Resolução Exe 2.12 – Monolítico ➔ Recursivo

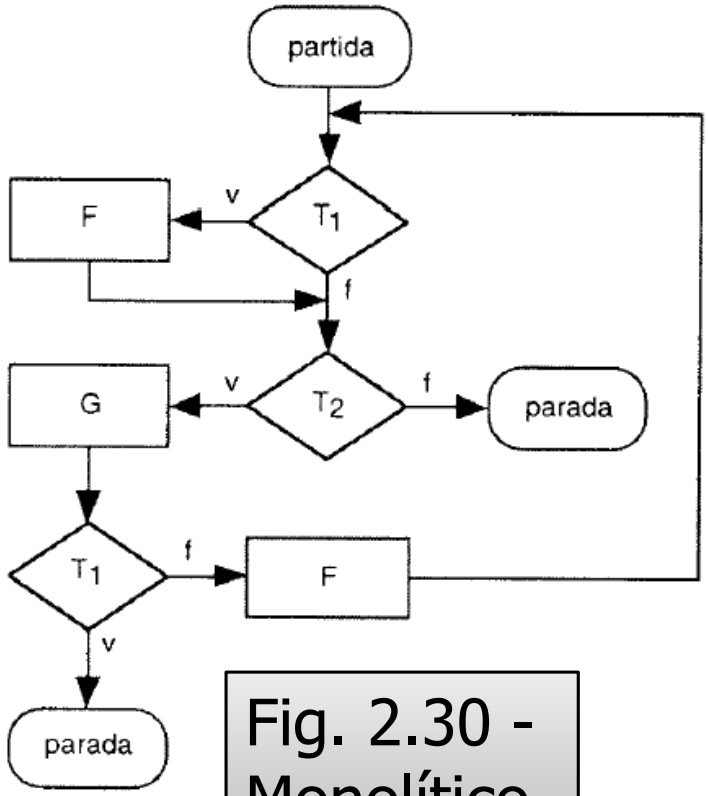


Fig. 2.30 - Monolítico

R_4 *def G;* R_5
R_5 *def (se T1 então* R_7 *senão* R_6 *)*
R_6 *def F;* R_1
R_7 *def* ✓

Resolução Exe 2.12 – Monolítico ➔ Recursivo

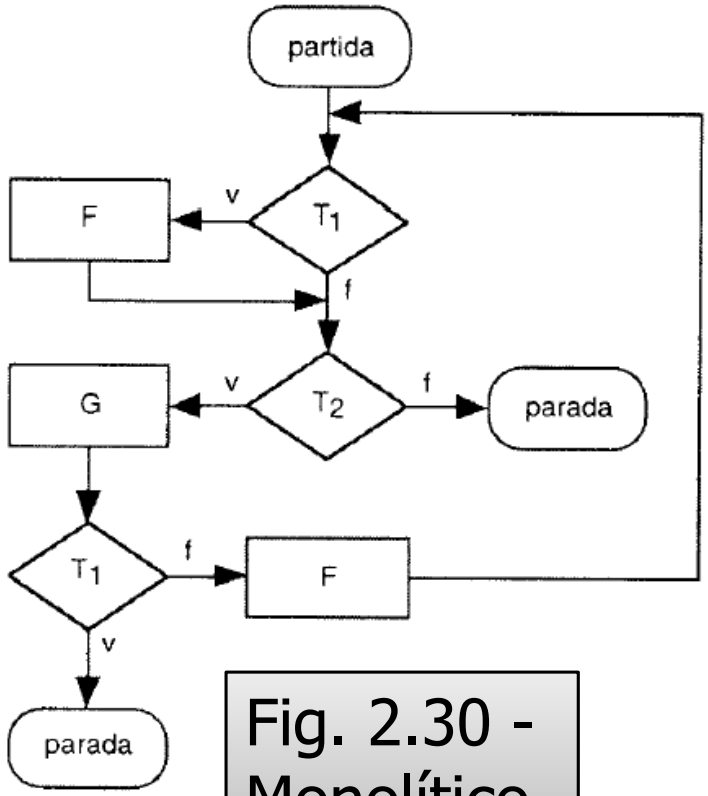


Fig. 2.30 - Monolítico

R_4 def G; R_5
R_5 def (se T1 então R_7 senão R_6)
R_6 def F; R_1
R_7 def ✓

Simplificado
P_R é R_1 onde
R_1 def (se T1 então F; R_2 senão R_2)
R_2 def (se T2 então G; R_3 senão ✓)
R_3 def (se T1 então ✓ senão F; R_1)

