Máquinas e Computação

Teoria da Computação

INF05501

Programas e Máquinas

- Até agora, foram definidos três tipos de **programas**
- Entretanto, esses programas são **incapazes de descrever uma computação**, pois **não se tem a natureza das operações ou dos testes**, mas apenas um conjunto de identificadores
- Os tipos das operações e dos testes são especificados na definição de uma máquina em que o programa será executado

Requisitos de Máquinas

- Uma máquina deve prover as **informações necessárias para que a computação de um programa** possa ser descrita:
  - Cada identificador de operação deve representar uma **transformação na estrutura de memória** da máquina
  - Cada identificador de teste deve ser associado a uma **função booleana**
  - Para cada identificador de operação ou teste definido para uma máquina, existe **somente uma função associada** a este identificador
  - A forma de **armazenamento e recuperação de informações** da memória deve ser descrita

Definição Formal de Máquina

*Uma **máquina** é uma 7-tupla* $M = (V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$*, onde*

- V *é o conjunto de valores de memória*
- X *é o conjunto de valores de entrada*
- Y *é o conjunto de valores de saída*
- π_X *é uma função de entrada, tal que* $\pi_X : X \to V$
- π_Y *é uma função de saída, tal que* $\pi_Y : V \to Y$
- Π_F *é um conjunto de interpretações de operações, tal que, para cada identificador de operação* F *interpretado por* M*, existe uma função única* $\pi_F : V \to V$ *em* Π_F
- Π_T *é um conjunto de interpretações de testes, tal que, para cada identificador de teste* T *interpretado por* M*, existe uma função única* $\pi_T : V \to \{\texttt{verdadeiro}, \texttt{falso}\}$ *em* Π_T

Exemplo de Máquina

- Suponha uma especificação de uma máquina $dois\_reg$ com dois registradores a e b, os quais assumem valores em $\mathbb{N}$, com duas operações e um teste:
  - **Subtração** de 1 de a, se $a > 0$
  - **Adição** de 1 a b
  - **Teste** se a é 0

- Os **valores de entrada** são armazenados em a (zerando b) e a **saída** retorna o valor de b

Definição Formal de $dois\_reg$

$dois\_reg = (\mathbb{N}^2, \mathbb{N}, \mathbb{N}, armazena\_a, retorna\_b, \{subtrai\_a, adiciona\_b\}, \{a\_zero\})$, onde:

$armazena\_a : \mathbb{N} \to \mathbb{N}^2$, tal que, $\forall n \in \mathbb{N}$, $armazena\_a(n) = (n, 0)$

$retorna\_b : \mathbb{N}^2 \to \mathbb{N}$, tal que, $\forall (n, m) \in \mathbb{N}^2$, $retorna\_b(n, m) = m$

$subtrai\_a : \mathbb{N}^2 \to \mathbb{N}^2$, 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 $n = 0$

$adiciona\_b : \mathbb{N}^2 \to \mathbb{N}^2$, tal que, $\forall (n, m) \in \mathbb{N}^2$, $adiciona\_b(n, m) = (n, m + 1)$

$a\_zero : \mathbb{N}^2 \to \{\texttt{verdadeiro}, \texttt{falso}\}$, tal que, $\forall (n, m) \in \mathbb{N}^2$, $a\_zero(n, m) = \texttt{verdadeiro}$, se $n = 0$; $a\_zero(n, m) = \texttt{falso}$, se $n \neq 0$.

Programas para Máquinas

- Diz-se que P **é um programa para uma máquina** M se todo identificador de operação ou teste em P tiver uma função correspondente de operação ou teste em M

- **Mais formalmente:**

  Sejam $M = (V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$ *uma máquina e* P *um programa onde* P_F *e* P_T *são os conjuntos de identificadores de operações e de testes de* P*, respectivamente*

  P é um Programa para a máquina M se, e somente, se:

  – $\forall F \in P_F$, *existe uma única função* $\pi_F : V \to V$ *em* Π_F, *e*
  – $\forall T \in P_T$, *existe uma única função* $\pi_T : V \to \{\mathtt{verdadeiro}, \mathtt{falso}\}$ *em* Π_T

Programas para Máquinas (cont.)

