3 VERIFICAÇÃO DA EQUIVALÊNCIA FORTE DE PROGRAMAS

3.1 Máquina de Traços

3.2 Instruções Rotuladas Compostas

3.3 Equivalência Forte de Programas Monolíticos

3.4 Conclusões

3 VERIFICAÇÃO DA EQUIVALÊNCIA FORTE DE PROGRAMAS

Relação Equivalência Forte de Programas

- permite identificar diferentes programas em uma mesma classe de equivalência, ou seja, identificar diferentes programas cujas funções computadas coincidem, para qualquer máquina;
- as funções computadas por programas fortemente equivalentes têm a propriedade de que as mesmas operações são efetuadas na mesma ordem, independentemente do significado das mesmas;
- Mostra-se que existe um algoritmo para decidir a equivalência forte entre programas monolíticos.
- Como todo programa iterativo possui um programa monolítico fortemente equivalente, o mesmo pode ser afirmado para programas iterativos.

A verificação de que dois programas monolíticos são fortemente equivalentes usa os seguintes conceitos:

a) *Máquina de Traços*. Produz um rastro ou histórico (denominado *traço*) da ocorrência das operações do programa. Neste contexto, dois programas são fortemente equivalentes se são equivalentes em qualquer máquina de traços.

b) *Programa Monolítico com Instruções Rotuladas Compostas*. Instruções rotuladas compostas constituem uma forma alternativa de definir programas monolíticos. Basicamente, uma instrução rotulada composta é um teste da seguinte forma :

r_1: se T então faça F vá_para r_2 senão faça G vá_para r_3

- De fato, usando máquinas de traços, é fácil verificar que, para um dado fluxograma, é possível construir um programa fortemente equivalente, usando instruções rotuladas compostas.

- Instruções rotuladas compostas induzem a noção de rótulos fortemente equivalentes, a qual é usada para determinar se dois programas são ou não fortemente equivalentes.

3.1 Máquina de traços

- Uma máquina de traços não executa as operações propriamente ditas, mas apenas produz um histórico ou rastro da ocorrência destas, denominado de *traço*.
- Essas máquinas são de grande importância para o estudo da equivalência de programas pois, se dois programas são equivalentes em qualquer máquina de traços, então são fortemente equivalentes.

Definição 3.1
Máquina de Traços.

Uma *Máquina de traços* é uma máquina

$$M = (Op^*, Op^*, Op^*, id_{Op^*}, id_{Op^*}, \Pi_F, \Pi_T)$$

onde:

Op^* *conjunto de palavras de operações* onde $Op = \{F, G, ...\}$ o qual corresponde, simultaneamente, aos conjuntos de valores de memória, de entrada e de saída;

id_{Op^*} *função identidade* em Op^*, a qual corresponde, simultaneamente, às funções de entrada e saída;

Π_F *conjunto de interpretações de operações* onde, para cada identificador de operação F de Op, a interpretação $\pi_F: Op^* \to Op^*$ é tal que, para qualquer $w \in Op^*$, $\pi_F(w)$ resulta na concatenação do identificador F à direita de w, ou seja:

$$\pi_F(w) = wF$$

Π_T *conjunto de interpretações de testes,* tal que, para cada identificador de teste T:

$$\pi_T: Op^* \to \{\text{verdadeiro}, \text{falso}\} \text{ é função de } \Pi_T$$

- efeito de cada operação interpretada por uma máquina de traços é simplesmente o de acrescentar o identificador da operação à direita do valor atual da memória.
- a função computada consiste em um histórico das operações executadas durante a computação.

Definição 3.2
Função induzida por um traço em uma máquina

Sejam $M = (V, X, Y, \pi_X, \pi_Y, \Pi_F, \Pi_T)$ uma máquina,

$Op = \{F, G, H, ...\}$ o conjunto de operações interpretadas em Π_F e

$w = FG...H$ um traço possível de M, ou seja, $w \in Op^*$.

A *Função Induzida pelo Traço* w *na Máquina* M, denotada por:

$$[w, M]: X \to V$$

é a função (total):

$$[w, M] = \pi_H \circ ... \circ \pi_G \circ \pi_F \circ \pi_X$$

A função $[w, M]$ aplicada a uma entrada $x \in X$ é denotada por:

$$[w_X, M] = \pi_H \circ ... \circ \pi_G \circ \pi_F \circ \pi_X(x)$$

- Portanto, a função induzida por um traço nada mais é do que a função resultante da composição das interpretações das diversas operações que constituem o traço.

Teorema 3.3
Equivalência forte de programas ⇔ equivalência de programas em máquinas de traços

Sejam P e Q dois programas arbitrários, não necessariamente do mesmo tipo. Então: $P \equiv Q$ se, e somente se, para qualquer máquina de traços M, $P \equiv_M Q$.

Corolário 3.4
Equivalência Forte de Programas ⇔ Equivalência de Programas em Máquinas de Traços.

Sejam P e Q dois programas arbitrários, não necessariamente do mesmo tipo. Então: $P \equiv Q$ se, e somente se, para qualquer máquina de traços M, $\langle P, M\rangle(\varepsilon) = \langle Q, M\rangle(\varepsilon)$.

3.2 Instruções rotuladas compostas

- Instruções rotuladas compostas possuem somente um formato, ao contrário das instruções rotuladas, as quais podem ter dois formatos: de operação e de teste.

