Proposta de Solução dos Exercícios do Capítulo 2

Exercício 2.1. Identifique e compare construções análogas às usadas nas definições de programas monolítico, iterativo e recursivo em linguagens de programação como: Pascal, C ou outra de seu conhecimento.

Exemplo de programa monolítico (Cobol):

```
PX.
  IF A > 0 THEN
          B = B + 1
          A = A - 1
          GOTO PX;
PX-EXIT.
```

Em Cobol normalmente os programas são codificados de forma mais estruturada (iterativos), mas também podem ser codificados conforme exemplo acima, com desvios incondicionais (GOTO) para labels de procedures .

Exemplo de programa Iterativo (Pascal):

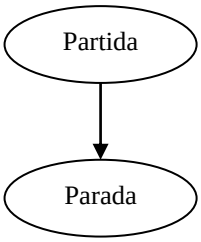
```
WHILE A > 0 DO
  BEGIN
          A := A - 1;
          B := B + 1;
  END;
```

Exemplo de programa Recursivo (Pascal):

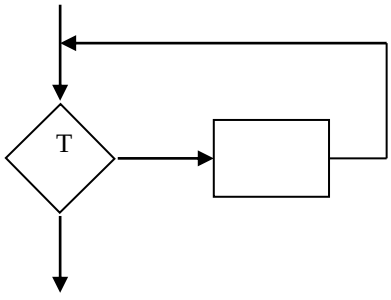
```
PROCEDURE P1;
  BEGIN
          IF A > 0 THEN
             BEGIN
                  A := A - 1;
                  B := B + 1;
                  R;
              END;
  END;
```

Exercício 2.2. Desenhe um fluxograma que corresponde a cada um dos seguintes programas

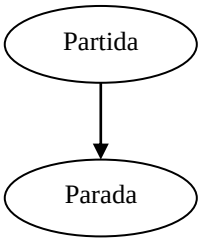
a) $P_2 = (\{r1:$ faça vá_para $r_2\}, r_1)$



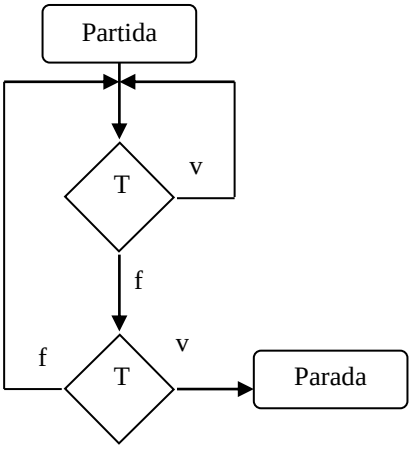
b) Composição até (programa iterativo)



c) Programa sem instrução alguma



d) Programa sem instrução de parada



Exercício 2.3. Relativamente a programas iterativos:

a) **Em que situação a execução de: enquanto T faça V**
V pode não ser executado?

V não será executado quando o resultado do teste T for falso na primeira vez em que o teste for executado.

b) Por que a operação constitui uma programa iterativo?

Por definição de Programa iterativo, temos que cada identificador de operação e a operação constitui um programa iterativo.

c) Por que pode-se afirmar que:
A tradução de um programa iterativo para um monolítico é trivial.
Porque para cada instrução iterativa existe um fluxograma correspondente. Cada instrução iterativa é definida por um fluxograma.

Exercício 2.4. Relativamente a computação:

a) Por que é possível afirmar que a computação de um programa monolítico em uma máquina, para um dado valor inicial de memória é determinístico?
b) Analogamente para um programa iterativo.
c) Analogamente para um programa recursivo.

Para qualquer destas estruturas de programas, dado um valor inicial, em uma mesma máquina, a computação será sempre a mesma (a seqüência de instruções executadas e os valores intermediários de memória, em diversas execuções do mesmo programa, para o mesmo valor de entrada, será sempre a mesma). O resultado da execução do mesmo programa, na mesma máquina, com mesmo valor inicial, será sempre o mesmo.

Exercício 2.5. Caracterize e diferencie computação e função computada.
Computação é a seqüência de instruções executadas para um programa e os diferentes valores da memória ao longo de sua execução (histórico do funcionamento da máquina para o programa). Função computada é o resultado obtido ao fim de uma computação, desde que a computação seja finita.

