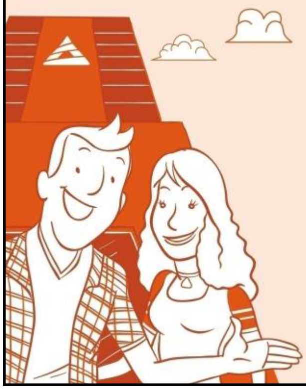

Teoria da Computação

Aula 03
Unidade Jundiaí

Anhanguera
Aqui o seu esforço ganha força.

Plano de Ensino

- Apresentação e Revisão
- Introdução à Teoria da Computação.
- Conceitos Básicos de Teoria da Computação.
- Programas.
- **Máquinas e Computações.**
- Modelos Computacionais.
- Máquinas Universais.
- Tese de Church.
- Máquina de Turing.

Livro-Texto

- Bibliografia Básica:
  - LEWIS, Harry R.; PAPADIMITRIOU, Christos H. **Elementos da Teoria da Computação. 2ª ed. Porto Alegre: Bookman, 2000.**
  - SIPSER, Michael. **Introdução à Teoria da Computação. 2ª ed. São Paulo: Cengage Learning, 2011.**

3. Máquinas – Introdução

- Uma máquina deve suprir todas as informações necessárias para que a computação de um programa possa ser descrita.
- A máquina deve suprir o significados dos identificadores de operações e testes.
- Cada identificador deve estar associado a uma transformação na estrutura de memória e a uma função verdade respectivamente. Note-se:
  - » Nem todo identificador de operação ou teste é definido em uma máquina.
  - » Para cada identificador de operação ou teste definido em uma máquina existe somente uma função associada.

3. Máquinas – Introdução

- **Definição:** uma Máquina **M** é uma 7-upla

$$M=(V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$$

  - » Onde:

V: conjunto de valores de Memória;

X: conjunto de valores de Entrada;

Y: conjunto de valores de Saída;

π_X: função de entrada tal que: $\pi_X: X \rightarrow V$;

π_Y: função de saída tal que: $\pi_Y: V \rightarrow Y$;

Π_O: conjunto de interpretações de operações onde, para cada identificador de operação F interpretado por M, existe uma única função: $\pi_Y: V \rightarrow V$ **em** Π_F;

Π_T: conjunto de interpretações de testes tal que, para cada identificador de teste T interpretado por M, existe uma única função: $\pi_T: V \rightarrow \{verdadeiro, falso\}$ **em** Π_T.

3. Máquinas – Introdução

- Máquina de 1 registrador M_{1reg} → suponha uma especificação de uma máquina com 1 registrador ***a*** o qual assume valores em $\mathbb{N}$, com duas operações e um teste, como segue:
  - » *Subtração de 1 em a, se a > 0;*
  - » *Adição de 1 em a;*
  - » *Teste se a é zero.*

3. Máquinas – Introdução

- Máquina $\mathbf{M_{1reg}}$

$$M_{1reg} = (\mathbb{N}, \mathbb{N}, \mathbb{N}, id_{\mathbb{N}}, id_{\mathbb{N}}, \{ad, sub\}, \{zero\})$$

- $\mathbb{N}$: corresponde ao conjunto de valores de memória, e/s;
- **$id_{\mathbb{N}}$**: $\mathbb{N} \rightarrow \mathbb{N}$ é a função de entrada e saída;
- **ad:** $\mathbb{N} \rightarrow \mathbb{N}$ é a interpretação tal que, $\forall n \in \mathbb{N}$, ad(n)=n+1;
- **sub:** $\mathbb{N} \rightarrow \mathbb{N}$ é a interpretação tal que, $\forall n \in \mathbb{N}$:
  sub(n)=n-1, n≠0; sub(n)=0, se n=0;
- **zero:** $\mathbb{N} \rightarrow$ {verdadeiro, falso} é a interpretação tal que, $\forall n \in \mathbb{N}$:
  zero(n)=verdadeiro, n=0; zero(n)=falso, se n≠0;

3. Máquinas – Introdução

- Máquina de 2 registradores M_{2reg} → suponha uma especificação de uma máquina com 2 registradores ***a*** e ***b*** os quais assumem valores em $\mathbb{N}$, com duas operações e um teste, como segue:
  - » *Subtração de 1 em a, se a > 0;*
  - » *Adição de 1 em b;*
  - » *Teste se a é zero.*