Definição 3.5
Instrução Rotulada Composta.

- Uma ***Instrução Rotulada Composta*** é uma seqüência de símbolos da seguinte forma (suponha que F e G são identificadores de operação e que T é um identificador de teste):
- r_1:se T então faça F vá_para r_2 senão faça G vá_para r_3
  - r_2 e r_3 são ditos *rótulos sucessores* de r_1;
  - r_1 é dito *rótulo antecessor* de r_2 e r_3.

Definição 3.6
Programa monolítico com instruções rotuladas compostas

- Um ***Programa Monolítico com Instruções Rotuladas Compostas*** **P** é um par ordenado P = (I, r) onde:
  I *Conjunto de Instruções Rotuladas Compostas,* o qual é finito;
  r *Rótulo Inicial* o qual distingue a instrução rotulada inicial em I.
- Adicionalmente, relativamente ao conjunto I, tem-se que:
  - não existem duas instruções diferentes com um mesmo rótulo;
  - um rótulo referenciado por alguma instrução o qual *não* é associado a qualquer instrução rotulada é dito um *Rótulo Final*.
- Para simplificar o entendimento da determinação da equivalência forte de programas monolíticos, é considerado somente o caso particular em que os programas possuem um *único* identificador de teste, denotado por T.
- Considerando-se um único identificador de teste, uma instrução rotulada composta da forma:

r_1:se T então faça F vá_para r_2 senão faça G vá_para r_3

pode ser abreviada simplesmente por:

$$r_1: \ (F, r_2), (G, r_3)$$

Definição 3.7
Algoritmo fluxograma → rotuladas compostas.

- ***Nós*** - componentes elementares de partida, parada e operação de um fluxograma
- algoritmo para traduzir um fluxograma P para um programa monolítico P' constituído por instruções rotuladas compostas:

a) ***Rotulação de Nós***. Rotula-se cada nó do fluxograma. Suponha que existe um único componente elementar de parada, ao qual é associado o identificador ε (palavra vazia). O rótulo correspondente ao nó partida é o Rótulo Inicial do programa P'.

b) ***Instruções Rotuladas Compostas***. A construção de uma instrução rotulada composta parte do nó partida e segue o caminho do fluxograma.

b.1) *Teste*. Para um teste, a correspondente instrução rotulada composta é: r_1: (F, r_2), (G, r_3)

b.2) *Operação*. Para uma operação, a correspondente instrução rotulada composta é:

$$r_1: (F, r_2), (F, r_2)$$

b.3) *Parada*. Para uma parada, a correspondente instrução rotulada composta é como segue:

$$r: (\text{parada}, \varepsilon), (\text{parada}, \varepsilon)$$

b.4) *Testes Encadeados*. No caso de testes encadeados, segue-se o fluxo até que seja encontrado um nó, resultando na seguinte instrução rotulada composta:

$$r_1: (F, r_2), (G, r_3)$$

b.5) *Testes Encadeados em Ciclo Infinito*. Para um ciclo infinito determinado por testes encadeados, a correspondente instrução rotulada composta é:

$$r_1: (F, r_2), (\text{ciclo}, \omega)$$

Neste caso, deve ser incluída, adicionalmente, uma instrução rotulada composta correspondente ao ciclo infinito:

$$\omega: (\text{ciclo}, \omega), (\text{ciclo}, \omega)$$

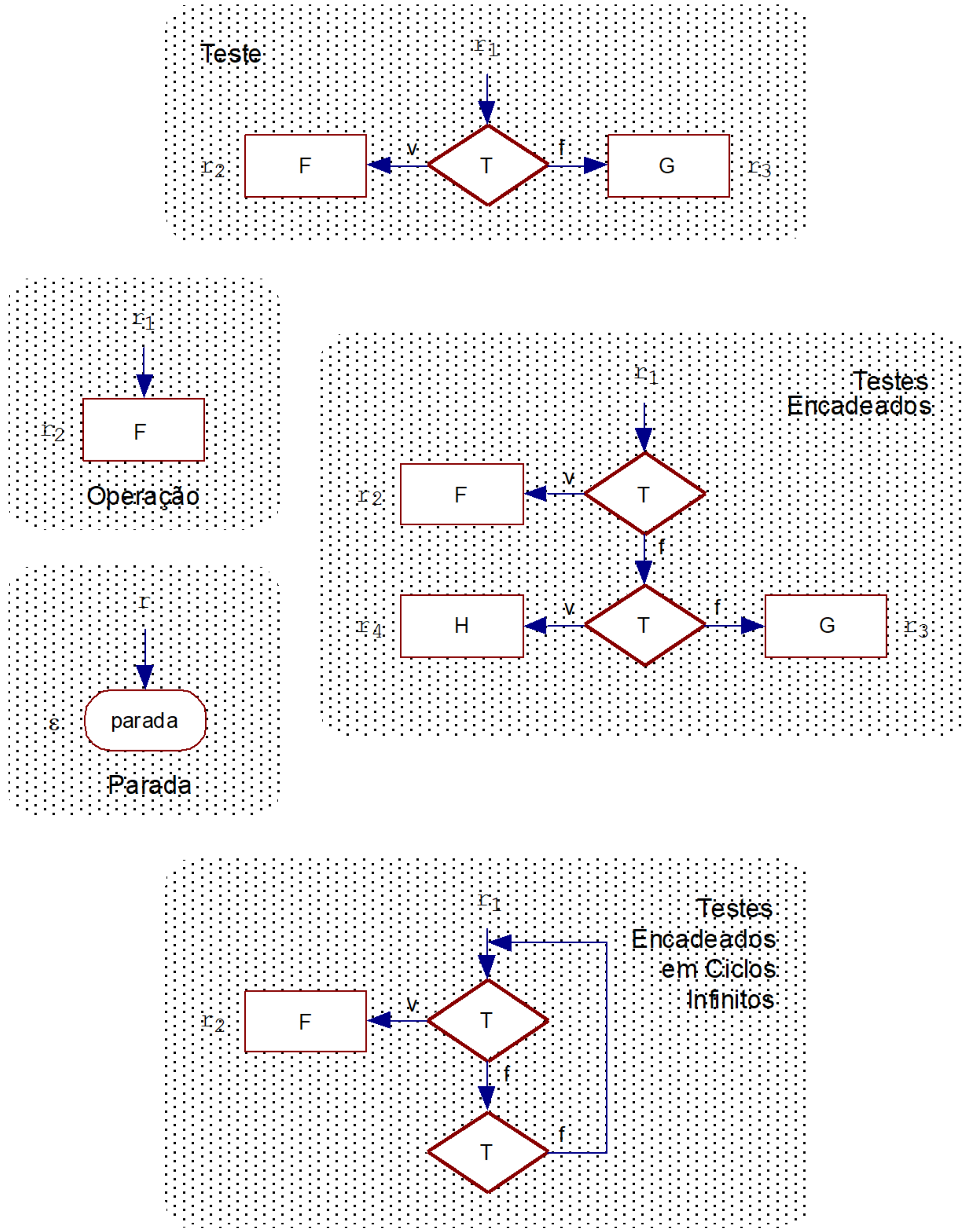

Figura 3.1 Instruções rotuladas compostas

Exemplo 3.1 Algoritmo fluxograma → Rotuladas compostas

- Considere o programa monolítico especificado na forma de fluxograma cujos **nós** já estão rotulados:

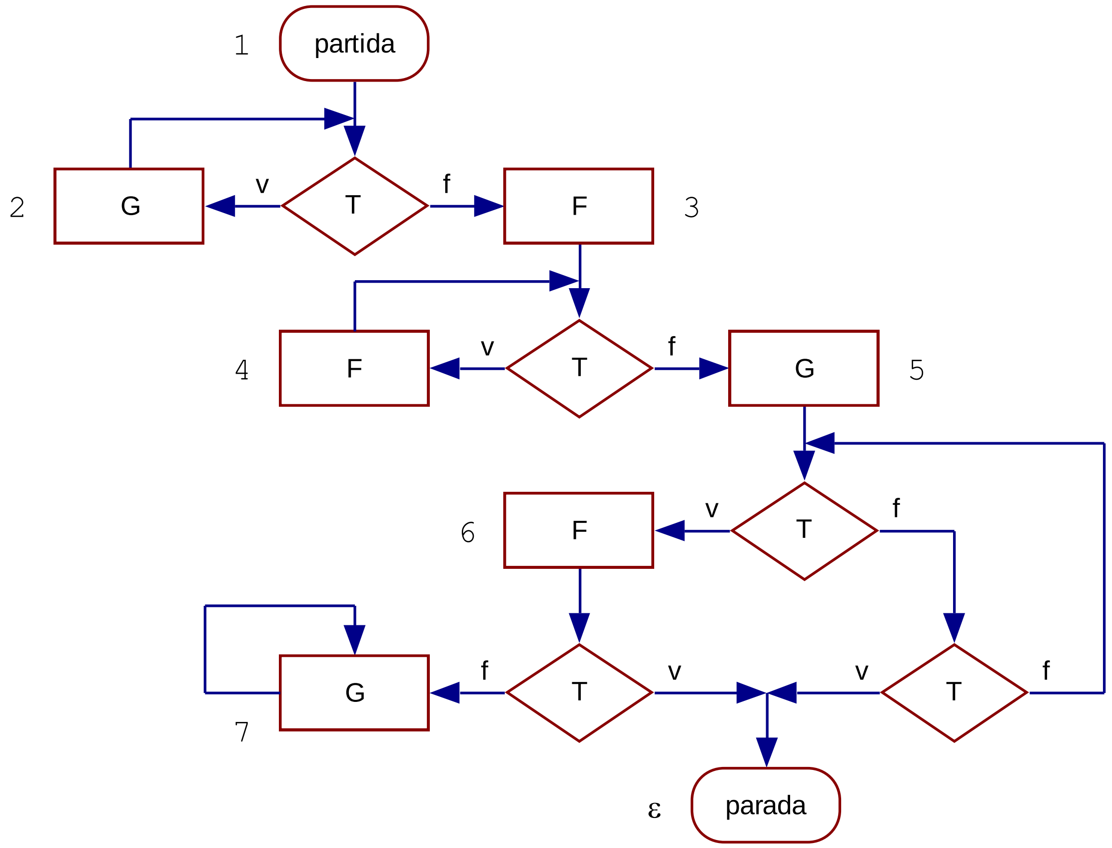

Figura 3.2 fluxograma Q com nós rotulados

- correspondente programa com instruções rotuladas compostas supondo que 1 é o rótulo inicial.

