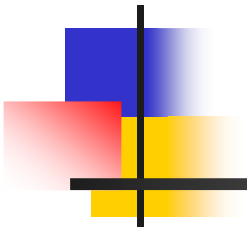
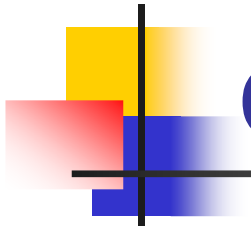


Teoria da Computação

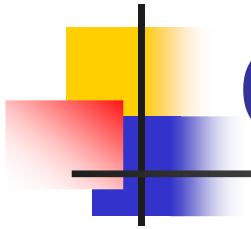
Unidade 1 – Conceitos Básicos

Referência – Teoria da Computação (Divério, 2000)



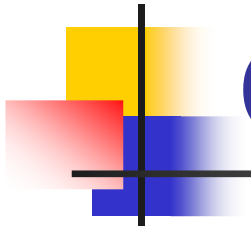
Conceitos Básicos

- Linguagem
  - Conceito fundamental
    - Forma precisa de expressar problemas
    - Permite um desenvolvimento formal adequado ao estudo da computabilidade
    - Será apresentada a solucionabilidade de um problema, analisando-a como a investigação da existência de um algoritmo que determine se uma palavra pertence ou não à linguagem que traduz um problema



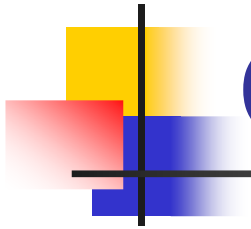
Conceitos Básicos

- Alfabeto
  - É um conjunto finito de *símbolos* ou *caracteres*
    - *Conjunto infinito não é um alfabeto*
    - *Conjunto vazio é um alfabeto*



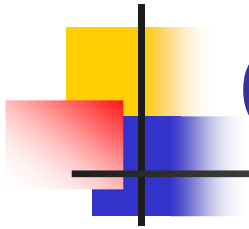
Conceitos Básicos

- Alfabeto
  - Exemplo de alfabeto:
    - {a,b,c}
    - ∅ (conjunto vazio)
  - Não são exemplos de alfabeto:
    - N (conjunto dos números naturais)
    - {a,b,aa, ab, ba, bb, aaa,...}



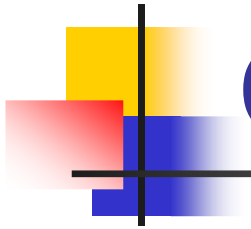
Conceitos Básicos

- Cadeia de símbolos
  - Uma *Cadeia de Símbolos* sobre um conjunto é uma sequência de zero ou mais símbolos (do conjunto) justapostos



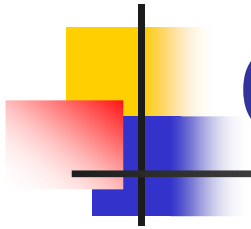
Conceitos Básicos

- Palavra
  - É uma *Cadeia de Símbolos* finita
    - Uma cadeia sem símbolos é uma palavra válida
    - ε representa uma cadeia vazia ou palavra vazia
    - Σ representa um alfabeto
      - Σ^* conjunto de todas as palavras possíveis sobre Σ
      - Σ^+ denota Σ^* **- { ε }**



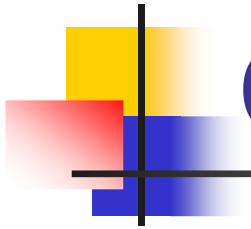
Conceitos Básicos

- Palavra
  - abc é uma palavra sobre o alfabeto {a, b, c}
  - Se Σ = {a, b}, então:
    - Σ^+ = { a, b, aa, ab, ba, bb, aaa, ... }
    - Σ^* = { ε, a, b, aa, ab, ba, bb, aaa, ... }



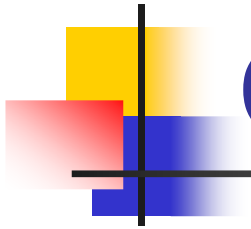
Conceitos Básicos

- Comprimento ou Tamanho de uma Palavra
  - Número de símbolos que compõem a palavra
    - Exemplos:
      - **|W|** Comprimento da palavra **W**
      - |abcd| = 4
      - | ε | = 0



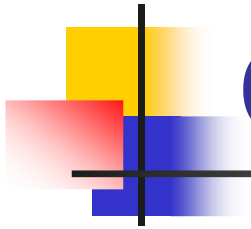
Conceitos Básicos

- Prefixo, Sufixo
  - ***Prefixo*** de uma palavra é qualquer sequência inicial de símbolos de uma palavra
  - ***Sufixo*** é qualquer sequência final de símbolos de uma palavra
    - Relativamente à palavra **abcb**, tem-se que:
      - ε, a, ab, abc, abcb são os *prefixos*
      - ε, b, cb, bcb, abcb são os respectivos *sufixos*



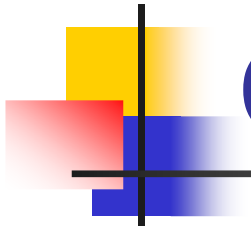
Conceitos Básicos

- Subpalavra
  - *Subpalavra* de uma palavra é qualquer sequência de símbolos contígua da palavra
    - Qualquer prefixo ou sufixo de uma palavra é uma subpalavra



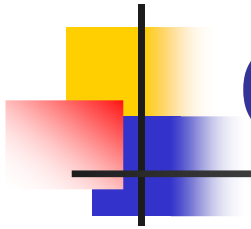
Conceitos Básicos

- Linguagem Formal (Linguagem)
  - É o conjunto de palavras sobre um alfabeto
    - **Dado o alfabeto** Σ **=** {a, b}. Então
      - O conjunto de palíndromos sobre Σ é um exemplo de uma linguagem infinita
        - ε, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, ... são palavras desta linguagem



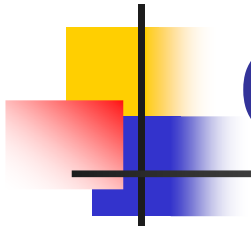
Conceitos Básicos

- Concatenação de Palavras (Concatenação)
  - É uma operação binária, definida sobre uma linguagem
  - Associa a cada par de palavras uma palavra formada pela justaposição da primeira com a segunda
  - Denotada pela justaposição dos símbolos que representam as palavras componentes



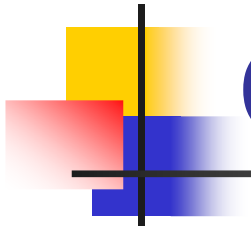
Conceitos Básicos

- Concatenação de Palavras
  - Satisfaz às seguintes propriedades:
  - Suponha v, w, t palavras
    - A) Associatividade
      - $v(wt) = (vw)\,t$
    - B) Elemento neutro à Esquerda e à Direita
      - $\varepsilon w = w = w\varepsilon$



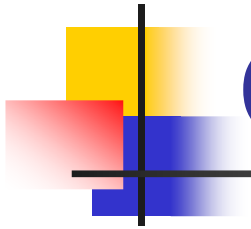
Conceitos Básicos

- Concatenação de Palavras
  - Suponha o alfabeto $\Sigma = \{a, b\}$
    - Então para as palavras v = baaaa e w = bb, tem-se que:
      - vw = baaaabb
      - $v\varepsilon$ = v = baaaa



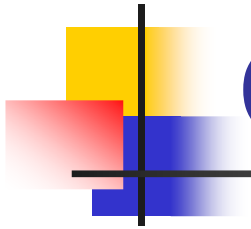
Conceitos Básicos

- Concatenação Sucessiva de uma Palavra
  - Concatenação com ela mesma
  - Representada na forma de expoente
    - $\mathbf{w^n}$ onde **n** é o número de concatenações sucessivas
  - É definida a partir da operação binária:
    - $w^0 = \varepsilon$
    - $w^n = ww^{n-1}$, para $n>0$



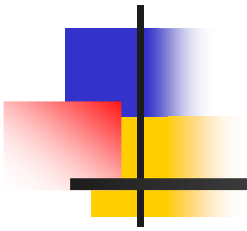
Conceitos Básicos

- Concatenação Sucessiva
  - Exemplo:
    - Sejam **w** uma palavra e **a** um símbolo
      - w^3 = www
      - w^1 = w
      - a^5 = aaaaa
      - a^n = aaa...a (o símbolo a repetido n vezes)



