Programas, Máquinas e Equivalências

Prof. Marcus Vinícius Midena Ramos

Universidade Federal do Vale do São Francisco

16 de maio de 2012

marcus.ramos@univasf.edu.br
www.univasf.edu.br/~marcus.ramos

Bibliografia

1. *Teoria da Computação (capítulo 2)*
   T. A. Diverio e P. B. Menezes
   Bookman, 2008, 2ª edição
2. *Programs & Machines - An Introduction to the Theory of Computation (capítulos 1 e 2)*
   R. Bird
   John Wiley & Sons, 1976

Roteiro

Conceitos básicos

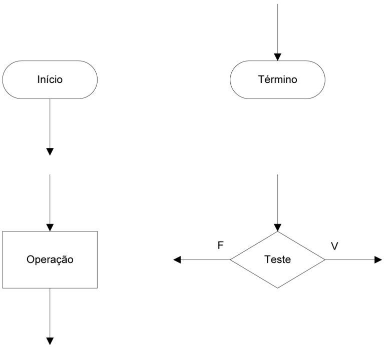
- Conjunto de instruções que estabelecem a seqüência em que certas "operações" e "testes" devem ser executados;
- Objetiva manipular dados de entrada, produzindo as saídas desejadas;
- A "estrutura de controle" do programa define a maneira como as operações e os testes são sequenciados no tempo;
- Tipos de estrutura de controle:
  - Monolítica ("flowchart programas");
  - Iterativa ("while programs");
  - Recursiva ("procedure programs").
- Composição "sequencial";
- Identificadores de operações: F, G, H, ...
- Identificadores de testes: T_1, T_2, T_3, ...
- Operação vazia: ✓

Programas monolíticos

Conceitos

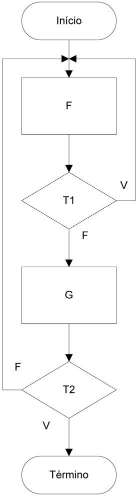
- "Programas com desvios arbitrários";
- Um único bloco;
- Elementos de estruturação:
  - Desvios condicionais;
  - Desvios incondicionais.
- Representações:
  - Gráfica (fluxograma);
  - Textual (conjunto de instruções rotuladas).

Programas monolíticos
Componentes



Programas monolíticos

Fluxograma



Programas monolíticos

Definição

Uma "instrução rotulada" é uma cadeia finita de caracteres que possui um dos seguintes formatos:

- r_1: faça F vá_para r_2 (operação simples)
- r_1: faça ✓ vá_para r_2 (operação vazia)
- r_1: se T então vá_para r_2 senão vá_para r_3 (teste)

onde r_1, r_2 e r_3 são rótulos numéricos, F é um identificador de operação e T é um identificador de teste.

Programas monolíticos

Definição

Um "programa monolítico" P é um par ordenado

$$P = (I, r)$$

onde:

- I é um conjunto (finito) de instruções rotuladas;
- r é o rótulo inicial.

Observações:

- Duas instruções não podem ter o mesmo rótulo;
- Rótulos finais são aqueles que são referenciados mas não estão associados a nenhuma instrução.

Programas monolíticos

Exemplos

Exemplos de programas monolíticos:

- $P_1 = (I_1, 1)$, onde $I_1 = \{$
  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 5 senão vá_para 1$\}$
- $P_2 = (\{1$: faça $\checkmark$ vá_para 2$\}$, 1)
- $P_3 = (\{1$: faça F vá_para 2, 2: se T vá_para 1 senão vá_para 3$\}$, 1)

Programas iterativos

Conceitos

- “Programas estruturados” sem subprogramas;
- Elementos de estruturação:
  - Execução sequencial;
  - Execução condicional (única entrada, única saída);
  - Execução iterativa (única entrada, única saída).
- Representações:
  - Gráfica (fluxograma estruturado);
  - Textual (conjunto de instruções que realizam os elementos de estruturação).

Programas iterativos
Definição

Definição de “programa iterativo”:

- A operação vazia ✓ e os identificadores de operação são programas iterativos;
- Se V e W são programas iterativos, então
$$V; W$$
é um programa iterativo (execução sequencial);
- Se V e W são programas iterativos, e T é um identificador de teste, então
$$\text{se } T \text{ então } V \text{ senão } W$$
é um programa iterativo (execução condicional);

Programas iterativos

Definição

- Se V é um programa iterativo, e T é um identificador de teste, então

$$\text{enquanto } T \text{ faça } V$$

  é um programa iterativo (execução iterativa do tipo "enquanto");
- Se V é um programa iterativo, e T é um identificador de teste, então

$$\text{até } T \text{ faça } V$$

  é um programa iterativo (execução iterativa do tipo "até", substitui o "enquanto" com a condição negada).
- Se V é um programa iterativo, então

$$(V)$$

  é um programa iterativo.

Programas iterativos

Exemplo

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

Programas recursivos

Conceitos

- Subprogramas recursivos, sem comando iterativo;
- Elementos de estruturação:
  - Execução sequencial;
  - Execução condicional (única entrada, única saída);
  - Definição de subprograma;
  - Chamada de subprograma.
- Representações:
  - Textual (conjunto de instruções que realizam os elementos de estruturação).

Considere que $R_1, R_2, \ldots$ são "identificadores de subprogramas".

Programas recursivos

Definição

Definição de "expressão":

- A operação vazia ✓ e os identificadores de operação são expressões;
- Os identificadores de subprograma são expressões;
- Se V e W são expressões, então

$$V;W$$

  é uma expressão (execução sequencial);
- Se V e W são expressões, e T é um identificador de teste, então

$$\text{se } T \text{ então } V \text{ senão } W$$

  é uma expressão (execução condicional);

Programas recursivos

Definição

Definição de "programa recursivo":

- P é E_0 onde
  R_1 def E_1, R_2 def $E_2, ..., R_n$ def E_n;
- $R_1, R_2, ..., R_n$ são identificadores de subprogramas;
- $E_1, E_2, ..., E_n$ são as expressões que definem, respectivamente, os subprogramas identificados por $R_1, R_2, ..., R_n$;
- E_0 é a "expressão inicial";
- Todos os identificadores de subprograma referenciados em P devem ser definidos em P.

Programas recursivos

Exemplo

P é $R; S$ onde

R def F; se T então R senão $G; S$

S def se T então $\checkmark$ senão $F; R$

Conceitos básicos

Uma máquina:

- Possui estrutura para armazenamento de dados (memória);
- Possui capacidade de ler e devolver dados para o meio externo;
- É responsável por atribuir significado para os identificadores de operação e de teste usados nos programas;
- Esses significados são representados na forma de funções que representam:
  - Alteração do conteúdo da memória (para os identificadores de operação);
  - A elaboração de um valor lógico a partir do conteúdo da memória, sem no entanto alterá-la (identificadores de teste).

Definição

Uma "máquina" é uma 7-upla:

$$M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$$

onde:

- V é o conjunto de valores que podem ser armazenados na memória;
- X é o conjunto de valores que podem ser lidos na entrada;
- Y é o conjunto de valores que podem ser escritos na saída;
- π_X é a "função de entrada", tal que $\pi_X : X \rightarrow V$;
- π_Y é a "função de saída", tal que $\pi_Y : V \rightarrow Y$

Definição

Sejam O_M e T_M, respectivamente, os conjuntos de identificadores de operações e testes definidos por M.

- Π_O é o conjunto de "interpretações de operações" tal que:

$$\forall o \in O_M, (\pi_o : V \to V) \in \Pi_O$$

- Π_T é o conjunto de "interpretações de testes" tal que:

$$\forall t \in T_M, (\pi_t : V \to \{\text{verdadeiro, falso}\}) \in \Pi_T$$

Exemplo

Máquina de Dois Registradores:

$$M_D = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$$

- $V = \mathbb{N}^2$
- $X = \mathbb{N}$
- $Y = \mathbb{N}$
- $\pi_X =$armazena_a
- $\pi_Y =$retorna_b
- $\Pi_O = \{$subtrai_a, adiciona_b$\}$
- $\Pi_T = \{$a_zero$\}$

Exemplo

- armazena_a: $\mathbb{N} \rightarrow \mathbb{N}^2$
  $\forall n \in \mathbb{N}$, armazena_a$(n) = (n, 0)$
- retorna_b: $\mathbb{N}^2 \rightarrow \mathbb{N}$
  $\forall (n, m) \in \mathbb{N}^2$, retorna_b$(n, m) = m$
- adiciona_b: $\mathbb{N}^2 \rightarrow \mathbb{N}^2$
  $\forall (n, m) \in \mathbb{N}^2$, adiciona_b$(n, m) = (n, m + 1)$
- subtrai_a: $\mathbb{N}^2 \rightarrow \mathbb{N}^2$
  $\forall (n, m) \in \mathbb{N}^2, n > 0$, subtrai_a$(n, m) = (n - 1, m)$
  se $n = 0$, subtrai_a$(n, m) = (0, m)$
- a_zero: $\mathbb{N}^2 \rightarrow$ {verdadeiro, falso}
  $\forall (n, m) \in \mathbb{N}^2$, se $n = 0$, a_zero$(n, m) =$ verdadeiro
  $\forall (n, m) \in \mathbb{N}^2$, se $n \neq 0$, a_zero$(n, m) =$ falso

