Blocos Básicos e Grafos de Fluxo de Controle

Sandro Rigo
sandro@ic.unicamp.br

Introdução

- Representação gráfica do código de 3 endereços é útil para entender os algoritmos de otimização
- Nós: computação
- Arestas: fluxo de controle
- Muito usado em coletas de informações sobre o programa

Blocos Básicos

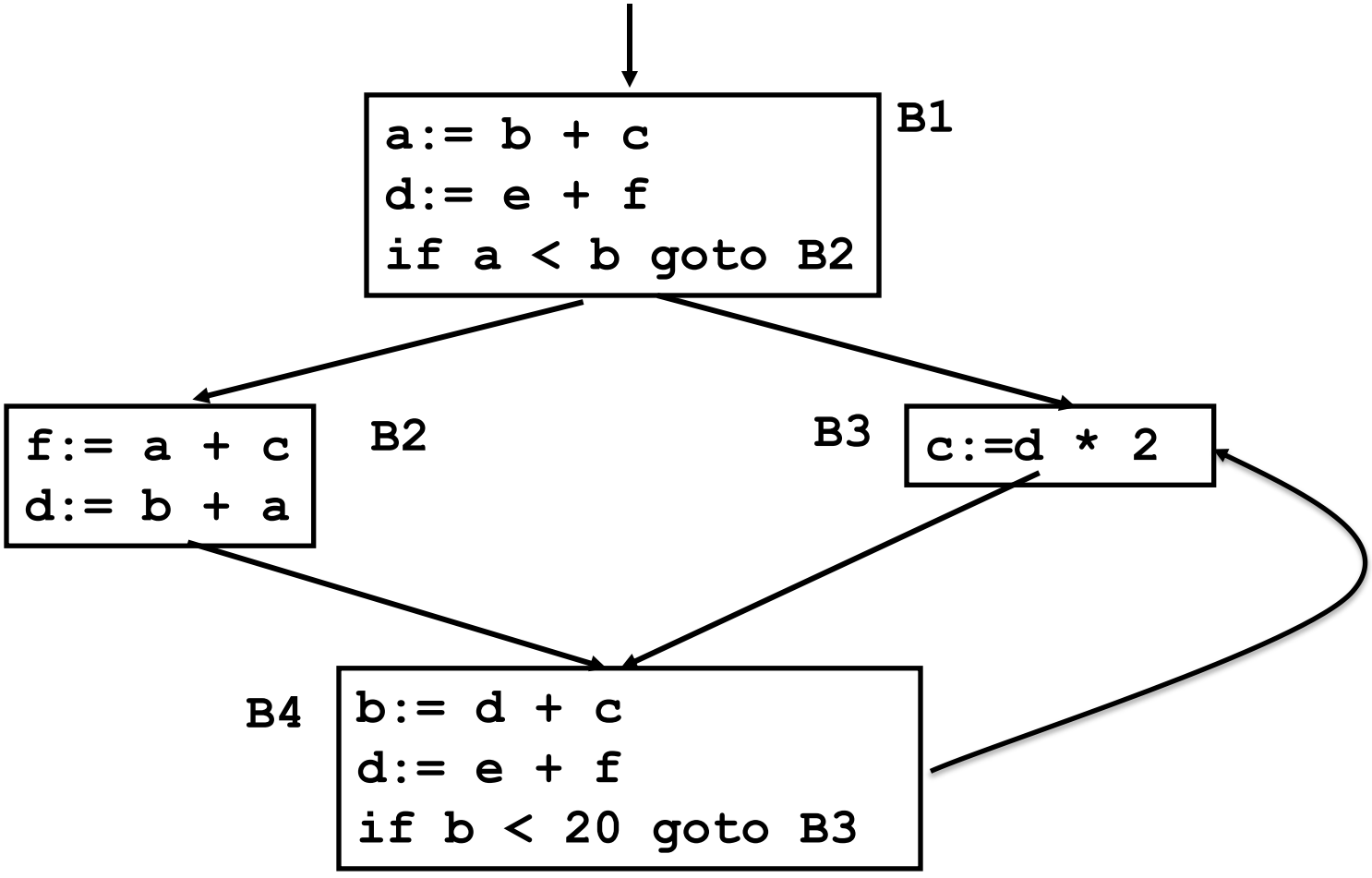
- Seqüência de instruções consecutivas
- Fluxo de Controle:
  - Entra no início
  - Sai pelo final
  - Não existem saltos para dentro ou do meio para fora da seqüência

```
t1 = a * a
t2 = a * b
t3 = b * 3
t4 = t2 - t3
```

Algoritmo para Quebrar em BBs

- Entrada: seqüência de código 3 endereços
- Defina os líderes (iniciam os BBs):
  - Primeira Sentença é um líder
  - Todo alvo de um goto, condicional ou incondicional, é um líder
  - Toda sentença que sucede imediatamente um goto, condicional ou incondicional, é um líder
- Os BBs são compostos pelos líderes e todas as instruções subsequentes até o próximo líder (exclusive)

Grafo de Fluxo de Controle (CFG)



Laços

- O que é um laço?
- Como encontrá-los?
- Na figura anterior: B3, B4
- Um laço é um conjunto de nós tais que:
  - Todos os nós são fortemente conectados
  - Têm uma única entrada
- Um laço que não contém outros laços é um *inner loop*

Directed Acyclic Graphs

- Úteis para transformações em BBs
- Mostra como os valores computados são usados em sentenças subseqüentes
- Common Sub-expression Elimination (CSE)
- Não confundir com o CFG
  - DAG: representa um BB
  - CFG: Nós são os BBs

Directed Acyclic Graphs

- Construção:
  - Folhas são identificadores únicos
    - variáveis, constantes
    - são os valores iniciais das variáveis
    - usa-se índices para não confundir com valor atual

Directed Acyclic Graphs

- Construção:
  - Nós internos são operadores:
    - valores computados
    - O rótulo é o operador associado
    - podem ter uma lista de variáveis associados
      - é o último valor computado para cada uma delas
    - um nó associado a cada sentença
    - filhos representam a última definição dos operandos

Exemplo - DAGs

```
(1)t1 := 4 * i
(2)t2 := a [ t1]
(3)t3 := 4 * i
(4)t4 := b [t3]
(5)t5 := t2 * t4
(6)t6 := prod + t5
(7)prod := t6
(8) t7 := i + 1
(9) i := t7
(10) if i <= 20 goto (1)
```

Aplicações

- Detectar sub-expressões comuns
  - automaticamente durante a construção
- Detectar os identificadores cujos valores são usados no bloco
  - São as folhas
- Detectar sentenças que geram valores que podem ser usados fora do bloco
  - São aquelas sentenças que geraram nó(x) = n durante a construção e ainda temos nó(x) =n ao final

Cuidados

- Arrays
  - `x := a[i]`
  - `a[j] := y`
  - `z := a[i]`
- Pode virar
  - `x := a[i]`
  - `z := x`
  - `a[j] := y`
- Ponteiros, problema similar
  - `*p=w`