Teoria da Computação

Máquinas, Computações e Funções Computadas

Cristiano Lehrer, M.Sc.

Introdução (1/2)

- O objetivo de uma máquina é suprir todas as informações necessárias para que a computação de um programa possa ser descrita.
- Portanto, cabe à máquina suprir o significado aos identificadores das operações e testes.
- Cada identificador de operação interpretado pela máquina deve ser associado a uma transformação na estrutura de memória.
- Cada identificador de teste interpretado pela máquina deve ser associado a uma função verdade.

Introdução (2/2)

- Observação:
  - Nem todo o identificador de operação ou teste é definido em uma máquina.
  - Para cada identificador de operação ou teste definido em um máquina, existe somente uma função associada.
- A máquina deve descrever o armazenamento ou recuperação de informações na estrutura de memória.

Definição de uma Máquina

- Uma máquina é uma 7-upla $M = (V, X, Y, \pi x, \pi y, \Pi F, \Pi T)$ onde:
  - V – conjunto de valores de memória.
  - X – conjunto de valores de entrada.
  - Y – conjunto de valores de saída.
  - πx – função de entrada, tal que $\pi x: X \to V$
  - πy – função de saída, tal que $\pi y: V \to Y$
  - ΠF – conjunto de interpretações de operações, onde para cada identificador de operação F interpretado por M, existe uma única função:
    - $\pi F: V \to V$ em ΠF
  - ΠT – conjunto de interpretações de testes, tal que para cada identificador de teste T interpretado por M, existe uma única função:
    - $\pi T: V \to \{\text{verdadeiro, falso}\}$ em ΠT

Máquina de Dois Registrados (1/3)

- Suponha uma especificação de uma máquina com dois registradores, a e b, os quais assumem valores em $\aleph$, com duas operações e um teste como segue:
  - Subtração de 1 em a, se $a \geq 0$.
  - Adição de 1 um b.
  - Teste se a é zero.
- Adicionalmente, valores de entrada são armazenados em a (zerando b) e a saída retorna o valor de b.
- Dois registradores com valores em $\aleph$ podem ser definidos pelo produto cartesiano $\aleph^2$ onde os registradores a e b são representados pela primeira e segunda componente, respectivamente.

Máquina de Dois Registrados (2/3)

- dois_reg = ($\aleph^2$, $\aleph$, $\aleph$, armazena_a, retorna_b, {subtrair_a, adicionar_b}, {a_zero}), onde:
  - $\aleph^2$ corresponde ao conjunto de valores de memória.
  - $\aleph$ corresponde, simultaneamente, aos conjuntos de valores de entrada e saída.
  - armazena_a: $\aleph \rightarrow \aleph^2$ é a função de entrada, tal que $\forall n \in \aleph$:
    - armazena_a(n) = (n, 0)
  - retorna_b: $\aleph \rightarrow \aleph^2$ é a função de saída, tal que $\forall (n, m) \in \aleph^2$:
    - retorna_b(n, m) = m
  - subtrair_a: $\aleph^2 \rightarrow \aleph^2$ é a interpretação, tal que $\forall (n, m) \in \aleph^2$:
    - subtrair_a(n, m) = (n – 1, m), se $n \neq 0$
    - subtrair_a(n, m) = (0, m), se n = 0

Máquina de Dois Registrados (3/3)

- continuação:
  - adicionar_b: $\aleph^2 \rightarrow \aleph^2$ é a interpretação, tal que $\forall(n, m) \in \aleph^2$:
    - adicionar_b(n, m) = (n, m + 1)
  - a_zero: $\aleph^2 \rightarrow$ {verdadeiro, falso} é a interpretação, tal que $\forall(n, m) \in \aleph^2$:
    - a_zero(n, m) = verdadeiro, se n = 0
    - a_zero(n, m) = falso, se $n \neq 0$
- Exemplo de um programa monolítico, para fazer b = a:
  - R1: se a_zero então vá_para Rx senão vá_para R2;
  - R2: faça subtrair_a vá_para R3;
  - R3: faça adicionar_b vá_para R1;

Computação (1/3)

- Uma computação é, resumidamente, um histórico do funcionamento da máquina para o programa, considerando um valor inicial.
- Computação referente aos programas monolíticos:
  - Basicamente, uma computação de um programa monolítico em uma máquina é um histórico das instruções executadas e o correspondente valor de memória.
  - O histórico é representado na forma de uma cadeia de pares onde:
    - Cada par reflete um estado da máquina para o programa, ou seja, a instrução a ser executada e o valor corrente de memória.
    - A cadeia reflete uma sequencia de estados possíveis a partir do estado inicial (instrução inicial e valor de memória considerado).