Exercício 2.6. Defina computação de programas iterativos em uma máquina.
Sejam $M=(V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$ uma máquina e P um programa iterativo para M. Uma computação do programa P na máquina M é uma cadeia finita ou infinita de pares:

$$(X_0,v_0)\ (X_1,v_1)\ (X_2,v_2)....$$

onde (X_0,v_0) é tal que X_0 é o programa iterativo inicial e v_0 é o valor inicial da memória e, para cada par (X_K,v_K) da cadeia, onde $K \in \{0,1,2,...\}$ tem-se que (suponha que F é um identificador de operação, T é um identificador de teste e V, W, X são programas iterativos):

(i) X_K são programas do tipo Iterativo, os quais, para executar, pega-se sempre programas em composição do tipo W; V onde V pode ser igual a □, que também por definição é do tipo iterativo.
a) Caso 1. Se X_K é um programa iterativo da forma: $X_K = F;W$
então $(X_{K+1}, v_{K+1}) = (W, \pi_F(v_K))$

b) Caso 2. Se X_K é um programa iterativo da forma:

X_K: se T então V senão W; X
então, (X_{K+1}, v_{K+1}) é par subseqüente de (X_K,v_K) na cadeia sendo que:
X_{K+1}=V;X se $\pi_T(v_K)$ = verdadeiro
X_{K+1}=W;X se $\pi_T(v_K)$ = falso
v_{K+1}= v_K

c) Caso 3. Se X_K é um programa iterativo da forma:

X_K: enquanto T faça (V); W
então, (X_{K+1}, v_{K+1}) é par subseqüente de (X_K,v_K) na cadeia sendo que:
X_{K+1}=V; enquanto T faça (V); W se $\pi_T(v_K)$ = verdadeiro
X_{K+1}=W se $\pi_T(v_K)$ = falso
v_{K+1}= v_K

d) Caso 4. Se X_K é um programa iterativo da forma:

X_K: até T faça (V); W
então, (X_{K+1}, v_{K+1}) é par subseqüente de (X_K,v_K) na cadeia sendo que:
X_{K+1}= W se $\pi_T(v_K)$ = verdadeiro
X_{K+1}=V; até T faça (V); W se $\pi_T(v_K)$ = falso
v_{K+1}= v_K

A cadeia 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;
□ Um teste não altera o valor corrente da memória;
□ Em uma computação finita a expressão □ ocorre no último par da cadeia (X_n = □);

Exercício 2.7. Dê a definição formal da função computada <W, M> de um programa iterativo W em uma máquina M.
Sejam $M=(V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$ uma máquina e P um programa iterativo para M. A Função Computada pelo Programa Iterativo P na Máquina M denotado por

$$< P, M > : X \square Y$$

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

$$(X_0,v_0)\ (X_1,v_1),\ldots,\ (X_n,v_n)$$

onde, o valor inicial da memória é dado pela função de entrada, ou seja, $v_0 = \pi_X(x)$ e $X_n = \square$. Nesse caso a imagem de x é dada pela função de saída aplicada ao último valor da memória na computação, ou seja:

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

Exercício 2.8. Defina computação finita para programas iterativos.
A definição de computação para programas iterativos está feita no exercício 2.6. Precisamos apenas considerar que a cadeia de pares é finita:

$$(X_0, v_0)\ (X_1, v_1)\ldots(X_n, v_s)$$

Exercício 2.9. Escreva um programa iterativo onde a computação seja finita.
enquanto a = a
faça F

Exercício 2.10. Relativamente à função computada por um programa em uma máquina:
a) Considere o programa monolítico mon_b←a (Figura 2.10) para a máquina dois_reg (Figura 2.7). A correspondente função computada <mon_b←a, dois-reg> é total?
Sim, pois para cada elemento do domínio (entrada) existe um único elemento da imagem (saída).

b) Considere o programa monolítico comp_infinita (Figura 2.12) para a máquina dois_reg (Figura 2.7). Para que valores do domínio a função computada <comp_infinita, dois_reg> é definida?
Esta função não é definida para valor algum do domínio, ou seja, para qualquer valor do domínio a função é indefinida)

c) Considere o programa recursivo qq_máquina (Figura 2.14) e uma máquina M = (V, X, Y, π_X, π_Y, Π_F, Π_T) qualquer. Por que a correspondente função computada <qq_máquina, M> é indefinida para qualquer entrada?
A correspondente função computada é indefinida para qualquer entrada porque não existe nenhuma condição que faça com que o programa R pare de ser chamado indefinidamente. A cadeia gerada pela computação deste programa é infinita.

Exercício 2.11. Relativamente aos seguintes teoremas:

a) Teorema Equivalência Forte de Programas ↔ Máquina de Traços;

a.1) Justifique a afirmação de que a prova (→) é imediata;

É imediata porque P = Q se, e somente se, para qualquer máquina M as correspondentes funções parciais computáveis são iguais. Logo, como a afirmação é válida para qualquer máquina, então será válida para a máquina de traços em específico, ou seja, P e Q são fortemente equivalentes para máquina de traços.

a.2) Esboce a prova (←) para programas iterativo e recursivos;

b) Justifique a afirmação de que a prova do Lema Equivalência Forte de programas Fluxograma → Instruções Rotuladas Compostas é imediata;

c) Por que o Lema Equivalência Forte de programas Fluxograma → Instruções Rotuladas Compostas garante que Pq≡Pr se, e somente se, Q≡R;

d) Justifique o Corolário Equivalência Forte de Programas ↔ Máquina de Traços.