3. Máquinas – Introdução

- Máquina $\mathbf{M_{2reg}}$

$$M_{2reg} = (\mathbb{N}^2, \mathbb{N}, \mathbb{N}, \text{armazena\_a}, \text{retorna\_b}, \{\text{subtrai\_a}, \text{adiciona\_b}\}, \{\text{a\_zero}\})$$

- $\mathbb{N}^2$: corresponde ao conjunto de valores de memória
- $\mathbb{N}$: corresponde aos conjuntos de valores de entrada e saída, simultaneamente.
- **armazena_a**: $\mathbb{N} \rightarrow \mathbb{N}^2$ é a função de entrada tal que, $\forall n \in \mathbb{N}$: armazena_a(n)=(n,0).
- **retorna_b**: $\mathbb{N}^2 \rightarrow \mathbb{N}$ é a função de saída tal que, $\forall (n,m) \in \mathbb{N}^2$: retorna_b(n,m)=(m).
- **subtrai_a**: $\mathbb{N}^2 \rightarrow \mathbb{N}^2$ é interpretação tal que, $\forall (n,m) \in \mathbb{N}^2$: subtrai_a(n,m)=(n-1,m), se n≠0; subtrai_a(n,m)=(0,m), se n=0.
- **adiciona_b**: $\mathbb{N}^2 \rightarrow \mathbb{N}^2$ é interpretação tal que, $\forall (n,m) \in \mathbb{N}^2$: adiciona_b(n,m)=(n,m+1).
- **a_zero**: $\mathbb{N}^2 \rightarrow$ {verdadeiro, falso} é interpretação tal que, $\forall (n,m) \in \mathbb{N}^2$: a_zero(n,m)=verdadeiro, se n=0; a_zero(n,m)=falso, se n≠0.

3. Máquinas – Computações

- Uma computação é resumidamente um histórico do funcionamento da máquina para o programa, considerando um valor inicial.
- A computação pode ser realizada por um:
  - » Programa monolítico;
  - » Programa iterativo;
  - » Programa recursivo.

3. Máquinas – Computações

- **Definição 1:** Sejam $M=(V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ uma máquina e $P=(I, r)$ um programa monolítico para M onde L é o seu correspondente conjunto de rótulos. Uma Computação do Programa Monolítico P na Máquina M é uma cadeia (finita ou infinita) de pares L X V:

$$\mathbf{(s_0, v_0)(s_1, v_1)...}$$

  - » Onde (s_0, v_0) é tal que $s_0 = r$ é o rótulo inicial do programa *P* e v_0 é o valor inicial de memória e, para cada par (s_k, v_k) da cadeia, onde $k \in \{0, 1, 2, ...\}$, tem-se que:

3. Máquinas – Computações

a) Operação:
  - » Se s_k é o rótulo de uma operação da forma:
    s_k: faça F vá_para r' então **$(s_{k+1}, v_{k+1})=(r', \pi_F(v_k))$** é par subsequente de (s_k, v_k) na cadeia;
  - » Se s_k é o rótulo de uma operação da forma:
    s_k: faça ✓ vá_para r' então **$(s_{k+1}, v_{k+1})=(r', v_k)$** é par subsequente de (s_k, v_k) na cadeia.

b) Teste:
  - » Se s_k é o rótulo de um teste da forma:
    s_k: se T então vá_para r' senão vá_para r'' então (s_{k+1}, v_{k+1}) é par subsequente de (s_k, v_k) na cadeia sendo que $v_{k+1}=v_k$ e:
    - s_{k+1} = r' se $\pi_F(v_k)$=verdadeiro
    - s_{k+1} = r'' se $\pi_F(v_k)$= falso

3. Máquinas – Computações

- Note-se:
  - » Para um dado valor inicial de memória, a correspondente cadeia de computação é única, ou seja, é determinística;
  - » Um teste e a operação vazia não alteram o valor corrente da memória;
  - » Em uma computação infinita, rótulo algum da cadeia é final.

