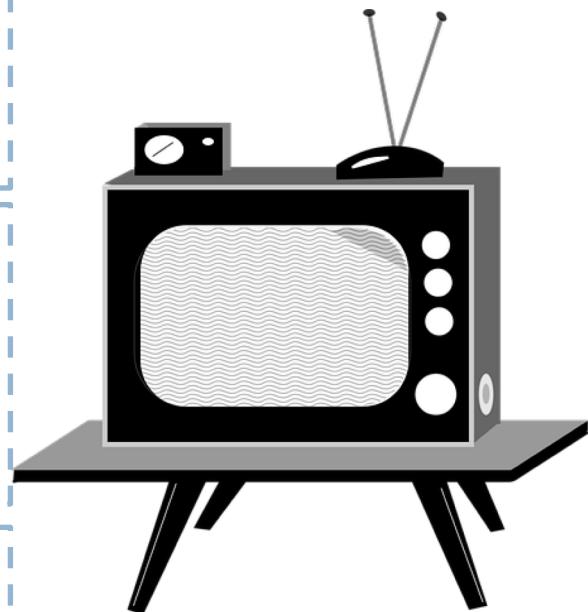
Um autômato consiste em **estados** (representados na figura por círculos), e **transições** (representadas por setas)

Quando o autômato recebe um símbolo de **entrada**, ele faz uma **transição** para outro estado, de acordo com sua **função de transição**, que tem como entradas o **estado atual** e o **símbolo** de entrada recente

Este autômato possui uma **memória bastante limitada**, i.e., baseia-se somente no estado atual

Exemplo de uma TV

- Um conceito fundamental nos autômatos é o conceito de **estado**
- Este conceito é aplicado a **qualquer sistema**, por exemplo, à uma **televisão**
- Uma televisão (simples) pode estar ligada (on) ou desligada (off), onde temos um sistema com **dois** estados
- Em um nível mais detalhado, podemos **diferenciar** os canais: **um** para desligada e os **restantes** significando ligada no **canal N**, existindo sempre um número **finito** de estados
- Apesar de podermos mudar a televisão de estado, em qualquer tempo, a televisão vai estar em **apenas um** dos estados possíveis



Exemplo de jogo de tabuleiro

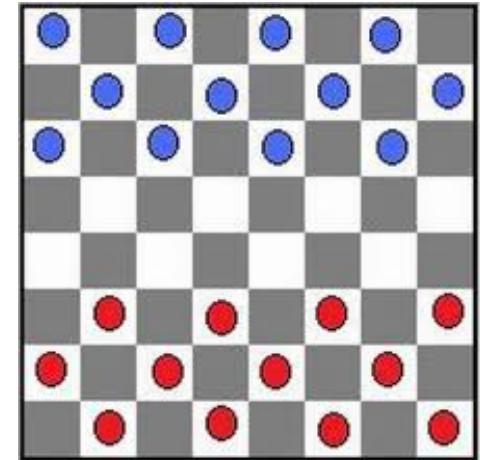
Um autômato parece muito com um jogo de tabuleiro em que cada espaço no tabuleiro representa um **estado**

Mover uma peça no tabuleiro é equivalente a uma **função de transição** de estados

Em um **novo estado** novas possibilidades de **abrem** e outras se **fecham**

Ou seja, em cada estado tem-se informações sobre **o que pode ser feito** quando uma entrada é recebida

Quando não há mais entradas, o autômato para e o estado onde está determina se o autômato **aceita** ou **rejeita** o conjunto de entradas





Teoria de Linguagens

Teoria de Linguagens

Teoria dos autômatos está profundamente relacionada à **teoria das linguagens formais** (compiladores)

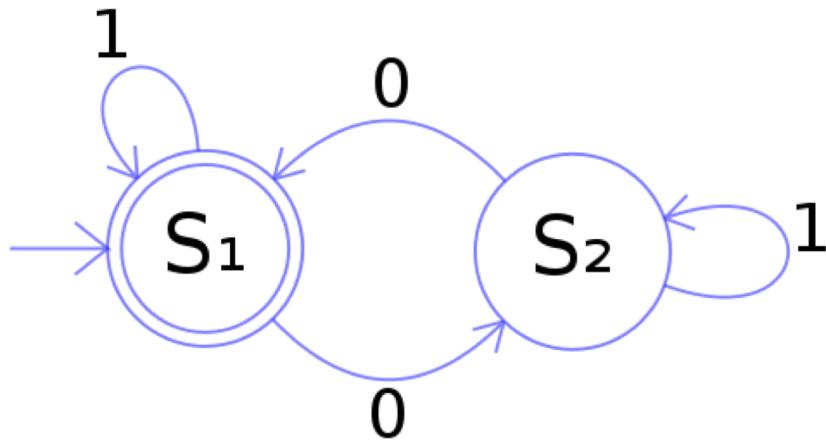
Um autômato é uma representação finita de uma **linguagem formal** que pode ser um **conjunto infinito**