Computação (2/3)

- Exemplos:
  - R1: Se a_zero então vá_para Rx senão vá_para R2;
  - R2: Faça subtrair_a vá_para R3;
  - R3: Faça adicionar_b vá_para R1;

(R1, (2, 0))
(R2, (2, 0))
(R3, (1, 0))
(R1, (1, 1))
(R2, (1, 1))
(R3, (0, 1))
(R1, (0, 2))
(Rx, (0, 2))

(R1, (3, 0))
(R2, (3, 0))
(R3, (2, 0))
(R1, (2, 1))
(R2, (2, 1))
(R3, (1, 1))
(R1, (1, 2))
(R2, (1, 2))
(R3, (0, 2))
(R1, (0, 3))
(Rx, (0, 3))

(R1, (4, 0))
(R2, (4, 0))
(R3, (3, 0))
(R1, (3, 1))
(R2, (3, 1))
(R3, (2, 1))
(R1, (2, 2))
(R2, (2, 2))
(R3, (1, 2))
(R1, (1, 3))
(R2, (1, 3))
(R3, (0, 3))
(R1, (0, 4))
(Rx, (0, 4))

Computação (3/3)

- Observações:
  - Para um dado valor inicial de memória, a correspondente cadeia de computação é única, ou seja, a computação é determinística.
  - Um teste e a operação vazia não alteram o valor corrente da memória.
  - Em uma computação infinita, rótulo algum da cadeia é final.

Função Computada (1/3)

- Em geral, a computação de um programa deve ser associada a uma entrada e uma saída.
- Adicionalmente, espera-se que a resposta (saída) seja gerada em um tempo finito.
- A função computada por um programa monolítico sobre uma máquina corresponde à noção intuitiva:
  - A computação inicia na instrução identificada pelo rótulo inicial com a memória contendo o valor inicial resultante da aplicação da função de entrada sobre o dado fornecido.
  - Executa, passo a passo, testes e operações, na ordem determinada pelo programa, até atingir um rótulo final, quando pára.
  - O valor da função computada pelo programa é o valor resultante da aplicação da função de saída ao valor da memória quando da parada.

Função Computada (2/3)

- Sejam M = (V, X, Y, πx, πy, ΠF, ΠT) uma máquina e P um programa monolítico para M.
- A Função Computada pelo Programa Monolítico P na Máquina M é denotada por:
  - <P, M>: X $\rightarrow$ Y, onde:
    - P – programa monolítico
    - M – máquina
    - X – entrada
    - Y – saída

Função Computada (3/3)

- Exemplos:
  - R1: Se a_zero então vá_para Rx senão vá_para R2;
  - R2: Faça subtrair_a vá_para R3;
  - R3: Faça adicionar_b vá_para R1;

(R1, (2, 0))
(R2, (2, 0))
(R3, (1, 0))
(R1, (1, 1))
(R2, (1, 1))
(R3, (0, 1))
(R1, (0, 2))
(Rx, (0, 2))

<MOV, 2_REG>:2 → 2

(R1, (3, 0))
(R2, (3, 0))
(R3, (2, 0))
(R1, (2, 1))
(R2, (2, 1))
(R3, (1, 1))
(R1, (1, 2))
(R2, (1, 2))
(R3, (0, 2))
(R1, (0, 3))
(Rx, (0, 3))

<MOV, 2_REG>:3 → 3

(R1, (4, 0))
(R2, (4, 0))
(R3, (3, 0))
(R1, (3, 1))
(R2, (3, 1))
(R3, (2, 1))
(R1, (2, 2))
(R2, (2, 2))
(R3, (1, 2))
(R1, (1, 3))
(R2, (1, 3))
(R3, (0, 3))
(R1, (0, 4))
(Rx, (0, 4))

<MOV, 2_REG>:4 → 4

Máquina 4_REG (1/2)

- 4_REG = (N^4, N, N, armazenar, retornar, {incA, decA, incB, decB, incC, decC, incD}, {nilA, nilB, nilC}), onde:
- armazenar → armazena o valor fornecido pelo usuário no registrador A, zerando os demais;
- retornar → retorna o valor armazenado no registrador D;
- `incA` → incrementa o registrador A em uma unidade;
- `decA` → decrementa o registrador A em uma unidade, caso o mesmo seja maior do que zero;
- `incB` → incrementa o registrador B em uma unidade;
- `decB` → decrementa o registrador B em uma unidade, caso o mesmo seja maior do que zero;

Máquina 4_REG (2/2)

- `incC` → incrementa o registrador C em uma unidade;
- `decC` → decrementa o registrador C em uma unidade, caso o mesmo seja maior do que zero;
- `incD` → incrementa o registrador D em uma unidade;
- `nilA` → retornar verdade caso o valor do registrador A seja zero, caso contrário, falso;
- `nilB` → retornar verdade caso o valor do registrador B seja zero, caso contrário, falso;
- `nilC` → retornar verdade caso o valor do registrador C seja zero, caso contrário, falso;