- Note que é possível que **certos identificadores de operação ou teste não sejam interpretados em uma dada máquina**
- Por outro lado, uma **máquina** pode possuir **operações e/ou testes sem correspondência em um programa**
- A operação vazia ✓ é sempre interpretada em qualquer máquina

Exemplos de programas para Máquina $dois\_reg$

- Programa iterativo $itv\_b \leftarrow a$

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

- Programa recursivo $rec\_b \leftarrow a$

$rec\_b \leftarrow a$ é $\mathcal{R}$ onde

$\mathcal{R}$ def (se a_zero então $\checkmark$ senão $\mathcal{S}$;$\mathcal{R}$),

$\mathcal{S}$ def $subtrai\_a; adiciona\_b$

Computação

- Uma computação representa um **histórico** do quê é realizado por uma máquina para executar um programa para esta máquina

- Desta forma, uma **computação depende do tipo do programa e da definição da máquina** para a qual este programa foi criado

Computação de Programas Monolíticos em uma Máquina

- Uma computação de um programa monolítico em uma máquina é um **histórico das instruções** executadas e o **correspondente valor de memória**

- O histórico é representado na forma de uma **cadeia de pares**
  - **Cada par** define um estado possível da máquina ao executar o programa, descrevendo a **instrução a ser executada** e o **valor atual da memória**
  - A **cadeia de pares** define uma **possível sequência de estados a partir de um estado inicial**, o qual é definido pela instrução inicial e pelo valor de memória inicial

Definição Formal de Computação de Programa Monolítico em uma Máquina

Sejam $M = (V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$ *uma máquina e* $P = (I, r_0)$ *um programa monolítico para* M*, onde* L *é seu conjunto de rótulos*

Uma computação do programa monolítico P *na máquina* M *é uma cadeia de pares de* $L \times V$*:*

$$(s_0, v_0)(s_1, v_1)(s_2, v_2)...$$

onde

- (s_0, v_0) *representa o estado inicial, tal que* $s_0 = r_0$ *e* v_0 *é o valor inicial da memória*

Definição Formal de Computação de Programa Monolítico em uma Máquina (cont.)

- *Para cada par* (s_k, v_k) *da cadeia, onde* $k \in \{0, 1, 2, ...\}$, *supondo-se que* F *é um identificador de operação,* T *é um identificador de teste e* r', r'' *são rótulos de* L, *tem-se uma computação correspondente da seguinte forma:*

  Se s_k : `faça` F `vá_para` r', então $(s_{k+1}, v_{k+1}) =$

  Se s_k : `faça` ✓ `vá_para` r', então $(s_{k+1}, v_{k+1}) =$

  Se s_k : `se` T `então vá_para` r' `senão vá_para` r'', então $(s_{k+1}, v_{k+1}) =$

Definição Formal de Computação de Programa Monolítico em uma Máquina (cont.)

- *Para cada par* (s_k, v_k) *da cadeia, onde* $k \in \{0, 1, 2, ...\}$*, supondo-se que* F *é um identificador de operação,* T *é um identificador de teste e* r'*,* r'' *são rótulos de* L*, tem-se uma computação correspondente da seguinte forma:*

Se s_k : $\quad$ `faça` F `vá_para` r', então $(s_{k+1}, v_{k+1}) = (r', \pi_F(v_k))$

Se s_k : $\quad$ `faça` $\checkmark$ `vá_para` r', então $(s_{k+1}, v_{k+1}) =$

Se s_k : $\quad$ `se` T `então vá_para` r' `senão vá_para` r'', então $(s_{k+1}, v_{k+1}) =$

Definição Formal de Computação de Programa Monolítico em uma Máquina (cont.)

- *Para cada par (s_k, v_k) da cadeia, onde $k \in \{0, 1, 2, ...\}$, supondo-se que F é um identificador de operação, T é um identificador de teste e r', r'' são rótulos de L, tem-se uma computação correspondente da seguinte forma:*

Se s_k: `faça` F `vá_para` r', então $(s_{k+1}, v_{k+1}) = (r', \pi_F(v_k))$

Se s_k: `faça` ✓ `vá_para` r', então $(s_{k+1}, v_{k+1}) = (r', v_k)$

Se s_k: `se` T `então vá_para` r' `senão vá_para` r'', então $(s_{k+1}, v_{k+1}) =$

Definição Formal de Computação de Programa Monolítico em uma Máquina (cont.)