Resolução Exe 2.12 – Monolítico ➔ Recursivo

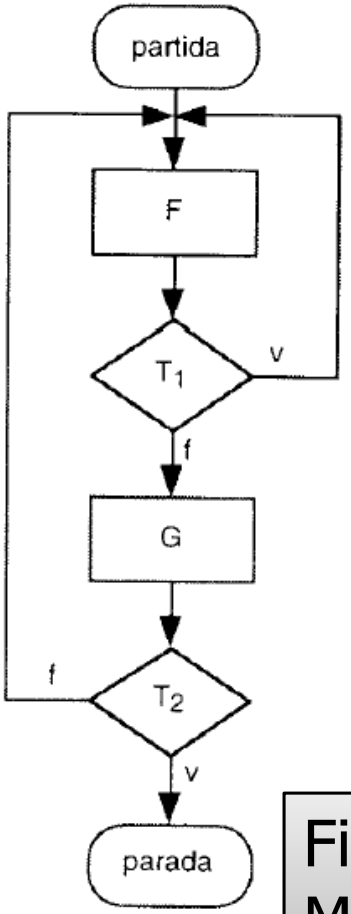


Fig. 2.32 - Monolítico

R_3 def G; R_4
R_4 def (se T2 então R_5 senão R_1)
R_5 def ✓

Resolução Exe 2.12 – Monolítico ➔ Recursivo

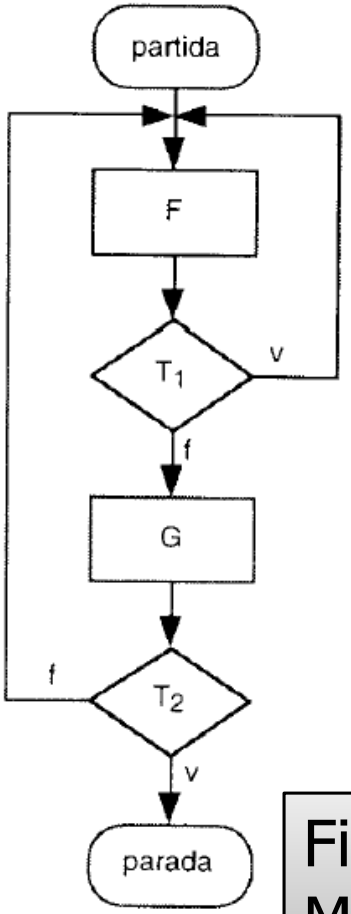


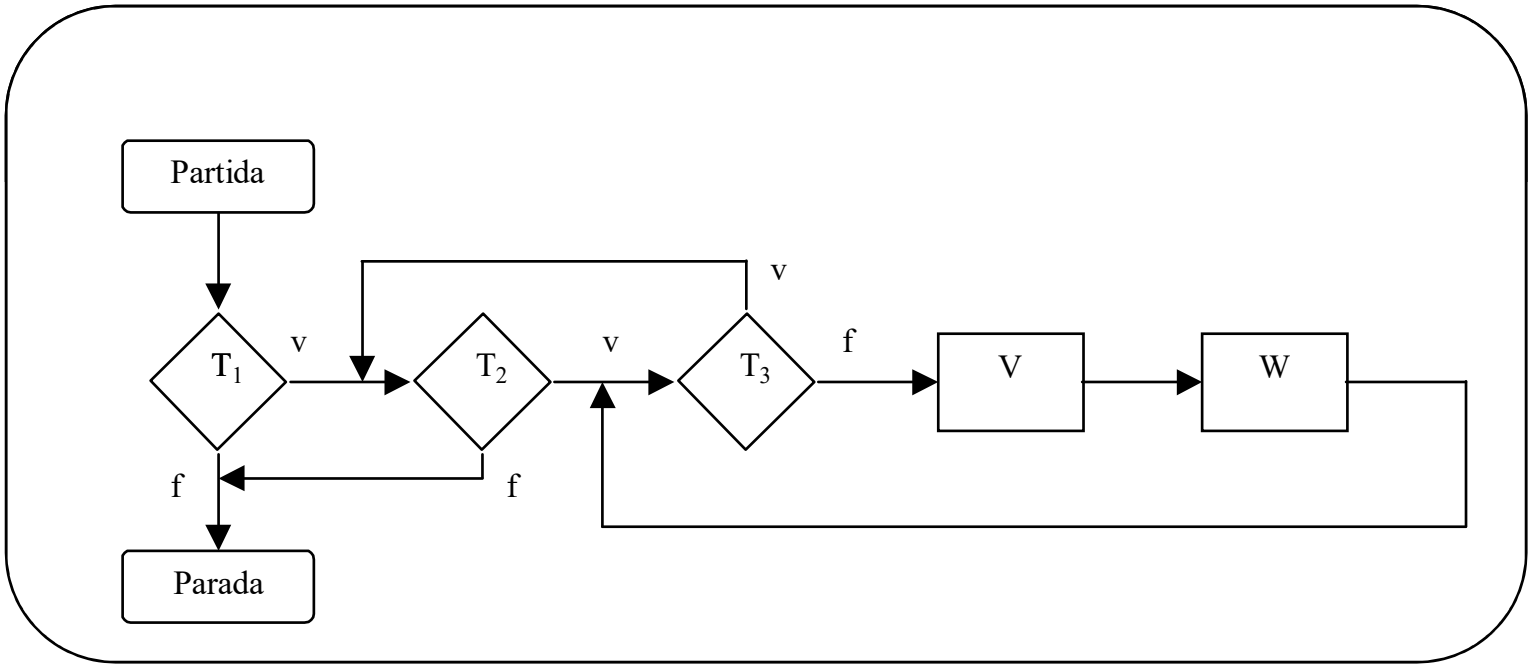
Fig. 2.32 - Monolítico

R_3 *def G;* R_4
R_4 *def (se T2 então* R_5 *senão* R_1 *)*
R_5 *def* ✓

Simplificado
P_R *é* R_1 *onde*
R_1 *def F; (se T1 então* R_1 *senão* R_2 *)*
R_2 *def G; (se T2 então* ✓ *senão* R_1 *)*

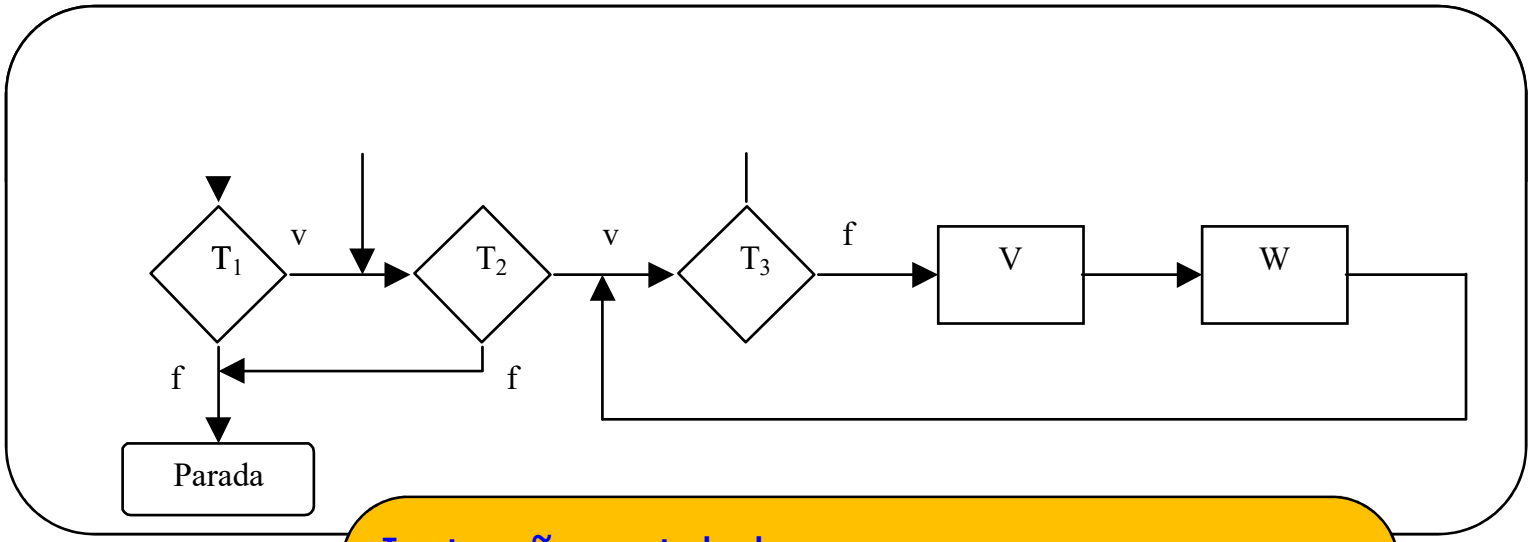
Resolução Exe 2.13 –Iterativo ➔ Monolítico

(se T_1
então enquanto T_2
faça (até T_3
faça (V;W))
senão (✔))



Resolução Exe 2.13 –Iterativo ➔ Monolítico

```
(se    T1
então enquanto T2
      faça  (até  T3
            faça  (V;W))
senão (✔))
```



Instruções rotuladas

1: Se T1 então vá-para 2 senão vá-para 6
2: Se T2 então vá-para 3 senão vá-para 6
3: Se T3 então vá-para 2 senão vá-para 4
4: Faça V vá-para 5
5: Faça W vá-para 3

Resolução Exe 2.17 – Equivalência de programas

Programa Iterativo P

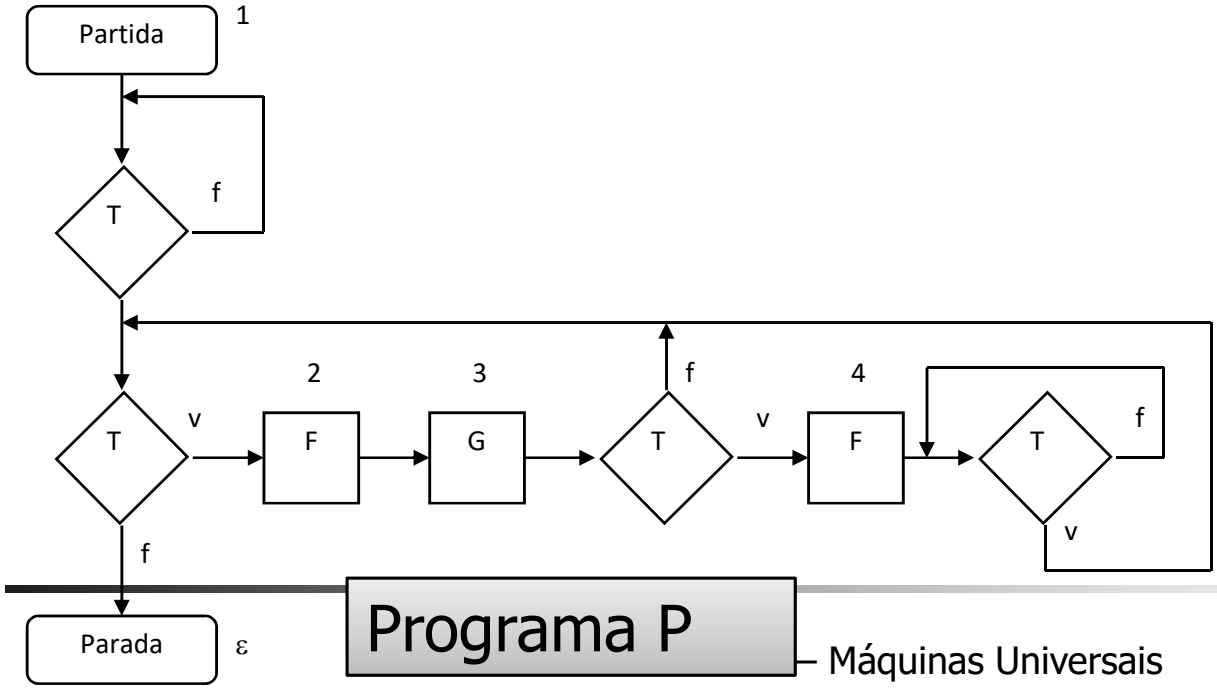
```
até    T
faça   (✔);
enquanto T
faça   (F;G;( se     T
              então F;
                    até    T
                    faça (✔)
              senão ✔))
```