Exemplo

Máquina de Um Registrador:

$$M_U = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$$

- $V = \mathbb{N}$
- $X = \mathbb{N}$
- $Y = \mathbb{N}$
- $\pi_X = id_{\mathbb{N}}$
- $\pi_Y = id_{\mathbb{N}}$
- $\Pi_O = \{\text{add, sub}\}$
- $\Pi_T = \{\text{zero}\}$

Exemplo

- $id_{\mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N}$
  $\forall n \in \mathbb{N}, id_{\mathbb{N}}(n) = n$
- add: $\mathbb{N} \rightarrow \mathbb{N}$
  $\forall n \in \mathbb{N}$, add$(n) = n + 1$
- sub: $\mathbb{N} \rightarrow \mathbb{N}$
  $\forall n \in \mathbb{N}, n > 0$, sub$(n) = n - 1$
  se $n = 0$, sub$(n) = 0$
- zero: $\mathbb{N} \rightarrow$ {verdadeiro, falso}
  se $n = 0$, zero$(n) =$ verdadeiro
  se $n \neq 0$, zero$(n) =$ falso

Programa para uma Máquina
Definição

Sejam $M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ e P um programa onde O_P e T_P são os respectivos conjuntos de identificadores de operações e testes de P. P é um programa para a máquina M se, e somente se:

- $\forall o \in O_P,$ existe uma única função $(\pi_o : V \to V) \in \Pi_O$
- $\forall t \in T_P,$ existe uma única função $(\pi_t : V \to \{\text{verdadeiro, falso}\}) \in \Pi_T$
- A operação vazia $\checkmark$ é sempre interpretada em qualquer máquina.

Portanto, P é um programa para uma máquina M se todos os identificadores de operações e testes utilizados em P estiverem definidos, em M, através de correspondentes funções de operações e testes.

Programa para uma Máquina

Exemplos

Programas para a Máquina de Dois Registradores:

- Programa monolítico:
  1: se a_zero vá_para 4 senão vá_para 2
  2: faça subtrai_a vá_para 3
  3: faça adiciona_b vá_para 1
- Programa iterativo:
  até a_zero
  faça (subtrai_a; adiciona_b)
- Programa recursivo:
  P é R onde
  R def se a_zero então ✓ senão $(S; R)$,
  S def subtrai_a; adiciona_b

Programa para uma Máquina

Exemplos

Programa para a Máquina de Um Registrador:

- Programa recursivo:
  P é R onde
  R def se zero então ✓ senão (sub;R;add;add)

Programa monolítico

Conceito

Sejam uma máquina $M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ e um programa monolítico $P = (I, r)$ para M, onde R é o conjunto de rótulos de P. Uma "computação do programa monolítico P na máquina M é uma cadeia de elementos de $R \times V$:

$$(r_0, v_0)(r_1, v_1)(r_2, v_2)...$$

onde $r_0 = r$ é o rótulo inicial de P e v_0 é o conteúdo inicial da memória de M.

- Essa cadeia indica a seqüência de estados que são assumidos pela máquina M durante a execução do programa P;
- Uma computação pode ser finita ou infinita.

Programa monolítico

Definição

Os pares $(r_{k+1}, v_{k+1}), k \geq 0$, são obtidos a partir dos pares (r_k, v_k), a partir da análise do tipo da instrução rotulada por r_k:

- r_k : faça F vá_para r'
  $(r_{k+1}, v_{k+1}) = (r', \pi_F(v_k))$
- r_k : faça ✓ vá_para r'
  $(r_{k+1}, v_{k+1}) = (r', v_k)$
- r_k : se T então vá_para r' senão vá_para r''
  se $\pi_T(v_k)$=verdadeiro, então $(r_{k+1}, v_{k+1}) = (r', v_k)$
  se $\pi_T(v_k)$=falso, então $(r_{k+1}, v_{k+1}) = (r'', v_k)$

Programa monolítico

Exemplo

- Programa monolítico P:
  1: se a_zero vá_para 4 senão vá_para 2
  2: faça subtrai_a vá_para 3
  3: faça adiciona_b vá_para 1
- Computação de P na Máquina de Dois Registradores M_D:
  (1,(3,0)) (2,(3,0))
  (3,(2,0)) (1,(2,1))
  (2,(2,1)) (3,(1,1))
  (1,(1,2)) (2,(1,2))
  (3,(0,2)) (1,(0,3))
  (4,(0,3))
- A computação é FINITA.

Programa monolítico
Exemplo

- Programa monolítico Q:
  1: faça adiciona_b vá_para 1
- Computação de Q na Máquina de Dois Registradores M_D:
  (1,(3,0)) (1,(3,1))
  (1,(3,2)) (1,(3,3))
  (1,(3,4)) (1,(3,5))
  (1,(3,6)) (1,(3,7))
  (1,(3,8)) (1,(3,9))
  ...
- A computação é INFINITA.

Programa iterativo

Conceito

Sejam uma máquina $M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ e um programa iterativo P para M. Uma "computação do programa iterativo P na máquina M é uma cadeia de elementos de $I \times V$:

$$(i_0, v_0)(i_1, v_1)(i_2, v_2)...$$

onde I é um conjunto de programas iterativos, $i_0 = P; \checkmark$ e v_0 é o conteúdo inicial da memória de M.

- Essa cadeia indica a seqüência de estados que são assumidos pela máquina M durante a execução do programa P;
- Uma computação pode ser finita ou infinita.

Programa iterativo
Definição

Os pares $(i_{k+1}, v_{k+1}), k \geq 0$, são obtidos a partir dos pares (i_k, v_k), a partir da análise do tipo da instrução inicial de i_k.
Considere que U, W e Z são programas iterativos, F é identificador de operação e T é identificador de teste.

- $i_k = \checkmark$
  A computação termina com o valor v_k na memória.
- $i_k = F; U$
  $(i_{k+1}, v_{k+1}) = (U, \pi_F(v_k))$
- $i_k =$se T então U senão $W; Z$
  se $\pi_T(v_k)$=verdadeiro, $(i_{k+1}, v_{k+1}) = (U; Z, v_k)$
  se $\pi_T(v_k)$=falso, $(i_{k+1}, v_{k+1}) = (W; Z, v_k)$

Programa iterativo

Definição

- i_k =enquanto T faça $U;W$
  se $\pi_T(v_k)$=verdadeiro, $(i_{k+1}, v_{k+1}) = (U;$ enquanto T faça $U;W, v_k)$
  se $\pi_T(v_k)$=falso, $(i_{k+1}, v_{k+1}) = (W, v_k)$
- i_k =até T faça $U;W$
  se $\pi_T(v_k)$=falso, $(i_{k+1}, v_{k+1}) = (U;$ até T faça $U;W, v_k)$
  se $\pi_T(v_k)$=verdadeiro, $(i_{k+1}, v_{k+1}) = (W, v_k)$

Programa iterativo
Exemplo

- Programa iterativo P:
  até a_zero
  faça (subtrai_a; adiciona_b)
- Computação de P na Máquina de Dois Registradores M_D:
  (até a_zero faça (subtrai_a; adiciona_b);✓,(2,0))
  (subtrai_a; adiciona_b; até a_zero faça (subtrai_a; adiciona_b);✓,(2,0))
  (adiciona_b; até a_zero faça (subtrai_a; adiciona_b);✓,(1,0))
  (até a_zero faça (subtrai_a; adiciona_b);✓,(1,1))
  (subtrai_a; adiciona_b; até a_zero faça (subtrai_a; adiciona_b);✓,(1,1))
  (adiciona_b; até a_zero faça (subtrai_a; adiciona_b);✓,(0,1))
  (até a_zero faça (subtrai_a; adiciona_b);✓,(0,2))
  (✓,(0,2))
- A computação é FINITA.

Programa recursivo

Conceito

Sejam uma máquina $M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ e P um programa recursivo para M, P é E_0 onde
R_1 def E_1, R_2 def $E_2, ..., R_n$ def E_n. Uma "computação do programa recursivo P na máquina M é uma cadeia de elementos de $J \times V$:

$$(j_0, v_0)(j_1, v_1)(j_2, v_2)...$$

onde J é um conjunto de expressões, $j_0 = E_0; \checkmark$ e v_0 é o conteúdo inicial da memória de M.

- Essa cadeia indica a seqüência de estados que são assumidos pela máquina M durante a execução do programa P;
- Uma computação pode ser finita ou infinita.

Programa recursivo
Definição

Os pares $(j_{k+1}, v_{k+1}), k \geq 0$, são obtidos a partir dos pares (j_k, v_k), a partir da análise do tipo da instrução inicial de j_k.
Considere que U, W e Z são expressões, F é identificador de operação, T é identificador de teste e R_i é identificador de subprograma.