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

Note-se que:

- o rótulo 2 é sucessor dele mesmo. O mesmo ocorre com os rótulos 4, 7 e ω;
- existem dois caminhos no fluxograma que atingem o nó parada. Só um é representado no conjunto de instruções rotuladas compostas.
- na instrução rotulada por 7, ocorre um ciclo infinito.

3.3 Equivalência forte de programas monolíticos

- A ***união disjunta*** de conjuntos garante que todos os elementos dos conjuntos componentes constituem o conjunto resultante, mesmo que possuam a mesma identificação.
- considera-se que os elementos são distintos, mesmo que possuam a mesma identificação.
- Exemplo: para os conjuntos $A = \{a, x\}$ e $B = \{b, x\}$, o conjunto resultante da união disjunta é: $\{a_A, x_A, b_B, x_B\} = \{a, x_A, b, x_B\}$

Corolário 3.9
Equivalência Forte: União Disjunta.
Sejam:

- $Q = (I_Q, q)$ e $R = (I_R, r)$ dois programa monolíticos especificados usando instruções rotuladas compostas
- $P_q = (I, q)$ e $P_r = (I, r)$ programas monolíticos onde I é o conjunto resultante da união disjunta de I_Q e I_R.
- Então: $P_q \equiv P_r$ se, e somente se, $Q \equiv R$

O algoritmo para verificação da equivalência forte de Q e R se resume à verificação se P_q e P_r são fortemente equivalentes.

- ***cadeia de conjunto***: seqüência de conjuntos ordenada pela relação de inclusão;
- ***programa monolítico simplificado***: instruções rotuladas compostas que determinam ciclos infinitos são excluídas (excetuando-se a instrução rotulada por ω, se existir). A simplificação baseia-se em cadeia de conjuntos;
- ***rótulos fortemente equivalentes***: o algoritmo de verificação se P_q e P_r são fortemente equivalentes baseia-se em rótulos fortemente equivalentes de programas simplificados.

Definição 3.10
Cadeia de conjuntos, cadeia finita de conjuntos, limite de uma cadeia finita de conjuntos.

Uma seqüência de conjuntos A_0A_1... é dita:

a) uma ***Cadeia de Conjuntos*** se, para qualquer **k ≥ 0**: $A_k \subseteq A_{k+1}$

b) uma ***Cadeia Finita de Conjuntos*** é uma cadeia de conjuntos onde existe n, para todo **k ≥ 0**, tal que: $A_n = A_{n+k}$

c) o ***Limite da Cadeia Finita de Conjuntos*** é: $\mathit{lim}\ A_k = A_n$

Lema 3.11
Identificação de ciclos infinitos em programa monolítico.

- *lema fornece um algoritmo para determinar se existem ciclos* infinitos em um conjunto de instruções rotuladas compostas.
- *A idéia básica é partir da instrução* **parada**, *rotulada por ε, determinando os seus antecessores.*
- *Por exclusão, uma instrução que não é antecessor da* **parada** *determina um ciclo infinito.*

- Seja I um conjunto de n instruções rotuladas compostas.
- Seja A_0A_1... uma seqüência de conjuntos de rótulos indutivamente definida como segue:

  $A_0 = \{\varepsilon\}$

  $A_{k+1} = A_k \cup \{r \mid$ r é rótulo de instrução antecessora de alguma instrução rotulada por $A_k\}$

- Então A_0A_1... é uma cadeia finita de conjuntos, e, para qualquer rótulo r de instrução de I, tem-se que

  $(I, r) \equiv (I, \omega)$ se, e somente se, $r \notin \mathit{lim}\ A_k$

O Lema acima proporciona uma maneira fácil de determinar se algum rótulo caracteriza ciclos infinitos.

Exemplo 3.2
Identificação de ciclos infinitos em programa monolítico.

- Considere o conjunto I de instruções rotuladas compostas representado na ***Figura 3.3*** abaixo

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

Figura 3.3 Conjunto de Instruções rotuladas compostas

- A correspondente cadeia finita de conjuntos é:

$A_0 = \{\varepsilon\}$

$A_1 = \{6, \varepsilon\}$

$A_2 = \{5, 6, \varepsilon\}$

$A_3 = \{3, 4, 5, 6, \varepsilon\}$

$A_4 = \{1, 2, 3, 4, 5, 6, \varepsilon\}$

$A_5 = \{1, 2, 3, 4, 5, 6, \varepsilon\}$

- Logo: $lim\ A_k = \{1, 2, 3, 4, 5, 6, \varepsilon\}$

- Simplificação de ciclos infinitos: $(I, 7) \equiv (I, \omega)$, pois $7 \notin lim\ A_k$.

Portanto, pode-se simplificar um conjunto de instruções rotuladas compostas eliminando-se qualquer instrução de rótulo $r \neq \omega$ que determine um ciclo infinito.

Definição 3.12
Algoritmo de simplificação de ciclos infinitos.

Seja I um conjunto finito de instruções rotulas compostas.

O *Algoritmo de Simplificação de Ciclos Infinitos* é:

a) determina-se a correspondente cadeia finita de conjuntos A_0A_1... como no lema 2.34;

b) para qualquer rótulo r de instrução de I tal que r $\not\subseteq$ *lim* A_k, tem-se que:
   - a instrução rotulada por r é excluída;
   - toda referência a pares da forma (F, r) em I é substituída por (ciclo, ω);
   - I = I ∪ {ω: (ciclo, ω), (ciclo, ω)}.