Programa Monolítico Q

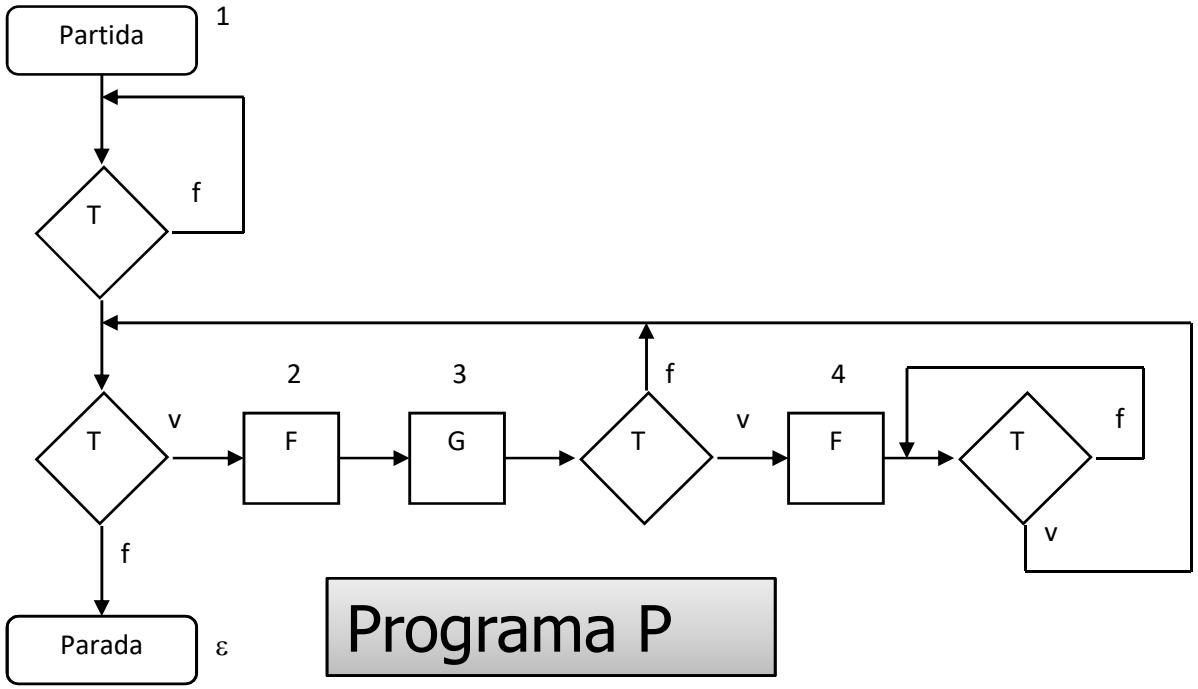
```
1:    se T então vá_para 2 senão vá_para 1
2:    faça F vá_para 3
3:    faça G vá_para 4
4:    se T então vá_para 5 senão vá_para 6
5:    faça F vá_para 1
```

Programa Iterativo P

```
até    T
faça   (✔);
enquanto T
faça   (F;G;( se     T
              então F;
                    até     T
                    faça (✔)
              senão ✔))
```



Programa P

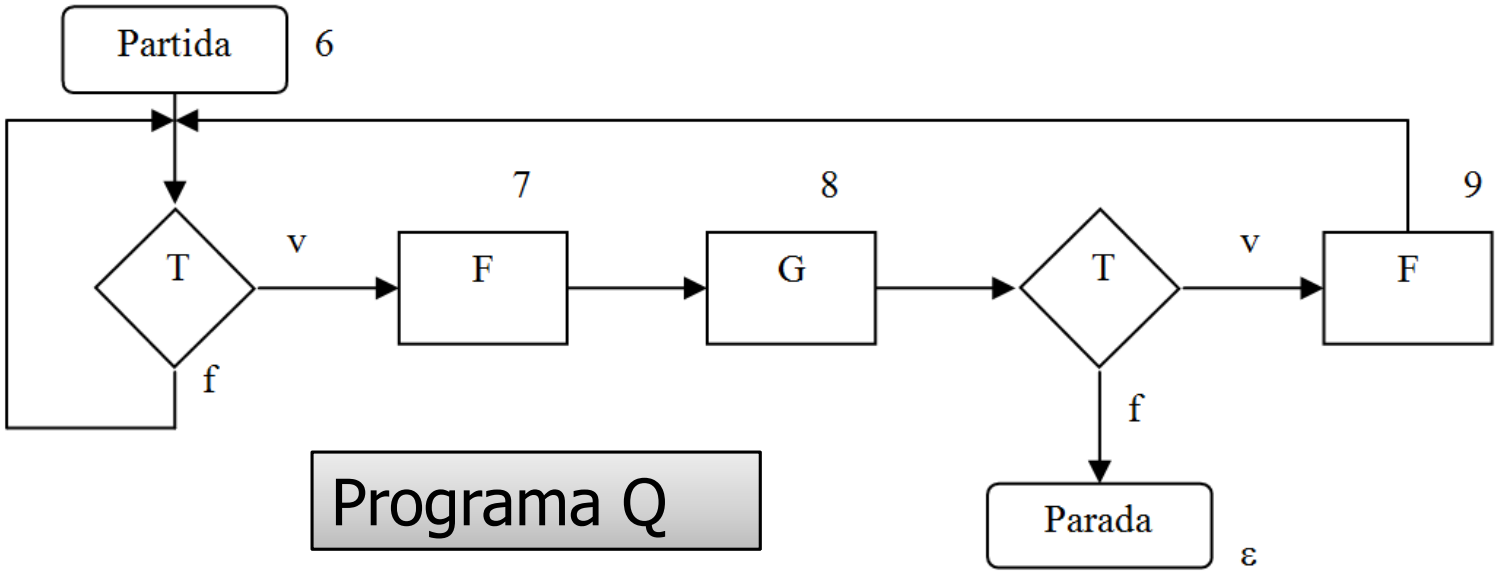


1: (F, 2) (Ciclo, ω)
2: (G, 3) (G, 3)
3: (F, 4) (Parada, ε)
4: (F, 2) (Ciclo, ω)
ω : (Ciclo, ω) (Ciclo, ω)

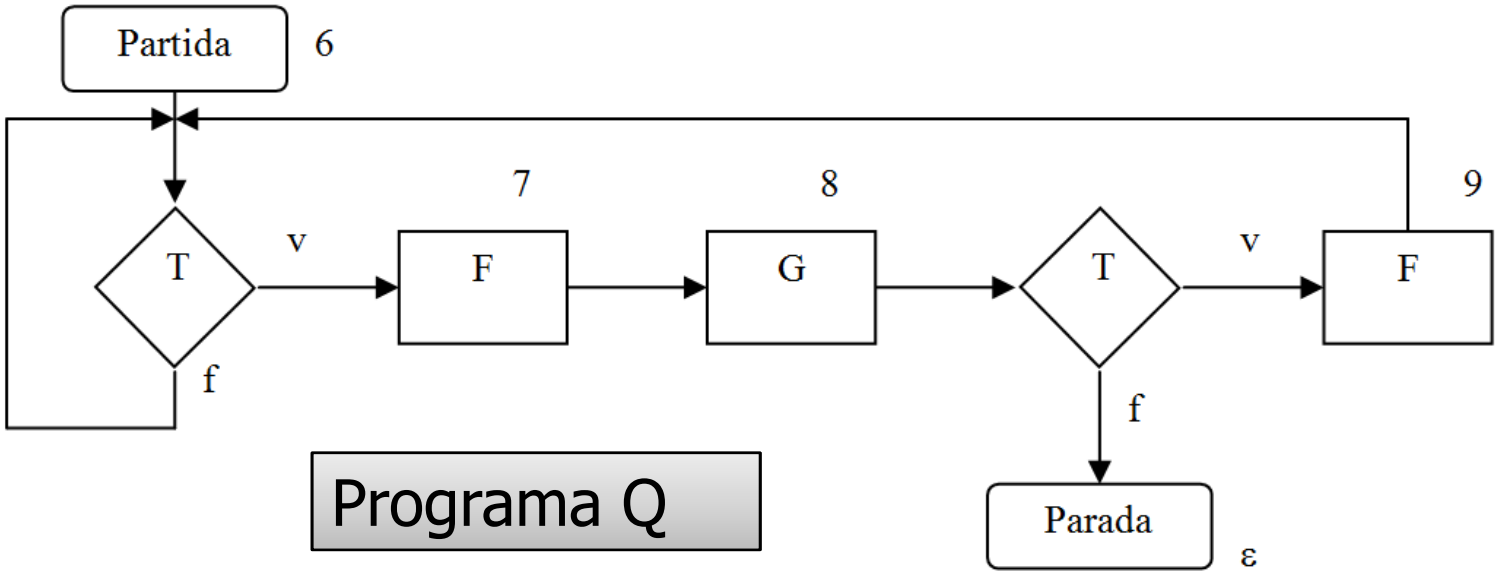
$A_0 = \{\varepsilon\}$
$A_1 = \{3, \varepsilon\}$
$A_2 = \{2, 3, \varepsilon\}$
$A_3 = \{1, 2,' 3, 4, \varepsilon\}$
$A_4 = A_3$