Os problemas computacionais são **redutíveis** para o problema de aceitação/rejeição de palavras em uma linguagem formal

Autômatos são classificados pela **classe** das linguagens formais que são capazes de reconhecer (**hierarquia de Chomsky**)

Autômatos desempenham um **papel importante** em teoria da computação, prova de computabilidade, elaboração e otimização de compiladores, análise estática de códigos, verificação formal, recuperação de informação (web), etc

Teoria de Linguagens



Qual a linguagem formal que o autômato acima reconhece?

R1: Um número binário

R2: Sequência finita de lâmpadas ligadas e desligadas

R3: Disjuntores ligados e desligados

etc

Alfabetos e Cadeias

Um **alfabeto** é um conjunto finito de elementos chamados **símbolos**.

Geralmente representados por letras gregas maiúsculas

Exemplos: $\Sigma = \{0,1\}$, $\Sigma_1 = \{a, b, c, \dots, z\}$, $\Sigma_2 = \{a, b\}$, $\Gamma = \{\#, a_1, bd\}$

Símbolos com **mais de um caractere** são **incomuns**, como é o caso de Γ

Dado um alfabeto Σ , uma **cadeia** (sobre Σ) é uma sequência $x_1 x_2 \dots x_n$, onde $x_i \in \Sigma$, para cada $1 \leq i \leq n$, ou seja, é uma sequência finita de símbolos

Geralmente a cadeia é representada por letras gregas minúsculas

Outros termos são **strings** ou **palavras**

Cadeias

O **comprimento** de uma cadeia w é a quantidade de posições na sequência e é denotado por $|w|$

Exemplos

$w = 011011$ é a cadeia sobre $\Sigma = \{0,1\}$ e $|w| = 6$

$\alpha = \text{linguagem}$ é a cadeia sobre $\Sigma_1 = \{a, b, c, \dots, z\}$ e $|\alpha| = 9$

$\beta = bd\#a1bd$ é a cadeia sobre $\Gamma = \{\#, a_1, bd\}$ e $|\beta| = 4$

Se $x \in \Sigma$ e w é cadeia sobre Σ , denotamos por $|w|_x$ a quantidade de vezes que o símbolo x aparece em w

Exemplos

$$|\alpha|_g = 2$$

$$|w|_1 = 4$$

$$|\beta|_{bd} = 2$$

Concatenação de cadeias

A **concatenação** de uma cadeia $\alpha = \alpha_1\alpha_2 \dots \alpha_n$ com uma cadeia $\beta = \beta_1\beta_2 \dots \beta_m$ é a cadeia $\alpha\beta = \alpha_1\alpha_2 \dots \alpha_n \beta_1\beta_2 \dots \beta_m$

Denotamos por α^k a concatenação de α **com ela mesma** k vezes

Exemplos: $\alpha = aba$ $\beta = caa$

$\alpha\beta = abacaa$

$\beta\alpha = caaaba$

$\alpha^3 = abaabaaba$

$\alpha^2\beta^3 = abaabacaacaacaa$

Cadeias importantes

O **reverso** da cadeia $w = w_1 w_2 \dots w_n$

é a cadeia $w^R = w_n w_{n-1} \dots w_1$

A cadeia vazia, denotada por ε , é a cadeia de **comprimento zero**

Para **qualquer cadeia** w sobre Σ , $w\varepsilon = \varepsilon w = w$

Subcadeias

Uma cadeia β é **subcadeia** de outra cadeia w se existem cadeias α e γ tais que $w = \alpha\beta\gamma$

Exemplo

$\beta = 01$ é subcadeia de $w = 011011$ pois $w = \alpha\beta\gamma$ com $\alpha = \varepsilon$ e $\gamma = 1011$ (**ou** $\alpha = 011$ e $\gamma = 1$)

Potência de alfabeto

Dado um alfabeto Σ , denotamos Σ^k o conjunto de todas as cadeias de **comprimento k** sobre Σ

Se $\Sigma = \{0,1\}$, então $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \{0,1\}$, $\Sigma^2 = \{00,01,10,11\}$, $\Sigma^3 = \{000,001,010,011,100,101,110,111\}$