Exercício 2.12. Traduza os programas monolíticos representados por fluxogramas em programas recursivos e simplifique se possível.
a) Figura 2.30
P_R é R_1 onde
R_1 def (se T1 então R_2 senão R_3)
R_2 def F; R_3

R_3 def (se T2 então R_4 senão R_7)
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)

b) Figura 2.31
P_R é R_1 onde
R_1 def (se T então R_2 senão R_3)
R_2 def G; R_6
R_3 def F; R_4
R_4 def (se T então R_6 senão R_5)
R_5 def F; R_1
R_6 def ✓

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

c) Figura 2.32
P_R é R_1 onde
R_1 def F; R_2
R_2 def (se T1 então R_1 senão R_3)
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)

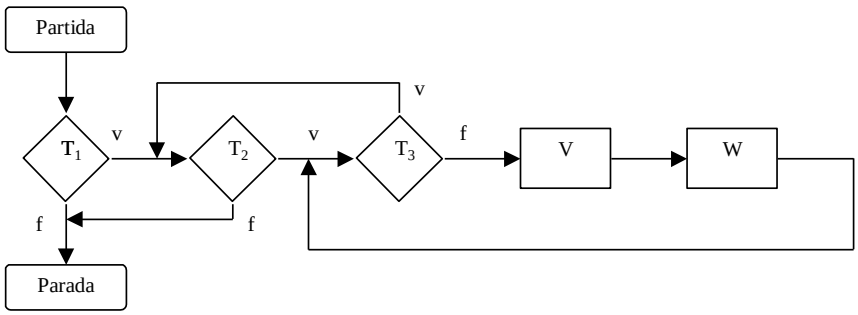
d) Figura 2.33
P_R é R_1 onde
R_1 def (se T então R_2 senão R_3)
R_2 def F; R_1
R_3 def ✓
Simplificado
P_R é R_1 onde
R_1 def (se T então F; R_1 senão ✓)

e) Figura 2.34
P_R é R_1 onde
R_1 def (se T então R_2 senão R_4)
R_2 def F; R_3
R_3 def (se T então R_1 senão R_4)
R_4 def ✓

Simplificado
P_R é R_1 onde
R_1 def (se T então F; R_1 senão ✓)

Exercício 2.13. Traduza o programa iterativo da Figura 2.35 em um programa monolítico na forma de fluxograma e em instruções rotuladas.

a) Fluxograma:



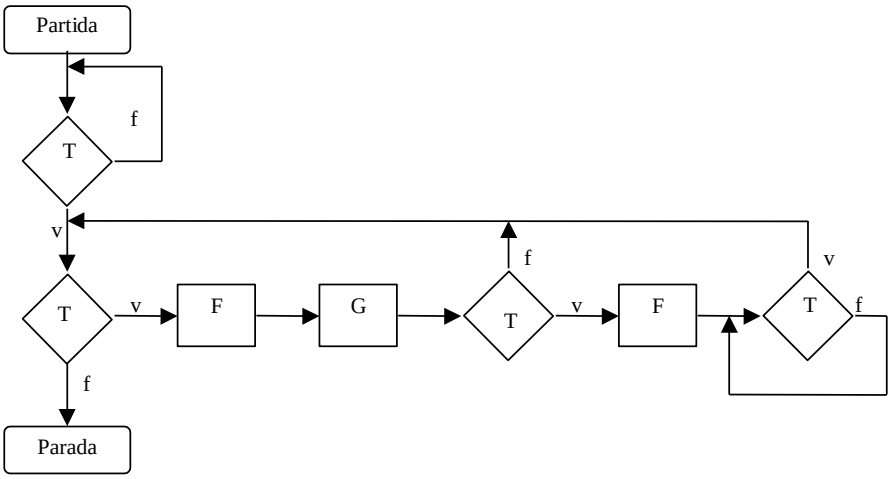
b) Instruções Rotuladas

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

Exercício 2.14. Traduza o programa recursivo representado na Figura 2.36 em programa iterativo.

Programa Monolítico:
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

Fluxograma:



Programa Iterativo:
até T faça ✓
enquanto T faça F; G; (se T então (F; até T faça ✓) senão faça ✓)

Exercício 2.15 Suponha que se escreva $M_1 \leq M_2$ se a máquina M_2 simula M_1, mas pode ter na sua definição outros testes e operações a mais, Qual é a relação entre $<P, M_1>$ e $<P', M_2>$ se $M_1 \leq M_2$? Qual a relação do poder computacional da máquina M_1 em relação à máquina M_2?

Se a máquina M_2 é compatível com a máquina M_1, podendo ter mais testes e operações, sabe-se que qualquer programa P que tenha uma computação finita em M_1 também o terá em M_2. Logo, suas funções computadas em ambas as máquinas serão iguais. Pode ocorrer, entretanto, que um programa Q tenha

uma computação finita em M_2 mas não em M_1. Neste caso, a computação de Q em M_1 pode não ser definida ou estar em loop. O poder computacional de M_1 não necessariamente é menor do que M_2. Os novos testes e operações de M_2 podem simplesmente facilitar a programação, mas não possibilitar a resolução de um novo problema.