Programa Monolítico Q

```
1:    se T então vá_para 2 senão vá_para 1
2:    faça F vá_para 3
3:    faça G vá_para 4
4:    se T então vá_para 5 senão vá_para 6
5:    faça F vá_para 1
```



Programa Q



Programa Q

6: (F, 7) (Ciclo, ω)
7: (G, 8) (G, 8)
8: (F, 9) (Parada, ε)
9: (F, 7) (Ciclo, ω)
ω : (Ciclo, ω) (Ciclo, ω)

$A_0 = \{\varepsilon\}$
$A_1 = \{8, \varepsilon\}$
$A_2 = \{7, 8, \varepsilon\}$
$A_3 = \{6, 7, 8, 9, \varepsilon\}$
$A_4 = A_3$

Resolução Exe 2.17 – Verificação de Equivalência

Verificação de equivalência:

União dos conjuntos de Instruções Rotuladas Compostas:

1: (F, 2) (Ciclo, ω)
2: (G, 3) (G, 3)
3: (F, 4) (Parada, ε)
4: (F, 2) (Ciclo, ω)

Programa P

6: (F, 7) (Ciclo, ω)
7: (G, 8) (G, 8)
8: (F, 9) (Parada, ε)
9: (F, 7) (Ciclo, ω)

Programa Q

ω : (Ciclo, ω) (Ciclo, ω)

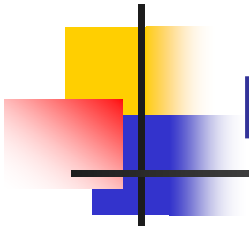
$B_0 = \{(1, 6)\}$
$B_1 = \{(2, 7)\ (\omega,\omega)\}$
$B_2 = \{(3, 8)\}$
$B_3 = (4, 9)\ (\varepsilon,\varepsilon)$
$B_3 = \phi$

Logo, os programas P e Q são *fortemente equivalentes.*

Teoria da Computação

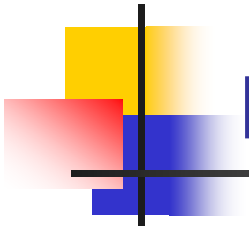
Unidade 3 – Máquinas Universais

Referência – Teoria da Computação (Divério, 2000)



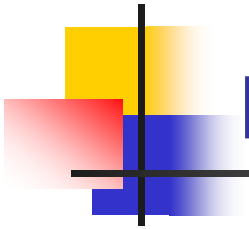
Máquinas Universais

- Algoritmo
  - É uma forma de descrever se determinada propriedade é verificada ou não a partir de uma classe de entrada



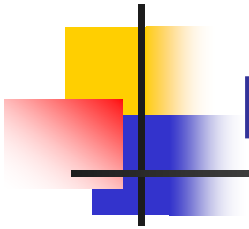
Máquinas Universais

- Noção intuitiva de algoritmo
  - Sua descrição deve ser finita e não-ambígua
  - Deve consistir de passos discretos, executáveis em tempo finito
  - Limitações de tempo ou espaço podem determinar se um algoritmo pode ou não ser utilizado na prática
  - É necessário definir a máquina a ser considerada



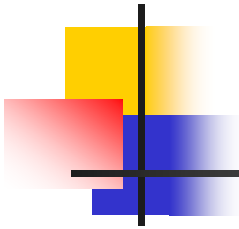
Máquinas Universais

- Máquinas – devem ser:
  - Simples: permite estabelecer conclusões gerais sobre a classe de funções computáveis
  - Poderosa: capaz de simular qualquer característica de máquinas reais ou teóricas, de tal forma que os resultados provados sejam válidos para modelos aparentemente com mais recursos.



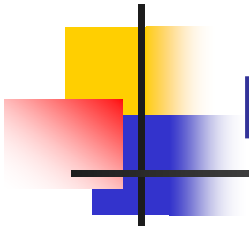
Máquinas Universais

- Máquina Universal - conceito
  - Uma máquina é dita universal se ela for capaz de representar qualquer algoritmo como um programa



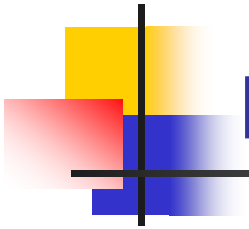
Máquinas Universais

- As evidências que permitem caracterizar uma máquina como universal:
  - Evidência Interna: Qualquer extensão das capacidades da máquina universal, computa, no máximo, a mesma classe de funções, ou seja, não aumenta o seu poder computacional
  - Evidência Externa: Consiste no exame de outros modelos que definem a noção de algoritmo, juntamente com a prova de que são, no máximo, computacionalmente equivalentes.



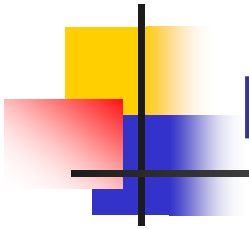
Máquinas Universais

- Máquina Universal – modelos a serem estudados
  - Máquina Norma
  - Máquina de Turing
  - Máquina de Post
  - Máquina com Pilhas



Máquinas Universais

- Exemplo de Máquina Universal
  - Máquina Norma
    - Conjunto de registradores naturais e somente três operações sobre eles
      - Adição e subtração do valor um
      - Teste se o valor armazenado é zero

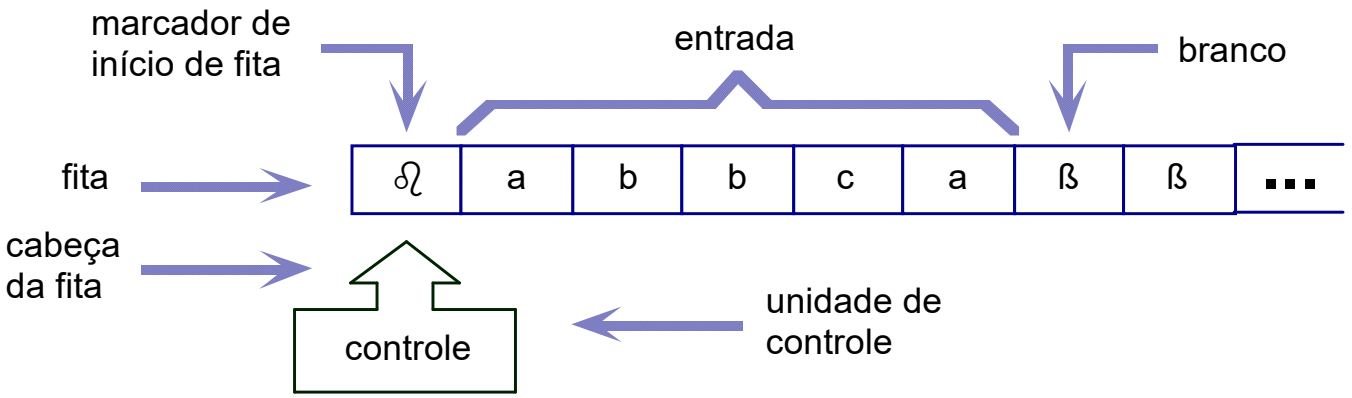


Máquinas Universais

- Máquina de Turing, proposta em 1936 por Alan Turing
  - é um mecanismo simples que formaliza a ideia de uma pessoa que realiza cálculos usando um instrumento de escrita e um apagador
  - Formada por:
    - uma fita (usada para entrada, saída e rascunho),
    - uma unidade de controle
    - um programa