Conceitos Básicos

- Exercícios sugeridos:
  - Página 7: 1.5
  - Página 8: 1.6, 1.7 e 1.8

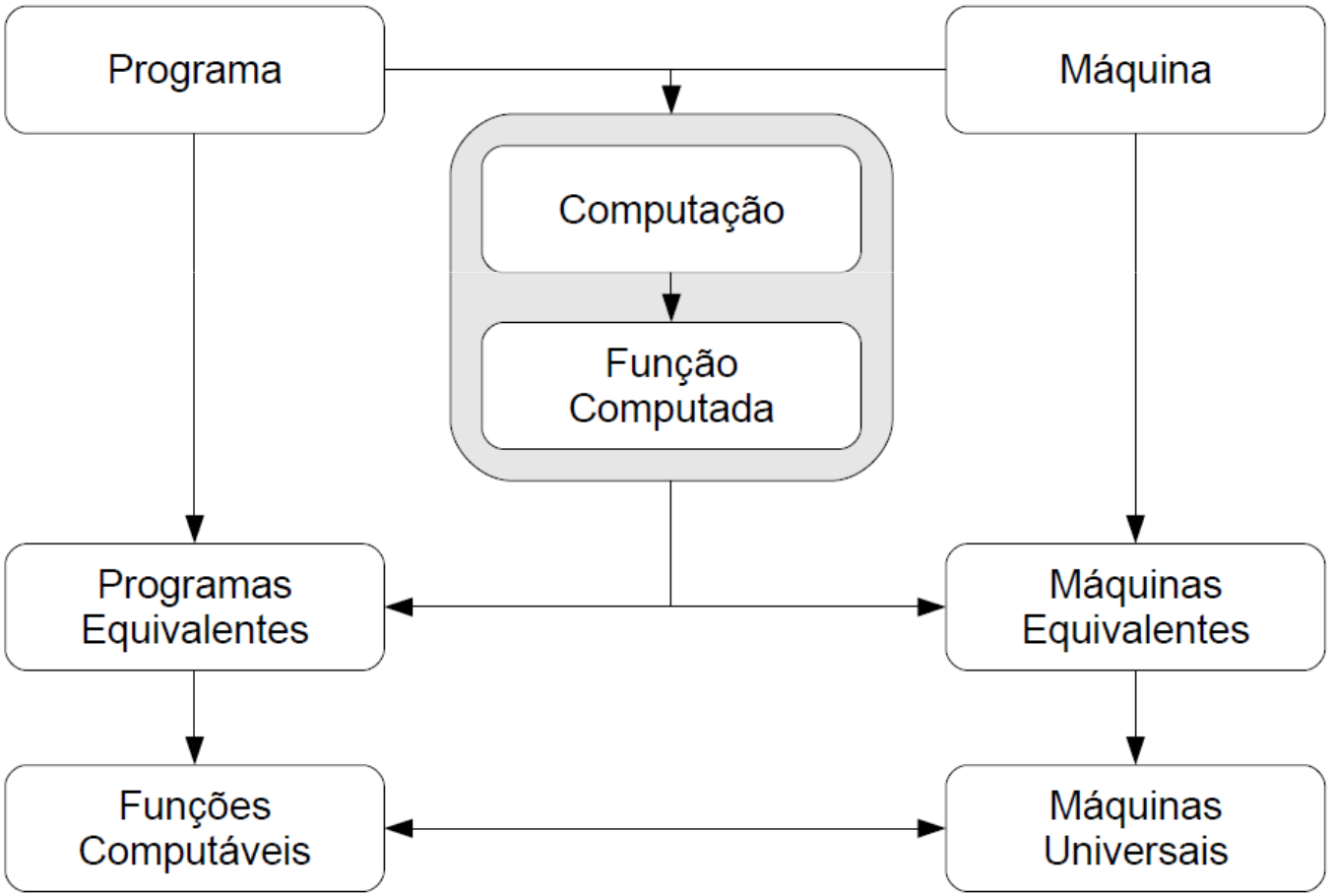


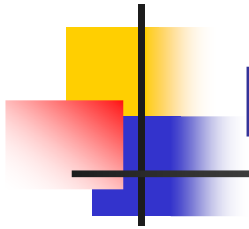
Teoria da Computação

Unidade 2 – Programas, Máquinas e Computações

Referência – Teoria da Computação (Divério, 2000)

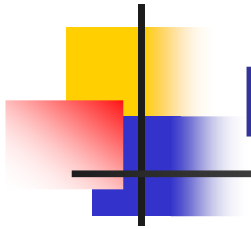
Programas, Máquinas e Computações





Programas, Máquinas e Computações

- Programas
  - *"Conjunto estruturado de instruções que capacitam uma máquina a aplicar sucessivamente certas operações básicas e testes sobre dados iniciais fornecidos, com o objetivo de transformar estes dados numa forma desejável."*



Programas, Máquinas e Computações

- *Programas*

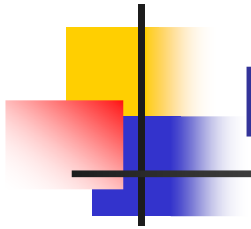
  Um conjunto de operações e testes compostos de acordo com uma *estrutura de controle*

- *Máquinas*

  Interpreta Programas e possui uma interpretação para cada operação ou teste que constituem o programa.

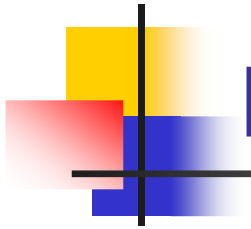
- *Computações*

  Histórico do funcionamento da máquina para o programa e um dado valor inicial.



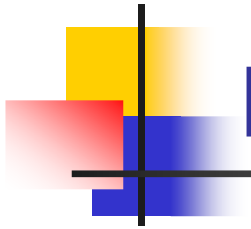
Programas, Máquinas e Computações

- Programa
  - Deve explicitar como as operações e testes devem ser *compostos*, ou seja, um programa deve possuir uma *estrutura de controle* de operações e *testes*.



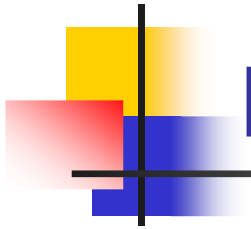
Programas, Máquinas e Computações

- Programas
  - Nas linguagens de programação atuais, existem várias formas de estruturação do controle, com destaque para:
    - Estruturação Monolítica
    - Estruturação Iterativa
    - Estruturação Recursiva



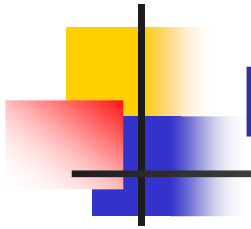
Programas, Máquinas e Computações

- Programas
  - Estruturação Monolítica
    - Baseada em desvios condicionais e incondicionais, não possuindo mecanismos explícitos de iteração, subdivisão ou recursão



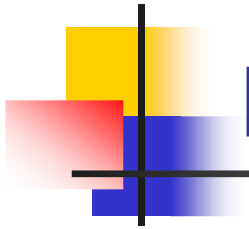
Programas, Máquinas e Computações

- Programas
  - Estruturação Iterativa
    - Possui mecanismos de controle de iterações de trechos de programas. Não permite desvios incondicionais.



Programas, Máquinas e Computações

- Programas
  - Estruturação Recursiva
    - Possui mecanismos de estruturação em sub-rotinas recursivas. Também não permite desvios incondicionais.

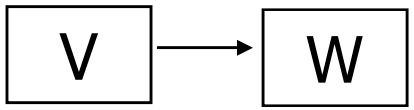


Programas, Máquinas e Computações

- Programas
  - Independentemente da estruturação de controle, duas ou mais operações ou testes podem ser compostos por:
    - Composição Sequencial
    - Composição Não-Determinística: composição de escolha
    - Composição Concorrente: execução em qualquer ordem, inclusive simultaneamente

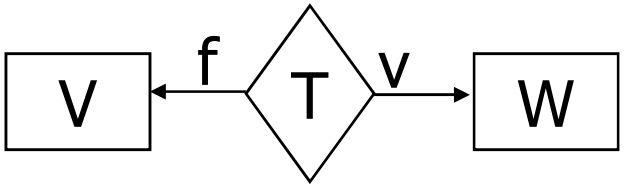
Programas, Máquinas e Computações

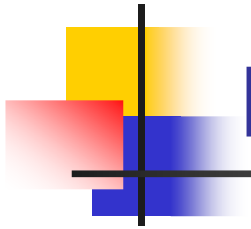
- Programas
  - Composição Sequencial
    - A execução da operação ou teste subsequente somente pode ser realizada após o encerramento da execução da operação ou teste anterior.