3. Máquinas – Computações

- **Exemplo**: computação finita de programa monolítico $\mathbf{P}_{\text{mon\_ab}}$ (*abaixo*) na máquina de 2 registradores $\mathbf{M}_{\text{2reg}}$ (*anterior*) e $\mathbf{v_0=(3,0)}$.

```
1:   se a_zero então vá_para 9 senão vá_para 2
2:   faça subtrai_a vá_para 3
3:   faça adiciona_b vá_para 1
```

(1, (3, 0))
(2, (3, 0))
(3, (2, 0))
(1, (2, 1))
(2, (2, 1))
(3, (1, 1))
(1, (1, 2))
(2, (1, 2))
(3, (0, 2))
(1, (0, 3))
(9, (0, 3))

3. Máquinas – Computações

- **Exemplo**: computação infinita de programa monolítico $\mathbf{P}_{\text{mon\_inf}}$ (*abaixo*) na máquina de 2 registradores $\mathbf{M}_{\text{2reg}}$ (*anterior*) e $\mathbf{v_0=(3,0)}$.

```
1:   faça adiciona_b vá_para 1
```

(1, (3, 0))
(1, (3, 1))
(1, (3, 2))
(1, (3, 3))
(1, (3, 4))
(1, (3, 5))
(1, (3, 6))
(1, (3, 7))
(1, (3, 8))
(1, (3, 9))
(1, (3, 10))
...

3. Máquinas – Computações

- **Definição 2:** Sejam M=(V, X, Y, π_X, π_Y, Π_O, Π_T) uma máquina e P um programa iterativo. Uma computação do programa iterativo P na máquina M é uma cadeia de elementos de I x V:

  $(i_0, v_0)\ (i_1, v_1)(i_2, v_2)$...

  » Onde *I* é um conjunto de programas iterativos, *i_0=P;*✓ e v_0 é o conteúdo inicial da memória de M.
- Essa cadeia indica a sequência de estados que serão assumidos pela máquina M durante a execução do programa P.
- Uma computação pode ser finita ou infinita.

3. Máquinas – Computações

- Os pares (i_{k+1}, v_{k+1}), $k > 0$, são obtidos a partir dos pares (i_k, v_k), a partir da análise do tipo de instrução inicial de i_k.
- Considere que U, W e Z são programa iterativos, F é identificador de operação e T é identificador de teste.
  - » i_k = ✓
    - a computação termina com valor v_k na memória.
  - » i_k = F; U
    - $(i_{k+1}, v_{k+1}) = (U, \pi_F(v_k))$

3. Máquinas – Computações

- » i_k = se T então U senão W;Z
  - *se $\pi_T(v_k)$ = verdadeiro,*
    $(i_{k+1}, v_{k+1}) = (U; Z, v_k)$
  - *se $\pi_T(v_k)$ = falso,*
    $(i_{k+1}, v_{k+1}) = (W; Z, v_k)$
- » i_k = enquanto T faça U; W
  - *se $\pi_T(v_k)$ = verdadeiro,*
    (i_{k+1}, v_{k+1}) = *(U; enquanto T faça U;W, v_k)*
  - *se $\pi_T(v_k)$ = falso,*
    $(i_{k+1}, v_{k+1}) = (W, v_k)$
- » i_k = até T faça U; W
  - *se $\pi_T(v_k)$ = falso,*
    (i_{k+1}, v_{k+1}) = *(U; até T faça U;W, v_k)*
  - *se $\pi_T(v_k)$ = verdadeiro,*
    $(i_{k+1}, v_{k+1}) = (W, v_k)$

3. Máquinas – Computações

- **Exemplo**: computação finita de programa iterativo $\mathbf{P}_{iter\_zera}$ (*abaixo*) na máquina de 2 registradores $\mathbf{M}_{2reg}$ (*anterior*) e $\mathbf{v_0=(2,0)}$.

```
até a_zero
  faça (subtrai_a; adiciona_b)
```

(até a_zero faça(subtrai_a; adiciona_b); ✓, (2,0))
(subtrai_a; adiciona_b; até a_zero faça(subtrai_a; adiciona_b); ✓, (2,0))
(adiciona_b; até a_zero faça(subtrai_a; adiciona_b); ✓, (1,0))
(até a_zero faça(subtrai_a; adiciona_b); ✓, (1,1))
(subtrai_a; adiciona_b; até a_zero faça(subtrai_a; adiciona_b); ✓, (1,1))
(adiciona_b; até a_zero faça(subtrai_a; adiciona_b); ✓, (0,1))
(até a_zero faça(subtrai_a; adiciona_b); ✓, (0,2))
(✓, (0,2))

3. Máquinas – Computações

- **Definição 3:** Sejam $M=(V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ uma *Máquina* e P um *Programa Recursivo* para M tal que:

  $$\mathbf{P} \textit{ é } \mathbf{E_0} \textit{ onde } \mathbf{R_1} \textit{ def } \mathbf{E_1, R_2} \textit{ def } \mathbf{E_2,\ldots,R_n} \textit{ def } \mathbf{E_n}$$