- *Para cada par* (s_k, v_k) *da cadeia, onde* $k \in \{0, 1, 2, ...\}$*, supondo-se que* F *é um identificador de operação,* T *é um identificador de teste e* r'*,* r'' *são rótulos de* L*, tem-se uma computação correspondente da seguinte forma:*

Se s_k : $\texttt{faça}$ F $\texttt{vá\_para}$ r', então $(s_{k+1}, v_{k+1}) = (r', \pi_F(v_k))$

Se s_k : $\texttt{faça}$ $\checkmark$ $\texttt{vá\_para}$ r', então $(s_{k+1}, v_{k+1}) = (r', v_k)$

Se s_k : $\texttt{se}$ T $\texttt{então vá\_para}$ r' $\texttt{senão vá\_para}$ r'', então $(s_{k+1}, v_{k+1}) = (?, v_k)$, sendo que

$s_{k+1} = r'$ se $\pi_T(v_k) = \texttt{verdadeiro}$

$s_{k+1} = r''$ se $\pi_T(v_k) = \texttt{falso}$

Computações de Programas Monolíticos

- Uma computação é dita finita se a **cadeia que a define é finita** e é dita infinita, caso contrário
- Em uma computação infinita, não existe rótulo final
- Para um **dado valor inicial de memória**, a correspondente **cadeia de computação é única**, ou seja, a computação é determinística
- Um teste não altera o valor corrente da memória

Exemplo de Computação Monolítica 1

- Dado o programa monolítico $mon\_b \leftarrow a$ para a máquina $dois\_reg$

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

como seria a computação finita para um valor de memória inicial $(3, 0)$?

Exemplo de Computação Monolítica 1 (cont.)

$(1,(3,0))$	instrução inicial e valor armazenado
$(2,(3,0))$	como $a \neq 0$, desvia para 2
$(3,(2,0))$	subtrai 1 de a e desvia para 3
$(1,(2,1))$	adiciona 1 a b e desvia para 1
$(2,(2,1))$	como $a \neq 0$, desvia para 2
$(3,(1,1))$	subtrai 1 de a e desvia para 3
$(1,(1,2))$	adiciona 1 a b e desvia para 1
$(2,(1,2))$	como $a \neq 0$, desvia para 2
$(3,(0,2))$	subtrai 1 de a e desvia para 3
$(1,(0,3))$	adiciona 1 a b e desvia para 1
$(9,(0,3))$	como $a = 0$, desvia para 9

Exemplo de Computação Monolítica 2

- Dado programa monolítico $comp\_infinita$ para a máquina $dois\_reg$

```
1 : faça adiciona_b vá_para 1
```

como seria a computação infinita para um valor de memória inicial $(3, 0)$?

Exemplo de Computação Monolítica 2 (cont.)

$(1, (3, 0))$	instrução inicial e valor armazenado
$(1, (3, 1))$	adiciona 1 a b e desvia para 1
$(1, (3, 2))$	adiciona 1 a b e desvia para 1
$(1, (3, 3))$	adiciona 1 a b e desvia para 1
...	repete 1 indefinidamente

Computação de Programas Recursivos em uma Máquina

- Uma computação de um programa recursivo em uma máquina é **semelhante à de um programa monolítico**

- O histórico também é representado na forma de uma **cadeia de pares**
  - **Cada par** define um estado possível da máquina ao executar o programa, descrevendo a **expressão de sub-rotina a ser executada** e o **valor atual da memória**
  - Assim como para programas monolíticos, a **cadeia de pares** descreve uma **possível sequência de estados a partir de um estado inicial**

Computações de Programas Recursivos

- Em uma **computação finita**, a **expressão** ✓ ocorre no **último par da cadeia** e **não ocorre em qualquer outro par**

- Em uma computação infinita, não existe a expressão ✓ em par algum da cadeia

- Para um **dado valor inicial de memória**, a correspondente **cadeia de computação é única**, ou seja, a computação é determinística

- Um teste ou uma referência a uma sub-rotina não alteram o valor corrente da memória

Definição Formal de Computação de Programa Recursivo em uma Máquina

Sejam $M = (V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$ *uma máquina e* P *um programa recursivo para* M*, tal que*

$$P \text{ é } E_0 \texttt{ onde } \mathcal{R}_1 \texttt{ def } E_1\texttt{, } \mathcal{R}_2 \texttt{ def } E_2\texttt{, ..., } \mathcal{R}_n \texttt{ def } E_n$$

Uma computação do programa recursivo P *na máquina* M *é uma cadeia de pares da forma:*

$$(D_0, v_0)(D_1, v_1)(D_2, v_2)...$$

onde

- (D_0, v_0) *representa o estado inicial, tal que* $D_0 = E_0$; ✓ *e* v_0 *é o valor inicial da memória*