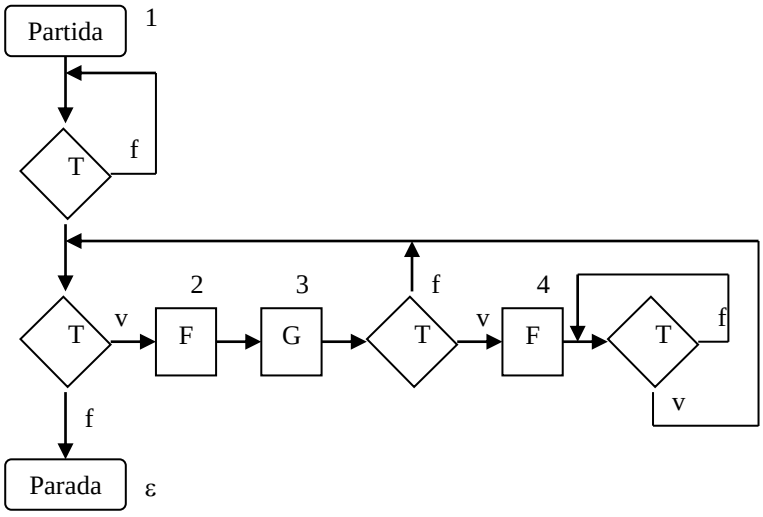
Exercício 2.16. Suponha que na definição de uma máquina sejam admitidas funções parciais na especificação dos indicadores de operações e testes. Dê a definição apropriada da função computada por um programa em uma máquina em tal caso.

Para os valores do conjunto de memória para os quais as operações e os testes estão definidos, se a computação termina, define-se a função computada como o valor contido na memória no instante em que a computação termina (função de saída). E quando a computação não termina, defini-se que a função computada fica em loop, representada pelo símbolo ω.

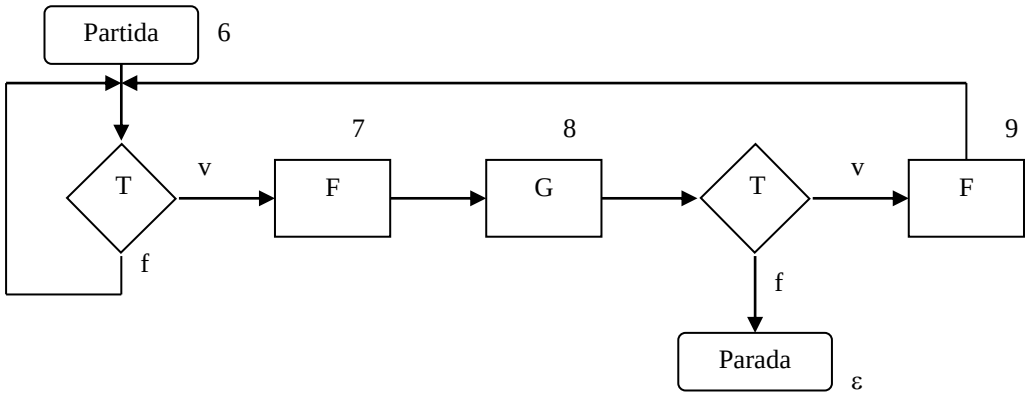
Exercício 2.17. Mostre que os seguintes programas P e Q representados na Figura 2.37 são equivalentes.

Programa P:

1: (F, 2) (Ciclo, ω) — $A_0 = \{\varepsilon\}$
2: (G, 3) (G, 3) — $A_1 = \{3, \varepsilon\}$
3: (F, 4) (Parada, ε) — $A_2 = \{2, 3, \varepsilon\}$
4: (F, 2) (Ciclo, ω) — $A_3 = \{1, 2, 3, 4, \varepsilon\}$
ω : (Ciclo, ω) (Ciclo, ω) — $A_4 = A_3$



Programa Q:

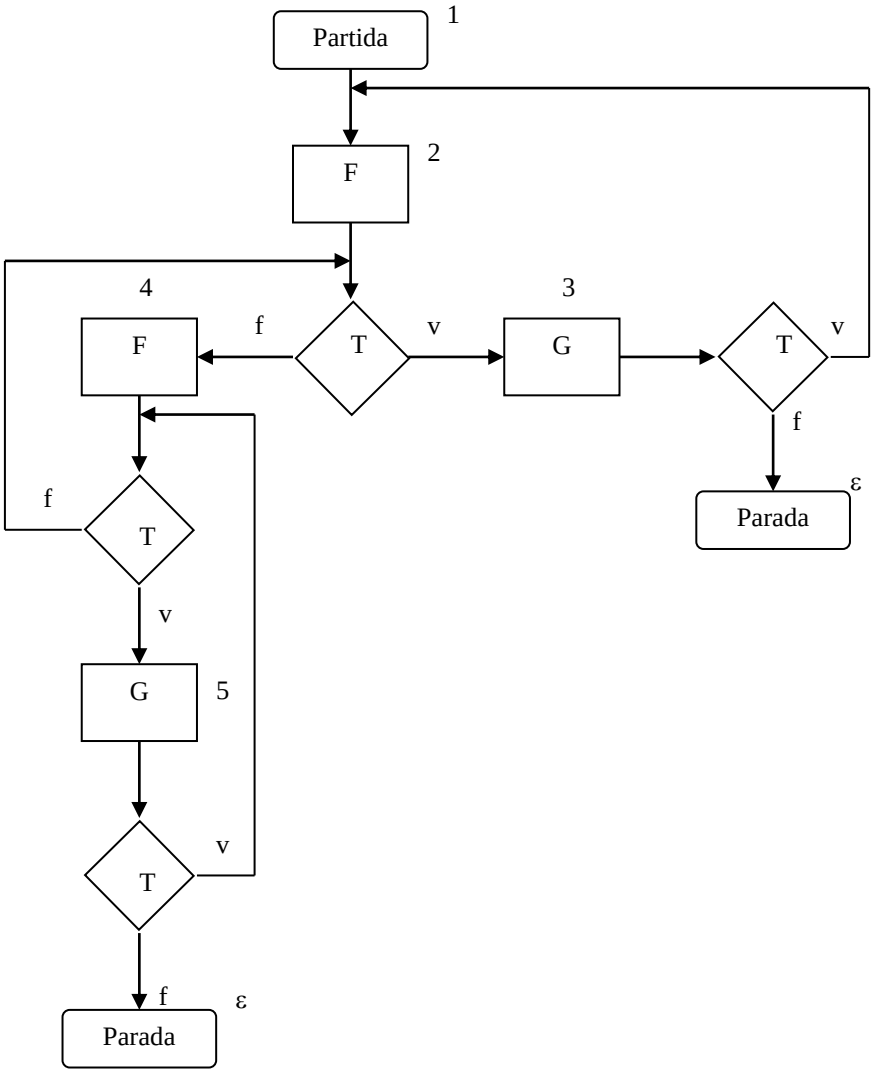


6: (F, 7) (Ciclo, ω) $A_0 = \{\varepsilon\}$
7: (G, 8) (G, 8) $A_1 = \{8, \varepsilon\}$
8: (F, 9) (Parada, ε) $A_2 = \{7, 8, \varepsilon\}$
9: (F, 7) (Ciclo, ω) $A_3 = \{6, 7, 8, 9, \varepsilon\}$
ω : (Ciclo, ω) (Ciclo, ω) $A_4 = A_3$

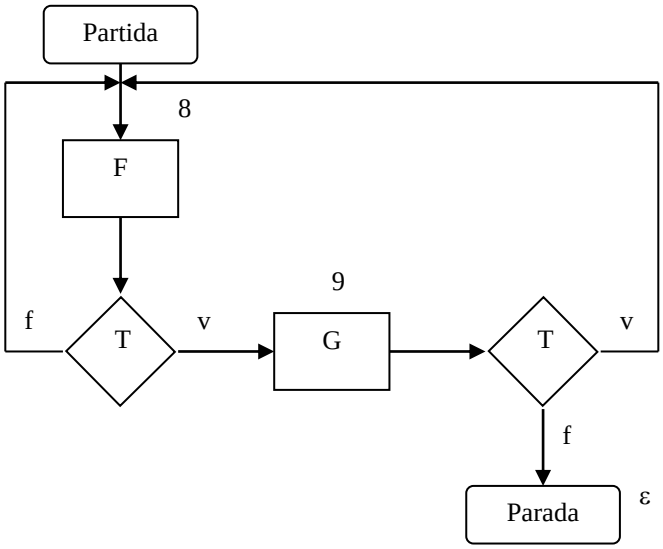
Verificação de equivalência:
União dos conjuntos de Instruções Rotuladas Compostas:
1: (F, 2) (Ciclo, ω)
2: (G, 3) (G, 3)
3: (F, 4) (Parada, ε)
4: (F, 2) (Ciclo, ω)
6: (F, 7) (Ciclo, ω)
7: (G, 8) (G, 8)
8: (F, 9) (Parada, ε)
9: (F, 7) (Ciclo, ω)
ω : (Ciclo, ω) (Ciclo, ω)

$B_0 = \{(1, 6)\}$
$B_1 = \{(2, 7) (\omega,\omega)\}$
$B_2 = \{(3, 8)\}$
$B_3 = (4, 9) (\varepsilon,\varepsilon)$
$B_3 = \phi$
Logo, os programas P e Q ***são*** *fortemente equivalentes.*