- Uma *Computação do Programa Recursivo* P *na Máquina* M é uma cadeia de pares da forma:

  $$\mathbf{(D_0, v_0)(D_1, v_1)(D_2, v_2)\ldots}$$

  - Onde:
    - (D_0, v_0) é tal que $D_0 = E_0$;✓ e v_0 é o valor inicial de memória;
    - Para cada par (D_k, v_k) da cadeia, onde $k \in \{0, 1, 2, \ldots\}$, tem-se que:

3. Máquinas – Computações

a) *Caso 1*. Se D_k é uma expressão de sub-rotina da forma:
  $\mathbf{D_k = (✓;C)}$
  - então $(D_{k+1}, v_{k+1}) = (C, v_k)$

b) *Caso 2*. Se D_k é uma expressão de sub-rotina da forma:
  $\mathbf{D_k = F;C}$
  - então $(D_{k+1}, v_{k+1}) = (C, p_F(v_k))$

c) *Caso 3*. Se D_k é uma expressão de sub-rotina da forma:
  $\mathbf{D_k = R_i;C}$
  - então $(D_{k+1}, v_{k+1}) = (E_i;C, v_k)$

d) *Caso 4*. Se D_k é uma expressão de sub-rotina da forma:
  $\mathbf{D_k =(C_1;C_2);C}$
  - então $(D_{k+1}, v_{k+1}) = (C_1;(C_2;C), v_k)$

e) *Caso 5*. Se D_k é uma expressão de sub-rotina da forma:
  $\mathbf{D_k = E_k}$ **= (se T então $\mathbf{C_1}$ senão $\mathbf{C_2}$);C**
  - então $(D_{k+1}, v_{k+1}) = (?, v_k)$

  $\mathbf{D_{k+1} = C_1;C \rightarrow}$ **se** $\mathbf{p_T(v_k)}$ **= verdadeiro**
  $\mathbf{D_{k+1} = C_2;C \rightarrow}$ **se** $\mathbf{p_T(v_k)}$ **= falso**

3. Máquinas – Computações

- **Exemplo**: computação finita de programa recursivo P_{rec_dup} (*abaixo*) na máquina de 1 registrador M_{1reg} (*anterior*) e v_0=(✓, 2).

```
Prec_dup é R onde
   R def (se zero então ✓ senão (sub;R;ad;ad))
```

(R; ✓, 2) //valor de entrada armazenado
((se zero então ✓ senão (sub; R; ad; ad)); ✓, 2)
((sub; R; ad; ad); ✓, 2) // como **n ≠ 0**, executa **senão**
(sub; (R; ad; ad); ✓, 2) // composicao sequencial
((R; ad; ad); ✓, 1) // **subtraiu 1** da memoria
(R; (ad; ad); ✓, 1) // composicao sequencial
((se zero então ✓ senão (sub; R; ad; ad)); (ad; ad); ✓, 1)

3. Máquinas – Computações

((sub; R; ad; ad); (ad; ad); ✓, 1) // como **n ≠ 0**, executa **senão**
(sub; (R; ad; ad); (ad; ad); ✓, 1) // composicao sequencial
((R; ad; ad); (ad; ad); ✓, 0) // **subtraiu 1** da memoria
(R; (ad; ad); (ad; ad); ✓, 0) // composicao sequencial
((se zero então ✓ senão(sub; R; ad; ad)); (ad; ad); (ad; ad); ✓, 0)
(✓; (ad; ad); (ad; ad); ✓, 0) // como **n = 0**, executa **então**
((ad; ad); (ad; ad); ✓, 0)
(ad; (ad; (ad; ad)); ✓, 0) // composicao sequencial
((ad; (ad; ad);); ✓, 1) // **adicionou 1** na memoria
(ad; ((ad; ad)); ✓, 1) // composicao sequencial
(((ad; ad)); ✓, 2) // **adicionou 1** na memoria
(ad; (ad); ✓, 2) // composicao sequencial
((ad); ✓, 3) // **adicionou 1** na memoria
(ad; ✓, 3) // composicao sequencial
(✓, 4) // **adicionou 1** na memoria e **fim**

Anhanguera
Aqui o seu esforço ganha força.

Teoria da Computação – Aula 03

Ciência da Computação

clayton.valdo@anhanguera.com