- *Para cada par (D_k, v_k) da cadeia, onde $k \in \{0, 1, 2, ...\}$, supondo-se que F é um identificador de operação, T é um identificador de teste e C, C' e C'' são expressões de sub-rotina, tem-se uma computação correspondente da seguinte forma:*

Caso 1. *Se* $D_k = (\checkmark; C)$, *então* $(D_{k+1}, v_{k+1}) = (C, v_k)$

Caso 2. *Se* $D_k = F; C$, *então* $(D_{k+1}, v_{k+1}) = (C, \pi_F(v_k))$

Caso 3. *Se* $D_k = \mathcal{R}_i; C$, *para* $i \in \{0, 1, 2, ...\}$, *então* $(D_{k+1}, v_{k+1}) = (E_i; C, v_k)$

Caso 4. *Se* $D_k = (C'; C''); C$, *então* $(D_{k+1}, v_{k+1}) = (C'; (C''; C), v_k)$

Caso 5. *Se* $D_k = E_k = (\texttt{se}\ T\ \texttt{então}\ C'\ \texttt{senão}\ C''); C$, *então* $(D_{k+1}, v_{k+1}) = (?, v_k)$, *sendo que*
$D_{k+1} = C'; C$ *se* $\pi_T(v_k) = \texttt{verdadeiro}$
$D_{k+1} = C''; C$ *se* $\pi_T(v_k) = \texttt{falso}$

Exemplo de Computação Recursiva 1

- Dado um programa recursivo $qq\_maquina$ para uma máquina qualquer

$$
\begin{array}{l}
qq\_maquina \text{ é } \mathcal{R} \text{ onde} \\
\qquad\qquad \mathcal{R} \text{ def } \mathcal{R}
\end{array}
$$

para qualquer valor inicial de memória, a **computação** correspondente é **infinita**:

$$(\mathcal{R}; \checkmark, v_0)(\mathcal{R}; \checkmark, v_0)(\mathcal{R}; \checkmark, v_0)\ldots$$

Exemplo de Computação Recursiva 2

- Para um dado conjunto A, a função identidade em A é aquela que associa cada elemento do conjunto a si próprio, ou seja:

$$id_A : A \to A$$

tal que, $\forall a \in A, id_A(a) = a$

Exemplo de Computação Recursiva 2 (cont.)

- Considere a definição da máquina de um registrador um_reg:

$um\_reg = (\mathbb{N}, \mathbb{N}, \mathbb{N}, id_{\mathbb{N}}, id_{\mathbb{N}}, \{add, sub\}, \{zero\})$, onde:

$id_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ é a função identidade em $\mathbb{N}$

$ad : \mathbb{N} \to \mathbb{N}$, tal que, $\forall n \in \mathbb{N}$, $ad(n) = n + 1$

$sub : \mathbb{N} \to \mathbb{N}$, tal que, $\forall n \in \mathbb{N}$, $sub(n) = n - 1$, se $n \neq 0$; $sub(n) = 0$, se $n = 0$

$zero : \mathbb{N} \to \{\texttt{verdadeiro}, \texttt{falso}\}$, tal que, $\forall n \in \mathbb{N}$, $zero(n) =$ $\texttt{verdadeiro}$, se $n = 0$; $zero(n) = \texttt{falso}$, se $n \neq 0$.

Exemplo de Computação Recursiva 2 (cont.)

- Considere o programa recursivo $duplica$ abaixo:

$duplica$ `é` $\mathcal{R}$ `onde`

$\mathcal{R}$ `def (se` $zero$ `então` $\checkmark$ `senão` $(sub; \mathcal{R}; ad; ad)$`)`

- Qual é a computação finita para um valor de memória inicial igual a 3?

$(\mathcal{R}; \checkmark, 3)$ → **valor de entrada armazenado**

$(\texttt{se}\ zero\ \texttt{então}\ \checkmark\ \texttt{senão}\ (sub; \mathcal{R}; ad; ad)); \checkmark, 3)$ → **caso 3**

$((sub; \mathcal{R}; ad; ad); \checkmark, 3)$ → **como** $n \neq 0$**, executa senão**