- $j_k = \checkmark; U$
  $(j_{k+1}, v_{k+1}) = (U, v_k)$
- $j_k = F; U$
  $(j_{k+1}, v_{k+1}) = (U, \pi_F(v_k))$
- $j_k = R_i; U$
  $(j_{k+1}, v_{k+1}) = (E_i; U, v_k)$
- $j_k =$se T então U senão $W; Z$
  se $\pi_T(v_k)$=verdadeiro, $(j_{k+1}, v_{k+1}) = (U; Z, v_k)$
  se $\pi_T(v_k)$=falso, $(j_{k+1}, v_{k+1}) = (W; Z, v_k)$

Programa recursivo

Exemplo

- Programa recursivo P:
  P é R onde
  R def se a_zero então ✓ senão $(S; R)$,
  S def subtrai_a; adiciona_b
- Computação de P na Máquina de Dois Registradores M_D:
  (R;✓,(2,0))
  ((se a_zero então ✓ senão (S; R));✓,(2,0))
  (S; R;✓,(2,0))
  (subtrai_a; adiciona_b;R;✓,(2,0))
  (adiciona_b;R;✓,(1,0))
  (R;✓,(1,1))
  ((se a_zero então ✓ senão (S; R));✓,(1,1))
  ...

Programa recursivo

Exemplo

- Continuação da computação de P em M:

  ...
  (S; R;✓,(1,1))
  (subtrai_a; adiciona_b;R;✓,(1,1))
  (adiciona_b;R;✓,(1,2))
  (R; ✓,(0,2))
  ((se a_zero então ✓ senão (S; R));✓,(0,2))
  (✓;✓,(0,2))
  (✓,(0,2))
- A computação é FINITA.

Programa recursivo
Exemplo

- Programa recursivo Q:
  Q é R onde R def R
- Computação de Q na Máquina de Dois Registradores M_D:
  (R; ✓,(2,0))
  (R; ✓,(2,0))
  (R; ✓,(2,0))
  (R; ✓,(2,0))
  (R; ✓,(2,0))
  (R; ✓,(2,0))
  (R; ✓,(2,0))
  ...
- A computação é INFINITA.

Programa recursivo
Exemplo

- Programa recursivo P:
  P é R onde R def se zero então ✓ senão (sub;R;add;add)
- Computação de P na Máquina de Um Registrador M_U:
  (R;✓,2)
  (se zero então ✓ senão (sub;R;add;add);✓,2)
  (sub;R;add;add;✓,2)
  (R;add;add;✓,1)
  (se zero então ✓ senão (sub;R;add;add);add;add;✓,1)
  (sub;R;add;add;add;add;✓,1)
  (R;add;add;add;add;✓,0)
  (se zero então ✓ senão (sub;R;add;add);add;add;add;add;✓,0)
  (✓;add;add;add;add;✓,0)
  ...

Programa recursivo
Exemplo

- Continuação da computação de P:
  ...
  (add;add;add;add;✓,0)
  (add;add;add;✓,1)
  (add;add;✓,2)
  (add;✓,3)
  (✓,4)
- A computação é FINITA.

Conceitos básicos

- Obtenção de uma saída a partir de uma entrada, em tempo finito;
- Definições para:
  - Programas monolíticos;
  - Programas iterativos;
  - Programas recursivos.
- Aplica-se a função de entrada π_X ao dado de entrada;
- Executa-se a computação (finita);
- Aplica-se a função de saída π_Y ao valor final de memória.

Programas monolíticos

Definição

Sejam uma máquina $M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ e um programa monolítico P para M. A "função computada pelo programa monolítico P na máquina M", denotada:

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

é uma função parcial definida para $x \in X$, se a cadeia:

$$(r_0, v_0)(r_1, v_1)...(r_n, v_n)$$

é uma computação finita de P em M, onde:

- r_0 é o rótulo inicial de P;
- $v_0 = \pi_X(x)$

A imagem de x, denotada $\langle P, M \rangle(x)$, é dada por $\pi_Y(v_n)$.

Programas monolíticos

Exemplo

Considere a Máquina de Dois Registradores M_D e o programa monolítico P abaixo:

1: se a_zero vá_para 4 senão vá_para 2
2: faça subtrai_a vá_para 3
3: faça adiciona_b vá_para 1

- $\langle P, M_D \rangle : \mathbb{N} \to \mathbb{N}$
- $\forall n \in \mathbb{N}, \langle P, M_D \rangle(n) = n$
- P reproduz na saída o dado de entrada
- Exemplo:
  - $\pi_X(3) = (3, 0)$
  - A computação de P em M_D produz o valor final $(0, 3)$
  - $\pi_Y(0, 3) = 3$
  - Portanto, $\langle P, M_D \rangle(3) = 3$

Programas iterativos
Definição

Sejam uma máquina $M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ e um programa iterativo P para M. A "função computada pelo programa iterativo P na máquina M", denotada:

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

é uma função parcial definida para $x \in X$, se a cadeia:

$$(i_0, v_0)(i_1, v_1)(i_2, v_2)...(i_n, v_n)$$

é uma computação finita de P em M, onde:

- $i_0 = P; \checkmark$
- $v_0 = \pi_X(x)$

A imagem de x, denotada $\langle P, M \rangle(x)$, é dada por $\pi_Y(v_n)$.

Programas iterativos

Exemplo

Considere a Máquina de Dois Registradores M_D e o programa iterativo P abaixo:
até a_zero faça (subtrai_a; adiciona_b)

- $\langle P, M_D \rangle : \mathbb{N} \rightarrow \mathbb{N}$
- $\forall n \in \mathbb{N}, \langle P, M_D \rangle (n) = n$
- P reproduz na saída o dado de entrada
- Exemplo:
  - $\pi_X(2) = (2, 0)$
  - A computação de P em M_D produz o valor final $(0, 2)$
  - $\pi_Y(0, 2) = 2$
  - Portanto, $\langle P, M_D \rangle (2) = 2$

Programas recursivos

Definição

Sejam uma máquina $M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$ e um programa recursivo P para M. A "função computada pelo programa recursivo P na máquina M", denotada:

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

é uma função parcial definida para $x \in X$, se a cadeia:

$$(j_0, v_0)(j_1, v_1)(j_2, v_2)...(j_n, v_n)$$

é uma computação finita de P em M, onde:

- $j_0 = E_0$; ✓
- $v_0 = \pi_X(x)$

A imagem de x, denotada $\langle P, M \rangle(x)$, é dada por $\pi_Y(v_n)$.

Programas recursivos
Exemplo

Considere a Máquina de Um Registrador M_U e o programa recursivo P abaixo:

P é R onde R def se zero então ✓ senão (sub;R;add;add)

- $\langle P, M_U \rangle : \mathbb{N} \to \mathbb{N}$
- $\forall n \in \mathbb{N}, \langle P, M_U \rangle(n) = 2n$
- P duplica na saída o dado de entrada
- Exemplo:
  - $\pi_X(2) = 2$
  - A computação de P em M_U produz o valor final 4
  - $\pi_Y(4) = 4$
  - Portanto, $\langle P, M_U \rangle(2) = 4$

Definição

Dois programas P e Q, de quaisquer tipos, são ditos "fortemente equivalentes", denotado:

$$P \equiv Q$$

se, e somente se:

$$\forall M, \langle P, M \rangle = \langle Q, M \rangle$$

Ou seja, P e Q são fortemente equivalentes se, e somente se, as respectivas funções computadas coincidem para qualquer máquina M que se possa considerar.

- Essa relação induz a uma partição do conjunto de todos os programas em classes de equivalências;
- Ela permite analisar, de forma comparativa, as propriedades exibidas pelos programas, como é o caso da sua complexidade estrutural.

Propriedade

As computações de programas (de quaisquer tipos) fortemente equivalentes executam as MESMAS operações na MESMA ordem.

- A justificativa para essa afirmação será apresentada mais adiante, após a introdução do conceito de "Máquina de Traços".

Exemplos

Os programas monolíticos P_1 e P_2 abaixo são equivalentes fortemente:

- P_1:
  1: se T vá_para 2 senão vá_para 3
  2: faça F vá_para 1
- P_2:
  1: se T vá_para 2 senão vá_para 4
  2: faça F vá_para 3
  3: se T vá_para 1 senão vá_para 4

Exemplos

Computação de P_1 com a entrada x:

- $(1, \pi_X(x))$
  $(2, \pi_X(x))$
  $(1, \pi_F(\pi_X(x)))$
  $(2, \pi_F(\pi_X(x)))$
  $(1, \pi_F^2(\pi_X(x)))$
  $(2, \pi_F^2(\pi_X(x)))$
  ...
  $(1, \pi_F^n(\pi_X(x)))$
  $(3, \pi_F^n(\pi_X(x)))$
- Portanto, $\langle P_1, M\rangle(x) = \pi_Y(\pi_F^n(\pi_X(x)))$

Exemplos

Computação de P_2 com a entrada x:

- $(1, \pi_X(x))$
  $(2, \pi_X(x))$
  $(3, \pi_F(\pi_X(x)))$
  $(1, \pi_F(\pi_X(x)))$
  $(2, \pi_F(\pi_X(x)))$
  $(3, \pi_F^2(\pi_X(x)))$
  ...
  $(1, \pi_F^{n-1}(\pi_X(x)))$
  $(2, \pi_F^{n-1}(\pi_X(x)))$
  $(3, \pi_F^{n}(\pi_X(x)))$
  $(4, \pi_F^{n}(\pi_X(x)))$
- Portanto, $\langle P_2, M\rangle(x) = \pi_Y(\pi_F^n(\pi_X(x)))$
- $P_1 \equiv P_2$

