

– INF01147 –
Compiladores

Geração de Código
Esquemas de Tradução para Arranjos
Expressões booleanas e Controle de fluxo

Prof. Lucas M. Schnorr
– Universidade Federal do Rio Grande do Sul –



Plano da Aula de Hoje

- ▶ Revisão e aprofundamento
- ▶ Geração de TAC para arranjos Multidimensionais
 - ▶ Esquema de Tradução
- ▶ Expressões Booleanas
- ▶ Controle do Fluxo de Execução
- ▶ Bison: atributos herdados

Geração de TAC
Arranjos com $n - 1$ dimensões

Arranjos de $n - 1$ dimensões com low_k e $high_k$

Declaração na forma: $A[low_0..high_0][low_1..high_1]..[low_{n-1}..high_{n-1}]$

$$A[i_0][i_1]...[i_{n-1}]$$

$$endereco = base + d_k * w$$

$$d_k = \begin{cases} i_k - low_k & \text{se } k = 0 \\ d_{k-1} * |high_k - low_k| + (i_k - low_k) & \text{se } k \geq 1 \end{cases} \quad (1)$$

Geração de TAC para Arranjos Multidimensionais

- ▶ TAC para arranjos multidimensionais
 - ▶ Múltiplas operações aritméticas
 - ▶ Cresce com a quantidade de dimensões
- ▶ Ilustrando o problema com um exemplo: $A[4..6][-2..10]$
 - ▶ Cálculo para $A[5][-1]$, supondo $base = 16$ e $w = 4$
$$base + ((i_0 - low_0) * n_1 + (i_1 - low_1)) * w$$
$$16 + ((5 - 4) * 12 + (-1 - -2)) * 4$$
- ▶ Uma melhor abordagem
 - ▶ Otimizar o cálculo de endereços
- ▶ Solução
 - ▶ Separar a fórmula de cálculo de endereço em duas
 - ▶ Uma com a parte **variável**
 - considera todos os índices i_k
 - ▶ Uma segunda parte **constante**
 - considera $base$, w , low_k e $high_k$ (e por consequência o n_k)

Isolando a parte constante e a variável

Exemplo com duas dimensões

- ▶ $base + [(i_0 - low_0) * n_1 + (i_1 - low_1)] * w$
 - ▶ $base + (i_0 - low_0) * n_1 * w + (i_1 - low_1) * w$
 - ▶ $base + i_0 * n_1 * w - low_0 * n_1 * w + i_1 * w - low_1 * w$
 - ▶ $base - low_0 * n_1 * w - low_1 * w + i_0 * n_1 * w + i_1 * w$
 - ▶ $base - (low_0 * n_1 + low_1) * w + (i_0 * n_1 + i_1) * w$
-
- ▶ Qual a característica das partes constante e variável?
 - ▶ **Recursiva**

Definições recursivas

- **Declaração**, seja $A[low_0..high_0][low_1..high_1]..[low_k..high_k]$
- Cálculo do valor constante C_A de um arranjo A

$$C_A = base - r_k * w \quad (2)$$

$$r_k = \begin{cases} low_k & \text{se } k = 0 \\ r_{k-1} * |high_k - low_k| + low_k & \text{se } k \geq 1 \end{cases} \quad (3)$$

- **Acesso**, seja $A[i_0][i_1]..[i_k]$
- Cálculo do endereço final

$$endereco = C_A + d_k * w \quad (4)$$

$$d_k = \begin{cases} i_k & \text{se } k = 0 \\ d_{k-1} * |high_k - low_k| + i_k & \text{se } k \geq 1 \end{cases} \quad (5)$$

Esquemas de tradução
Arranjos com $n - 1$ dimensões

Esquemas de Tradução (Declaração)

- Uma gramática para declaração de arranjos multidimensionais

$$\begin{aligned} \text{decl} &\rightarrow \text{T id [L]} \\ \text{T} &\rightarrow \text{float} \\ \text{L} &\rightarrow \text{L , D} \\ \text{L} &\rightarrow \text{D} \\ \text{D} &\rightarrow \text{N..N} \\ \text{N} &\rightarrow \text{num} \end{aligned}$$

- Precisamos um esquema de tradução para definir C_A
(repetindo o cálculo recursivo que permite obtê-lo)

$$C_A = \text{base} - r_k * w \quad (6)$$

$$r_k = \begin{cases} low_k & \text{se } k = 0 \\ r_{k-1} * |high_k - low_k| + low_k & \text{se } k \geq 1 \end{cases} \quad (7)$$

