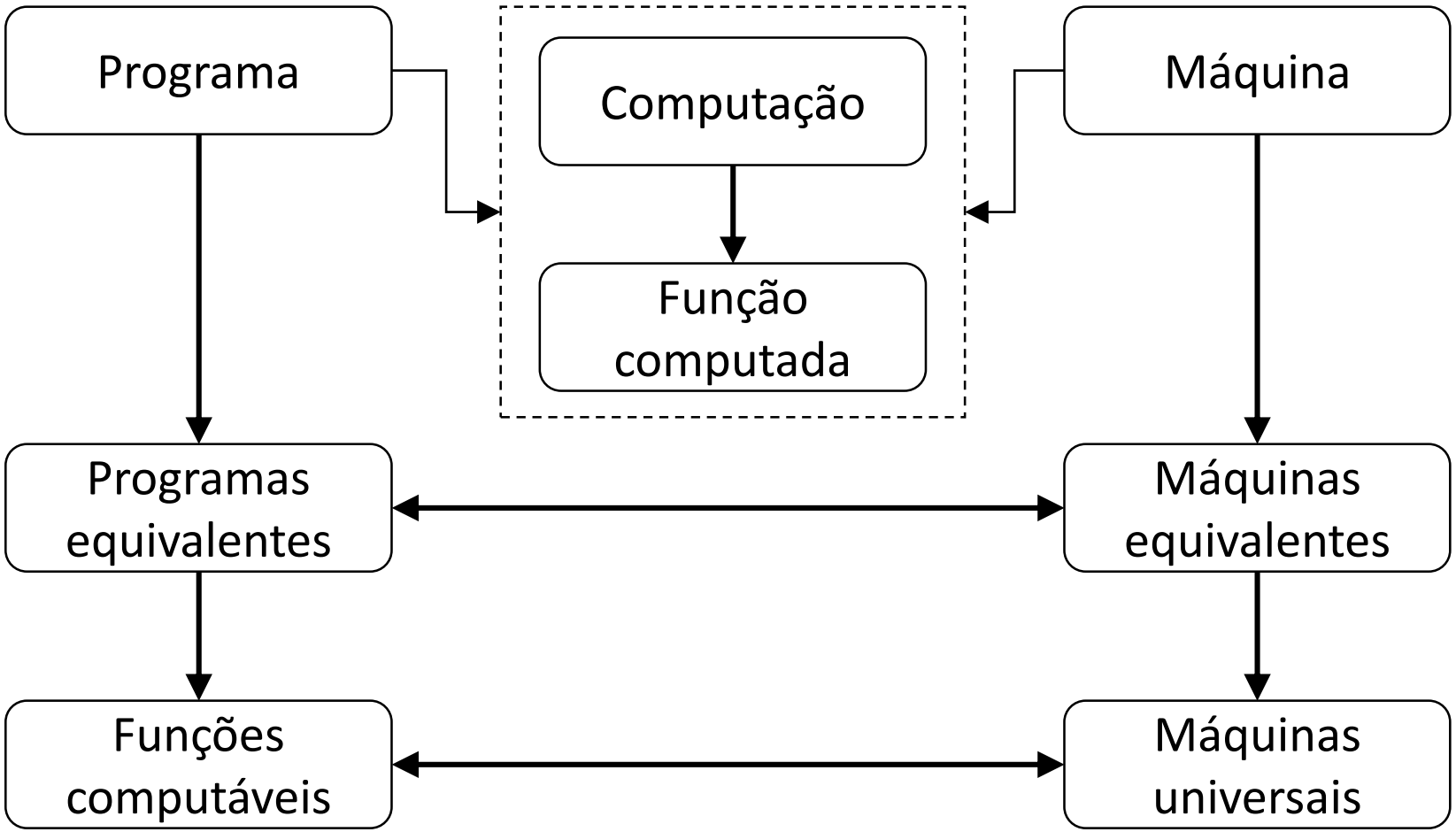
Programas, máquinas e computações

Teoria da computação

Prof. Allan Rodrigo Leite

Programas, máquinas e computações



Definições

- Programa
  - Conjunto estruturado de instruções para uma máquina aplicar certas operações básicas e testes sucessivamente sobre um dado valor inicial fornecido
  - Tais sequências de operações objetivam transformar os dados iniciais em outra forma desejável, de acordo com uma estrutura de controle
- Máquina
  - Interpretador de programas, 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