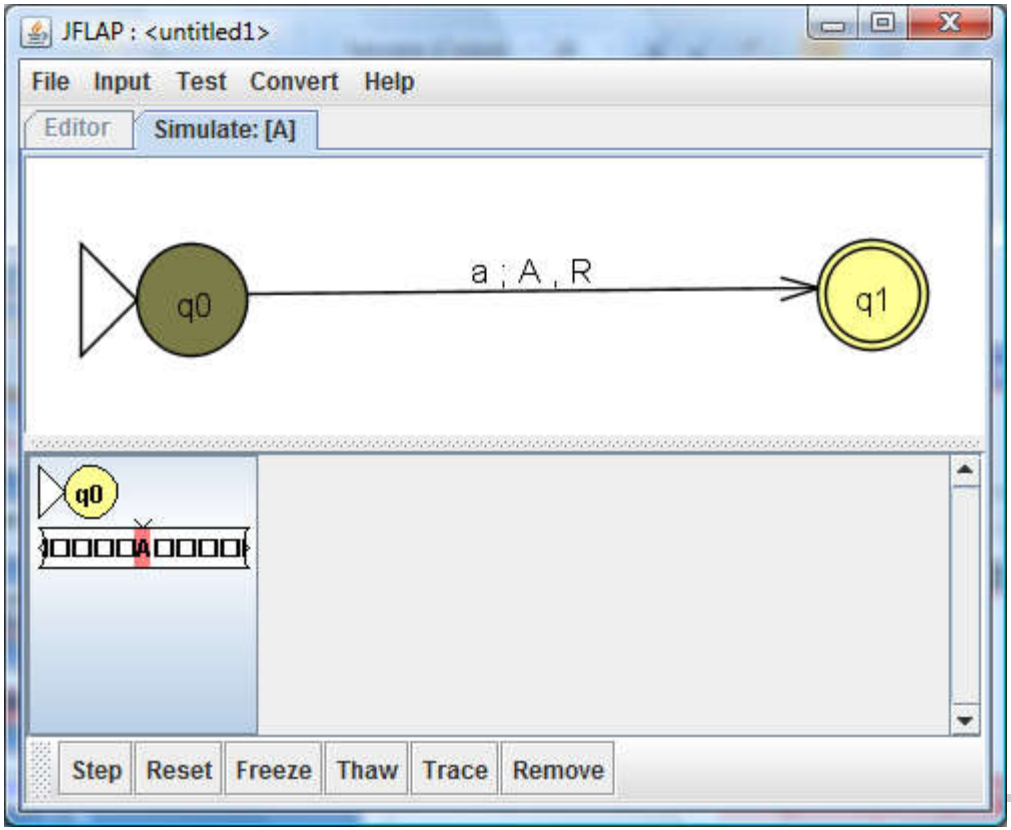
Máquinas Universais

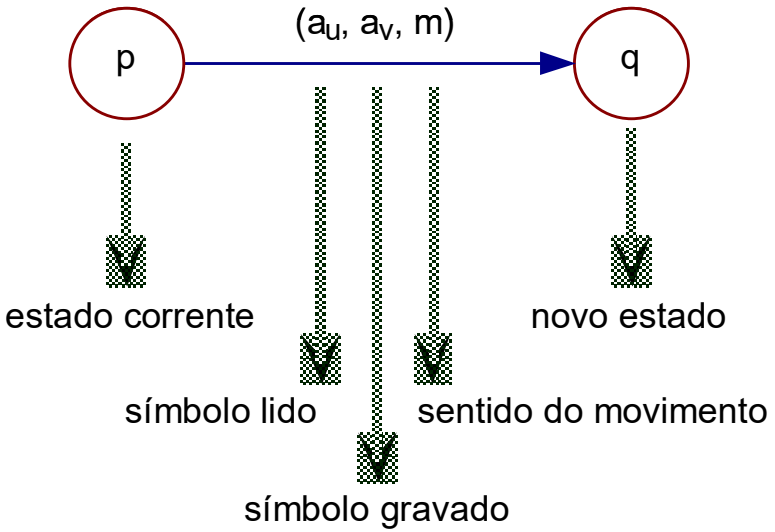
- Máquina de Turing



Máquinas Universais

- Máquina de Turing

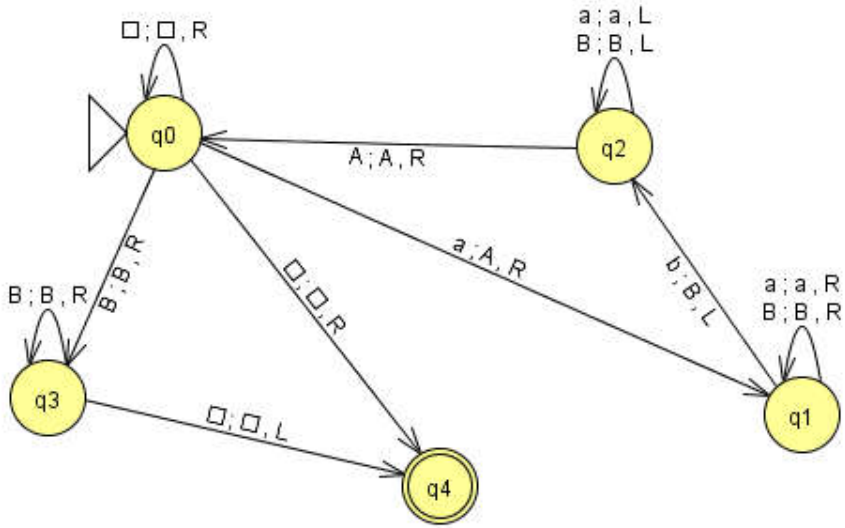




Máquinas Universais

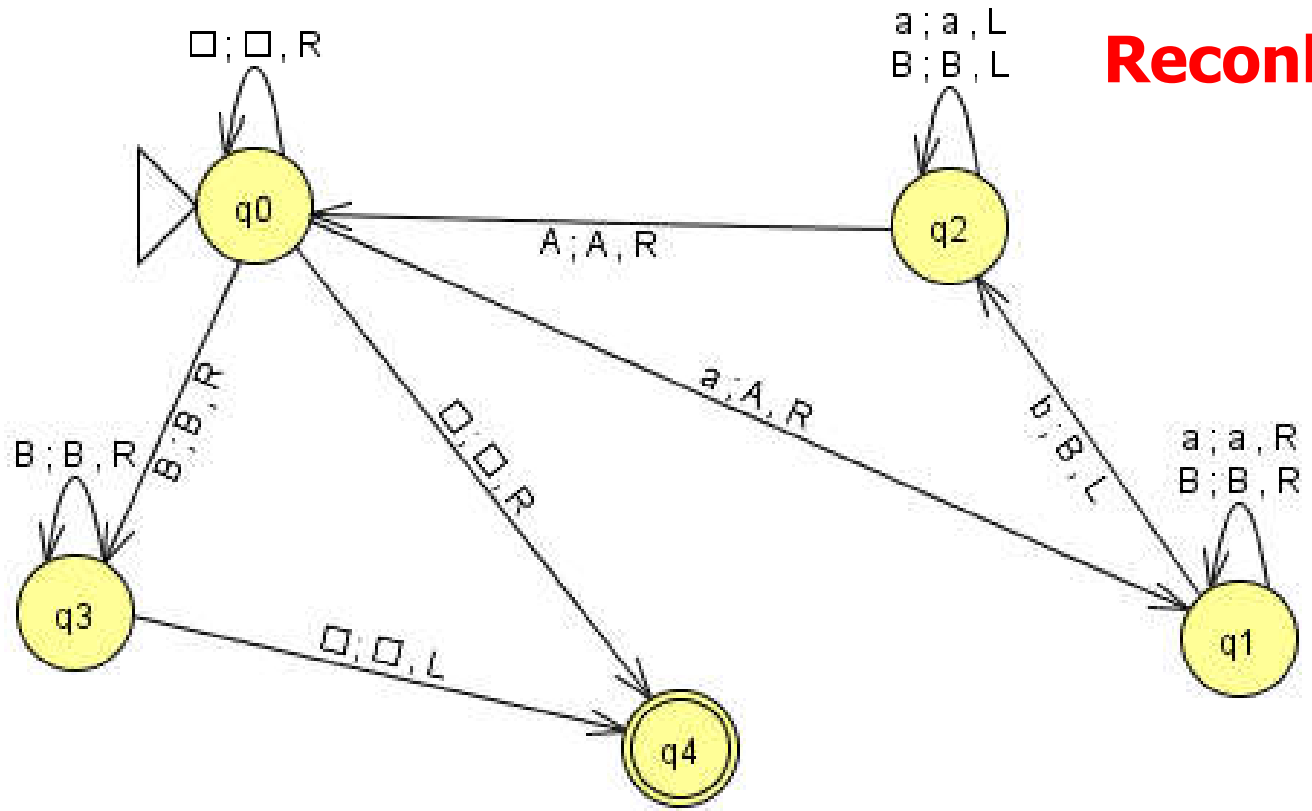
- Máquina de Turing – formas de representação