Exemplo 3.3 Simplificação de ciclos infinitos

```
1:   (G, 2), (F, 3)
2:   (G, 2), (F, 3)
3:   (F, 4), (G, 5)
4:   (F, 4), (G, 5)
5:   (F, 6), (ciclo, ω)
6:   (parada, ε), (ciclo, ω)
ω:   (ciclo, ω), (ciclo, ω)
```

Figura 3.4 Conjunto de Instruções rotuladas compostas

Lema 3.13
Rótulos Consistentes.

- Seja I um conjunto finito de instruções rotuladas compostas e simplificadas.
- Sejam r e s dois rótulos de instruções de I, ambos diferentes de ε.
- Suponha que as instruções rotuladas por r e s são da seguinte forma, respectivamente:

  r: $(F_1, r_1), (F_2, r_2)$

  s: $(G_1, s_1), (G_2, s_2)$

Então r e s são **consistentes** se, e somente se,

($F_1 = G_1$ e $F_2 = G_2$) e ((r_1, s_1) e (r_2, s_2) forem consistentes).

Definição 3.14
Rótulos Fortemente Equivalentes.

- Seja I um conjunto finito de instruções rotuladas compostas e simplificadas.
- Sejam r e s dois rótulos de instruções de I.
- Suponha que as instruções rotuladas por r e s são da seguinte forma, respectivamente:

$$r: (F_1, r_1), (F_2, r_2)$$
$$s: (G_1, s_1), (G_2, s_2)$$

Então, r e s são ***Rótulos Fortemente Equivalentes*** se, e somente se:

- ou $r = s = \varepsilon$;
- ou r e s são ambos diferentes de ε e consistentes, ou seja

$$(F_1 = G_1 \text{ e } F_2 = G_2) \text{ e } ((r_1, s_1) \text{ e } (r_2, s_2)).$$

Teorema 3.15
Determinação de rótulos fortemente equivalentes

- Seja I um conjunto de n instruções compostas e simplificadas.
- Sejam r e s dois rótulos de instruções de I.
- Define-se, indutivamente, a seqüência de conjuntos $B_0 B_1 \ldots$ por:

$$B_0 = \{(r, s)\}$$
$$B_{k+1} = \{(r'', s'') \mid r'' \text{ e } s'' \text{são rótulos sucessores de } r' \text{ e } s', \text{respectivamente, } (r', s') \in B_k \text{ e } (r'', s'') \notin B_i, (0 \leq i \leq k)\}$$

- Então $B_0 B_1 \ldots$ é uma seqüência que converge para o conjunto vazio, e r, s são rótulos fortemente equivalentes se, e somente se, qualquer par de B_k é constituído por rótulos **consistentes**.

Definição 3.16
Algoritmo de equivalência forte de programas monolíticos

- Sejam $Q = (I_Q, q)$ e $R = (I_R, r)$ dois programas monolíticos especificados usando instruções rotuladas compostas e simplificado.
- ***Algoritmo de Equivalência Forte de Programas Monolíticos*** Q *e* R é determinado pelos passos:

***Passo 1*.** Sejam $P_q = (I, q)$ e $P_r = (I, r)$ programas monolíticos onde I é o conjunto resultante da união disjunta de I_Q e I_R, excetuando-se a instrução rotulada ω, se existir, a qual ocorre, no máximo, uma vez em I.

Passo 2*.** Se q e r são rótulos fortemente equivalentes, então $B_0=\{(q, r)\}$. Caso contrário, Q e R ***não são fortemente equivalentes, e o **algoritmo termina.**

***Passo 3*.** Para $k \geq 0$, define-se o conjunto B_{k+1}, contendo somente os pares (q'', r'') de rótulos sucessores de cada $(q', r') \in B_k$, tais que:

- $q' \neq r'$;
- q' e r' são ambos diferentes de ε;
- os pares sucessores (q'', r'') não são elementos de B_0, B_1, ... , B_k.

***Passo 4*.** Dependendo de B_{k+1}, tem-se que:

a) $B_{k+1} = \emptyset$: Q *e* R ***são fortemente equivalentes***, e o **algoritmo termina**;

b) $B_{k+1} \neq \emptyset$: se todos os pares de rótulos de B_{k+1} são fortemente equivalentes, então vá para o ***Passo 3***; caso contrário, Q e R ***não são fortemente equivalentes***, e o **algoritmo termina**.

Exemplo 3.4
Algoritmo de equivalência forte de programas monolíticos.

Considere os programas monolíticos especificados na forma de fluxograma Q (*figura 3.2*) e R (*figura 3.5*):

