# Alias Analysis

**Sandro Rigo**
**sandro@ic.unicamp.br**

# Introdução

- Alias
  - Duas ou mais expressões que denotam o mesmo endereço de memória
- Complicam as análises de fluxo de dados
  - Causam incertezas sobre o que é definido e usado
  - Se não sabemos nada sobre onde o apontador p pode apontar
    - Uma atribuição através deste apontador pode mudar qualquer variável
    - Um uso pode usar qualquer variável

# Introdução

- Complicam as análises de fluxo de dados
  - Geram mais variáveis vivas e definições alcançantes do que o necessário
  - Geram menos expressões disponíveis do que o real
- Objetivo
  - Fazer uma análise para limitar os locais para onde os ponteiros podem estar apontando
  - Usar essa informação para tornar as outras análises mais precisas

# Introdução

- Conservativo
  - O conjunto de valores que podem ser apontados tem que incluir todas que realmente são
  - Porém, pode incluir algumas que não são
  - Não podemos inserir erro no programa

# Linguagem

- Tipos de 1 palavra
  - Integer e reals
  - Arrays destes tipos
- Ponteiros
  - Para variáveis
  - Para arrays
  - Mas não para outros ponteiros
  - Sabemos que aponta para o array a, mas não para qual elemento de a
  - Declarados como ponteiros ou são temporários que receberam um ponteiro mais ou menos alguma constante

# Regras

1. S: p =&a
   - Imediatamente após s, p aponta apenas para a
   - Se a é um array
     - S: p = &a ± c
     - p aponta para um elemento de a
2. s: p= q ± c, onde q é ponteiro
   - Depois de s, p pode apontar para qualquer array que q apontava antes de s, mas nada mais.
3. s: p= q
   - Depois de s, p pode apontar para qualquer coisa que q apontava antes de s

# Regras

4. Após qualquer outra atribuição a p
   - Não há objeto para o qual p possa apontar
   - Instrução provavelmente inútil
5. Após qualquer outra atribuição a uma variável que não seja p
   - p continua apontando para onde apontava anteriormente
   - Assume que não existe apontador para apontador

# Análise de Fluxo de Dados

- In[B]
  - Para cada apontador p
    - Determina o conjunto de variáveis para as quais p pode apontar no início de B
  - Conjunto de pares (p, a)
    - p apontador
    - a variável
    - p pode apontar para a
  - Na prática
    - Uma lista para cada apontador

# Análise de Fluxo de Dados

- out[B]
  - O mesmo para o final de B
- Funções $trans_B$:
  - Definem o efeito do bloco B
  - Argumentos: conjunto de pares S da forma (p,a)
  - Resultado: outro conjunto de pares T
- Computamos trans para sentenças
  - $trans_B$ é a composição de $trans_s$ para cada s do bloco B

# Computando Funções *trans*

1. Se s é p = &a ou p = &a ± c

$$trans_s(S) = (S - \{(p,b) \mid \text{any variable } b\}) \bigcup \{(p,a)\}$$

2. Se s é p = &q ± c, q apontador

$$trans_s(S) = (S - \{(p,b) \mid \text{any variable } b\})$$
$$\bigcup \{(p,b) \mid (q,b) \text{ is in } S \text{ and } b \text{ is an array variable}\}$$

# Computando Funções *trans*

3. Se s é p = q

$$trans_s(S) = (S - \{(p,b) \mid \text{any variable } b\}) \bigcup \{(p,b) \mid (q,b) \text{ is in } S\}$$

4. Se s atribui a p qualquer outra expressão

$$trans_s(S) = S - \{(p,b) \mid \text{any variable } b\}$$

5. Se S não atribui apontador

$$trans_s(S) = S$$

# Análise de Fluxo de Dados

- Para um bloco B temos:

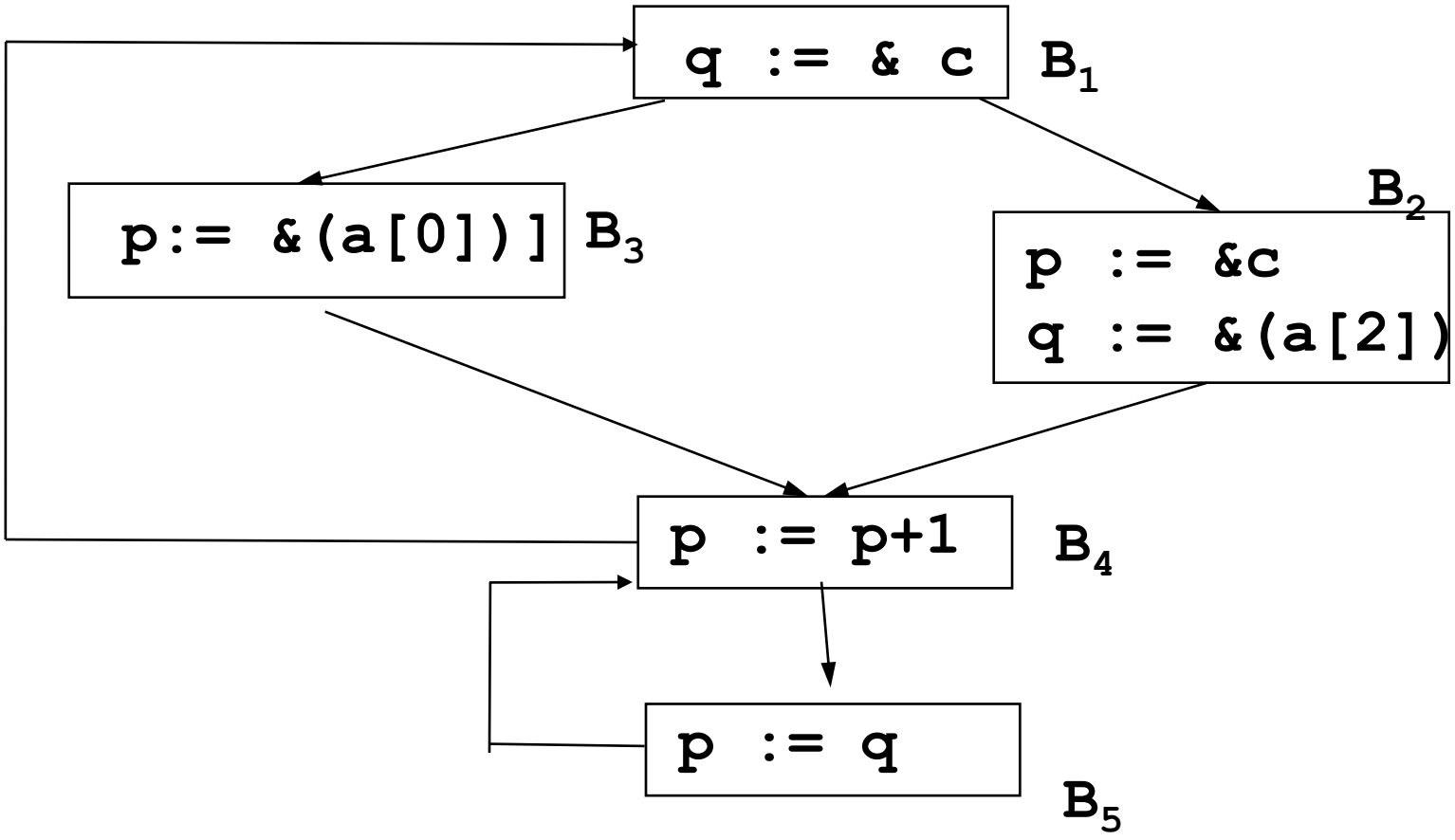
$$trans_B(S) = trans_{s_k}(trans_{s_{k-1}}(...trans_{s_2}(trans_{s_1}(S)))...))$$

- Equações DFA:

$$out[B] = trans_B(in[B])$$

$$in[B] = \bigcup_{P \in Pred(B)} out[P]$$

# Exemplo



# Como usar essa informação?

- in[B] é o conjto de variáveis apontadas por cada ponteiro em B
- usando trans podemos propagar essa informação a cada sentença
- Devemos usar essa informação de maneira conservativa

# Reaching Definitions

- Usa o mesmo algoritmo
- Gen e Kill os mesmo para sentenças que não usam ponteiros
- s:*p=a
  - gera definição de toda variável b tal que p possa apontar para b
  - mata definições de b somente se b não é um array e é a única variável que p pode apontar
    - permite que definições de b passem por s a menos que tenha certeza da redefinição de b

# Variáveis Vivas

- Usa o mesmo algoritmo
- def e use precisam considerar:
  - s:*p=a
    - usa a e p
    - define a variável b se b é a única variável que p pode apontar
      - permite que usos de b passem por s a menos que tenha certeza da redefinição de b

# Variáveis Vivas

- def e use precisam considerar:
  - s: a=*p
    - define a
    - usa p
    - usa toda variável b para qual p pode apontar
      - maximizando os usos possíveis estamos sendo conservativos