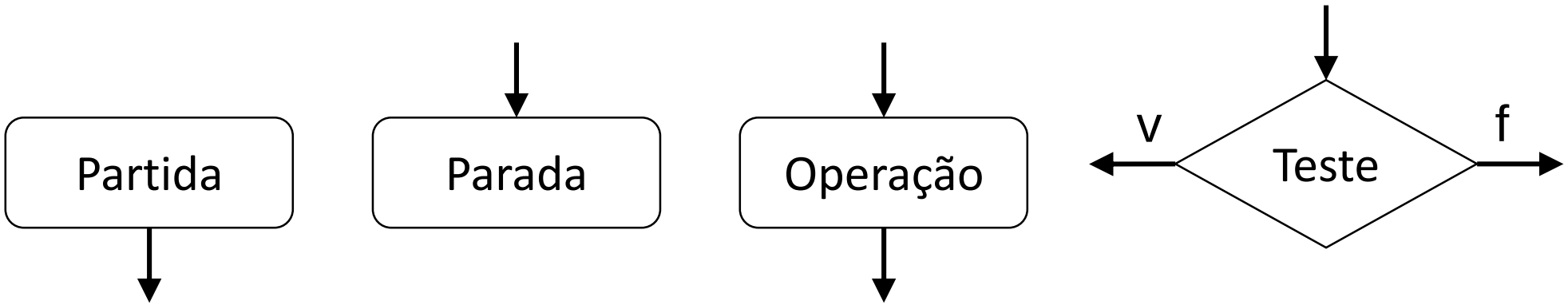
Programas

- Estruturação de programas
  - Um programa é formado por dois conjuntos de identificadores:
    - Identificadores de operações { `F, G, H, ...` }
    - Identificadores de testes: verdadeiro ou falso { `T1, T2, T3, ...` }
    - Tipo especial de operação: operação vazia { • }

  - As linguagens de programação atuais dispõem de várias formas de estruturação do controle do programa, destacando-se:
    - Estruturação monolítica
    - Estruturação iterativa
    - Estruturação recursiva