Π	⊙	a	b	A	B	ß
Q_0	$(q_0, \odot, D)$	(q_1, A, D)			(q_3, B, D)	$(q_4, ß, D)$
q_1		(q_1, a, D)	(q_2, B, E)		(q_1, B, D)	
q_2		(q_2, a, E)		(q_0, A, D)	(q_2, B, E)	
q_3					(q_3, B, D)	$(q_4, ß, E)$
q_4						

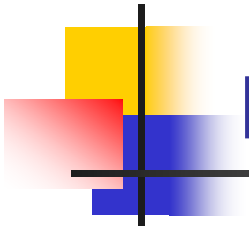


Máquinas Universais

- Máquina de Turing – exemplo

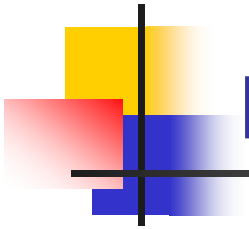


Reconhece $a^n b^n$



Máquinas Universais

- Máquina Norma X Máquina de Turing
  - Os registradores da máquina Norma podem assumir qualquer valor natural tão grande quanto necessário e, existem tantos registradores quanto necessários
  - A máquina de Turing possui tantas células de armazenamento de dados quanto necessário

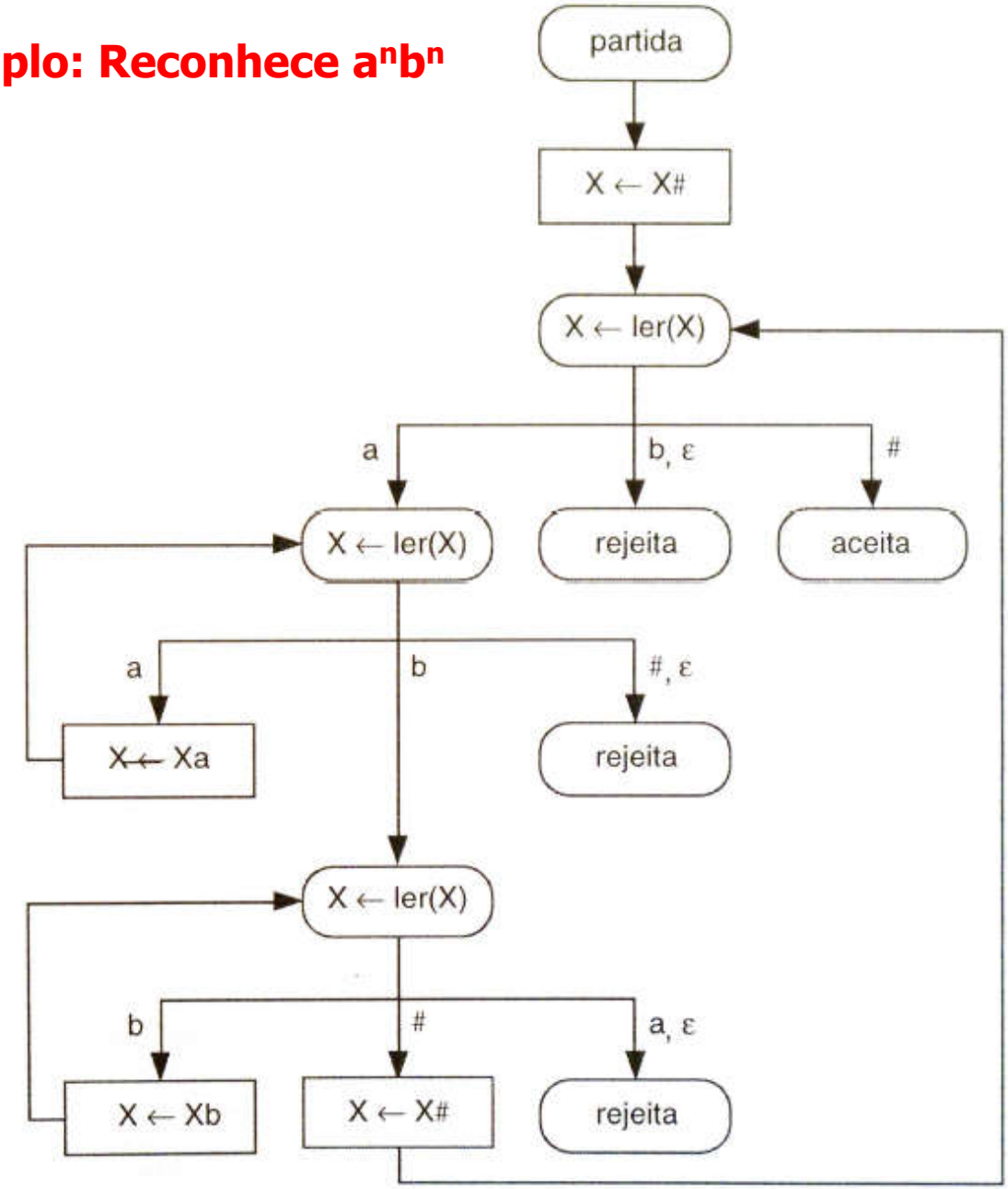


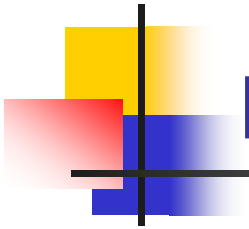
Máquinas Universais

- Máquina de Post: Baseada na estrutura de dados do tipo *fila* (o primeiro dado armazenado é o primeiro a ser recuperado)
  - Fila
    - Entrada, saída e trabalho
    - Tamanho inicial igual ao tamanho da cadeia de entrada
    - Tamanho pode variar sem restrições
  - Possui um programa (fluxograma) associado
    - Partida, parada, desvio condicional e atribuição

Exemplo: Reconhece $a^n b^n$

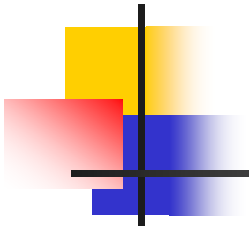
- Máquina de Post
  - X contém entrada
  - # indica final de X
  - ler(x) lê e remove da fila
  - Xn insere n na fila



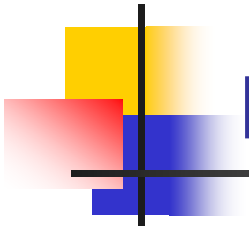


Máquinas Universais

- Máquinas com Pilhas: Baseada na estrutura de dados do tipo *pilha* (o último dado armazenado é o primeiro a ser recuperado), onde são necessárias pelo menos duas pilhas para simular o mesmo poder computacional de uma fita ou fila

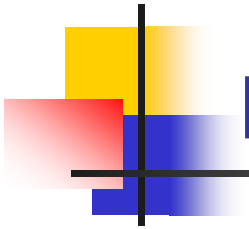


MÁQUINA NORMA COMO MÁQUINA UNIVERSAL



Máquina Norma

- Definida por Richard Bird em 1976
- Norma - **N**umber The**o**retic **R**egister **Ma**chine – nome da esposa dele
- A Máquina Universal Norma possui como memória um conjunto infinito de registradores naturais e três instruções sobre cada registrador: adição e subtração (se 0, continua com 0) do valor um e teste se o valor armazenado é zero.
- Para evitar subscritos, os registradores são denotadas por letras maiúsculas como A, B, X, Y, ...