Exemplos

Os programas P_3 (iterativo) e P_4 (recursivo) abaixo são equivalentes fortemente:

- P_3:
  enquanto T
  faça F
- P_4:
  P_4 é R onde
  R def se T então $(F; R)$ senão ✓

Exemplos

Computação de P_3 com a entrada x:

- (enquanto T faça F; $\checkmark, \pi_X(x)$)
  (F;enquanto T faça F; $\checkmark, \pi_X(x)$)
  (enquanto T faça F; $\checkmark, \pi_F(\pi_X(x))$)
  (F;enquanto T faça F; $\checkmark, \pi_F(\pi_X(x))$)
  (enquanto T faça F; $\checkmark, \pi_F^2(\pi_X(x))$)
  ...
  (enquanto T faça F; $\checkmark, \pi_F^n(\pi_X(x))$)
  ($\checkmark, \pi_F^n(\pi_X(x))$)
- Portanto, $\langle P_3, M\rangle(x) = \pi_Y(\pi_F^n(\pi_X(x)))$
- $P_1 \equiv P_2 \equiv P_3$

Exemplos

Computação de P_4 com a entrada x:

- $(R; \checkmark, \pi_X(x))$
  $((\text{se } T \text{ então } (F; R) \text{ senão } \checkmark); \checkmark, \pi_X(x))$
  $(F; R; \checkmark, \pi_X(x))$
  $(R; \checkmark, \pi_F(\pi_X(x)))$
  $((\text{se } T \text{ então } (F; R) \text{ senão } \checkmark); \checkmark, \pi_F(\pi_X(x)))$
  $(F; R; \checkmark, \pi_F(\pi_X(x)))$
  $(R; \checkmark, \pi_F^2(\pi_X(x)))$
  ...
  $(R; \checkmark, \pi_F^n(\pi_X(x)))$
  $((\text{se } T \text{ então } (F; R) \text{ senão } \checkmark); \checkmark, \pi_F^n(\pi_X(x)))$
  $(\checkmark; \checkmark, \pi_F^n(\pi_X(x)))$
  $(\checkmark, \pi_F^n(\pi_X(x)))$
- Portanto, $\langle P_4, M\rangle(x) = \pi_Y(\pi_F^n(\pi_X(x)))$
- $P_1 \equiv P_2 \equiv P_3 \equiv P_4$

Teorema 1

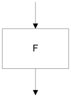
Iterativos $\subseteq$ Monolíticos

- Seja P_I um programa iterativo. Então, existe um programa monolítico P_M tal que $P_M \equiv P_I$.
- Justificativa: a obtenção de um programa P_M monolítico a partir de P_I é direta, a partir do mapeamento das construções elementares de um programa iterativo em seqüências de construções equivalentes em um programa monolítico. Como as mesmas operações são executadas na mesma ordem em ambos os programas, as funções computadas são idênticas e $P_M \equiv P_I$.

Teorema 1

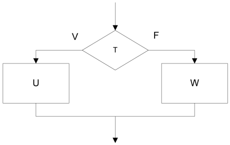
Iterativos $\subseteq$ Monolíticos

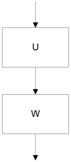
✓



F

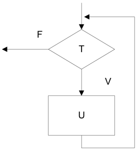
se T então faça U senão faça W

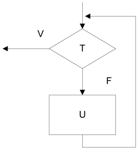




U;W

enquanto T faça U





até T faça U

Teorema 1

Exemplo

- Considere o seguinte programa iterativo P_I:
  até a_zero faça (subtrai_a; adiciona_b)
- O programa monolítico P_M abaixo, obtido por mapeamento direto, é fortemente equivalente à P_I:
  1: se a_zero vá_para 4 senão vá_para 2
  2: faça subtrai_a vá_para 3
  3: faça adiciona_b vá_para 1

Teorema 2

Monolíticos $\subseteq$ Recursivos

- Seja P_M um programa monolítico. Então, existe um programa recursivo P_R tal que $P_R \equiv P_M$.
- Justificativa: Suponha que $L = \{r_1, r_2, ..., r_n\}$ seja o conjunto de rótulos de P_M, r_1 seja o rótulo inicial e que r_n seja o (único) rótulo final. Então:

  $$P_R \text{ é } R_1 \text{ onde } R_1 \text{ def } E_1, R_2 \text{ def } E_2, ..., R_n \text{ def } \checkmark$$

  e, $\forall k, 1 \leq k < n$, E_k é definido da seguinte forma:
  - r_k : faça F vá_para r_i
    $E_k = F; R_i$
  - r_k : se T então vá_para r_i senão vá_para r_j
    E_k=(se T então R_i senão R_j)
- $P_R \equiv P_M$

Teorema 2

Exemplo

- Considere o programa monolítico Q:
  1: se a_zero vá_para 4 senão vá_para 2
  2: faça subtrai_a vá_para 3
  3: faça adiciona_b vá_para 1
- O programa recursivo R abaixo, obtido por mapeamento direto, é fortemente equivalente à Q:
  R é R_1 onde
  R_1 def (se a_zero R_4 senão R_2)
  R_2 def (subtrai_a;R_3)
  R_3 def (adiciona_b;R_1)
  R_4 def $\checkmark$

Corolário

Iterativos $\subseteq$ Recursivos

Para qualquer programa iterativo P_I existe um programa recursivo P_R tal que:

$$P_R \equiv P_I$$

Teorema 3

Monolíticos $\neq$ Recursivos

- Dado um programa recursivo P_R qualquer, não necessariamente existe um programa monolítico P_M tal que:

$$P_M \equiv P_R$$

- Justificativa: É suficiente mostrar que existe pelo menos um programa recursivo que, para uma determinada máquina, não apresente nenhum programa monolítico que seja fortemente equivalente.
- Considere o programa recursivo P_R abaixo e a máquina de um registrador M_U:
  P_R é R onde
  R def se $zero$ então ✓ senão $(sub; R; add; add)$
- $\langle P_R, M_U \rangle(n) = 2n$

Teorema 3

Monolíticos $\neq$ Recursivos

- A seqüência de operações contida na computação de P_R para um valor de entrada n é:

$$\underbrace{sub; sub; ...; sub}_{n}; \underbrace{add; add; ...; add}_{2n}$$

- Suponha que P_M contém k operações add (cada uma numa instrução diferente);
- Suponha um valor de entrada $n > k/2$;
- Como, por hipótese, $P_M \equiv P_R$, a mesma seqüência de operações é executada por P_M;

Teorema 3

Monolíticos $\neq$ Recursivos

- Como $2n > k$, pelo menos uma mesma instrução add é executada mais de uma vez na computação de P_M;
- Isso significa que há um desvio incondicional que permite a execução repetida dessa instrução, pois não seria possível, com um único registrador, controlar a execução do loop e ainda assim dobrar o valor da entrada;
- Há, portanto, um ciclo infinito em P_M envolvendo essa instrução add;
- Logo, a computação de P_M não pode ser finita e isso contradiz a hipótese da existência de P_M;
- Não existe P_M tal que $P_M \equiv P_R$.
- Não é possível construir um programa monolítico para M_U que dobre o valor da entrada!

Teorema 4

Iterativos $\neq$ Monolíticos

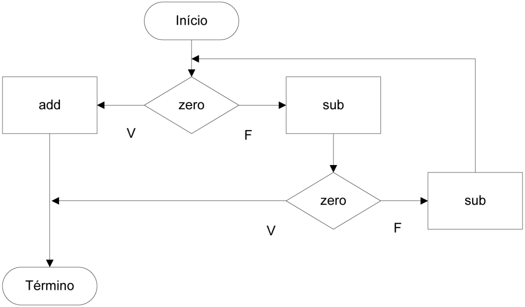
- Dado um programa monolítico P_M qualquer, não necessariamente existe um programa iterativo P_I tal que:

$$P_I \equiv P_M$$

- Justificativa: É suficiente mostrar que existe pelo menos um programa monolítico que, para uma determinada máquina, não apresente nenhum programa iterativo que seja fortemente equivalente.
- Considere o programa monolítico P_M da figura seguinte e a máquina de um registrador M_U
- $\langle P_M, M_U \rangle(n) = 1$ se n é par ou 0 se n é ímpar.