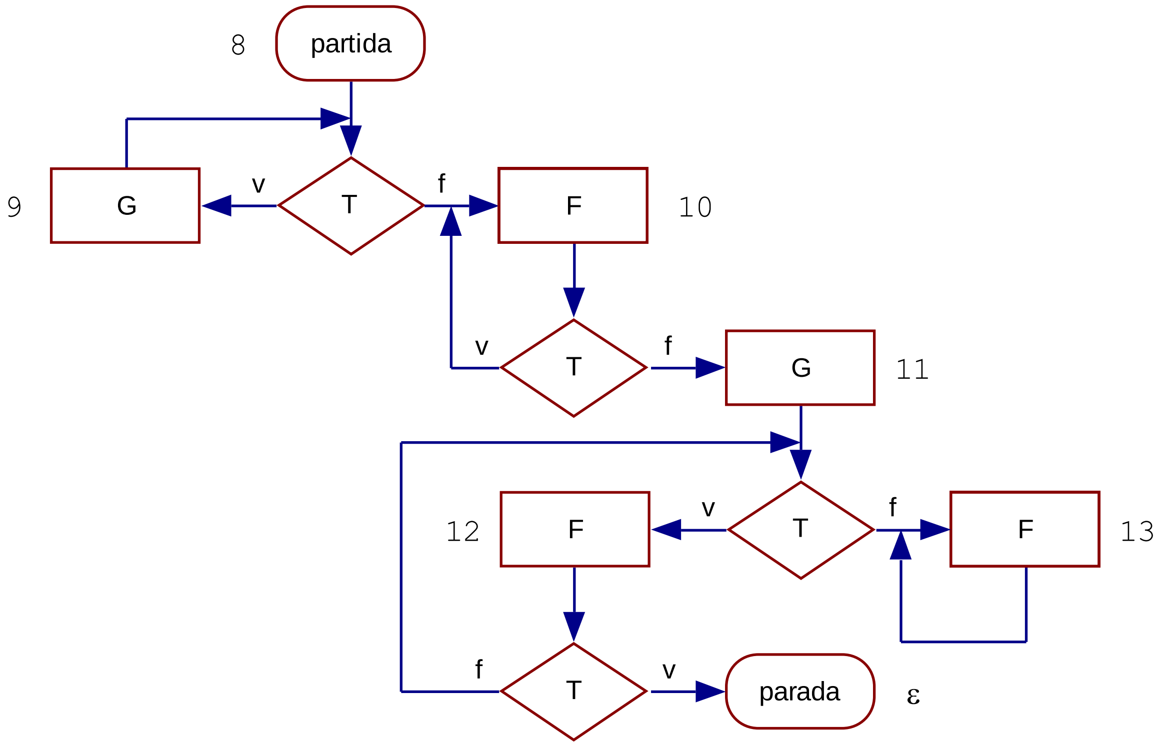

figura 3.5 Programa Monolítico R

a) ***A especificação do programa Q usando instruções rotuladas compostas, já simplificado, já foi feita na Figura 2.26.***

```
1: (G, 2), (F, 3)
2: (G, 2), (F, 3)
3: (F, 4), (G, 5)
4: (F, 4), (G, 5)
5: (F, 6), (ciclo, ω)
6: (parada, ε), (ciclo, ω)
ω: (ciclo, ω), (ciclo, ω)
```

b) ***Em relação ao programa R, tem-se que:***

b.1) ***Conjunto de instruções rotuladas compostas.***

```
8:      (G, 9), (F, 10)
9:      (G, 9), (F, 10)
10:     (F, 10), (G, 11)
11:     (F, 12), (F, 13)
12:     (parada, ε), (F, 13)
13:     (F, 13), (F, 13)
```

b.2) ***Identificação de ciclos infinitos.***

$A_0 = \{\varepsilon\}$
$A_1 = \{12, \varepsilon\}$
$A_2 = \{11, 12, \varepsilon\}$
$A_3 = \{10, 11, 12, \varepsilon\}$
$A_4 = \{8, 9, 10, 11, 12, \varepsilon\}$
$A_5 = \{8, 9, 10, 11, 12, \varepsilon\}$

Portanto:

$lim\ A_k = \{8, 9, 10, 11, 12, \varepsilon\}$
$(I_R, 13) \equiv (I, \omega)$, pois $13 \notin lim\ A_k$.

b.3) ***Simplificação de ciclos infinitos.***

```
8:      (G, 9), (F, 10)
9:      (G, 9), (F, 10)
10:     (F, 10), (G, 11)
11:     (F, 12), (ciclo, ω)
12:     (parada, ε), (ciclo, ω)
ω:      (ciclo, ω), (ciclo, ω)
```

c) Relativamente à aplicação do algoritmo, tem-se que:

Passo 1. Seja I a união disjunta dos conjuntos I_Q e I_R, excetuando-se a instrução rotulada ω, como segue:

```
1:  (G, 2), (F, 3)
2:  (G, 2), (F, 3)
3:  (F, 4), (G, 5)
4:  (F, 4), (G, 5)
5:  (F, 6), (ciclo, ω)
6:  (parada, ε), (ciclo, ω)
8:  (G, 9), (F, 10)
9:  (G, 9), (F, 10)
10: (F, 10), (G, 11)
11: (F, 12), (ciclo, ω)
12: (parada, ε), (ciclo, ω)
ω:  (ciclo, ω), (ciclo, ω)
```

Para verificar se **Q ≡ R**, é suficiente verificar se (I, 1) ≡ (I, 8).

Passo 2. Como 1 e 8 são rótulos fortemente equivalentes, então:

$$B_0 = \{(1, 8)\}$$

Passos 3 e 4. Para $k \geq 0$, construção de B_{k+1} é como segue:

$B_1 = \{(2, 9), (3, 10)\}$ pares de rótulos fortemente equivalentes

$B_2 = \{(4, 10), (5, 11)\}$ pares de rótulos fortemente equivalentes

$B_3 = \{(6, 12), (\omega, \omega)\}$ pares de rótulos fortemente equivalentes

$B_4 = \{(\varepsilon, \varepsilon)\}$ pares de rótulos fortemente equivalentes

$B_5 = \varnothing$

Logo (I, 1) ≡ (I, 8), e, portanto, **Q ≡ R**. □

Exemplo 3.5
Algoritmo de equivalência forte de programas monolíticos.