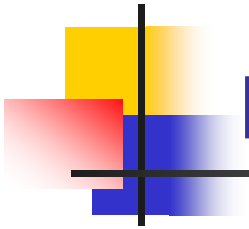
Máquina Norma

Norma = (N^{∞}, X, Y, ent, sai, { ad_K, sub_K }, { $zero_K$ })

onde:

- Cada elemento do conjunto de valores de memória N^{∞} denota uma configuração de seus infinitos registradores, os quais são denotados por: A, B, ..., X, Y

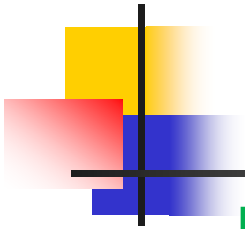
- A função de entrada: ent: $N \rightarrow N^{\infty}$ carrega no registrador denotado por X o valor de entrada, iniciando todos os demais registradores com zero;



Máquina Norma

Norma = (N^{∞}, X, Y, ent, sai, { ad_K, sub_K }, { $zero_K$ })

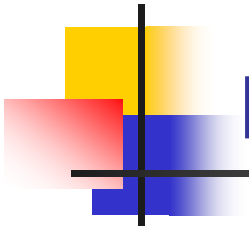
- A função de saída: sai: $N^{\infty} \rightarrow N$ é tal que retorna o valor corrente do registrador denotado por Y;
- O conjunto de interpretações de operações é uma família de operações indexada pelos registradores, na qual, para cada registrador K $\in$ { A, B, X, Y,... }, tem-se que:
  - ad_K: $N^{\infty} \rightarrow N^{\infty}$ adiciona um à componente correspondente ao registrador K, deixando as demais com seus valores inalterados. K:=K+1



Máquina Norma

Norma = (N^{∞}, X, Y, ent, sai, { ad_K, sub_K }, { $zero_K$ })

- O conjunto de interpretações de operações é uma família de operações indexada pelos registradores, na qual, para cada registrador K $\in$ { A, B, X, Y,... }, tem-se que:
  - sub_K: $N^{\infty} \rightarrow N^{\infty}$ subtrai um da componente correspondente ao registrador K, se o seu valor for maior que zero (caso contrário, mantém o valor zero), deixando as demais com seus valores inalterados. K:=K-1
- O conjunto de interpretações de testes é indexada pelos registradores na qual, para cada registrador K, tem-se que:
  - $zero_K$: $N^{\infty} \rightarrow$ { verdadeiro, falso } resulta em verdadeiro, se a componente correspondente ao registrador K for zero e em falso, caso contrário. K=0?



Máquina Norma

- É uma máquina extremamente simples, e o seu poder computacional é, no mínimo, o de qualquer computador moderno

- Características de máquinas reais são simuladas usando a Máquina Norma, reforçando as evidências de que se trata de uma máquina universal.

Máquina Norma

Simulações suportadas

a) Operações e Testes: Definição de operações e testes mais complexos como adição, subtração, multiplicação e divisão de dois valores e tratamento de valores diversos como os números primos;

b) Valores Numéricos: Armazenamento e tratamento de valores numéricos de diversos tipos como inteiros (negativos e não-negativos) e racionais;

c) Dados Estruturados: Armazenamento e tratamento de dados estruturados como em arranjos (vetores uni e multidimensionais), pilhas, etc;

d) Endereçamento Indireto e recursão: Desvio para uma instrução determinada pelo conteúdo de um registrador;

e) Cadeia de Caracteres: Definição e manipulação de cadeias de caracteres.

Máquina Norma

a) Operações e Testes

- Atribuição do Valor Zero a um Registrador (A:=0)
  - Programa Iterativo

```
A := 0
até     A = 0
   faça (A := A – 1)
```

  - Representada pela macro A := 0
  - Usando a macro A := 0, é fácil construir macros para definir operações de atribuição de um valor qualquer.

Máquina Norma

- Atribuição de um Valor Natural a um Registrador (macro: A:=n)
  - Programa Iterativo (para n=3)

A := n

```
A := 0;
A := A+1;
A := A+1;
A := A+1
```

Máquina Norma

- Adição de Dois Registradores

(macro: A:=A+B)

- Programa Iterativo

A := A + B

até **B = 0**
faça **(A:=A+1; B:=B-1)**

  - Observe que, ao somar o valor de B em A, o registrador B é zerado!

Máquina Norma

- Adição de Dois Registradores Preservando o Conteúdo (B) (macro: A := A + B usando C)
  - Programa Iterativo

A := A + B usando C

```
C := 0;
até     B = 0
faça    (A := A + 1; C := C + 1; B := B - 1);
até     C = 0
faça    (B := B + 1; C := C - 1)
```

Máquina Norma

- Atribuição do Conteúdo de um Registrador (macro: A := B usando C)
  - Programa Iterativo

A := B usando C

```
A := 0;
A := A + B usando C
```

  - B permanece inalterado após a atribuição

Máquina Norma

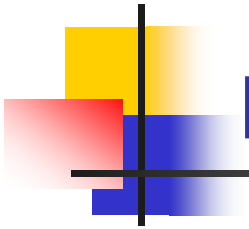
- Multiplicação de Dois Registradores

(macro: A := A x B usando C, D)

- Programa Iterativo

A := A x B usando C, D

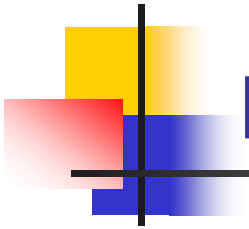
```
C := 0;
até     A = 0
faça    (C := C + 1; A := A - 1);
até     C = 0
faça    (A := A + B usando D; C := C - 1)
```



Máquina Norma

b) Valores Numéricos

- Os tipos de dados não definidos na Norma:
  - inteiros (negativos e positivos)
  - racionais



Máquina Norma

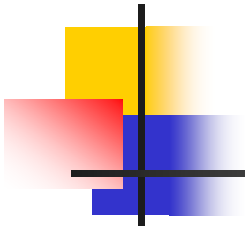
- Inteiros
  - Um valor inteiro m pode ser representado como um par (s, |m|), onde:
    - $|m|$ denota magnitude dada pelo valor absoluto de m;
    - s denota o sinal de m: se m < 0, então s = 1 (negativo) senão s = 0 (positivo)

Máquina Norma

- Inteiros
  - Supondo que o registrador inteiro A é representado pelo par (A_1, A_2) na representação conhecida como *sinal-magnitude*, ou seja, A1 (sinal) e A2 (magnitude).
  - Programa em Norma para executar a operação A := A+1
  - Programa Iterativo

A := A+1

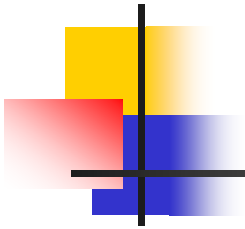
```
(se  A1 = 0
     então   A2 := A2 + 1
     senão   A2 := A2 - 1;
             (se      A2 = 0
             então    A1 := A1 – 1
             senão    V) )
```



Máquina Norma

b) Valores Numéricos

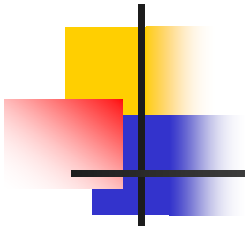
- Racionais
  - Um valor racional r pode ser denotado como um par ordenado: (a, b) tal que b > 0 e r = a/b.
  - A representação não é única pois, por exemplo, o valor racional 0.75 pode ser representado pelos pares (3, 4) e (6, 8), entre outros.