Teorema 4

Iterativos $\neq$ Monolíticos



Teorema 4

Iterativos $\neq$ Monolíticos

- A seqüência de operações contida na computação de P_M para um valor de entrada n é:

$$\underbrace{sub; sub; ...; sub}_{n}, \text{ se } n \text{ é ímpar}$$

$$\underbrace{sub; sub; ...; sub}_{n}; add, \text{ se } n \text{ é par}$$

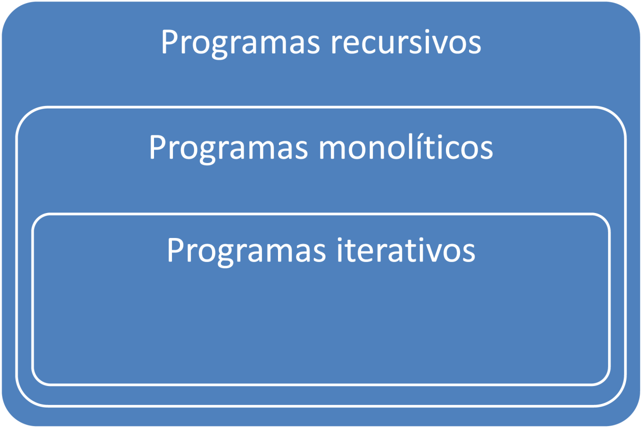
- Suponha que P_I contém k operações sub;
- Suponha um valor de entrada $n > k$;
- Como, por hipótese, $P_I \equiv P_M$, a mesma seqüência de operações é executada por P_I;

Teorema 4

Iterativos $\neq$ Monolíticos

- Como $n > k$, pelo menos uma instrução sub é executada mais de uma vez na computação de P_I;
- Logo, existe uma instrução iterativa do tipo "enquanto" ou "até" que controla a execução dessa operação sub;
- Ao término dessa instrução, no entanto, não é possível contabilizar a quantidade de execuções da operação sub que foram executadas no loop e, conseqüentemente, distinguir a condição "par" ou "ímpar" do valor n de entrada;
- Note que em P_M a avaliação do primeiro (segundo) teste implica a execução de um número par (ímpar) de subtrações;
- Logo, P_I não é capaz de produzir o resultado desejado;
- Portanto, não existe P_I que tal que $P_I \equiv P_M$.
- Não é possível construir um programa iterativo para M_U que determine se a entrada é par!

Hierarquia das classes de programas



Poder computacional

- Equivalência forte de programas $\neq$ poder computacional;
- Os três formalismos possuem o mesmo poder computacional:
  - Para qualquer programa recursivo e para qualquer máquina, existe um programa monolítico e uma máquina tal que as funções computadas coincidem;
  - Para qualquer programa monolítico e para qualquer máquina, existe um programa iterativo e uma máquina tal que as funções computadas coincidem.
- O Teorema de Böhm-Jacopini, por exemplo, mostra como gerar programas iterativos equivalentes a programas monolíticos dados como entrada. Não necessariamente as mesmas operações e a mesma ordem são usadas, tampouco o resultado vale para qualquer máquina.

Definição

Dois programas P e Q, de quaisquer tipos, são ditos "equivalentes na máquina M", denotado:

$$P \equiv_M Q$$

se, e somente se as correspondentes funções computadas na máquina M são iguais, ou seja:

$$\langle P, M \rangle = \langle Q, M \rangle$$

P e Q, nesse caso, são ditos "programas equivalentes na máquina M", ou "programas M-equivalentes".

Definição

Duas máquinas são ditas "equivalentes" se uma pode simular a outra e vice-versa. Sejam

$$M = (V_M, X, Y, \pi_{X_M}, \pi_{Y_M}, \Pi_{O_M}, \Pi_{T_M})$$

e

$$N = (V_N, X, Y, \pi_{X_N}, \pi_{Y_N}, \Pi_{O_N}, \Pi_{T_N})$$

- N "simula fortemente" M:
  $\forall P, \exists Q \mid \langle P, M \rangle = \langle Q, N \rangle$
- N "simula" M:
  $\forall P, \exists (Q, c : X_M \rightarrow X_N, d : Y_N \rightarrow Y_M) \mid \langle P, M \rangle = d \circ \langle Q, N \rangle \circ c$
- $(M \equiv N) \Leftrightarrow ((M \text{ simula } N) \wedge (N \text{ simula } M))$

Conceitos básicos

É possível determinar se dois programas, de quaisquer tipos, são fortemente equivalentes?

- Como programas iterativos e monolíticos podem ser transformados em programas recursivos, a resposta geral para essa questão envolveria a demonstração da equivalência forte de programas recursivos;
- Entretanto, esse problema é indecidível para esse tipo de programas (ele é, no entanto, decidível para uma classe especial de programas recursivos, aqueles que não contém a operação vazia);
- Por outro lado, ele é decidível para programas monolíticos (e, conseqüentemente, programas iterativos também);
- Pré-requisitos:
  - Máquina de traços;
  - Instruções rotuladas compostas.

Máquina de Traços

Conceitos

Máquina que gera, como saída, uma cadeia composta pelos identificadores das operações executadas durante a computação.

- Produz um histórico da ocorrência das operações no programa que está sendo executado;
- Esse histórico é representado na forma de uma cadeia de identificadores de operações;
- A memória armazena a cadeia que representa esse histórico;
- Para cada nova operação encontrada, a máquina de traços concatena o identificador da mesma no final da cadeia armazenada na memória;
- Ao término da execução a cadeia armazenada na memória é escrita na saída;
- Máquinas de Traços são importantes para demonstrar a equivalência forte de programas.

Máquina de Traços

Definição

Seja M uma Máquina de Traços:

$$M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$$

e, além disso, considere O o conjunto de identificadores das operações interpretadas por M e T o conjunto de identificadores de teste interpretados por M. Então:

- $V = O^*$
- $X = O^*$
- $Y = O^*$
- $\pi_X = id_{O^*}$
- $\pi_Y = id_{O^*}$

Máquina de Traços

Definição

$$M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$$

- Π_O é o conjunto de "interpretações de operações" tal que:

$$\forall o \in O, (\pi_o : O^* \to O^*) \in \Pi_O \text{ é tal que}, \forall \gamma \in O^*, \pi_o(\gamma) = \gamma o$$

- Π_T é o conjunto de "interpretações de testes" tal que:

$$\forall t \in T, (\pi_t : O^* \to \{\text{verdadeiro, falso}\}) \in \Pi_T$$

Para definir uma Máquina de Traços é necessário especificar apenas as interpretações dos testes, uma vez que as interpretações das operações são especificadas à priori.

Máquina de Traços

Função induzida por um traço em uma máquina

Seja:

$$N = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$$

Considere-se $O = \{o_1, o_2, ..., o_n\}$ o conjunto de operações interpretadas em Π_O e $\gamma = o_1 o_2 ... o_n$ um traço de N. A "função induzida pelo traço γ na máquina N", denotado:

$$[\gamma, N] : X \rightarrow V$$

é a função total:

$$[\gamma, N] = \pi_{o_n} \circ ... \pi_{o_2} \circ \pi_{o_1} \circ \pi_X$$

Máquina de Traços

Função induzida por um traço em uma máquina

A função $[\gamma, N]$ aplicada a uma entrada $x \in X$ é denotada:

$$[\gamma_x, N] = \pi_{o_n} \circ ...\pi_{o_2} \circ \pi_{o_1} \circ \pi_X(x)$$

Se $\gamma = \epsilon$, então:

$$[\epsilon_x, N] = \pi_X(x)$$

Observar que $[\gamma_x, N] \in V$.

Máquina de Traços

Definição de Π_T

- Em princípio, a definição de Π_T é livre;
- Há interesse, no entanto, em fazer com que uma Máquina de Traços M produza, como resultado da computação de um programa P, a mesma seqüência de operações que seria realizada por uma outra máquina N, durante a execução de P com uma entrada x;
- Nesse caso, é importante que as funções de teste em M produzam os mesmos resultados que produziriam em N;
- Para isso, é necessário considerar que:

$$\forall t \in T, \forall \gamma \in O^*, \pi_{t_M}(\gamma) = \pi_{t_N}([\gamma_x, N])$$

Máquina de Traços

Exemplo

$$M = (V, X, Y, \pi_X, \pi_Y, \Pi_O, \Pi_T)$$

$O = \{F, G, H\}$ é o conjunto de identificadores das operações interpretadas por M;
$T = \{T\}$ é o conjunto de identificadores de teste interpretados por M.
Então:

- $V = \{F, G, H\}^*$
- $X = \{F, G, H\}^*$
- $Y = \{F, G, H\}^*$
- $\pi_X = id_{\{F,G,H\}^*}$
- $\pi_Y = id_{\{F,G,H\}^*}$

Máquina de Traços
Exemplo

- $\Pi_O = \{\pi_F, \pi_G, \pi_H\}$

$$\forall \gamma \in \{F, G, H\}^*, \pi_F(\gamma) = \gamma F$$

$$\forall \gamma \in \{F, G, H\}^*, \pi_G(\gamma) = \gamma G$$

$$\forall \gamma \in \{F, G, H\}^*, \pi_H(\gamma) = \gamma H$$

- $\Pi_T = \{\pi_{t_M}\}$

$$\pi_{t_M}(\gamma) = \pi_{t_N}([\gamma_x, N])$$

Máquina de Traços

Exemplo

- Considere o programa monolítico P abaixo:
  1: faça F vá_para 2
  2: faça G vá_para 3
  3: faça G vá_para 4
  4: se T vá_para 5 senão vá_para 1
- Considere $x = \epsilon$
- Portanto, $\pi_X(\epsilon) = \epsilon$
- A computação de P na Máquina de Traços M é (supondo duas iterações):
  $(1, \epsilon)(2, F)(3, FG)(4, FGG)(1, FGG)$
  $(2, FGGF)(3, FGGFG)(4, FGGFGG)(5, FGGFGG)$
- Assim, $\pi_Y(FGGFGG) = FGGFGG$ e $\langle P, M\rangle(\epsilon) = FGGFGG$

Teorema 5

Equivalência de programas em Máquinas de Traços

Sejam P e Q dois programas de tipos quaisquer. Então:

$$(P \equiv Q) \Leftrightarrow (P \equiv_M Q) \text{ para toda Máquina de Traços } M$$

- ($\longrightarrow$) Se dois programas são fortemente equivalentes, então eles são equivalentes em qualquer Máquina de Traços (trivial, pois decorre diretamente da definição);
- ($\longleftarrow$) Se dois programas são equivalentes em qualquer Máquina de Traços, então eles são fortemente equivalentes (necessita de demonstração).