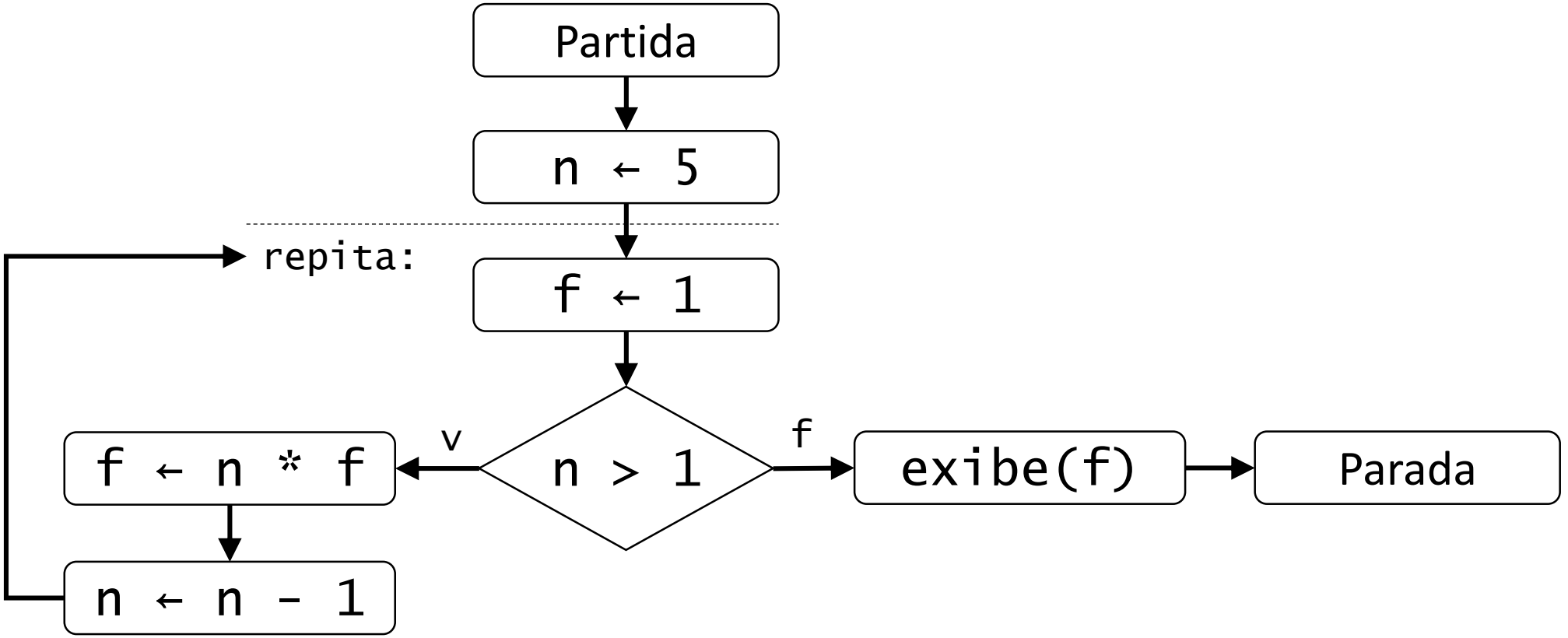
Programas

- Estruturação monolítica
  - Baseada em desvios condicionais e incondicionais
  - Não possui mecanismos explícitos de iteração, subdivisão ou recursão
  - Operações ou testes podem ser rotulados
  - Estrutura básica utilizada em linguagens de baixo nível (Assembly)



Programas

- Estruturação monolítica: calcular o fatorial de um número



Programas

- Estruturação monolítica: calcular o fatorial de um número

```
int n = 5;                  //operação
int f = 1;                  //operação

repita:                     //rótulo

if (n > 1) {                //teste
    f = f * n;              //operação
    n = n - 1;              //operação
    goto repita;            //operação
}

printf("Resultado: %d", f); //operação
```

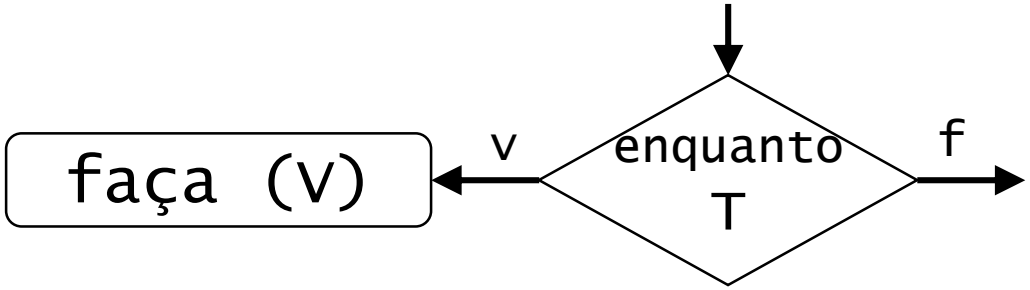
Programas

- Estruturação monolítica (cont.)
  - Otimizada para máquinas interpretarem as instruções
  - Dificuldade para um humano efetuar manutenção e entender o algoritmo
  - Liberdade dos desvios incondicionais causa a “quebra de lógica”, diminuindo a legibilidade e compreensão do algoritmo por um humano