$(sub; (\mathcal{R}; ad; ad); \checkmark, 3)$ → **caso 4**

$((\mathcal{R}; ad; ad); \checkmark, 2)$ → **subtrai 1 da memória**

$(\mathcal{R}; (ad; ad); \checkmark, 2)$ → **caso 4**

$(((\texttt{se}\ zero\ \texttt{então}\ \checkmark\ \texttt{senão}\ (sub; \mathcal{R}; ad; ad)); (ad; ad); \checkmark, 2)$ → **caso 3**

$((sub; \mathcal{R}; ad; ad); (ad; ad); \checkmark, 2)$ → **como** $n \neq 0$**, executa senão**

$(sub; (\mathcal{R}; ad; ad); (ad; ad); \checkmark, 2)$ → **caso 4**

$((\mathcal{R}; ad; ad); (ad; ad); \checkmark, 1)$ → **subtrai 1 da memória**

$(\mathcal{R}; (ad; ad); (ad; ad); \checkmark, 1)$ → **caso 4**

$((\texttt{se}\ zero\ \texttt{então}\ \checkmark\ \texttt{senão}\ (sub; \mathcal{R}; ad; ad)); (ad; ad); (ad; ad); \checkmark, 1)$ → **caso 3**

$((sub; \mathcal{R}; ad; ad); (ad; ad); (ad; ad); \checkmark, 1)$ → **como** $n \neq 0$**, executa senão**

$(sub; (\mathcal{R}; ad; ad); (ad; ad); (ad; ad); \checkmark, 1)$ → **caso 4**

$((\mathcal{R}; ad; ad); (ad; ad); (ad; ad); \checkmark, 0)$ → **subtrai 1 da memória**

$(\mathcal{R}; (ad; ad); (ad; ad); (ad; ad); \checkmark, 0)$ → **caso 4**

$\langle$(se $zero$ então $\checkmark$ senão $(sub; \mathcal{R}; ad; ad)); (ad; ad); (ad; ad); (ad; ad); \checkmark, 0\rangle$
$\rightarrow$ **caso 3**
$\langle \checkmark; (ad; ad); (ad; ad); (ad; ad); \checkmark, 0\rangle \rightarrow$ **como $n = 0$, executa então**
$\langle (ad; ad); (ad; ad); (ad; ad); \checkmark, 0\rangle \rightarrow$ **caso 1**
$\langle ad; (ad; (ad; ad); (ad; ad); \checkmark), 0\rangle \rightarrow$ **caso 4**
$\langle (ad; (ad; ad); (ad; ad); \checkmark), 1\rangle \rightarrow$ **adiciona 1 à memória**
$\langle ad; ((ad; ad); (ad; ad); \checkmark), 1\rangle \rightarrow$ **caso 4**
$\langle ((ad; ad); (ad; ad); \checkmark), 2\rangle \rightarrow$ **adiciona 1 à memória**
$\langle ad; (ad; (ad; ad); \checkmark), 2\rangle \rightarrow$ **caso 4**
$\langle (ad; (ad; ad); \checkmark), 3\rangle \rightarrow$ **adiciona 1 à memória**
$\langle ad; ((ad; ad); \checkmark), 3\rangle \rightarrow$ **caso 4**
$\langle ((ad; ad); \checkmark), 4\rangle \rightarrow$ **adiciona 1 à memória**
$\langle ad; ((ad); \checkmark), 4\rangle \rightarrow$ **caso 4**
$\langle (ad; \checkmark), 5\rangle \rightarrow$ **adiciona 1 à memória**
$\langle ad; ((\checkmark), 5\rangle \rightarrow$ **caso 4**
$\langle (\checkmark), 6\rangle \rightarrow$ **adiciona 1 à memória**
$\langle \checkmark, 6\rangle \rightarrow$ **fim da recursão**

Exemplo de Computação Recursiva 2 (cont.)

- Observe que o programa $duplica$ termina com o **valor de memória igual ao dobro do valor inicial**

- Note também que, com o uso de recursão, não foi preciso usarem-se dois registros (um para controlar o ciclo e outro para cálculos)

Função Computada

- Vimos que **programas** são definidos para executarem sobre uma **máquina** e que um histórico da execução de um programa sobre uma máquina representa uma **computação**
- Tal computação deve ser sempre associada a uma **entrada** (valor inicial) e à **saída** correspondente (resultado da computação)
- Preferencialmente, quer-se obter o resultado de uma computação em um **tempo finito**
- As ideias de entrada e saída e de tempo de resposta levam à definição de função computada

Função Computada por Um Programa Monolítico sobre uma Máquina

- A **computação inicia** na instrução identificada pelo **rótulo inicial**, com a **memória contendo o valor inicial resultante da aplicação da função de entrada** sobre o dado fornecido