Programas, Máquinas e Computações

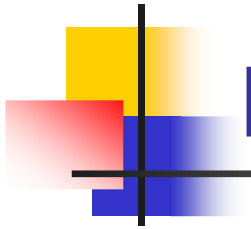
- Programas
  - Composição Não-Determinística
    - Uma das operações ou testes compostos é escolhido para ser executada. A composição não-determinista também é denominada de escolha.





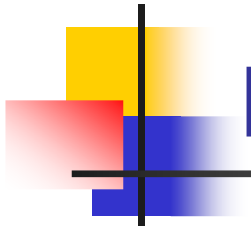
Programas, Máquinas e Computações

- Programas
  - Composição Concorrente
    - As operações ou testes compostos podem ser executados em qualquer ordem inclusive simultaneamente. Ou seja, a ordem de execução é irrelevante.



Programas, Máquinas e Computações

- Programas
  - Formado por dois conjuntos de identificadores:
    - Identificadores de Operações
      - F, G, H, ...
    - Identificadores de Testes
      - T_1, T_2, T_3, ...
    - Tipo especial de operação: operação vazia
      - ✓

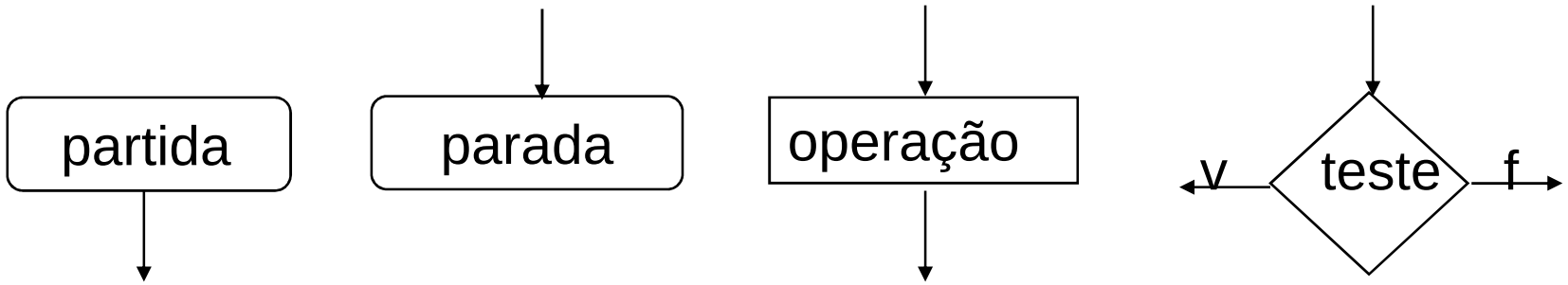


Programas, Máquinas e Computações

- Programa Monolítico
  - Estrutura com desvios condicionais e incondicionais
  - Não faz uso explícito de mecanismos como iteração, subdivisão ou recursão.
  - Forma tradicional de especificar programas monolíticos é através de fluxogramas

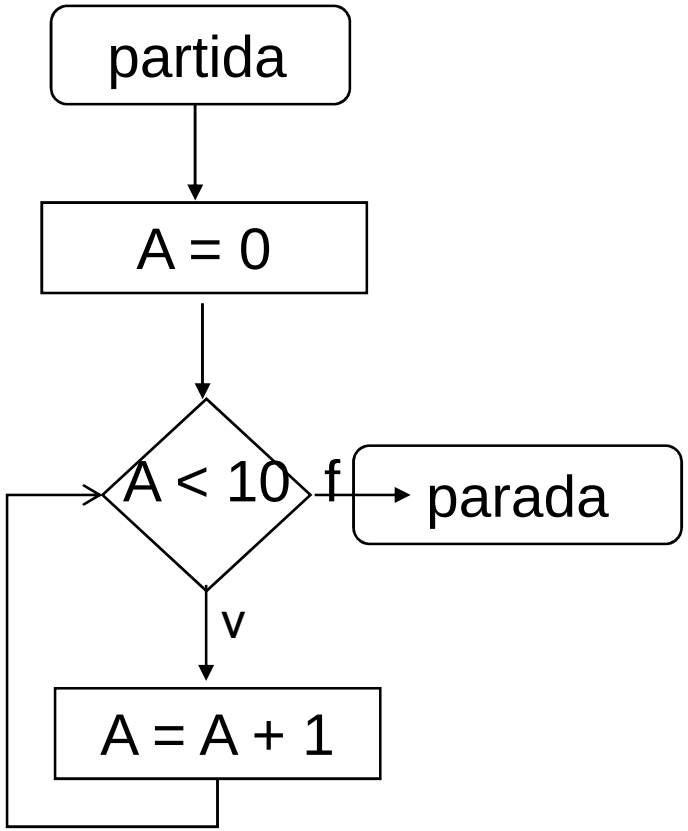
Programas, Máquinas e Computações

- Programa Monolítico
  - Fluxograma – componentes elementares



Programas, Máquinas e Computações

- Exemplo: Programa Monolítico



Programas, Máquinas e Computações

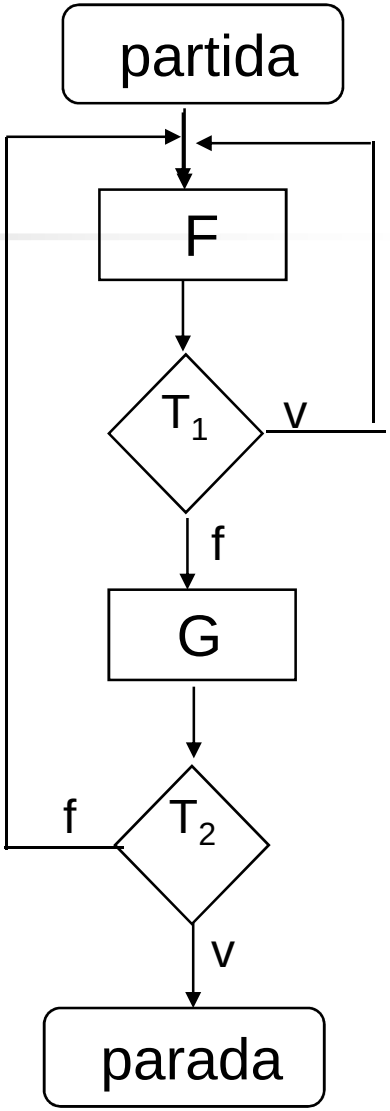
- Programa Monolítico: também pode ser representado por instruções rotuladas
  - Um rótulo é uma cadeia finita de caracteres constituída de letras ou dígitos
    - **Operação:** indica a operação a ser executada seguida de um desvio incondicional para a instrução subsequente
    - **Teste:** Determina um desvio condicional, ou seja, que depende da avaliação de um teste

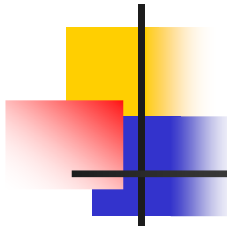
Obs.: Parada é especificada usando um desvio incondicional para um rótulo sem instrução correspondente.