Teorema 5

Equivalência de programas em Máquinas de Traços

Suponha que R é um programa monolítico qualquer, N é uma máquina e M é uma Máquina de Traços. Deseja-se, inicialmente, demonstrar que:

$$\langle R, N\rangle(x) = \pi_{Y_N} \circ [\langle R, M\rangle(\epsilon), N]$$

- Ou seja, que a função computada pelo programa R na máquina N é igual ao resultado obtido pela composição da função de saída da máquina N com a função induzida pelo traço de R na máquina N.

Teorema 5

Equivalência de programas em Máquinas de Traços

- Computação de R em N com entrada x:
  $(r_0, v_0)(r_1, v_1)$..., com $v_0 = \pi_{X_N}(x)$
- Computação de R em M com entrada ϵ:
  $(m_0, \gamma_0)(m_1, \gamma_1)$..., com $\gamma_0 = \pi_{X_M}(\epsilon) = \epsilon$
- Prova-se, por indução, que $\forall k \geq 0$, $r_k = m_k$ e $v_k = [\gamma_k, N]$

Teorema 5

Equivalência de programas em Máquinas de Traços

- Base $k = 0$:
  - $r_0 = m_0$, o rótulo inicial de R
  - $v_0 = \pi_{X_N}(x)$
  - $[\gamma_0, N] = [\epsilon, N] = \pi_{X_N}(x)$
  - Portanto, $v_0 = [\gamma_0, N]$
- Hipótese $\forall k \geq 0$:
  - $r_k = m_k$
  - $v_k = [\gamma_k, N]$

Teorema 5

Equivalência de programas em Máquinas de Traços

- Passo de indução (conforme o tipo da instrução referenciada por $r_k = m_k$):
  - faça F vá_para r'
    $r_{k+1} = r'$
    $m_{k+1} = r'$
    $v_{k+1} = \pi_F(v_k) = \pi_F([\gamma_k, N]) = [\gamma_k F, N]$
    $\gamma_{k+1} = \gamma_k F$
    Portanto, $r_{k+1} = m_{k+1}$ e $v_{k+1} = [\gamma_{k+1}, N]$
  - se T então vá_para r' senão vá_para r''
    $r_{k+1} = m_{k+1}$ pois $T_M(\gamma_k) = T_N([\gamma_k, N]) = T_N(v_k)$
    $v_{k+1} = v_k = [\gamma_k, N] = [\gamma_{k+1}, N]$
    Portanto, $r_{k+1} = m_{k+1}$ e $v_{k+1} = [\gamma_{k+1}, N]$
- Demonstrações similares podem ser feitas para outros tipos de programas.

Teorema 5

Equivalência de programas em Máquinas de Traços

- $(P \equiv_M Q)$ para toda Máquina de Traços $M \Rightarrow (P \equiv Q)$;
- Prova pela contrapositiva;
- $\neg(P \equiv Q) \Rightarrow \neg((P \equiv_M Q)$ para toda Máquina de Traços $M)$;
- Suponha que P e Q não são fortemente equivalentes;
- Logo, existe uma máquina:

$$N = (V, X, Y, \pi_{X_N}, \pi_{Y_N}, \Pi_{O_N}, \Pi_{T_N})$$

e uma entrada $x \in X$ tal que:

$$\langle P, N\rangle(x) \neq \langle Q, N\rangle(x)$$

Teorema 5

Equivalência de programas em Máquinas de Traços

- Considere a Máquina de Traços:

$$M = (O^*, O^*, O^*, id_{O^*}, id_{O^*}, \Pi_{O_M}, \Pi_{T_M})$$

- Considere que:
  $\forall \gamma \in O^*, \pi_{t_M} \in \Pi_{T_M}, \pi_{t_M}(\gamma) = \pi_{t_N}([\gamma_x, N]) = \pi_{t_N}([\gamma, N])$
- Como $\langle R, N\rangle(x) = \pi_{Y_N} \circ [\langle R, M\rangle(\epsilon), N]$, então:
  $\langle P, N\rangle(x) \neq \langle Q, N\rangle(x) \Leftrightarrow$
  $\pi_{Y_N} \circ [\langle P, M\rangle(\epsilon), N] \neq \pi_{Y_N} \circ [\langle Q, M\rangle(\epsilon), N] \Leftrightarrow$
  $[\langle P, M\rangle(\epsilon), N] \neq [\langle Q, M\rangle(\epsilon), N] \Leftrightarrow$
  $\langle P, M\rangle(\epsilon) \neq \langle Q, M\rangle(\epsilon)$
- Portanto, existe uma Máquina de Traços M tal que P e Q não tem a mesma função computada.

Teorema 5

Equivalência de programas em Máquinas de Traços

Em outras palavras:

- Suponha que P e Q sejam equivalentes em qualquer Máquina de Traços;
- Considere como hipótese que P e Q não sejam fortemente equivalentes;
- A demonstração anterior permite concluir, por construção direta, que P e Q não são equivalentes em qualquer Máquina de Traços;
- Isso contradiz a suposição original;
- Logo, a hipótese é falsa e P e Q são fortemente equivalentes.

Corolário

$$P \equiv Q$$

se e somente se, para toda Máquina de Traços M,

$$\langle P, M \rangle(\epsilon) = \langle Q, M \rangle(\epsilon)$$

Portanto, como Máquinas de Traços produzem histórico das operações executadas pelos programas, segue que a propriedade apresentada anteriormente é válida, ou seja:

"As computações de programas fortemente equivalentes executam as MESMAS operações na MESMA ordem."

Instruções rotuladas compostas

Uma "instrução rotulada composta" é uma instrução do tipo:

$$r_1: \text{ se } T \text{ então faça } F \text{ vá\_para } r_2 \text{ senão faça } G \text{ vá\_para } r_3$$

- A "instrução rotulada composta" combina, em uma única instrução, testes e operações, e dispensa, portanto, a necessidade de uso de instruções distintas para executar operações e desviar o fluxo do controle;
- r_1 é dito "rótulo antecessor" de r_2 e r_3;
- r_2 e r_3 são ditos "rótulos sucessores" de r_1.

Programas monolíticos com instruções rotuladas compostas

Definição

Um "programa monolítico *com instruções rotuladas compostas*" P é um par ordenado:

$$P = (I, r)$$

onde:

- I é um conjunto (finito) de instruções rotuladas *compostas*;
- r é o rótulo inicial.

Observações:

- Duas instruções não podem ter o mesmo rótulo;
- Rótulos finais são aqueles que são referenciados mas não estão associados a nenhuma instrução;

Programas monolíticos com instruções rotuladas compostas

Definição

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

- A instrução rotulada composta acima será abreviada por:

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

- Para simplificar a demonstração da verificação da equivalência forte de programas monolíticos, considera-se que exista apenas um único identificador de teste, denotado T;
- Os resultados podem ser estendidos para o caso de programas com mais de um identificador de teste.

Programas monolíticos com instruções rotuladas compostas

Extensão

- Suponha que o programa contenha n identificadores de teste, denotados $T_1, T_2, ..., T_n$. Nesse caso, existem 2^n combinações possíveis para os resultados desses testes;
- A instrução rotulada composta poderia ser representada:

$$r : (O_1, r_1)(O_2, r_2)...(O_{2^n}, r_{2^n})$$

  onde cada ramo da instrução corresponderia a uma particular combinação dos valores desses testes, ou seja, (O_1, r_1) se $T_1 = T_2 = ... = T_n =$falso, ..., (O_{2^n}, r_{2^n}) se $T_1 = T_2 = ...T_n =$verdadeiro;
- Como os testes são avaliados isoladamente, apenas dois ramos distintos (O, r) seriam usados em cada instrução com 2^n pares.