- São executados, passo-a-passo, testes e operações, na ordem determinada pelo programa

- A **computação termina** quando um **rótulo final** é atingido

- O valor da função computada pelo programa é aquele resultante da **aplicação da função de saída ao valor da memória quando da parada**

Definição Formal

Sejam $M = (V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$ *uma máquina e* P *um programa monolítico para* M*, a função computada por* P *em* M*, denotada por:*

$$\langle P, M \rangle : X \to Y$$

é uma ***função parcial definida para*** $x \in X$ *se a cadeia* $(s_0, v_0)(s_1, v_1)...(s_n, v_n)$ *é uma computação finita de* P *em* M*, onde:*

- *O valor inicial da memória é* $v_0 = \pi_X(x)$
- *A imagem de* x *é dada por* $\langle P, M \rangle(x) = \pi_Y(v_n)$

Exemplo de Função Computada $\langle mon\_b \leftarrow a, dois\_reg \rangle$

- Considere o programa monolítico $mon\_b \leftarrow a$ para a máquina $dois\_reg$

- A função computada correspondente é a **função identidade**

$$\langle mon\_b \leftarrow a, dois\_reg \rangle : \mathbb{N} \to \mathbb{N}$$

tal que, $\forall n \in \mathbb{N}$, tem-se que

$$\langle mon\_b \leftarrow a, dois\_reg \rangle (n) = n$$

Exemplo de Função Computada $\langle mon\_b \leftarrow a, dois\_reg \rangle$

- Dessa forma, dado o valor de entrada 3:
  - $\pi_X(3) = (3, 0)$
  - $\langle mon\_b \leftarrow a, dois\_reg \rangle(3) = \pi_Y(0, 3) = 3$
  - Portanto, a **função** $\langle mon\_b \leftarrow a, dois\_reg \rangle$ **é definida para o valor** 3

Exemplo de Função Computada $\langle comp\_infinita, dois\_reg \rangle$

- Considere o programa monolítico $comp\_infinita$ para a máquina $dois\_reg$
- A função computada correspondente é

$$\langle comp\_infinita, dois\_reg \rangle : \mathbb{N} \to \mathbb{N}$$

- Dessa forma, dado o valor de entrada 3:
  - $\pi_X(3) = (3, 0)$
  - Como a cadeia da computação é infinita, a **função computada <u>não</u> é definida para o valor de entrada** 3

Função Computada por Um Programa Recursivo sobre uma Máquina

- A **computação inicia** na **expressão inicial**, com a **memória contendo o valor inicial resultante da aplicação da função de entrada** sobre o dado fornecido

- São executados, passo-a-passo, testes e operações, na ordem determinada pelo programa

- A **computação termina** quando a **expressão de sub-rotina** resultante for a **expressão vazia**

- O valor da função computada pelo programa é aquele resultante da **aplicação da função de saída ao valor da memória quando da parada**

Definição Formal

Sejam $M = (V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$ *uma máquina e* P *um programa recursivo para* M*, a função computada por* P *em* M*, denotada por:*

$$\langle P, M \rangle : X \to Y$$

*é uma **função parcial definida para*** $x \in X$ *se a cadeia* $(D_0, v_0)(D_1, v_1)...(D_n, v_n)$ *é uma computação finita de* P *em* M*, onde:*

- $D_0 = E_0$; ✓ é a expressão inicial de P
- *O valor inicial da memória é* $v_0 = \pi_X(x)$
- $E_n = \checkmark$, *o que resulta que* $\langle P, M \rangle(x) = \pi_Y(v_n)$

Exemplo de Função Computada $\langle qq\_maquina, M\rangle$

- Considere o programa recursivo $qq\_maquina$ e uma máquina M qualquer
- A função computada correspondente é $\langle qq\_maquina, M\rangle : X \to Y$
- Esta função é **indefinida para qualquer entrada**

Exemplo de Função Computada $\langle duplica, um\_reg \rangle$

- Considere o programa recursivo $duplica$ na máquina um_reg
- A função computada correspondente é $\langle duplica, um\_reg \rangle : \mathbb{N} \to \mathbb{N}$
- Para esta função, para qualquer $n \in \mathbb{N}$,

$$\langle duplica, um\_reg \rangle(n) = 2n$$

Exercício

1. Apresente como seria a definição formal da computação de um programa iterativo em uma máquina