Programas, Máquinas e Computações

- Programa Monolítico
  - Instruções rotuladas:

r1: faça F vá_para r2
r2: se T_1 então vá_para r1 senão vá_para r3
r3: faça G vá_para r4
r4: se T_2 então r5 (✓) senão vá_para r1





Programas, Máquinas e Computações

- Programa Monolítico
  - *Rótulo* ou *Etiqueta*: cadeia de caracteres finita (palavra) constituída de letras ou dígitos
  - *Instrução Rotulada* i é uma cadeia de caracteres finita (palavra) das seguintes formas:
    - *Operação*
      - r_1: faça F vá_para r_2 ou
      - r_1: faça ✓ vá_para r_2
    - *Teste*
      - r_1: se T então vá_para r_2 senão vá_para r_3

Programas, Máquinas e Computações

- Programa Monolítico
  - Definição
    - Um programa Monolítico P é um par ordenado

      P = (I,r)
    - Onde,
      - **I** Conjunto de Instruções Rotuladas o qual é finito
      - **r** Rótulo Inicial o qual distingue a instrução rotulada inicial em **I**

Observações:

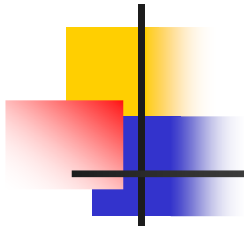
• Não existem duas instruções diferentes com um mesmo Rótulo;

• Um rótulo referenciado por alguma instrução e que não está associado a qualquer instrução rotulada é dito um Rótulo Final

Programas, Máquinas e Computações

- Programa Monolítico
  - Exemplos:
    - $P_1 = (I_1 , r_1)$

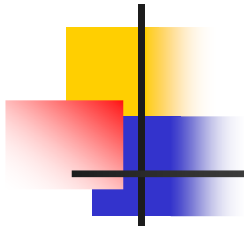
r_1: faça F vá_para r_2
r_2: se T_1 então vá_para r_1 senão vá_para r_3
r_3: faça G vá_para r_4
r_4: se T_2 então r_5 (✓) senão vá_para r_1



Programas, Máquinas e Computações

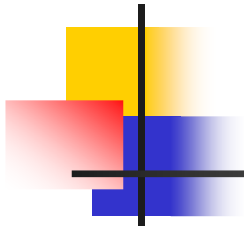
- Programa Monolítico
  - Exemplos:
    - P_2 = ({r_1: faça ✓ vá_para r_2 }, r_1)

R_2 é um rótulo final



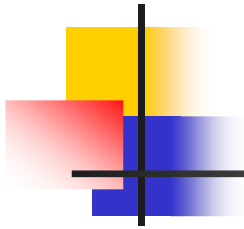
Programas, Máquinas e Computações

- Programa Monolítico
  - Problema dos programas monolíticos:
    - Dificuldade na manutenção e entendimento;
    - Liberdade dos desvios incondicionais: Causando a "quebra de lógica"
  - Solução:
  - Programação Estruturada: Substituição dos desvios incondicionais por estruturas de controle de ciclos ou repetições.



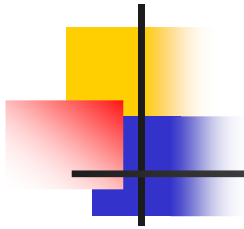
Programas, Máquinas e Computações

- Programa Iterativo
  - Substitui desvios incondicionais por estruturas de controle de ciclos (testes) ou repetições resultando em uma melhor estruturação de desvios



Programas, Máquinas e Computações

- Programa Iterativo
  - Definição:
    - a) A operação vazia ✓ constitui um programa iterativo
    - b) Cada identificador de operação constitui um programa iterativo
    - c) *Composição Sequencial*: V;W
    - d) *Composição Condicional*: se T então V senão W
    - e) *Composição Enquanto*: enquanto T faça (V)
    - f) *Composição Até*: até T faça (V)

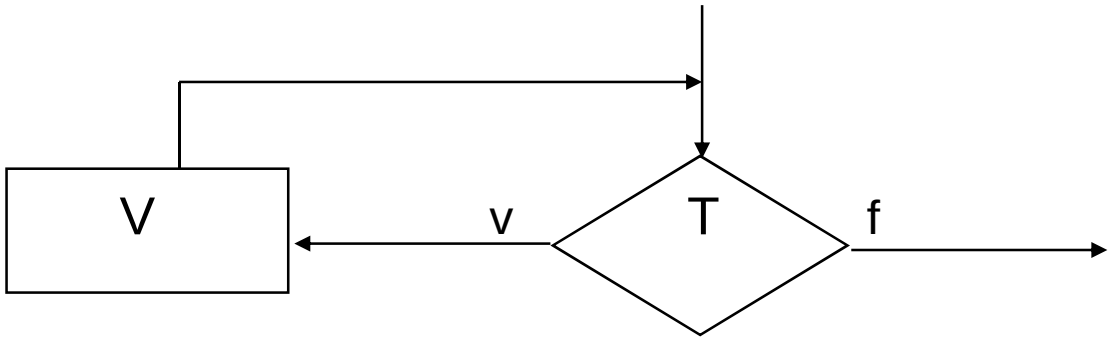


Programas, Máquinas e Computações

- Programa Iterativo
  - Exemplo de uso dos parênteses para mudar uma interpretação
    - Enquanto T faça V;W
    - Admite 2 interpretações:
      - (enquanto T faça V); W
      - enquanto T faça (V;W)

Programas, Máquinas e Computações

- Programa Iterativo
  - Fluxograma que simula a composição enquanto:



Programas, Máquinas e Computações

- Programa Iterativo
  - Exemplo:

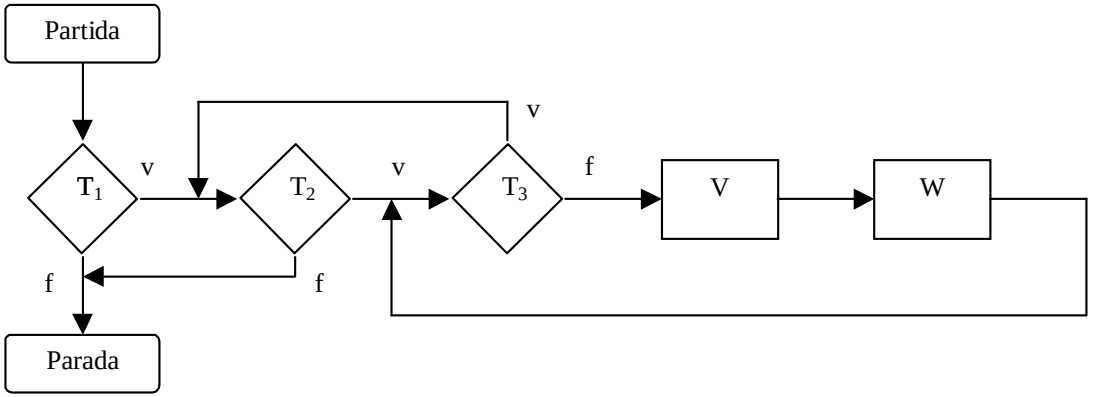
    (se T_1

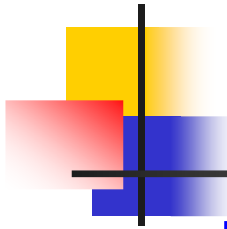
    então enquanto T_2

    faça (até T_3

    faça (V; W))

    senão ✓)



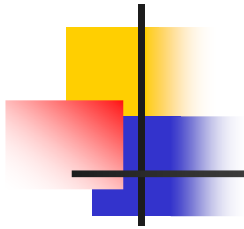


Programas, Máquinas e Computações

- Programa Recursivo
  - Admite a definição de sub-rotinas recursivas
  - *Recursão* é uma forma indutiva de definir programas.
  - Sub-rotinas permitem a estruturação hierárquica de programas, possibilitando níveis diferenciados de abstração.
  - Suponha um conjunto de Identificadores de sub-rotinas os quais são descritos por:
    - R_1, R_2, ...

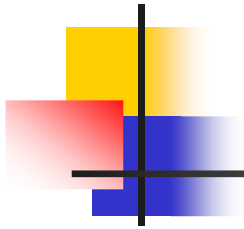
Programas, Máquinas e Computações

- Programa Recursivo
  - Expressão de sub-rotinas, definida como:
    - a) A operação vazia ✓ constitui um programa recursivo
    - b) Cada identificador de operação constitui uma expressão de sub-rotinas;
    - c) Cada identificador de sub-rotina constitui uma expressão de sub-rotinas
    - d) *Composição Sequencial*: $D_1;D_2$
    - e) *Composição Condicional*: (se T então D_1 senão D_2)



Programas, Máquinas e Computações

- Programa Recursivo
  - Tem a seguinte forma:
    - P é E_0 onde R_1 def E_1, R_2 def E_2 , ..., R_n def E_n
    - Onde (suponha K {1, 2, ..., n}):
    - E_0 Expressão inicial a qual é uma expressão de sub-rotinas;
    - E_k Expressão que define R_k, ou seja, a expressão que define a sub-rotina identificada por R_k