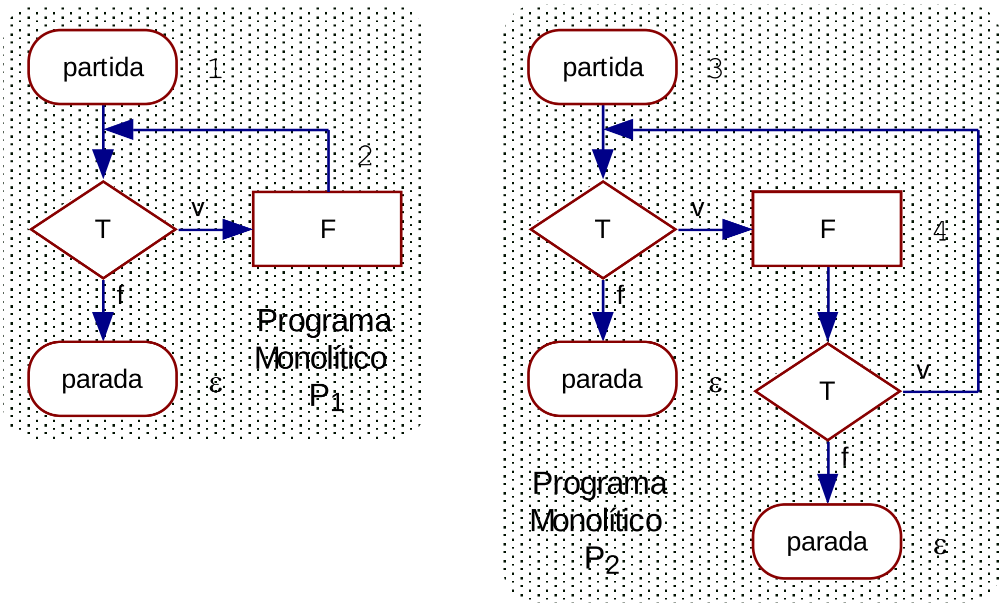

Figura 3.6 Fluxogramas P_1 e P_2 fortemente equivalentes

a) *A especificação do programa P_1 usando instruções rotuladas compostas (e já simplificadas) é:*

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

b) *A especificação de P_2 usando instruções rotuladas compostas (e já simplificadas) é:*

```
3: (F, 4), (parada, ε)
4: (F, 4), (parada, ε)
```

- os correspondentes conjuntos de instruções rotuladas compostas são iguais, a menos dos rótulos.
- aplicação do algoritmo: é fácil verificar que (I, 1) ≡ (I, 3).
- a relação equivalência forte fornece subsídios para analisar a complexidade estrutural de programas.
- P_1 é estruturalmente "mais otimizado" que P_2.

3.4 Conclusões

- Mostrou-se a existência de um algoritmo para verificar se programas monolíticos (ou iterativos) são fortemente equivalentes.

3.5 Exercícios

Exercício 3.1

Relativamente aos seguintes corolários e teoremas:

a) Corolário 3.4 Equivalência forte de Programas ⇔ Equivalência de programas em máquinas de Traços;
   a.1) Justifique a afirmação de que a prova (→) é imediata;
   a.2) Esboce a prova (←) para programas iterativo e recursivos;

b) Justifique a afirmação de que a prova do Lema 3.8 Equivaencia forte: fluxogramas → rotuladas compostas é imediata;

c) Por quê o Lema 3.8 Equivaencia forte: fluxogramas → rotuladas compostas garante que $P_q \equiv P_r$ se, e somente se, $Q \equiv R$?

d) Justifique Corolário 3.4 Equivalência forte de Programas ⇔ Equivalência de programas em máquinas de Traços;

Exercício 3.2

Mostre que os seguintes programas P e Q representados na ***Figura 3.7*** são fortemente equivalentes.

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 l
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
```

Figura 3.7 Programas iterativo P e monolítico Q

Exercício 3.3

Verifique se os programas monolíticos M_1 e M_2 representados na ***Figura 3.8*** são fortemente equivalentes.

Programa Monolítico M_1

```
1:   faça F vá_para 2
2:   se T então vá_para 3 senão vá_para 5
3:   faça G vá_para 4
4:   se T então vá_para 1 senão vá_para 0
5:   faça F vá_para 6
6:   se T então vá_para 7 senão vá_para 2
7:   faça G vá_para 8
8:   se T então vá_para 6 senão vá_para 0
```

Programa Monolítico M_2