Exercício 2.18. Verifique se os programas monolíticos M_1 e M_2 representados na Figura 2.38 são fortemente equivalentes.
Programa M1:
1: (F, 2) (F, 2) $A_0 = \{\varepsilon\}$
2: (G, 3) (F, 4) $A_1 = \{3, 5, \varepsilon\}$
3: (F, 2) (Parada, ε) $A_2 = \{2, 3, 4, 5, \varepsilon\}$
4: (G, 5) (F, 4) $A_3 = \{1, 2, 3, 4, 5, \varepsilon\}$
5: (G, 5) (Parada, ε) $A_4 = A_3$
ε: *(Parada, ε),(Parada, ε)*



<u>Programa M2:</u>



7: (F, 8) (F, 8)
8: (G, 9) (F, 8)
9: (F, 8) (Parada, ε)

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

União dos conjuntos de Instruções Rotuladas Compostas:
1: (F, 2) (F, 2)
2: (G, 3) (F, 4)
3: (F, 2) (Parada, ε)
4: (G, 5) (F, 4)
5: (G, 5) (Parada, ε)
7: (F, 8) (F, 8)
8: (G, 9) (F, 8)
9: (F, 8) (Parada, ε)

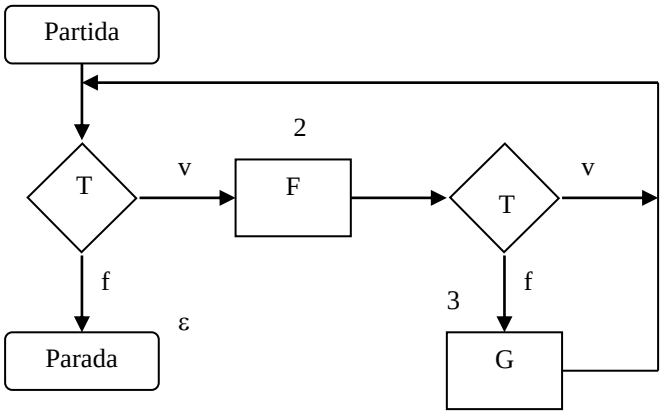
B_0 *= {(1, 7)}*
B_1 *= {(2, 8)}*
B_2 *= {(3, 9) (4, 8)}*
B_3 *= {(5,9), (ε, ε)}*
o par (5,9) não é consistente, pois G não é equivalente a F,
portanto M1 não é fortemente equivalente a M2

Exercício 2.19. Qual a importância da relação Equivalência Forte de Programas?
A importância da Relação Equivalência Forte de programas se deve a várias razões:

- *Permite identificar diferentes programas em uma mesma classe equivalência, ou seja, identificar diferentes programas cujas funções computadas coincidem, para qualquer máquina;*
- *As funções computadas por programas equivalentes fortemente têm a mesma propriedade de que os mesmos testes e as mesmas operações são efetuados na mesma ordem, independentemente do significado dos mesmos;*
- *Fornece subsídios para analisar a complexidade estrutural de programas.*

Exercício 2.20. Verifique se o programas iterativos W_1 e W_2 definidos na Figura 2.39 e na Figura 2.40 são fortemente equivalentes (mostre os passos intermediários – fluxogramas, instruções rotuladas, simplificação A_k, conjunto final de instruções rotuladas e verificação de equivalência).

Programa W_1:

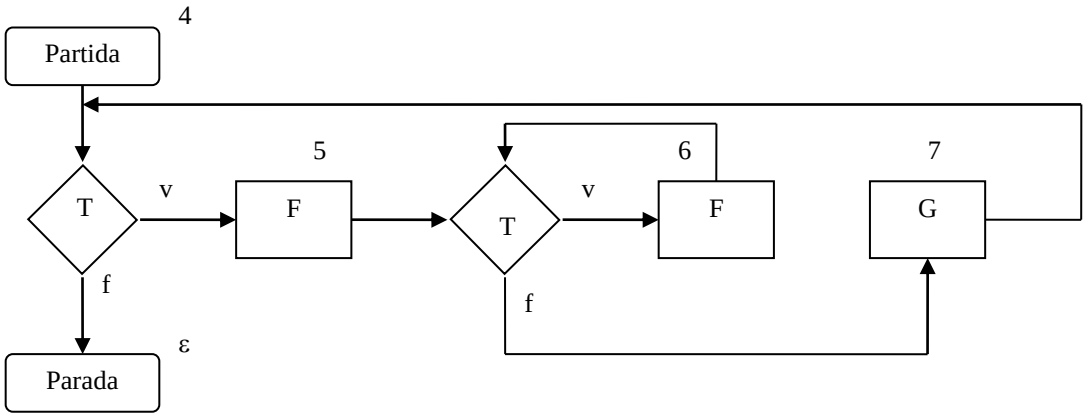


1: (F,2) (parada,ε)
2: (F,2) (G,3)
3: (F,2) (parada,ε)

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

Logo, já está simplificado.