Obs: $|\Sigma^k| = |\Sigma|^k$

Fecho de Kleene e Fecho positivo

O **fecho de Kleene** de Σ , denotado por Σ^* , é o conjunto de todas as cadeias sobre Σ , isto é

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots = \bigcup_{i=0}^{\infty} \Sigma^i$$

O **fecho positivo** de Σ , denotado por Σ^+ , é o conjunto de todas as cadeias não vazias sobre Σ , isto é

$$\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i$$

Logo, $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Não esqueça que $\{\varepsilon\} \neq \{\} = \emptyset$

Fecho de Kleene e Fecho positivo

Seja $\Sigma = \{0,1,2\}$, então

$$\Sigma^* = \{\varepsilon, 0, 1, 2, 00, 01, 02, 10, 11, 12, 20, 21, 22, 000, 001, \dots\}$$

$$\Sigma^+ = \{0, 1, 2, 00, 01, 02, 10, 11, 12, 20, 21, 22, 000, 001, \dots\}$$

Observação

Agora, ao invés de escrever “**w é uma cadeia sobre o alfabeto Σ** ”, basta escrever “**w $\in \Sigma^*$** ”

Exemplo: $\Sigma^k = \{w \in \Sigma^* : |w| = k\}$

Linguagens

Uma **linguagem** sobre um alfabeto Σ é um subconjunto de Σ^* , geralmente representados por **letras maiúsculas**

“ L é uma linguagem sobre Σ ” equivale a “ $L \subseteq \Sigma^*$ ”

Exemplos:

$L = \{010, 11, 0, 1011\}$ é linguagem sobre $\Sigma = \{0,1\}$

$A = \{abacate, uva, cafe, banana, \varepsilon\}$ é linguagem sobre $\Sigma = \{a, b, \dots z\}$

$B = \{w \in \{x, y\}^* : |w| \text{ é par}\}$

Exemplos de linguagens

$L_1 = \{w \in \{0,1\}^*: |w|_0 = |w|_1\}$ é linguagem sobre $\Sigma = \{0,1\}$

$L_2 = \{w \in \{a,b\}^*: |w|_a \text{ é par}\}$

$L_3 = \{0^n 1^n : n \geq 1\}$

$L_4 = \{a^i b^j : 0 \leq i \leq j\}$

$L_5 = \{\varepsilon\}$

$L_6 = \{\}$

$L_7 = \{w \in \{0,1,2\}^* : o \text{ quinto elemento de } w \text{ é } 2\}$

$L_8 = \{w : w \text{ é um programa sintaticamente correto em } C\}$

Linguagens x Problemas

Dado $L \subseteq \Sigma^*$ e $w \in \Sigma^*$, w pertence a L ?

Qualquer problema pode ser expresso com linguagens

$P = \{w \in \{0,1,2,\dots,9\}^* : \text{se vista como número, } w \text{ é primo?}\}$

Ou seja, suponha que $w=137$, a pergunta é: **137 é primo?** Em outras palavras $w \in P$?

$Q = \{w\mathbf{0} : w \in \{0,1\}^*\}$

Ou seja, são as cadeias que **terminam em zero**. Neste caso, em outras palavras, se são números pares

$S = \{a^i b^j c^k : i, j, k \in \mathbb{Z}_{\geq 1} \text{ e } i + j = k\}$

Ou seja, pela cadeia **aabccc** dá para saber se $2 + 1 = 3$

Autômatos Finitos

São dispositivos **reconhecedores** de linguagens

Dado $w \in \Sigma^*$, **aceita** se $w \in L$ e **rejeita** caso contrário, sendo L a linguagem projetada para reconhecer

Possuem estados e transições entre os estados

A ideia é processar um símbolo por vez de w , começando no estado inicial, seguindo as transições e, eventualmente, **aceitar** ($w \in L$) ou **rejeitar** ($w \notin L$)

Referências

Newton José Vieira. Introdução aos Fundamentos da Computação. Editora Thompson, 2006.

SIPSER, M. Introdução à Teoria da Computação. São Paulo: Thomson Pioneira, 2007

MENEZES, P. B. Linguagens Formais e Autômatos. 6 ed. Porto Alegre: Artmed, 2011.

Paulo Blauth Menezes. Linguagens Recursivamente Enumeráveis e Sensíveis ao Contexto. Notas de aula. Disponível em:
<http://www.ic.uff.br/~ueverton/files/LF/aula08.pdf>.

**WHAT IF THE ENTIRE MICROSERVICES
IMPLEMENTATION**

WAS JUST A STATE MACHINE