```
1:   faça F vá_para 2
2:   se T então vá_para 3 senão vá_para 1
3:   faça G vá_para 4
4:   se T então vá_para 1 senão vá_para 0
```

Figura 3.8 Programas monolíticos M1 e M2

Exercício 3.4
Qual a importância da relação de Equivalência Forte de Programas?

Exercício 3.5
Verifique se os programas iterativos W_1 e W_2 definidos na ***Figura 3.9*** e ***Figura 3.10***, respectivamente, são fortemente equivalentes.

Programa Iterativo W_1
enquanto T
faça (F;(se T então faça √ senão faça G))

Figura 3.9 Programa iterativo W1

Programa Iterativo W_2
enquanto T
faça (F; enquanto T faça (F);G)

Figura 3.10 Programa iterativo W2

Exercício 3.6
Traduza o programa monolítico da ***Figura 3.11*** 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 testes T_1 e T_2.

Programa Monolítico dois_testes	
1:	faça F vá_para 2
2:	se T_1 então vá_para 1 senão vá_para 3
3:	faça G vá_para 4
4:	se T_2 então vá_para 0 senão vá_para 1

Figura 3.11 Programa Monolítico dois_testes

Exercício 3.7
Adapte para o caso do programa monolítico da ***Figura 3.11***, os seguintes itens:
a) **Lema 3.11** Identificação de ciclos infinitos em programas monolíticos;
b) **Teorema 3.15** Determinação de rótulos fortemente equivalentes;
c) **Definição 3.16** Algoritmo de equivaencia forte de programas monolíticos.

Exercício 3.8
Generalize a definição de instruções rotuladas compostas para o caso de três testes distintos.

Exercício 3.9

Traduza os fluxogramas da ***Figura 3.12*** e da ***Figura 3.13*** em instruções rotuladas compostas.

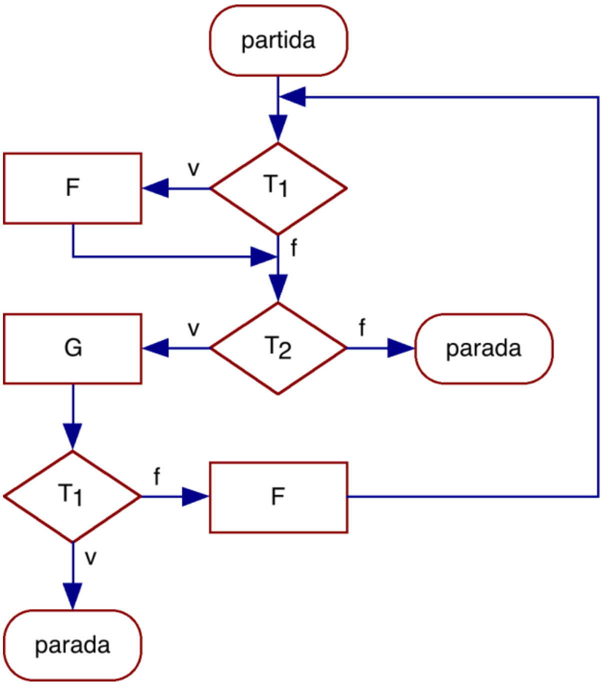

Figura 3.12 Fluxograma

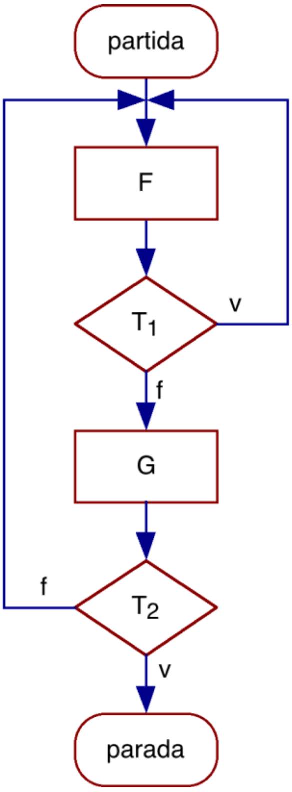

Figura 3.13 Fluxograma

Exercício 3.10

Reescreva o programa monolítico da ***Figura 3.14*** na forma de Instruções Rotuladas Compostas. Apresente o Fluxograma com os nós identificados.

Oservação: Atenção ao número de testes!

Programa Monolítico	
1:	se T então vá-para 2 senão vá-para 3
2:	faça F vá-para 6
3:	se U então vá-para 5 senão vá-para 4
4:	faça G vá-para 0
5:	faça F vá-para 0
6:	se U então vá-para 4 senão vá-para 1

Figura 3.14 Programa Monolítico

Exercício 3.11

Reescreva o programa monolítico da ***Figura 3.15*** na forma de Instruções Rotuladas Compostas. Apresente o Fluxograma com os nós identificados.

Oservação: Atenção ao número de testes!

Programa Monolítico	
1:	faça F vá-para 2
2:	se T_1 então vá-para 3 senão vá-para 1
3:	faça G vá-para 4
4:	se T_2 então vá-para 1 senão vá-para 5
5:	faça H vá-para 6
6:	se T_2 então vá-para 0 senão vá-para 5

Figura 3.15 Programa Monolítico

Exercício 3.12

Reescreva o programa iterativo abaixo na forma de Instruções Rotuladas Compostas. Apresente o Fluxograma com os nós identificados. Atenção ao número de testes!

Oservação: Atenção ao número de testes!

enquanto T_1 faça (F_1);
F_2;
(se T_2 então faça até T_1 faça (F_3; F_1);√ senão faça √)

Exercício 3.13

Reescreva o programa monolítico da ***Figura 3.16*** na forma de Instruções Rotuladas Compostas. Apresente o Fluxograma com os nós identificados.

Oservação: Atenção ao número de testes!

Programa Monolítico	
1:	se T_1 então vá-para 4 senão vá-para 3
2:	faça F_1 vá-para 1
3:	se T_1 então vá-para 2 senão vá-para 1

Figura 3.16 Programa Monolítico