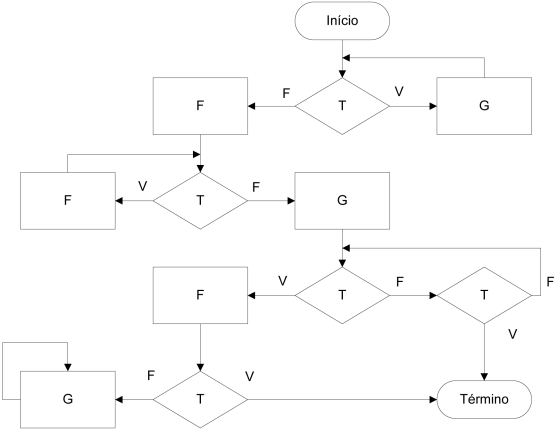
Conversão para instruções rotuladas compostas

Algoritmo

- Representar o programa monolítico na forma de um fluxograma;
- Atribuir rótulos numéricos para todas as operações;
- Atribuir o rótulo ϵ para o nó "Término" (que deve ser único);
- Considerar que os rótulos seguem os nós;
- Para cada rótulo numérico i, criar $i : (F, i')(G, i'')$ se:
  - A condição verdadeira para o teste T implica a execução da operação F;
  - Após a execução de F o próximo rótulo atingido é i';
  - A condição falsa para o teste T implica a execução da operação G;
  - Após a execução de G o próximo rótulo atingido é i''.
- Se um certo ramo da execução conduzir o programa a um loop infinito, deve-se usar (ciclo, w) e acrescentar a instrução rotulada composta $w : (\text{ciclo}, w)(\text{ciclo}, w)$ ao programa.

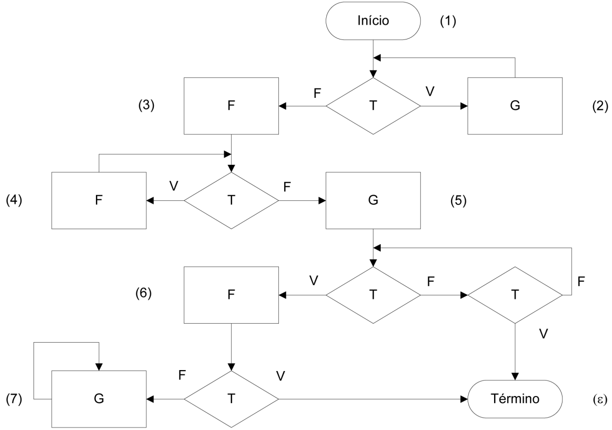
Conversão para instruções rotuladas compostas

Exemplo — Q



Conversão para instruções rotuladas compostas

Exemplo — Q



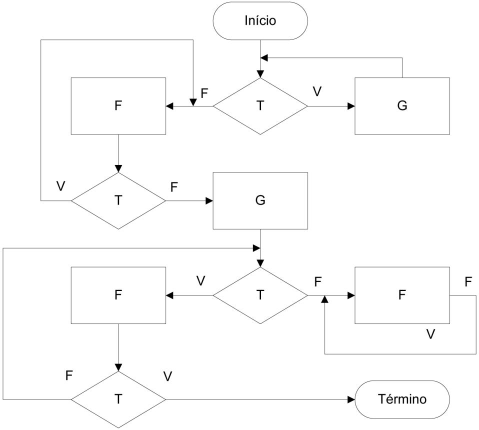
Conversão para instruções rotuladas compostas

Exemplo — Q

$1 : (G, 2)(F, 3)$
$2 : (G, 2)(F, 3)$
$3 : (F, 4)(G, 5)$
$4 : (F, 4)(G, 5)$
$5 : (F, 6)(\text{ciclo}, w)$
$6 : (\text{Término}, \epsilon)(G, 7)$
$7 : (G, 7)(G, 7)$
$w : (\text{ciclo}, w)(\text{ciclo}, w)$

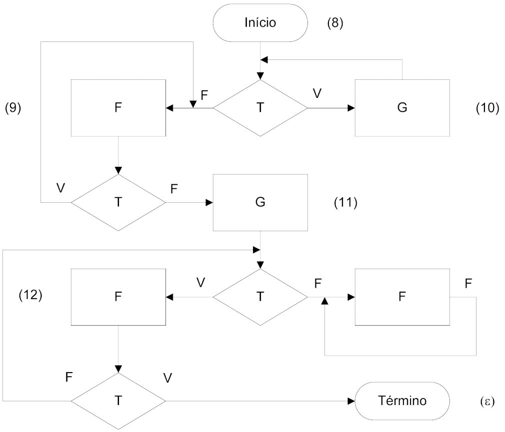
Equivalência forte de programas monolíticos

Exemplo — R



Equivalência forte de programas monolíticos

Exemplo — R



Conversão para instruções rotuladas compostas

Exemplo — R

$8 : (G, 10)(F, 9)$
$9 : (F, 9)(G, 11)$
$10 : (G, 10)(F, 9)$
$11 : (F, 12)(F, 13)$
$12 : (\text{Término}, \epsilon)(F, 13)$
$13 : (F, 13)(F, 13)$

Conversão para instruções rotuladas compostas

Equivalência forte

Sejam P um programa monolítico qualquer e P' o programa monolítico equivalente com instruções rotuladas compostas, obtido a partir do algoritmo apresentado anteriormente. Então:

$$\langle P, M\rangle(\epsilon) = \langle P', M\rangle(\epsilon)$$

para qualquer Máquina de Traços M e, portanto:

$$P \equiv P'$$

União disjunta de conjuntos

Definição e exemplo

A "união disjunta" de dois conjuntos A e B, denotado $A \sqcup B$, é o conjunto formado pelos elementos de A e de B, devidamente indexados com os nomes dos conjuntos de origem. Diferentemente da união simples, na união disjunta elementos repetidos não são representados por uma única cópia. Caso existam elementos repetidos em ambos os conjuntos, todos eles deverão fazer parte de $A \sqcup B$, porém devidamente identificados com o nome do conjunto de origem. Considere $A = \{a, x\}$ e $B = \{b, x\}$. Então a união disjunta de A e B resulta em:

$$\{a_A, x_A, b_B, x_B\}$$

ou, simplesmente:

$$\{a, x_A, b, x_B\}$$

União disjunta de conjuntos

Equivalência forte

- Sejam $Q = (I_Q, q)$ e $R = (I_R, r)$ dois programas monolíticos especificados usando instruções rotuladas compostas, e sejam $P_q = (I, q)$ e $P_r = (I, r)$ programas monolíticos onde $I = I_Q \sqcup I_R$. Então:

$$(P_q \equiv P_r) \Leftrightarrow (Q \equiv R)$$

- A verificação da equivalência forte de Q e R corresponde à verificação da equivalência forte de P_q e P_r.

Cadeia de conjuntos

Definições

Seja $A_0 A_1 ...$ uma seqüência de conjuntos.

- Ela é dita “cadeia de conjuntos”, se $\forall k \geq 0$,

$$A_k \subseteq A_{k+1}$$

- Ela é dita “cadeia finita de conjuntos”, se $\exists n \; \forall k \geq 0$, tal que:

$$A_n = A_{n+k}$$

- O “limite de uma cadeia finita de conjuntos” é A_n, onde n é o menor inteiro obtido acima. Denota-se:

$$\lim A_k = A_n$$

Cadeia de conjuntos

Exemplo

Seja $A_0 \subseteq A_1 \subseteq A_2 \subseteq A_3 \subseteq A_4 \subseteq A_5 = A_6 = A_7 ...,$

- Trata-se de uma cadeia de conjuntos, pois $\forall k \geq 0$,

$$A_k \subseteq A_{k+1}$$

- Trata-se de uma cadeia finita de conjuntos, pois $n = 5, 6, 7, ...$ são tais que $\forall k \geq 0$, $A_n = A_{n+k}$
- O limite dessa cadeia finita de conjuntos é A_5, pois $n = 5$ é o menor inteiro obtido acima.

$$\lim A_k = A_5$$

Ciclos infinitos

Identificação em programas monolíticos

Seja I um conjunto formado por n instruções rotuladas compostas e considere-se a seqüência de conjuntos $A_0 A_1 ...$ definida de forma indutiva:

- $A_0 = \{\epsilon\}$
- $\forall k \geq 0, A_{k+1} = A_k \cup$ $\{r | r$ é rótulo antecessor de instrução rotulada por elemento de $A_k\}$

Prova-se que $A_0 A_1 ...$ é uma seqüência finita de conjuntos e que, para qualquer rótulo r de instrução de I:

$$((I, r) \equiv (I, w)) \Leftrightarrow (r \notin \lim A_k)$$

Ou seja, todos os rótulos $r \neq w$ tais que $r \notin \lim A_k$ caracterizam ciclos infinitos.

