Alias Analysis

Alias Analysis

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
- Caso não se sabia nada sobre onde um apontador **p** pode apontar:
  - Uma atribuição através deste apontador pode mudar qualquer variável
  - Um uso deste apontador pode usar qualquer variável
- Geram mais variáveis vivas e definições alcançantes do que o necessário
- Geram menos expressões disponíveis do que o real

Alias Analysis

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

Análise Conservativa

- O conjunto de valores que podem ser apontados tem que incluir todos os que realmente são.
- Porém, pode incluir alguns valores que não o são de fato.
- Não se pode inserir erros no programa.

Alias Analysis: Linguagem Exemplo

Linguagem Exemplo - Considerar:

Tipos de 1 palavra

- `inteiro` e `real`
- Arrays destes tipos

Ponteiros

- Para variáveis
- Para arrays
- Não há ponteiros para outros ponteiros
- Pode apontar para o array `a`, mas não se sabe para qual elemento de `a`

Alias Analysis: Linguagem Exemplo - Regras

1. s: p = &a, onde a é uma variável:

- Imediatamente após **s**, **p** aponta apenas para 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, nada mais.

3. s: p = &a, onde a é um array:
s: p = &a ± c, onde a é um array:

- **p** aponta para um elemento do array **a**, sem ser um em específico.

4. s: p = q, onde q é ponteiro

- Depois de s, p aponta para qualquer coisa que q apontava antes de s

Alias Analysis: Linguagem Exemplo - Regras

5. Após qualquer outra atribuição a p

- Não há objeto para o qual p possa apontar
  - **Ex.: p = NULL;**

6. Após qualquer outra atribuição a uma variável que não seja p

- p continua apontando para onde apontava anteriormente

Alias Analysis: 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

Alias Analysis: 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

Computa-se *trans* para sentenças do bloco básico:

- $trans_B$ é a composição de $trans_s$ para cada sentença s do bloco B

Alias Analysis: Funções *trans*

1. **Se S é p = &a, a é variável**

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

2. **Se S é p = q ± c, q apontador**

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

3. **Se S é p = &a ou p = &a ± c, a array**

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

Alias Analysis: Funções *trans*

4. **Se S é p = q, q é ponteiro**

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

5. **Se S atribui a p qualquer outra expressão**

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

6. **Se S não atribui a um apontador**

$$trans_s(S) = S$$

Alias Analysis: Análise de Fluxo de Dados

Para um bloco B tem-se:

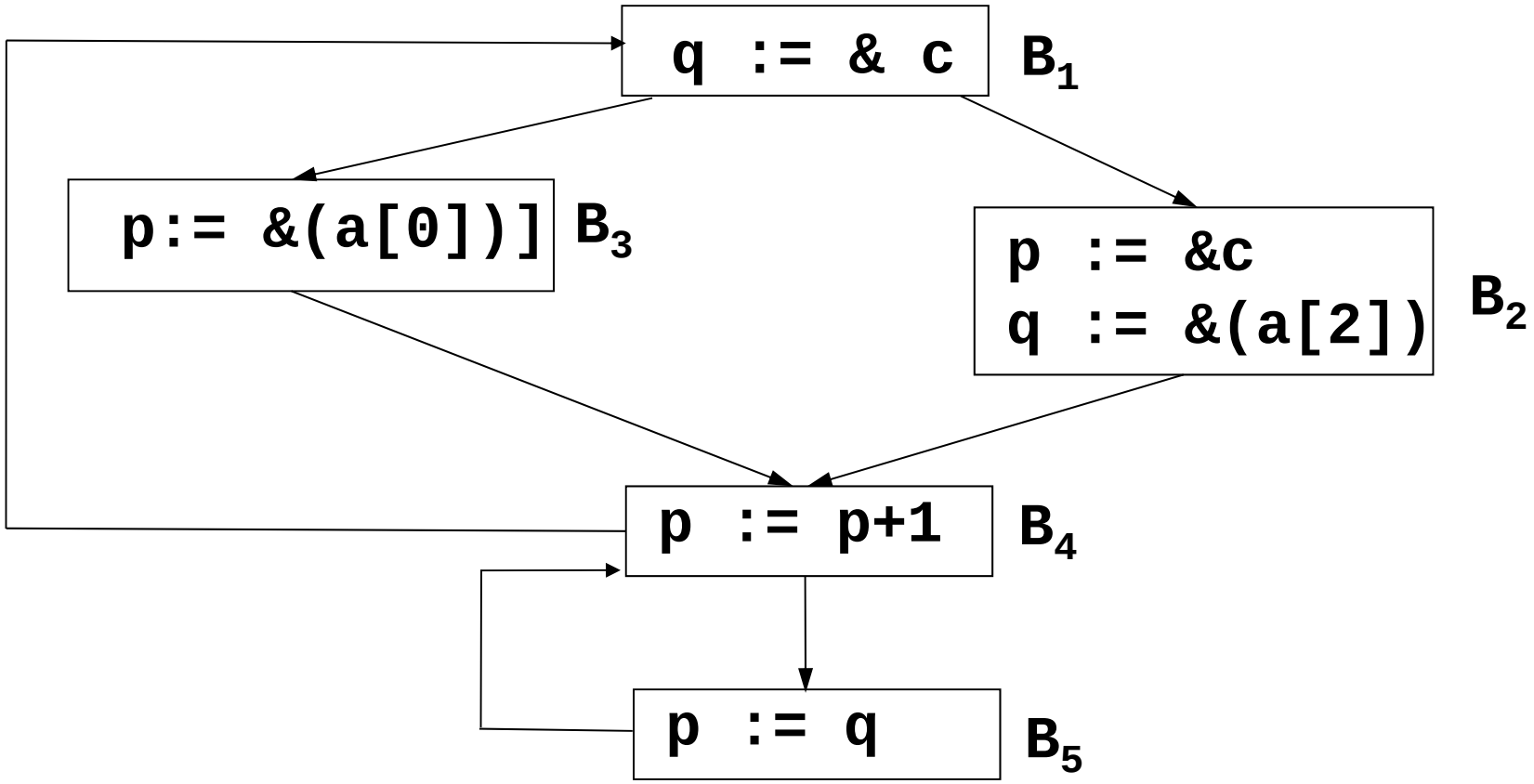
$$trans_B(S) = trans_{s_k}(trans_{s_{k-1}}(...trans_{s_2}(trans_{s_1}(S)))...))$$

Equações DFA:

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

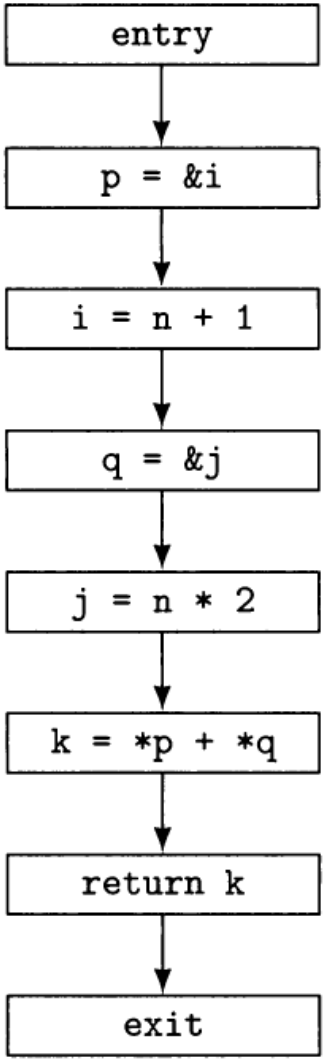
$$in[B] = \bigcup_{P \in Pred(B)} out[P]$$

Alias Analysis: Exemplo



Alias Analysis: Otimização de Ponteiros

```
int arith(n)
    int n;
{   int i, j, k, *p, *q;
    p = &i;
    i = n + 1;
    q = &j;
    j = n * 2;
    k = *p + *q;
    return k;
}
```



Alias Analysis: Como usar essa informação?