Programas

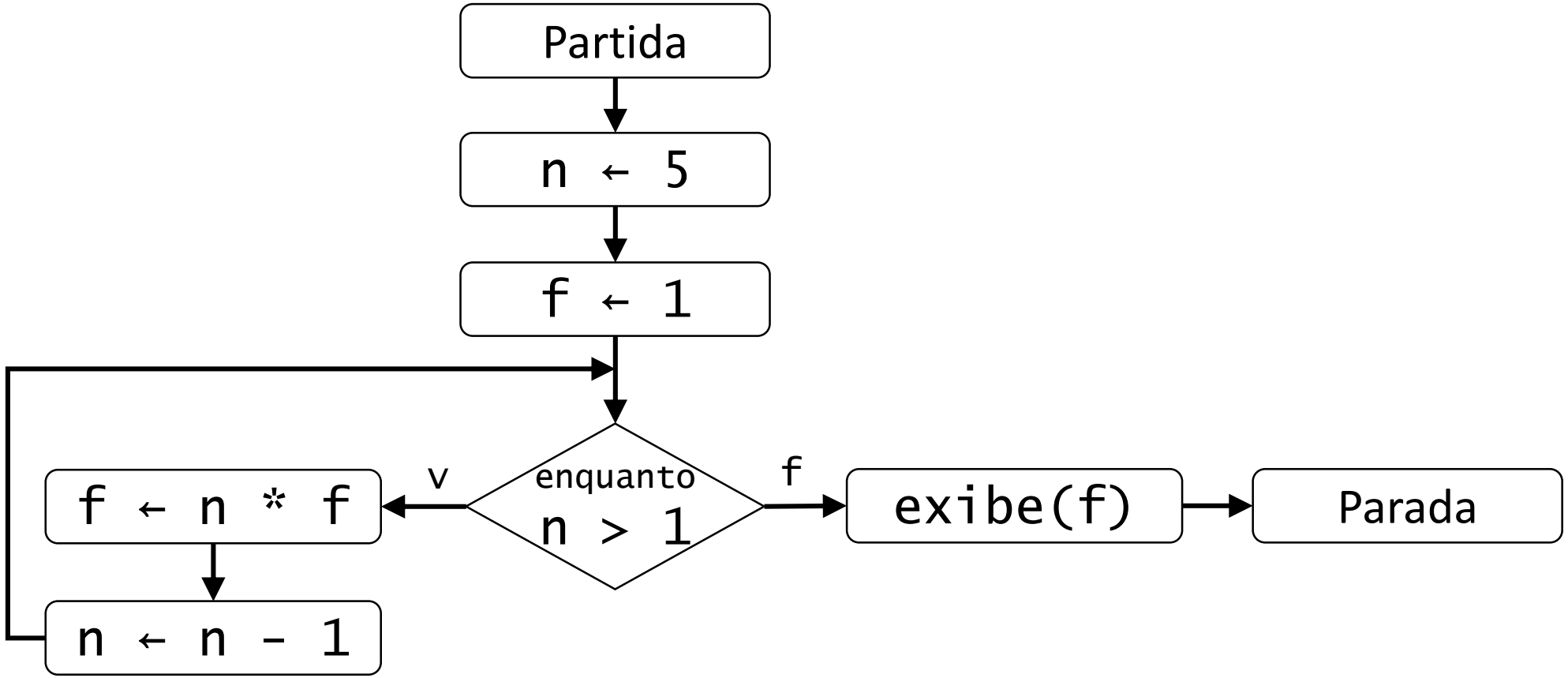
- Estruturação iterativa
  - Dispõe de mecanismos de controle de iterações de trechos de programas
  - Não permite desvios incondicionais

- Diferenças entre a estruturação iterativa e monolítica
  - Além da composição sequencial e condicional, há também:
    - Composição enquanto: `enquanto T faça (V)`
    - Composição até: `até T faça (V)`



Programas

- Estruturação iterativa: calcular o fatorial de um número



Programas

- Estruturação iterativa: calcular o fatorial de um número

```
int n = 5;              //operação
int f = 1;              //operação

while (n > 1) {         //enquanto teste
    f = f * n;          //operação
    n = n - 1;          //operação
}

printf("Resultado: %d", f); //operação
```

Programas

- Estruturação recursiva
  - Permite a definição de sub-rotinas recursivas
  - Recursão é uma forma indutiva de definir programas
  - Sub-rotinas permitem a estruturação hierárquica de programas em níveis
  - diferenciados de abstração