Programas, Máquinas e Computações

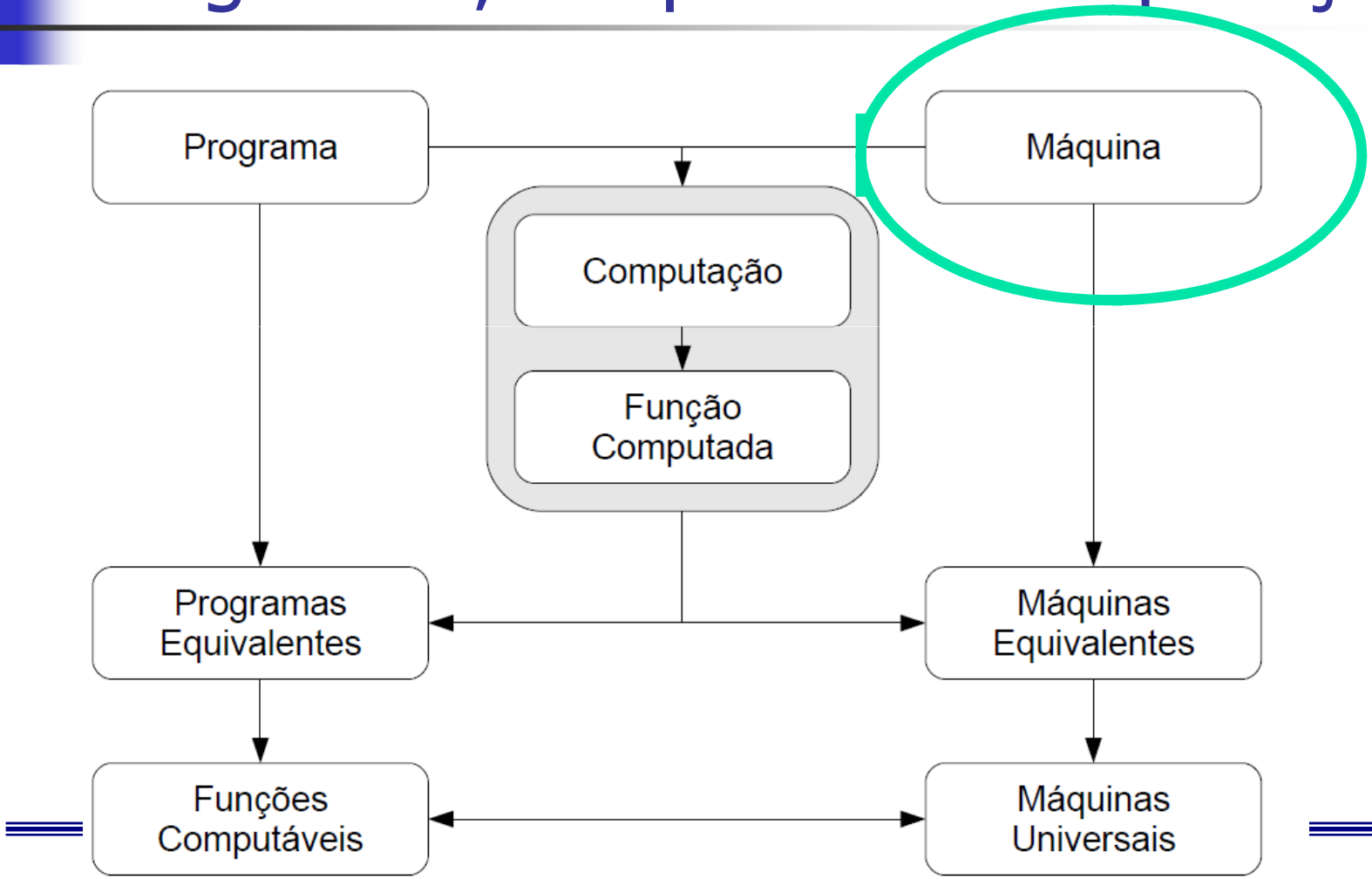
- Programa Recursivo
  - Exemplo

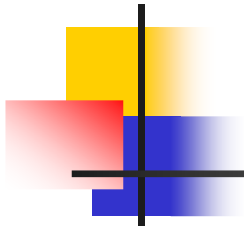
    P é R; S onde

    R def F; (se T então R senão G; S),

    S def (se T então ✓ senão F; R)

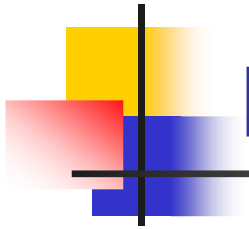
Programas, Máquinas e Computações





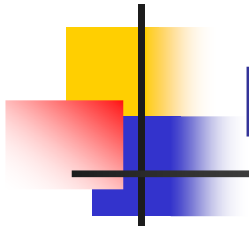
Programas, Máquinas e Computações

- Máquinas
  - Cabe a máquina suprir o significado aos identificadores das operações e testes para que a computação de um programa possa ser descrita → dar semântica
  - Assim, cada identificador de operação e de teste interpretado pela máquina deve ser associado a uma transformação na estrutura de memória e a uma função verdade (teste)



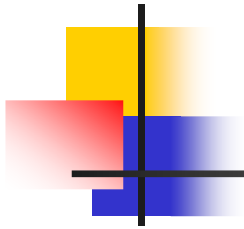
Programas, Máquinas e Computações

- Máquinas
- **É capaz de interpretar um programa**, desde que possua uma interpretação para cada operação ou teste que constitui o programa;



Programas, Máquinas e Computações

- Máquinas
- Observação:
  - Nem todo identificador de operação ou teste é definido em uma máquina;
  - Quando definido, existe somente uma função associada a ele.

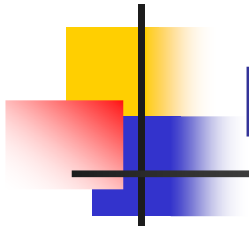


Programas, Máquinas e Computações

- Máquinas
- Uma máquina deve **descrever** o **armazenamento** ou **recuperação** de informações na estrutura de memória

Programas, Máquinas e Computações

- Máquinas - Definição: É uma 7-upla
  - M = (V, X, Y,πx, πy , $\prod_F$, $\prod_T$) onde:
  - V *Conjunto de Valores de Memória*
  - X *Conjunto de Valores de Entrada*
  - Y *Conjunto de Valores de Saída*
  - πx *Função de Entrada* tal que:
    - πx: X→ V
  - πy *Função de Saída* tal que:
    - πx: V→ Y
  - $\prod_F$ *Conjunto de Interpretações de Operações:*
    - $\prod_F$: V→ V em $\prod_F$
  - $\prod_T$ *Conjunto de Interpretações de Testes*
    - $\prod_T$: V→ V em $\prod_T$



Programas, Máquinas e Computações

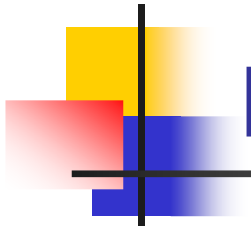
- Exemplo: Máquina de Dois Registradores
  - Dois registradores **a** e **b** assumem valores em **N** (naturais), com duas operações e um teste:
    - `subtração de 1 em a, se a>0;`
    - `adição de 1 em b;`
    - `teste se a é zero;`
    - Adicionalmente, valores de entrada são armazenados em **a** (zerando **b**) e a saída retorna o valor de **b**

Programas, Máquinas e Computações

- Máquina de Dois Registradores
  - Dois_reg=(N^2, N, N, armazena_a, retorna_b, {subtrai_a,adiciona_b},{a_zero})
  - N^2 corresponde ao conjunto de valores de memória
  - N corresponde aos conjuntos de valores de entrada e saída
  - **armazena_a**: N → N^2 é a função de entrada tal que,$\forall$ n $\in$ N:
    - armazena_a(n) = (n, 0)
  - **retorna_b**: N^2 → N é a função de saída tal que,$\forall$ (n, m) $\in$ n^2 :
    - retorna_b(n, m) = m
  - **subtrai_a**: N^2 → N^2 é interpretação tal que,$\forall$ (n, m) $\in$ n^2 :
    - subtraia_a(n,m) = (n-1, m), se n $\neq$0; subtrai_a(n, m) = (0, m), se n=0
  - **adiciona_b**: N^2 → N é interpretação tal que,$\forall$ (n, m) $\in$ N^2 :
    - Adiciona_b(n,m) = (n, m+1)
  - **a_zero**: N^2 → {verdadeiro, falso} é interpretação tal que,$\forall$ (n, m) $\in$ N^2 :
    - a_zero(n,m) = verdadeiro, se n=0; a_zero(n, m) = falso, se n $\neq$0