Ciclos infinitos

Exemplo — Q

$1 : (G, 2)(F, 3)$
$2 : (G, 2)(F, 3)$
$3 : (F, 4)(G, 5)$
$4 : (F, 4)(G, 5)$
$5 : (F, 6)(\mathsf{ciclo}, w)$
$6 : (\mathsf{Término}, \epsilon)(G, 7)$
$7 : (G, 7)(G, 7)$
$w : (\mathsf{ciclo}, w)(\mathsf{ciclo}, w)$

Ciclos infinitos

Exemplo — Q

- $A_0 : \{\epsilon\}$
  $A_1 : \{6, \epsilon\}$
  $A_2 : \{5, 6, \epsilon\}$
  $A_3 : \{3, 4, 5, 6, \epsilon\}$
  $A_4 : \{1, 2, 3, 4, 5, 6, \epsilon\}$
  $A_5 : \{1, 2, 3, 4, 5, 6, \epsilon\}$
- Portanto:
  $\lim A_k = A_4 = \{1, 2, 3, 4, 5, 6, \epsilon\}$
  $(I, 7) \equiv (I, w)$ pois $7 \notin A_4$

Ciclos infinitos

Exemplo — R

$8 : (G, 10)(F, 9)$
$9 : (F, 9)(G, 11)$
$10 : (G, 10)(F, 9)$
$11 : (F, 12)(F, 13)$
$12 : (\text{Término}, \epsilon)(F, 13)$
$13 : (F, 13)(F, 13)$

Ciclos infinitos

Exemplo — R

- $A_0 : \{\epsilon\}$
  $A_1 : \{12, \epsilon\}$
  $A_2 : \{11, 12, \epsilon\}$
  $A_3 : \{9, 11, 12, \epsilon\}$
  $A_4 : \{8, 9, 10, 11, 12, \epsilon\}$
  $A_5 : \{8, 9, 10, 11, 12, \epsilon\}$
- Portanto:
  $\lim A_k = A_4 = \{8, 9, 10, 11, 12, \epsilon\}$
  $(I, 13) \equiv (I, w)$ pois $13 \notin A_4$

Ciclos infinitos

Algoritmo de simplificação

Seja I um conjunto de instruções rotuladas compostas. A "simplificação de ciclos infinitos" em I é feita em três passos:

- Calcular a seqüência finita de conjuntos de rótulos $A_0 A_1 ...$;
- $\forall r \notin \lim A_k$:
  - Excluir a instrução rotulada por r de I;
  - Todos os pares (F, r) em I são substituídos por (ciclo, w);
- Incluir a instrução $w : (\text{ciclo}, w)(\text{ciclo}, w)$ caso a mesma não faça parte do programa.

Ciclos infinitos

Exemplo — Q

Em função do resultado anterior, a aplicação do algoritmo resulta em:

- $1 : (G,2)(F,3)$
  $2 : (G,2)(F,3)$
  $3 : (F,4)(G,5)$
  $4 : (F,4)(G,5)$
  $5 : (F,6)(\text{ciclo},w)$
  $6 : (\text{Término},\epsilon)(\text{ciclo},w)$
  $w : (\text{ciclo},w)(\text{ciclo},w)$

Note:

- A eliminação da instrução $7 : (G,7)(G,7)$
- A substituição da instrução $6 : (\text{Término},\epsilon)(G,7)$ por $6 : (\text{Término},\epsilon)(\text{ciclo},w)$

Ciclos infinitos

Exemplo — R

Em função do resultado anterior, a aplicação do algoritmo resulta em:

- $8 : (G, 10)(F, 9)$
  $9 : (F, 9)(G, 11)$
  $10 : (G, 10)(F, 9)$
  $11 : (F, 12)(\text{ciclo}, w)$
  $12 : (\text{Término}, \epsilon)(\text{ciclo}, w)$
  $w : (\text{ciclo}, w)(\text{ciclo}, w)$

Note:

- A eliminação da instrução $13 : (F, 13)(F, 13)$
- A substituição da instrução $11 : (F, 12)(F, 13)$ por $11 : (F, 12)(\text{ciclo}, w)$ e da instrução $12 : (\text{Término}, \epsilon)(F, 13)$ por $12 : (\text{Término}, \epsilon)(\text{ciclo}, w)$
- A inclusão da instrução $w : (\text{ciclo}, w)(\text{ciclo}, w)$

Rótulos consistentes

Definição

- Seja I um conjunto finito de instruções rotuladas compostas e simplificadas.
- Sejam r e s dois rótulos de instruções do conjunto I, ambos diferentes de ϵ.

Suponha que as instruções rotuladas por r e s sejam:

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

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

Então r e s são ditos "rótulos consistentes" se e somente se:

$$(F_1 = G_1) \wedge (F_2 = G_2)$$

Rótulos equivalentes fortemente

Definição

- Seja I um conjunto finito de instruções rotuladas compostas e simplificadas.
- Sejam r e s dois rótulos de instruções do conjunto I.

Então r e s são ditos "rótulos fortemente equivalentes" se, e somente se:

- $r = s = \epsilon$, ou
- r e s são consistentes.

Rótulos equivalentes fortemente

Determinação

- Seja I um conjunto finito de instruções rotuladas compostas e simplificadas.
- Sejam r e s dois rótulos de instruções do conjunto I.

A seqüência de conjuntos $B_0 B_1 \ldots$ é definida indutivamente da seguinte forma:

- $B_0 = \{(r, s)\}$
- $\forall k \geq 0, B_{k+1} = \{(r'', s'') | (r', s') \in B_k,$
  r'' é sucessor de r',
  s'' é sucessor de s' e
  $\forall 0 \leq i \leq k, (r'', s'') \notin B_i\}$

Equivalência forte de programas monolíticos

Algoritmo

- Sejam $Q = (I_Q, q)$ e $R = (I_R, r)$ dois programas monolíticos especificados usando instruções rotuladas compostas e simplificadas.
- O algoritmo apresentado a seguir verifica se Q e R são equivalentes fortemente.

Equivalência forte de programas monolíticos

Algoritmo

1. Obter $P_q = (I, q)$ e $P_r = (I, r)$ onde $I = I_Q \sqcup I_R$. A instrução w, se existir, deverá ocorrer no máximo uma vez em I;
2. Se q e r são rótulos fortemente equivalentes, então $B_0 = \{(q, r)\}$. Caso contrário, $Q \not\equiv R$ e FIM;
3. $k \leftarrow 0$;
4. Obter B_{k+1} contendo os pares (q'', r'') de rótulos sucessores de cada $(q', r') \in B_k$, tais que:
   - $q' \neq r'$
   - $(q' \neq \epsilon) \wedge (r' \neq \epsilon)$
   - $\forall 0 \leq i \leq k, (q'', r'') \notin B_i$
5. Considere B_{k+1}:
   - $B_{k+1} = \emptyset$: $Q \equiv R$ e FIM;
   - $B_{k+1} \neq \emptyset$: Se todos os pares de B_{k+1} são fortemente equivalentes, $k \leftarrow k + 1$ e vá para 4. Senão, $Q \not\equiv R$ e FIM.

Equivalência forte de programas monolíticos

Exemplo — (I_Q, q)

(I_Q, q), com I_Q abaixo (já simplificado) e $q = 1$

- $1 : (G, 2)(F, 3)$
  $2 : (G, 2)(F, 3)$
  $3 : (F, 4)(G, 5)$
  $4 : (F, 4)(G, 5)$
  $5 : (F, 6)(\text{ciclo}, w)$
  $6 : (\text{Término}, \epsilon)(\text{ciclo}, w)$
  $w : (\text{ciclo}, w)(\text{ciclo}, w)$

Equivalência forte de programas monolíticos

Exemplo — (I_R, r)

(I_R, r), com I_R abaixo (já simplificado) e $r = 8$

- $8 : (G, 10)(F, 9)$
  $9 : (F, 9)(G, 11)$
  $10 : (G, 10)(F, 9)$
  $11 : (F, 12)(\text{ciclo}, w)$
  $12 : (\text{Término}, \epsilon)(\text{ciclo}, w)$
  $w : (\text{ciclo}, w)(\text{ciclo}, w)$

Equivalência forte de programas monolíticos

Exemplo — $I_Q \sqcup I_R$

- $1 : (G, 2)(F, 3)$
  $2 : (G, 2)(F, 3)$
  $3 : (F, 4)(G, 5)$
  $4 : (F, 4)(G, 5)$
  $5 : (F, 6)(\text{ciclo}, w)$
  $6 : (\text{Término}, \epsilon)(\text{ciclo}, w)$
  $8 : (G, 10)(F, 9)$
  $9 : (F, 9)(G, 11)$
  $10 : (G, 10)(F, 9)$
  $11 : (F, 12)(\text{ciclo}, w)$
  $12 : (\text{Término}, \epsilon)(\text{ciclo}, w)$
  $w : (\text{ciclo}, w)(\text{ciclo}, w)$

Equivalência forte de programas monolíticos

Exemplo

- 1 e 8 são rótulos equivalentes fortemente $\Rightarrow B_0 = \{(1, 8)\}$;
- $B_1 = \{(2, 10), (3, 9)\}$, $(2, 10)$ e $(3, 9)$ são dois pares de rótulos equivalentes fortemente;
  $B_2 = \{(4, 9), (5, 11)\}$, $(4, 9)$ e $(5, 11)$ são dois pares de rótulos equivalentes fortemente;
  $B_3 = \{(6, 12), (w, w)\}$, $(6, 12)$ e (w, w) são dois pares de rótulos equivalentes fortemente;
  $B_4 = \{(\epsilon, \epsilon)\}$, (ϵ, ϵ) é um par de rótulos equivalentes fortemente;
  $B_5 = \{\}$;
- Portanto, $(I, 1) \equiv (I, 8)$ e, conseqüentemente, $Q \equiv R$.

Exercícios

Do livro "Teoria da Computação", de T.A. Diverio e P.B. Menezes:

- 2.12
- 2.13
- 2.14
- 2.17
- 2.18
- 2.20
- 2.21
- 2.22
- 2.23
- 2.24
- 2.25