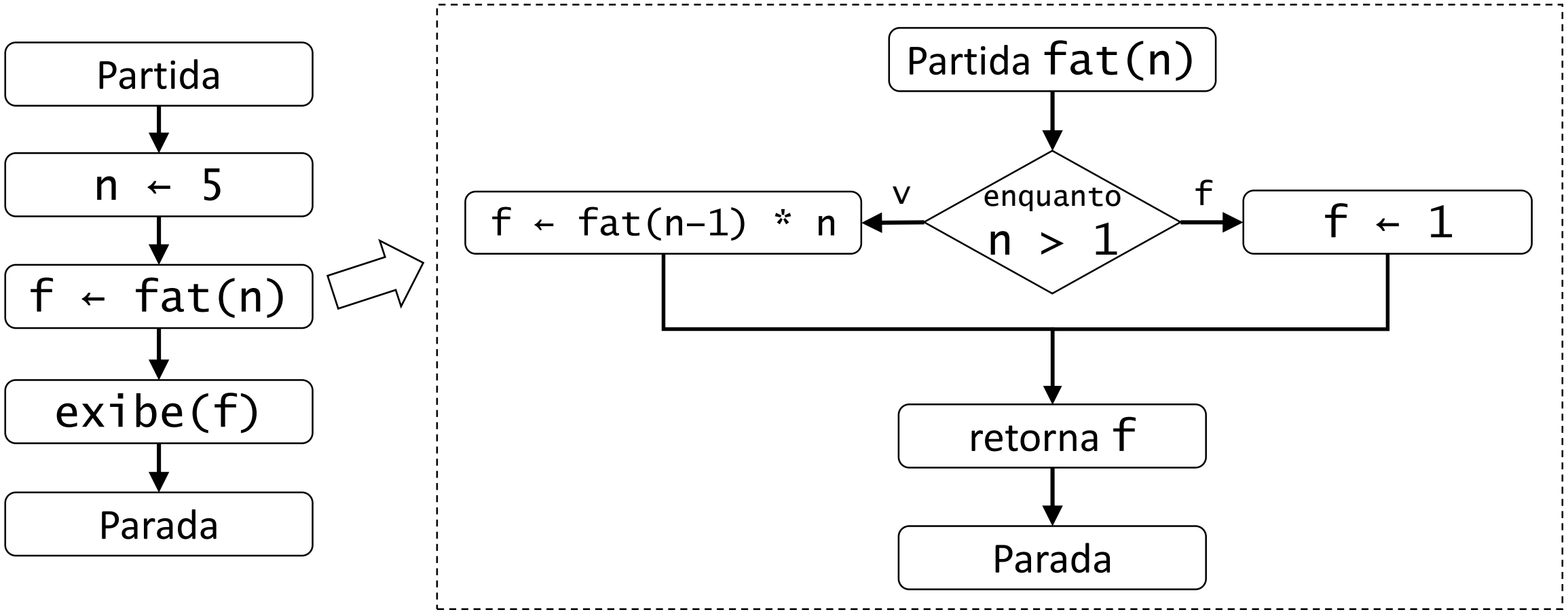
Programas

- Estruturação recursiva (cont.)
  - A operação vazia • constitui um programa recursivo
  - Cada identificador de operação constitui uma expressão de sub-rotinas
  - Cada identificador de sub-rotina constitui uma expressão de sub-rotinas
  - Conjunto de identificadores de sub-rotinas são descritos por:
    - `R1, R2, ...`

- Um programa recursivo tem a forma:
  - P é E^0 onde R^1 `def` E^1, R^2 `def` E^2, `...`, R^n `def` E^n
  - E^0 é a expressão inicial que é uma expressão de sub-rotinas
  - E^k é a expressão que define R^k, ou seja, a expressão que define a sub-rotina identificada por R^k

Programas

- Estruturação recursiva: calcular o fatorial de um número



Programas

- Estruturação iterativa: calcular o fatorial de um número

```
int n = 5;                          //operação
int f = fat(n);                     //operação

printf("Resultado: %d", f);         //operação

int fat(int n) {                    //sub-rotina
    if (n > 1) {                    //teste
        return fat(n - 1) * n;      //operação
    }
    return 1;                       //operação
}
```

Máquinas

- Propósito
  - Interpreta o significado dos identificadores de operações e testes
  - Possibilita que a computação de um programa possa ser descrita

- Funcionamento
  - Para cada identificador de operação ou teste interpretado pela máquina:
    - Está associado a uma transformação na estrutura de memória e a uma função verdade
    - Existe somente uma função associada

Definição formal de máquinas

- Tupla $M = < V, X, Y, \pi x, \pi y, \Pi_F, \Pi_T >$
  - V é o conjunto de valores em memória
  - X é o conjunto de valores de entrada
  - Y é o conjunto de valores de saída
  - πx é a função de entrada tal que $\pi x: X \rightarrow V$
  - πy é a função de saída tal que $\pi y: V \rightarrow Y$
  - Π_F é o conjunto de operações para cada F interpretado por M
    - $\pi_F: V \rightarrow V$ em Π_F
  - Π_T é o conjunto de testes para cada T interpretado por M
    - $\pi_T: V \rightarrow \{$ verdadeiro, falso $\}$ em Π_T

Máquinas

- Exemplo de uma máquina de dois registradores
  - Os dois registradores, a e b, podem assumir números inteiros não negativos (números naturais N)
  - Esta máquina é capaz de executar duas operações e um teste
    - Subtração de 1 em a, se a > 0
    - Adição de 1 em b
    - Teste se a = 0
  - Valores de entrada são armazenados em a (zerando b)
  - Valores de saída utiliza-se o valor de b

Máquinas

- Exemplo de uma máquina de dois registradores (cont.)
  - M2R =

```
<
    N², //conjunto de valores em memória (V)
    N, //conjunto de valores de entrada (X)
    N, //conjunto de valores de saída (Y)
    armazena_a, //função de entrada  X → V (πx)
    retorna_b, //função de saída V → Y (πy)
    {subtrai_a, adiciona_b}, //operações V → V em Π_F
    {a_zero} //teste V → { v, f } em Π_T
>
```