Programas, Máquinas e Computações

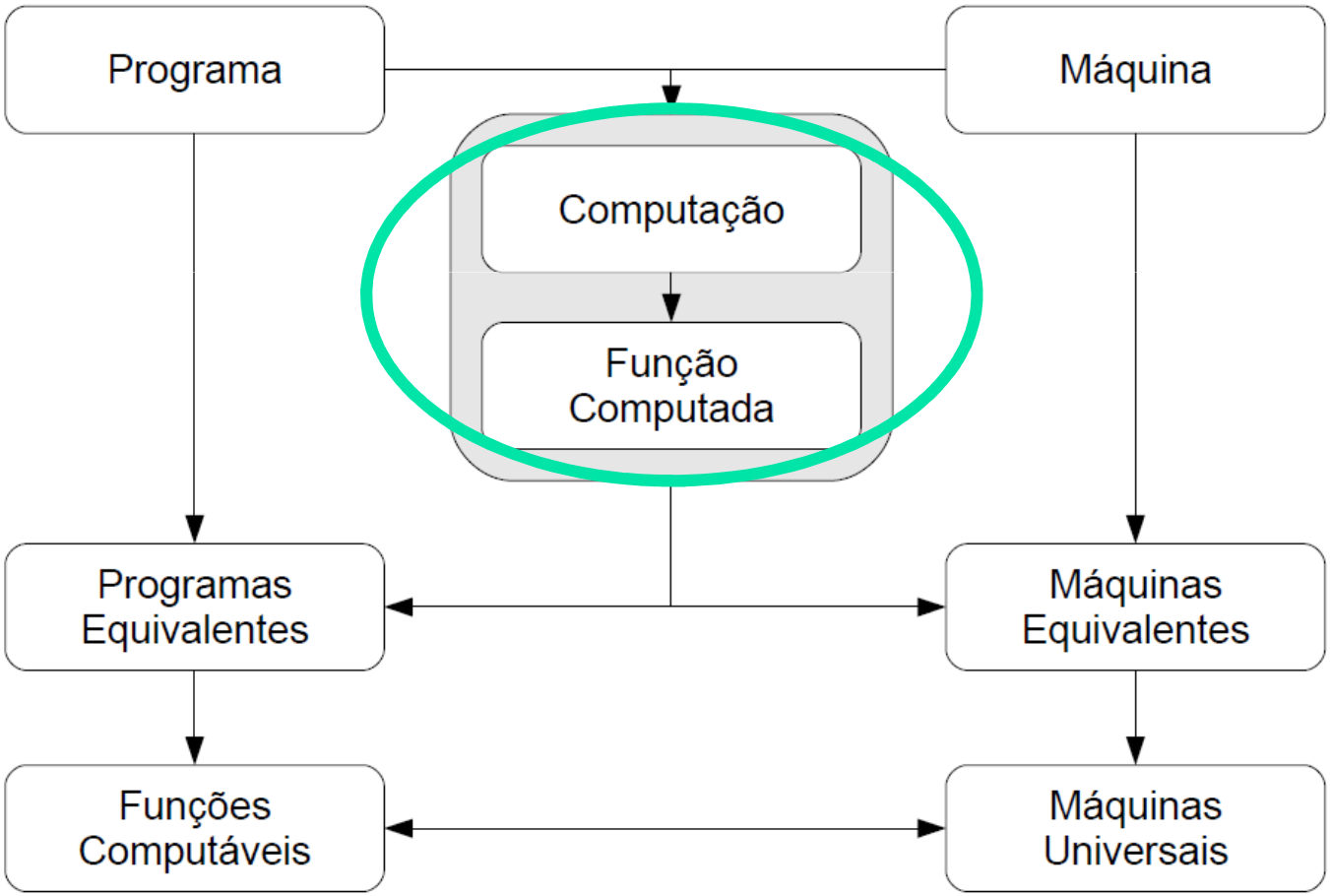
- Definição formal
  - **Programa para uma máquina**
  - Seja M = (V, X, Y,πx, πy , $\prod_F$, $\prod_T$) uma máquina e P um programa onde P_F e P_T são os conjuntos de identificadores de operações e testes de P.
  - P é um programa para a Máquina M, se, e somente se:
    - Para qualquer F $\in$ P_F, existe uma função π_F: V → V em $\prod_F$
    - Para qualquer T $\in$ P_T, existe uma função π_T: V → {verdadeiro, falso} $\prod_T$
  - Adicionalmente, a operação vazia $\sqrt{}$ sempre é interpretada em qualquer máquina

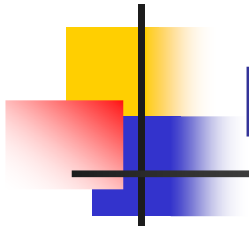


Programas, Máquinas e Computações

- Exemplos: Programas para a Máquina de Dois Registradores
  - Programa Iterativo itv_b ← a
    - até **a_zero**
    - faça (**subtrai_a**; **adiciona_b**)
  - Programa Recursivo rec_b ← a
    - **rec_b** ← a é R onde
      - R def (se **a_zero** então √ senão S; R),
      - S def **subtrai_a**; **adiciona_b**

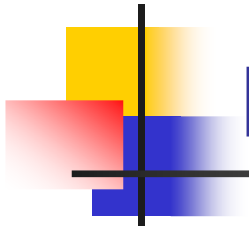
Programas, Máquinas e Computações





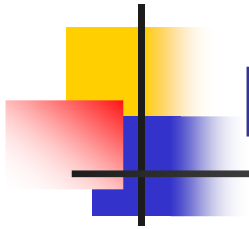
Programas, Máquinas e Computações

- Computações e Funções Computadas
  - Computação
    - É um histórico do funcionamento da máquina para o programa, considerando um valor inicial.
  - Função computada
    - É o resultado obtido após o término da computação (finita)



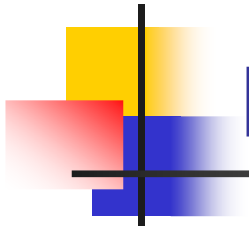
Programas, Máquinas e Computações

- Computação de um Programa Monolítico
  - É um histórico do funcionamento da máquina para o programa, considerando um valor inicial
  - Histórico representado na forma de cadeia de pares:
    - **Cada par** reflete um estado da máquina para o programa, ou seja, a instrução a ser executada e o valor corrente da memória;
    - A **cadeia** reflete uma sequência de estados possíveis a partir do estado inicial (instrução inicial e valor de memória considerado)



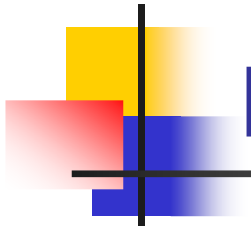
Programas, Máquinas e Computações

- Computação de um Programa Monolítico
  - Seja M = (V, X, Y,πx, πy , $\prod_F$, $\prod_T$) uma máquina e P = (I, r) um programa Monolítico para M onde L é o correspondente conjunto de rótulos.
  - Uma computação do *Programa Monolítico P* na *Máquina M* é uma cadeia (finita ou infinita de pares) de L x V:
    - $(s_0, v_0)(s_1,v_1)$...
  - 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



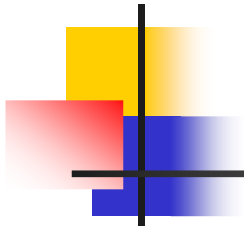
Programas, Máquinas e Computações

- Computação de um Programa Monolítico
  - **a)Operação**
    - a1) 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
    - a2) Se s_k é o rótulo de uma operação da forma:
      - s_k : faça $\surd$ vá_para r´
      - então(s_{k+1}, v_{k+1}) = (r´,V_K) é par subsequente de (s_k, v_k) na cadeia



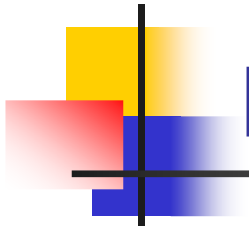
Programas, Máquinas e Computações

- Computação de um Programa Monolítico
  - **b)Teste.** Se s_k é o rótulo de um teste na 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 $\pi T(V_K)$ = verdadeiro
    - s_{k+1} =r´´ se $\pi T(V_K)$ = falso



Programas, Máquinas e Computações

- Computação de um Programa Monolítico
  - Uma computação é Finita ou Infinita, se a cadeia que a define é finita ou infinita, respectivamente
  - Notar que:
    - Para um dado valor inicial de memória, a correspondente cadeia de computação é única, ou seja, a computação 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



Programas, Máquinas e Computações

- Computação de um Programa Monolítico
  - Exemplo:
  - **1)** Fazer a computação do programa monolítico mon_b←a para a máquina **dois_reg**. Para o valor inicial de memória (3, 0), e verificar se a computação é finita ou infinita.