Programa W_2:



4: (F,5) (parada,ε)
5: (F,6) (G,7)
6: (F,6) (G,7)
7: (F,5) (parada,ε)

$A_0 = \{\varepsilon\}$
$A_1 = \{\varepsilon,4,7\}$
$A_2 = \{\varepsilon,4,5,6,7\}$
$A_3 = \{\varepsilon,4,5,6,7\}$

Logo, já está simplificado.
a) Seja I a união disjunta dos conjuntos:

1: (F,2) (parada,ε)
2: (F,2) (G,3)
3: (F,2) (parada,ε)
4: (F,5) (parada,ε)
5: (F,6) (G,7)
6: (F,6) (G,7)
7: (F,5) (parada,ε)

b) Como 1 e 4 são rótulos equivalentes fortemente então: $B_0=\{(1,4)\}$
c) Para $K \geq 0$, construção de B_{k+1} é como segue:
$B_1=\{(2,5)\ (\varepsilon,\varepsilon)\}$
$B_2=\{(2,6)\ (3,7)\}$
$B_3=\varnothing$

Portanto, (I,1)≡(I,4) e portanto $W_1 \equiv W_2$.

Exercício 2.21 Traduza os programas iterativos W_1 e W_2 definidos na Figura 2.39 e na Figura 2.40 para programas recursivos.
Fluxograma já feito no exercício 2.20.

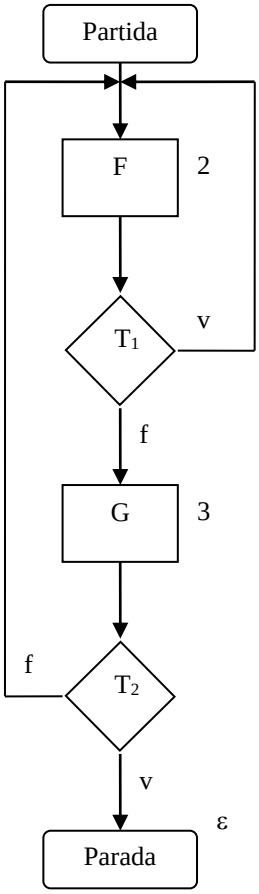
1: Se T então vá para 2 senão vá para 6
2: Faça F vá para 3
3: Se T então vá para 4 senão vá para 5
4: Faça F vá para 3
5: Faça G vá para 1

P_1 é R_1 onde
R_1 def E_1 = Se T então R_2 senão R_6
R_2 def E_2 = F;R_3
R_3 def E_3 = Se T então R_4 senão R_5
R_4 def E_4 = F;R_3
R_5 def E_5 = G;R_1
R_6 def E_6 = √

<u>Simplificando</u>:
P_1 é R_1 onde
R_1 def E_1 = Se T então F;R_2 senão √
R_2 def E_2 = Se T então F;R_2 senão G;R_1

Exercício 2.22 Traduza o seguinte programa monolítico da Figura 2.41 na forma de instruções rotuladas compostas. Como existem dois testes, cada instrução rotulada composta terá quatro possíveis sucessores, um para cada possível combinação de valores-verdade dos teste T_1 e T_2.

Vamos admitir esta ordem, conseqüência da combinação dos valores-verdade: (V, V) (V, F) (F, V) (F, F)
1: (F,2) (F,2) (F,2) (F,2)
2: (F,2) (F,2) (G,3) (G,3)
3: (parada,ε) (F,2) (parada,ε) (F,2)



Exercício 2.23 Adapte para o caso do programa monolítico da Figura 2.41, os seguintes itens:

a) Lema 2.34 - Identificação de Ciclos Infinitos em Programas Monolíticos;

b) Teorema 2.38 - Determinação de Rótulos Equivalentes Fortemente;

c) Definição 2.39 - Algoritmo de Equivalência Forte de Programas Monolíticos.

Exercício 2.24 Generalize a definição de instruções rotuladas compostas para o caso de três testes distintos.

Exercício 2.25 Traduza os fluxogramas da Figura 2.30 e da Figura 2.32 em instruções rotuladas compostas.

Proposta de Exercícios e Soluções Complementares do Capítulo 2

Exercício c2.1 Dê os possíveis prefixos e sufixos de cada uma das seguintes palavras:

a) Grêmio

prefixo = {ε, G, Gr, Grê, Grêm, Grêmi, Grêmio}

sufixo = {ε,o, io, mio, êmio, rêmio, Grêmio}

b) Campeão

prefixo = {ε, c, ca, cam, camp, campe, campeã, campeão}

sufixo = {ε, o, ão, eão, peão, mpeão, ampeão, campeão}

Exercício c2.2 Por que nem toda palavra é um prefixo ou um sufixo? Exemplifique.

Porque subpalavra é qualquer seqüência contígua de símbolos da palavra e prefixo (sufixo) é qualquer seqüência inicial (final) de símbolos da palavra.

Exemplo:

Palavra: matemática

Subpalavra: atemá (não é prefixo nem sufixo da palavra)

Exercício c2.3 Considere o alfabeto $\Sigma = \{1, 2\}$ e a linguagem $L = \{\omega \mid \omega \in \Sigma^* \text{ e } |\omega| = 3\}$.

a) Quantos elementos possui o conjunto Σ^*? E o conjunto L? (Justifique)

O conjunto Σ^* possui infinitos elementos, pois Σ^* é o conjunto de todas as palavras possíveis de ser formadas com o alfabeto Σ, incluindo ε.