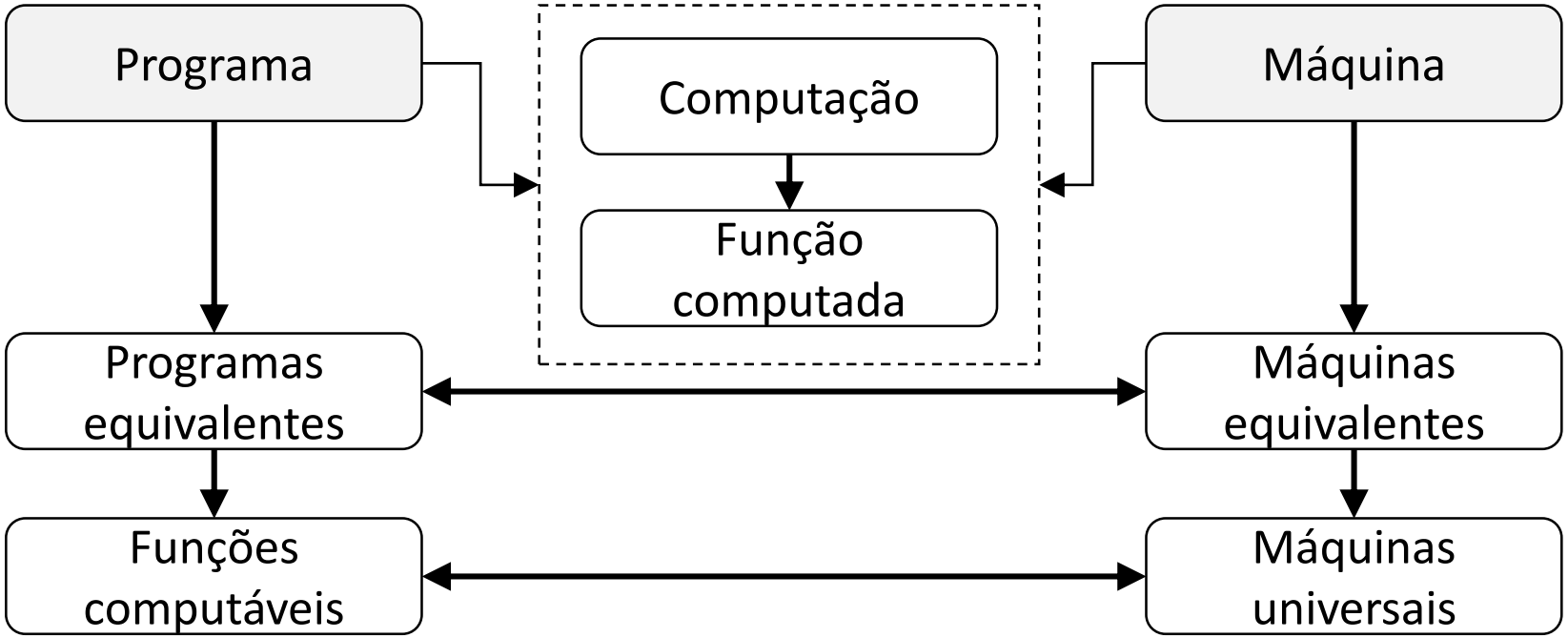
Máquinas

- Exemplo de uma máquina de dois registradores (cont.)
  - M2R =

```
<
    N², //Registradores (n,m)
    N, //n
    N, //m
    armazena_a, //armazena_a(n): (n, 0)
    retorna_b,  //retorna_b(m): (n, m)
    {
        subtrai_a, //subtrai_a(n,m): se n ≠ 0 (n – 1,m) senão (0,m)
        adiciona_b //adiciona_b(n,m): (n,m + 1)
    },
    {
          a_zero //a_zero(n, m): se n = 0 verdadeiro senão falso
    }
>
```

Máquinas

- Relação entre máquinas e programas
  - P é um programa para uma máquina M se:
    - Cada identificador de operação e teste em P possuir uma função correspondente de operação e teste em M



Computação e função computada

- Computação
  - Histórico do funcionamento da máquina para o programa, considerando um determinado valor inicial
- Função computada
  - Resultado obtido após o término da computação, quando finita

Computação

- Computação de um programa monolítico P
  - 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)
    - A cadeia é identificada pelo rótulo de cada operação ou teste do programa P

- Exemplo de programa monolítico P1 para a máquina M2R

```
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
```

Computação

- Exemplo de programa monolítico `P1` para a máquina M2R (cont.)

```
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
```

- `P1` possui `L = {1, 2, 3}` rótulos
  - Como seria a execução de `P1` dados os valores iniciais `(n, m) = (2, 0)` para os registradores?