Programas, Máquinas e Computações

- **Programa Monolítico mon_b ← a**
  - 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

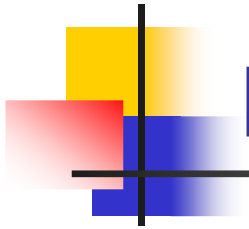
(1,(3,0)) → instrução inicial e valor inicial armazenado
(2,(3,0)) → em 1, como a # 0, desviou para 2
(3,(2,0)) → em 2, subtraiu de a e desviou para 3

(1,(2,1)) → em 3, adicionou em b e desviou para 1
(2,(2,1)) → em 1, como a # 0, desviou para 2
(3,(1,1)) → em 2, subtraiu de a e desviou para 3

(1,(1,2)) → em 3, adicionou em b e desviou para 1
(2,(1,2)) → em 1, como a # 0, desviou para 2
(3,(0,2)) → em 2, subtraiu de a e desviou para 3

(1,(0,3)) → em 3, adicionou em b e desviou para 1
(9,(0,3)) → em 1, como a = 0, desviou para 9

computação é finita



Programas, Máquinas e Computações

- Computação de um Programa Monolítico
  - **2)** Fazer a computação do programa monolítico mon_comp para a máquina **dois_reg**. Para o valor inicial de memória (3, 0), e verificar se a computação é finita ou infinita.
  - **Programa Monolítico comp_infinita**
    - 1: faça **adiciona_b** vá_para 1

(1,(3,0)) → instrução inicial e valor inicial armazenado
(1,(3,1)) → adicionou em b e permanece em 1
(1,(3,2)) → adicionou em b e permanece em 1
(1,(3,3)) → adicionou em b e permanece em 1
... indefinidamente

computação é infinita

Programas, Máquinas e Computações

- Computação de um Programa Recursivo
  - Seja M = (V, X, Y,πx, πy , $\prod_F$, $\prod_T$) uma máquina e P um programa recursivo para M tal que
    - P é E_0 onde R_1 def E_1, R_2 def E_2 , ... ,R_n def E_n
  - Uma computação do *Programa Recursivo* P na *Máquina M* é uma cadeia (finita ou infinita de pares) na forma:
    - $(D_0, v_0)(D_1,v_1)$...
  - Onde (D_0, v_0) é tal que $D_0= E_0;\sqrt{}$ e v_0 é o valor inicial de memória e para cada para $(D_k= V_k)$ da cadeia, onde k={0, 1, 2,...}, tem-se que:

Programas, Máquinas e Computações

- Computação de um Programa Recursivo
  - *Caso 1*: Se D_k é uma expressão de sub-rotina da forma:
    - $D_k = \sqrt{};C$

  então $(D_{k+1}, V_{k+1}) = (C, V_k)$ é par subsequente de (D_k, V_k) na cadeia
  - *Caso 2*: Se D_k é uma expressão de sub-rotina da forma:
    - $D_k = F;C$

  então$(D_{k+1},V_{k+1}) = (C,\pi F(V_k))$ é par subsequente de (D_k,V_k) na cadeia
  - *Caso 3*: Se D_k é uma expressão de sub-rotina da forma:
    - $D_k = R_i ;C$

  então$(D_{k+1},V_{k+1}) = (E_i;C,V_k)$ é par subsequente de (D_k,V_k) na cadeia

Programas, Máquinas e Computações

- Computação de um Programa Recursivo
  - *Caso 4*: Se D_k é uma expressão de sub-rotina da forma:
    - $D_k = (C_1,C_2), C$

então$(D_{k+1},V_{k+1}) = ((C_1;(C_2,C),V_k)$ é par subsequente de (D_k,V_k) na cadeia

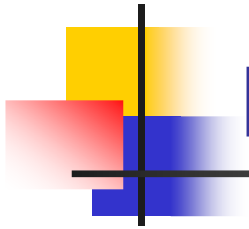
  - *Caso 5*: Se D_k é uma expressão de sub-rotina da forma:
    - $D_k = E_k$ = (se T então C_1 senão C_2);C

então(D_{k+1},V_{k+1}) é par subsequente de (D_k,V_k) na cadeia sendo que:

$V_{k+1} = V_k$

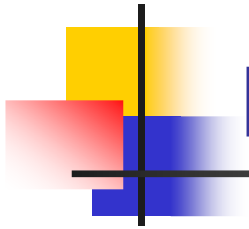
$D_{k+1} = C_1;C$ se $\pi T(V_K)$ = verdadeiro

$D_{k+1} = C_2;C$ se $\pi T(V_K)$ = falso



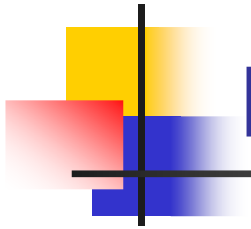
Programas, Máquinas e Computações

- Computação de um Programa Recursivo
  - A computação é dita finita ou infinita, se a cadeia que a define é finita ou infinita, respectivamente, portanto:
    - para um dado valor inicial de memória, a correspondente cadeia de computação é única, ou seja, a computação é determinística
    - teste ou referência a uma sub-rotina não alteram o valor da memória;
    - em uma computação finita, a expressão $\surd$ ocorre no último par da cadeia e não ocorre em qualquer outro par;
    - em uma computação infinita, expressão alguma da cadeia é $\surd$



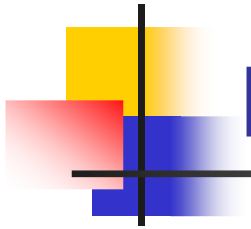
Programas, Máquinas e Computações

- Computação Infinita de um Programa Recursivo
  - Programa Recursivo qq_máquina
  - **qq_máquina** é R onde
    - R def R



Programas, Máquinas e Computações

- ## Máquina de Um Registrador
  - um_reg=(N, N, N, id_N, id_N, {ad, sub}, {zero})
  - N corresponde ao conjunto de valores de memória, de entrada e saída
  - id $_N$ N→ N é a função de entrada e de saída
  - ad: N → N é interpretação tal que,$\forall$ n $\in$ N, ad (n) = n+1
  - sub: N → N é interpretação tal que,$\forall$ n $\in$ N:
    - sub(n) = n-1, se n $\neq$0; sub(n) = 0, se n = 0
  - zero N → {verdadeiro, falso} é interpretação tal que $\forall$ n $\in$ N
    - zero(n) =verdadeiro, se n=0; zero(n)= falso, caso contrário



Programas, Máquinas e Computações

- Exercício - Fazer a computação e verificar se a computação é finita ou infinita
- programa Recursivo duplica

  duplica é R onde

  R def (se **zero** então √ senão **sub**; R; **ad**; **ad**)

Programas, Máquinas e Computações

duplica é R onde

R def (se **zero** então √ senão **sub**; R; **ad**; **ad**)

Computação de **duplica**

(R, √, 2)

(se **zero** então √ senão (**sub**; R; **ad**; **ad**); √; 2)

(**sub**; R; **ad**; **ad**); √; 2)

(R; **ad**; **ad**); √; 1)

(se **zero** então √ senão (**sub**; R; **ad**; **ad**); **ad**; **ad** √; 1)

(**sub**; R; **ad**; **ad**; **ad**; **ad** √; 1)

(R; **ad**; **ad**; **ad**; **ad** √; 0) ...