O conjunto L possui 8 elementos, pois L é formado por todas as palavras de Σ^* que tem 3 elementos.

b) Quais os prefixos e sufixos da palavra 121?

Prefixos: ε, 1, 12, 121

Sufixos: ε, 1, 21, 121

Exercício c2.4 Responda as questões abaixo:

a) Qual a diferença entre programa e máquina? Por que os dois são necessários para definir computações?

Programa é um conjunto de operações e testes compostos de acordo com uma estrutura de controle enquanto a Máquina interpreta este programa desde que possua uma interpretação para cada operação ou teste que constitui o programa.

Porque a computação, uma vez que, é um histórico do funcionamento da máquina para o programa, considerando o valor inicial, utiliza-se da máquina e do programa.

b) Qual a diferença entre computação e função computada?

A computação é o histórico do funcionamento da máquina para o programa enquanto a função computada é o valor de saída, quando a computação é finita (a máquina pára)

Exercício c2.5 Traduza o fluxograma abaixo para um programa recursivo.

P_1 é R_1, onde

R_1 def Se T então R_2 senão R_3

R_2 def G; R_6

R_3 def F;R_4

R_4 def Se T então R_6 senão R_5

R_5 def F;R_1

R_6 def √

Pode-se transformar qualquer programa monolítico em um programa recursivo fortemente equivalente, mas o contrário não é verdadeiro. Isto significa que, dados uma, máquina M e um programa recursivo P_r que não pode ser transformado em um programa monolítico fortemente equivalente P_m, então não existe um programa monolítico para M que tem $\langle P_r, M\rangle$ como função computada? Explique.

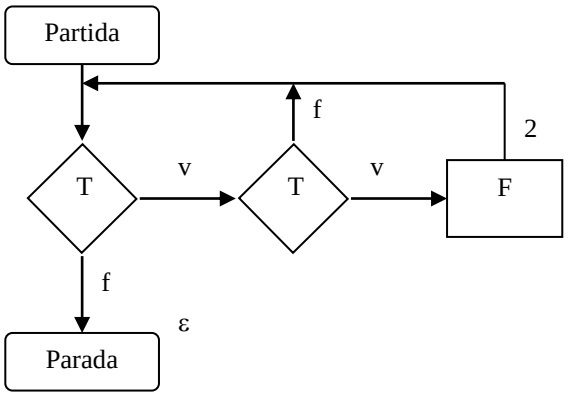
Em M não existe P_{M1}, então não pode existir $P_{M2} \equiv P_{M1}$, $P_{M3} \equiv P_{M1}$, etc.

Exercício c2.6 Verifique se os programas abaixo são fortemente equivalentes usando o algoritmo de equivalência forte de programas monolíticos.

P_1: enquanto T faça (se T então F senão √)

P_2: enquanto T faça (até T faça (G);F)

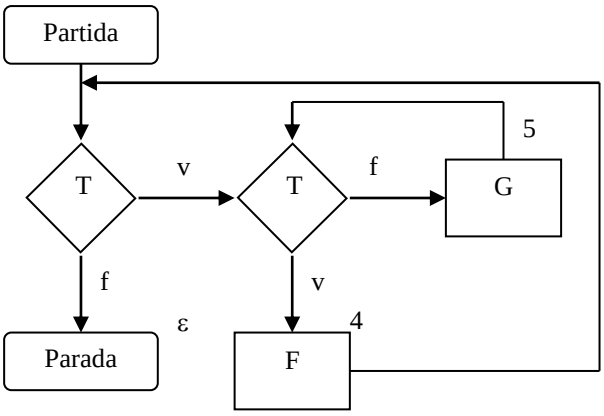
Programa P_1



1: (F, 2) (parada, ε)
2: (F, 2) (parada, ε)

$A_0 = \{\varepsilon\}$
$A_1 = \{1, 2, \varepsilon\}$
$A_2 = A_1$

Programa P_2



4: (F, 5) (parada, ε)
5: (F, 5) (parada, ε)
6: (F, 5) (G, 6)

$A_0 = \{\varepsilon\}$
$A_1 = \{4, 5, \varepsilon\}$
$A_2 = \{4, 5, 6, \varepsilon\}$
$A_3 = A_2$

Verificação de equivalência:

União dos conjuntos de Instruções Rotuladas Compostas:

1: (F, 2) (parada, ε)
2: (F, 2) (parada, ε)
4: (F, 5) (parada, ε)
5: (F, 5) (parada, ε)
6: (F, 5) (G, 6)

$B_0 = \{(1, 4)\}$
$B_1 = \{(2, 5)\ (\varepsilon, \varepsilon)\}$
$B_2 = \varnothing$

Logo, os programasP_1 e P_2 são fortemente equivalentes.