Computação

- Programa monolítico P1 para a máquina M2R (cont.)

```
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, (2, 0))   //instrução inicial e valor de entrada armazenado
(2, (2, 0))   //em 1, como a ≠ 0, desviou para 2
(3, (1, 0))   //em 2, subtraiu 1 do registrador a e desviou para 3
(1, (1, 1))   //em 3, adicionou 1 no registrador b e desviou para 1
(2, (1, 1))   //em 1, como a ≠ 0, desviou para 2
(3, (0, 1))   //em 2, subtraiu 1 do registrador a e desviou para 3
(1, (0, 2))   //em 3, adicionou 1 no registrador b e desviou para 1
(9, (0, 2))   //em 1, como a = 0, desviou para 9
```

Computação

- Programa monolítico P2 para a máquina M2R com computação infinita

```
1:  faça adiciona_b vá-para 1
```

- P2 possui L = { 1 } rótulo
  - Como seria a execução de P dados os valores iniciais (n, m) = (2, 0)?

```
(1, (2, 0))   //instrução inicial e valor de entrada armazenado
(1, (2, 1))   //em 1, adicionou 1 no registrador b e desviou para 1
(1, (2, 2))   //em 1, adicionou 1 no registrador b e desviou para 1
(1, (2, 3))   //em 1, adicionou 1 no registrador b e desviou para 1
...
```

Função computada

- Função computada de 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 as operações e testes na ordem determinada pelo programa até atingir um rótulo final (condição de parada)
  - 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

- Dadas uma máquina $M = \langle V, X, Y, \pi x, \pi y, \Pi_F, \Pi_T \rangle$ e um programa P, a função computada é determinada por:

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

  - Ou seja, $\langle P, M \rangle$ é uma função parcial definida para a cadeia de computações (estados)

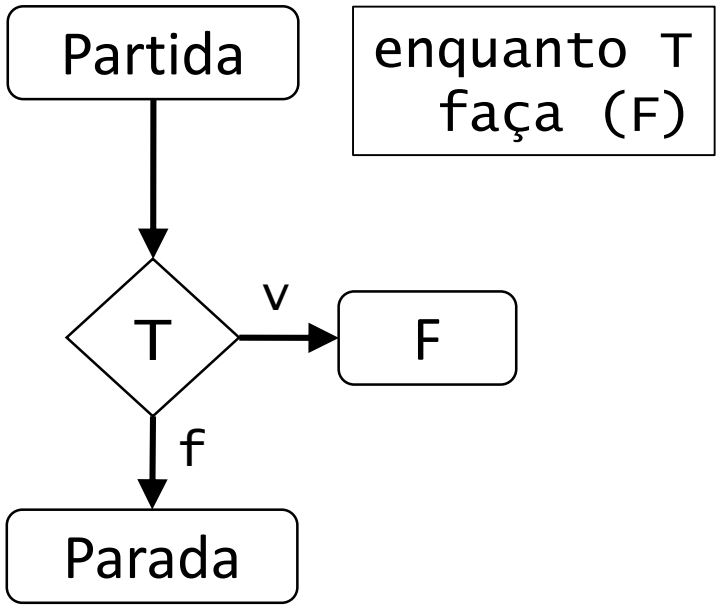
- Para o exemplo de programa P1 para a máquina M2R, temos:
  - Função de entrada $\pi x(2) = (2,0)$
  - Função computada $\langle P1, M2R \rangle(2) = \pi y(0,2)$
  - Função de saída $\pi y(0,2) = 2$