Programas, Máquinas e Computações

(R; **ad**; **ad**; **ad**; **ad** √; 0)

(se **zero** então √ senão (**sub**; R; **ad**; **ad**); **ad**; **ad**; **ad**; **ad** √; 0)

(√; **ad**; **ad**; **ad**; **ad** √; 0)

(**ad**; **ad**; **ad** √; 1)

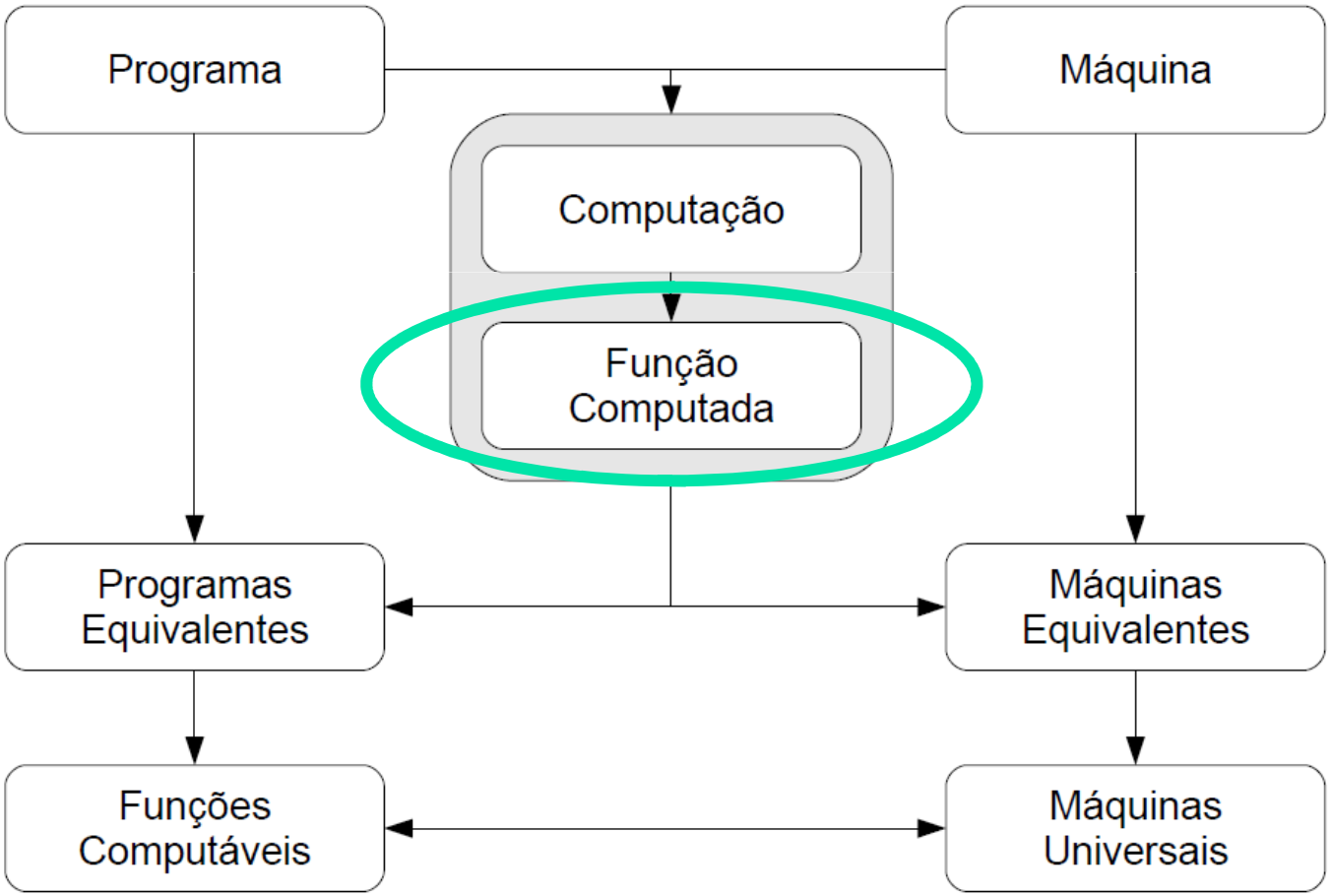
(**ad**; **ad** √; 2)

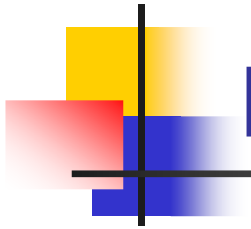
(**ad** √; 3)

(√; 4)

A computação é finita

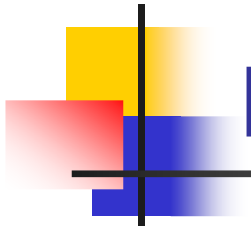
Programas, Máquinas e Computações





Função Computada

- Função computada por um programa monolítico:
  - 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;
  - Executa, passo a passo, testes e operações, na ordem determinada pelo programa, até atingir um rótulo final, quando para;
  - O valor da função computada pelo programa é o valor resultante da aplicação da *função de saída* ao *valor da memória* (máquina parada)

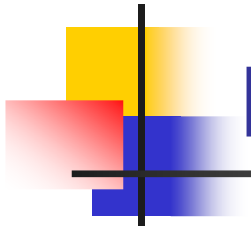


Função Computada

- Função computada por um programa monolítico em uma Máquina:
  - Seja M = (V, X, Y, π_x, π_y, $\prod_F$, $\prod_T$) uma máquina e P um programa monolítico para M.
  - A *Função Computada* pelo *Programa Monolítico* P na Máquina M denotada por:

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

  - é uma função parcial definida por x ϵ X se a cadeia:

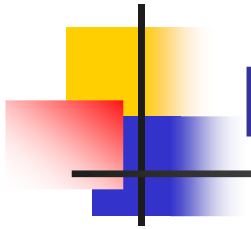


Função Computada

$(s_0, v_0)(s_1,v_1)\ldots (s_n,v_n)$

- É uma computação finita de P em M, onde o valor inicial da memória é dado pela função de entrada, ou seja, $v_0 = \pi_X(X)$.
- Neste caso, a imagem de x é dada pela função de saída aplicada ao último valor da memória na computação, ou seja

$$<P, M>(x) = \pi_Y(V_n)$$



Função Computada

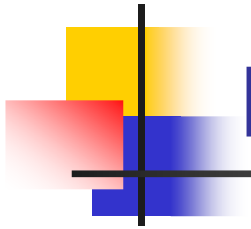
- Função computada por um programa monolítico na máquina de dois registradores:

- Considere mon_b←a para a máquina dois_reg.
- A correspondente função computada é a função identidade, ou seja,

<mon_b←a, dois_reg>: N →N

- Tal que, para qualquer n ∈ N (naturais), tem-se que:

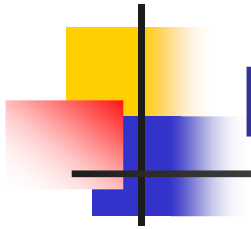
<mon_b←a, dois_reg>(n) = n



Função Computada

- Função computada por um programa monolítico na máquina de dois registradores:

- Exemplo:
  - para o valor de entrada 3, tem-se que:
    - $\pi_x(3) = (3,0)$
    - **<mon_b←a, dois_reg>(3) =** $\pi_x(0,3) = 3$



Função Computada

- Função computada por um programa recursivo em uma Máquina:
  - Seja M = (V, X, Y, π_x, π_y, $\prod_F$, $\prod_T$) uma máquina e P um programa recursivo para M.
  - A *Função Computada* pelo *Programa Recursivo* P na *Máquina* M denotada por:

$$\mathbf{<P, M>: X \rightarrow Y}$$

É uma função parcial definida para x $\in$ X se a cadeia é uma computação finita de P em M

$$\bullet(D_0, v_0)(D_1,v_1)...(D_n,v_n)$$

Função Computada

- Função computada por um programa recursivo em uma Máquina:
  - Onde:
    - $D_0 = E_0$; ✓ é expressão inicial de P
    - $v_0 = \pi_x(x)$
    - $E_n = ✓$
  - Tem-se que: $\langle P,M\rangle(x) = \pi_y(v_n)$

Exemplo:

duplica é R onde

R def (se **zero** então √ senão **sub**; R; **ad**; **ad**)

- A correspondente função computada é: <duplica, um_reg>: N → N
- e é tal que, para qualquer n ϵ N: <duplica, um_reg>(n) = 2n

Função Computada

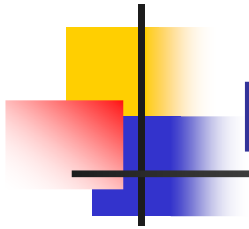
- Função computada por um programa recursivo em uma Máquina:
  - Onde:
    - $D_0 = E_0$; ✓ é expressão inicial de P
    - $v_0 = \pi_x(x)$
    - $E_n = ✓$
  - Tem-se que: $\langle P,M\rangle(x) = \pi_y(v_n)$

Exemplo:

duplica é R onde

R def (se **zero** então √ senão **sub**; R; **ad**; **ad**)

- A correspondente função computada é: <duplica, um_reg>: N → N
- e é tal que, para qualquer n ϵ N: <duplica, um_reg>(n) = 2n



Programas, Máquinas e Computações

- Exercícios sugeridos:
  - 2.1 ao 2.10
  - Enviar até a próxima aula – pode ser feito em dupla.