Máquina Norma

b) Valores Numéricos

- Racionais: Neste contexto, as operações de adição, subtração, multiplicação e divisão, bem como o teste de igualdade, podem ser definidos como segue:
  - (a, b) + (c, d) = (a*d + b*c, b*d)
  - (a, b) - (c, d) = (a*d – b*c, b*d)
  - (a, b) * (c, d) = (a*c, b*d)
  - (a, b) / (c, d) = (a*d, b*c) (com c ≠0)
  - (a, b) = (c, d) se, e somente se, a*d = b*c



Máquina Norma

b) Valores Numéricos

- Racionais: exemplos
  - Adição 0,25 + 0,75 = 1

    (1, 4) + (3, 4) = (4 + 12, 16) = (16, 16)

    (a, b) + (c, d) = (a*d + b*c, b*d)

  - Multiplicação 0,25 * 0,75 = 0,1875

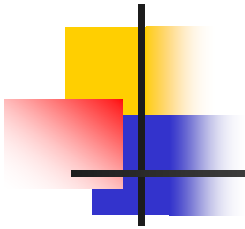
    (1, 4) * (3, 4) = (3, 16)

    (a, b) * (c, d) = (a*c, b*d)

  - Teste de igualdade

    (1, 4) = (3, 12) e verdadeiro pois, 1*12 = 4*3

    (a, b) = (c, d) se, e somente se, a*d = b*c



Máquina Norma

c) Codificação de conjuntos estruturados

- Elementos de tipos de dados estruturados podem ser representados como números naturais, através do teorema fundamental da aritmética. Cada número natural é decomposto em seus fatores primos;

- Cadeias de caracteres podem ser representados através da codificação de n-uplas naturais

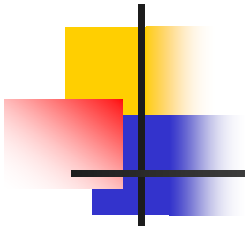
Máquina Norma

c) Dados Estruturados

- Arranjo Unidimensional
  - Uma estrutura da forma A(1), A(2), ..., pode ser representada por um único registrador A, usando a codificação de n-uplas naturais – Codificação de conjuntos estruturados. Exemplo:

A(1)=5; A(2)=2; A(3)=0

Então o valor armazenado em A é: $2^5 \mathbf{x} 3^2 \mathbf{x} 5^0 \mathbf{x} 7^0 \mathbf{x} 11^0 \mathbf{x} 13^0 \ldots = 32x9x1 = 288$

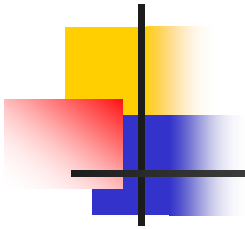


Máquina Norma

c) Dados Estruturados

- Arranjo Unidimensional
  - Não necessita ter tamanho máximo pré-definido;
  - A função de entrada carrega o valor de entrada no registrador X, zerando os demais, incluindo, portando, o arranjo.
  - Em uma estrutura do tipo arranjo, é desejável indexar as posições de forma direta (número natural) ou indireta (conteúdo de um registrador)

    - Operações:
      - O arranjo é implementado usando o registrador A
      - p_n denota o **n**-ésimo número primo
      - Adiciona 1 à posição indexada
      - Subtrai 1 de uma posição indexada
      - Testa se uma posição indexada tem o valor 0



Máquina Norma

c) Dados Estruturados

- Arranjo Unidimensional
  - indexação direta. As macros:

    $ad_{A(n)}$ usando C

    $sub_{A(n)}$ usando C

    $zero_{A(n)}$ usando C

    Onde $_{A(n)}$ denota a n-ésima posição do arranjo A

Máquina Norma

- Arranjo Unidimensional – exemplos
  - indexação direta.

- Programa iterativo $ad_{A(n)}$ usando C

  C:=p_n;
  A:=AxC

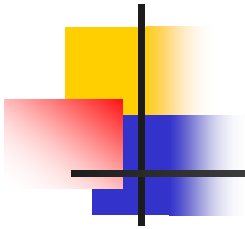
- Programa iterativo $sub_{A(n)}$ usando C

  C:=p_n;
  (se teste_div(A,C)
  então A:=A/C
  senão v)

- Programa iterativo $zero_{A(n)}$ usando C

  C:=p_n;
  (se teste_div(A,C)
  então falso
  senão verdadeiro)

teste_div(A,C) → é uma macro que retorna Verdadeiro se C é um divisor de A



Máquina Norma

c) Dados Estruturados

- Arranjo Unidimensional
  - indexação indireta. As macros:

    $ad_{A(B)}$ usando C

    $sub_{A(B)}$ usando C

    $zero_{A(B)}$ usando C

    Onde $_{A(B)}$ denota a b-ésima posição do arranjo A, onde B é o conteúdo do registrador B

Máquina Norma

- Arranjo Unidimensional – exemplos
  - indexação indireta.

- Programa iterativo $ad_{A(B)}$ usando C

  C:=primo(B)
  A:=AxC

- Programa iterativo $sub_{A(B)}$ usando C

  C:=primo(B)
  (se teste_div(A,C)
  então A:=A/C
  senão v)

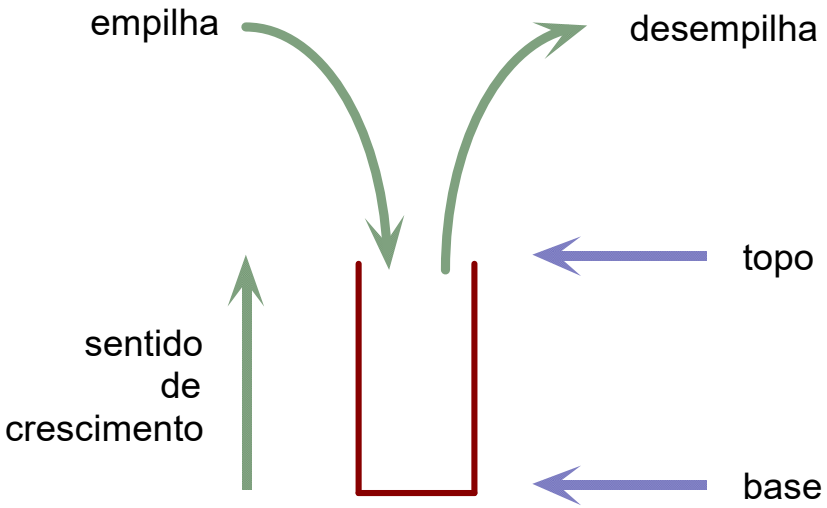
- Programa iterativo $zero_{A(n)}$ usando C

  C:=primo(B)
  (se teste_div(A,C)
  então falso
  senão verdadeiro)

Máquina Norma

c) Dados Estruturados

- Pilha: pode ser simulada usando 2 registradores
  - O primeiro representa o conteúdo da pilha, considerado como um vetor
  - O segundo contém o número do elemento que corresponde ao topo da pilha
  - Operações:
    - *Empilha*
    - *desempilha*



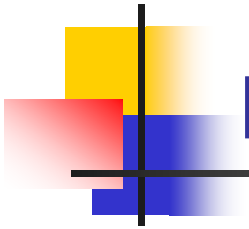
Máquinas Universais

d) Cadeias de Caracteres

- Cadeia de caracteres é outro tipo de dado não pré-definido na Máquina Norma.
- O tratamento da definição e da manipulação de cadeias de caracteres será realizado através de uma outra Máquina Universal, denominada Máquina de Turing, a qual prova-se, é equivalente à Norma.

F = 6	"FADA" = $2^6 * 3^1 * 5^4 * 7^1$
A = 1	"FADA" = $64 * 3 * 625 * 7$
D = 4	"FADA" = 840000
A = 1	

Codificação da palavra "FADA"



Máquinas Universais

- Implemente uma máquina norma – qualquer linguagem – que realize as seguintes operações:
  - Adição entre dois registradores
    - Sem preservar conteúdo – obs: utilize números positivos/negativos
    - Preservando o conteúdo
  - Multiplicação entre dois registradores.
  - Testes *A<B* e *A<=B*
  - Testar se o valor de um registrador é um número primo.
  - Fatorial e potenciação de um número
  - Resto da divisão de valores dois registradores
  - Demais exemplos apresentados
  - Implemente as operações da estrutura Pilha
- Entregar até dia 26/03