Equivalência de programas e máquinas

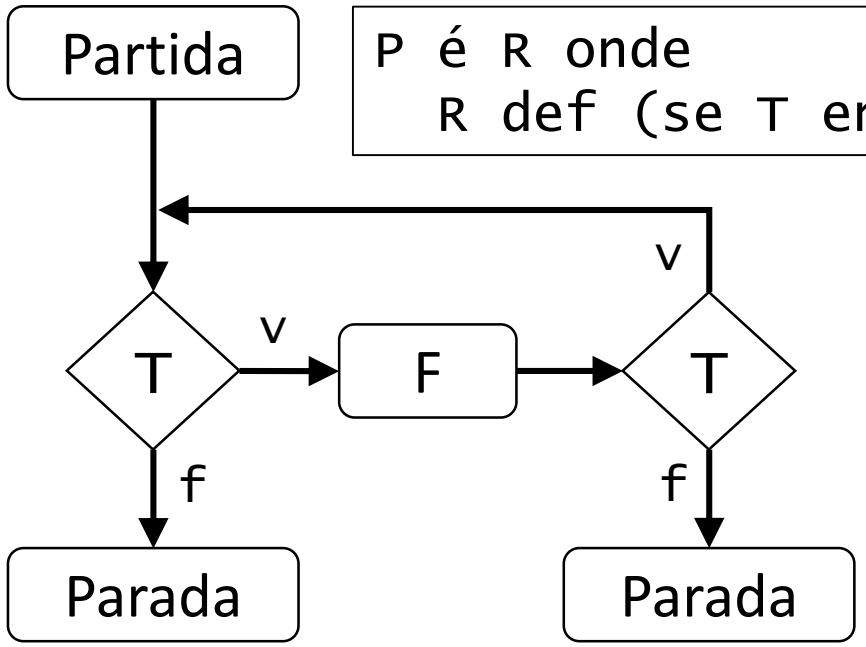
- Equivalência forte de programas
  - Um par de programas pertence à esta relação se as correspondentes funções computadas coincidem para qualquer máquina
- Equivalência de programas em uma máquina
  - Um par de programas pertence à esta relação se as correspondentes funções computadas coincidem para uma dada máquina
- Equivalência de máquinas
  - Um par de máquinas pertence à esta relação se as máquinas podem se simular mutuamente
  - A simulação de uma máquina por outra pode ser feita usando programas diferentes

Equivalência de programas e máquinas

- Equivalência forte de programas
  - Sejam P e Q dois programas, o par (P,Q) estão nesta relação por: P ≡ Q
  - Se e somente se para qualquer máquina M as correspondentes funções computadas são iguais <P,M> = <Q,M>




```
enquanto T
  faça (F)
```



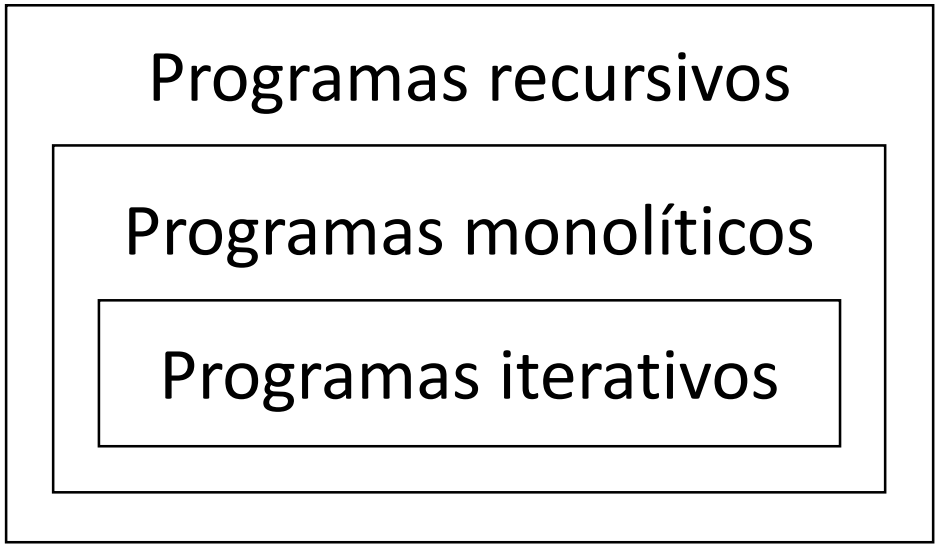

```
P é R onde
  R def (se T então F;R senão •)
```

Equivalência de programas e máquinas

- Equivalência forte de programas (cont.)
  - Permite identificar diferentes programas em uma mesma classe de equivalência, ou seja, identificar diferentes programas cujas funções computadas coincidem, para qualquer máquina
  - Em funções computadas por programas equivalentes fortemente, os mesmos testes e operações são efetuados na mesma ordem, independentemente do significado de cada um
  - Fornece subsídios para analisar a complexidade estrutural de programas
    - Analisando os dois exemplos anteriores, o iterativo é mais otimizado que o recursivo, pois contém um teste a menos

Equivalência de programas e máquinas

- Equivalência forte de programas (cont.)
  - Para todo iterativo, existe um monolítico equivalente fortemente
  - Para todo monolítico, existe um recursivo equivalente fortemente
  - Para todo iterativo, existe um recursivo equivalente fortemente



Programas, máquinas e computações