- in[B] é o conjunto de variáveis apontadas por cada ponteiro no início de B
- Usando *trans* pode-se propagar essa informação a cada sentença
- Deve-se usar essa informação de maneira conservativa

Alias Analysis: Reaching Definitions

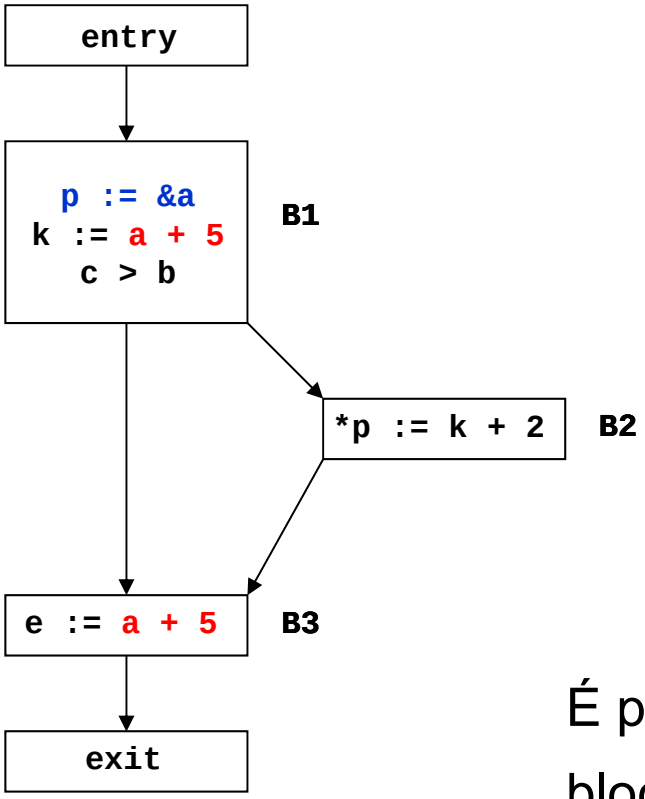
- Usa o mesmo algoritmo
- gen e kill são os mesmo para sentenças que não usam ponteiros
- **s : *p=a**
  - gera definição de toda variável **b** tal que **p** possa apontar para **b**
  - mata definições de **b** somente se **b** não é um array e é a única variável que **p** pode apontar
    - permite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

Alias Analysis: Variáveis Vivas

- Usa o mesmo algoritmo

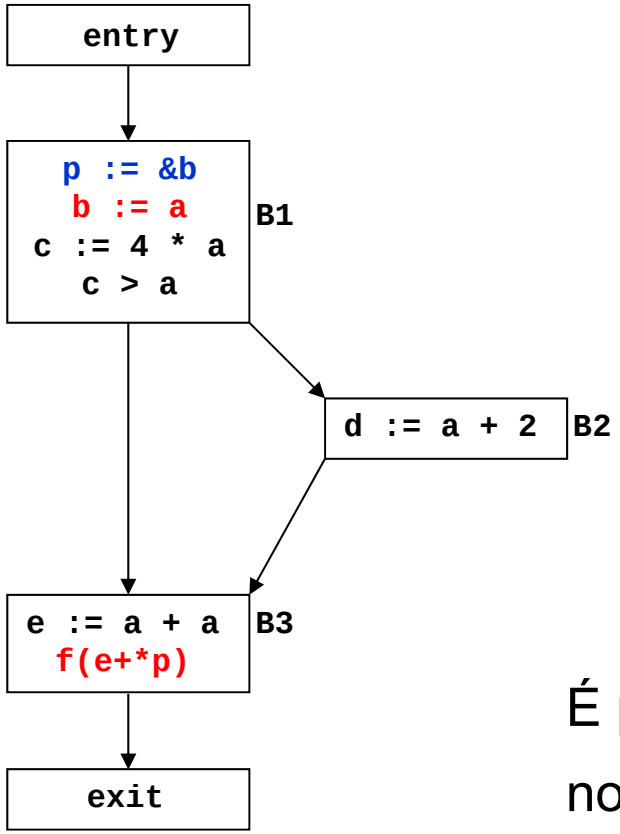
- def e use precisam considerar:
  - **`s:*p=a`**
    - usa **a** e **p**
    - define a variável **b** se **b** é a única variável que **p** pode apontar
      - permite que usos de **b** passem por **s** a menos que tenha certeza da redefinição de **b**
  - **`s: a=*p`**
    - define **a**
    - usa **p**
    - usa toda variável **b** para qual **p** pode apontar
      - maximizar os usos possíveis é a estratégia conservativa

Alias: CSE



É possível otimizar a expressão **a+5** no bloco B3?

Alias: Dead Code Elimination



É possível remover a definição **b := a** no bloco B1?