- Exemplo com a declaração: **int cubo[3..5][-2..3][-4..10]**

Esquemas de Tradução (Declaração)

Cálculo da constante C_A

decl \rightarrow T **id** [L]

T \rightarrow **float**

L \rightarrow L , D

L \rightarrow D

D \rightarrow N..N

N \rightarrow **num**

Esquema de Tradução (Acesso)

- Uma gramática de acesso em arranjos multidimensionais

acesso \rightarrow **id** [L]

L \rightarrow L , D

L \rightarrow D

D \rightarrow **num**

- Precisamos um esquema de tradução para calcular o endereço (repetindo o cálculo recursivo que permite obtê-lo)

$$endereco = C_A + d_k * w \quad (8)$$

$$d_k = \begin{cases} i_k & \text{se } k = 0 \\ d_{k-1} * |high_k - low_k| + i_k & \text{se } k \geq 1 \end{cases} \quad (9)$$

- Exemplo com a declaração: **cubo**[4][-1][0]

Esquema de Tradução (Acesso com expressões)

- Uma gramática de acesso em arranjos multidimensionais

| | | |
|--------|---|------------------------|
| acesso | → | id [L] |
| L | → | L , E |
| L | → | E |
| E | → | E + N |
| E | → | N |
| N | → | num id |

- Precisamos um esquema de tradução para calcular o endereço
(repetindo o cálculo recursivo que permite obtê-lo)

$$endereco = C_A + d_k * w \quad (10)$$

$$d_k = \begin{cases} i_k & \text{se } k = 0 \\ d_{k-1} * |high_k - low_k| + i_k & \text{se } k \geq 1 \end{cases} \quad (11)$$

- Exemplo com a declaração: **cubo[a+b][c+d][e+4]**

Esquema de Tradução (Acesso com expressões)

Geração de TAC para acesso a arranjos multidimensionais

acesso \rightarrow **id** [L]

L \rightarrow L , E

L \rightarrow E

E \rightarrow E + N

E \rightarrow N

N \rightarrow **num** | **id** | acesso

Expressões Booleanas

Expressões Booleanas

- ▶ Servem para calcular valores lógicos
- ▶ Usadas frequentemente em expressões condicionais
 - ▶ while
 - ▶ if
 - ▶ for
- ▶ Compostas por
 - ▶ Operadores Booleanos (and, or, not)
 - ▶ Variáveis e constantes booleanas
 - ▶ Operadores Relacionais (entre expressões aritméticas)
- ▶ Exemplo
`(2*x >= y) or (x == false)`

Expressões Booleanas

- ▶ Um exemplo de gramática para expressões booleanas
→ Regras de precedência e associatividade

B → B or B

B → B and B

B → not B

B → (B)

B → E relop E

B → true

B → false

- ▶ Dois métodos de avaliação
 - ▶ **Numérico** → tratamento como expressões aritméticas
 - ▶ Falso é 0, Verdadeiro é diferente de 0
 - ▶ **Fluxo de controle** → tratamento por desvios
 - ▶ Também conhecido por avaliação de curto-circuito

Expressões Booleanas – Numérica

- Expressão booleana em uma linguagem fictícia

`x = a or b and not c`

- Tradução em TAC

099: ...

100: `t1 = not c`

101: `t2 = b and t1`

102: `t3 = a or t2`

103: `x = t3`

104: ...

Expressões Booleanas – Numérica

- Expressão booleana em uma linguagem fictícia

$x = a < b$

- Tradução em TAC

```
099:  ...
100:  if (a < b) goto 103
101:  t1 = 0
102:  goto 104
103:  t1 = 1
104:  x = t1
105:  ...
```

Expressões Booleanas – Numérica Esquema

| | |
|--------------------------------------|--|
| $E \rightarrow E_1 \text{ or } E_2$ | <pre>{ E.nome = temp(); gera(E.nome = E₁.nome or E₂.nome)</pre> |
| $E \rightarrow E_1 \text{ and } E_2$ | <pre>{ E.nome = temp(); gera(E.nome = E₁.nome and E₂.nome)</pre> |
| $E \rightarrow \text{not } E_1$ | <pre>{ E.nome = temp(); gera(E.nome = not E₁.nome)</pre> |
| $E \rightarrow (E_1)$ | <pre>{ E.nome = E₁.nome }</pre> |
| $E \rightarrow E_1 \text{ op } E_2$ | <pre>{ E.nome = temp(); gera(if E₁.nome op.simb E₂.nome goto proxq+3); gera(E.nome = 0); gera(goto proxq+2); gera(E.nome = 1); }</pre> |
| $E \rightarrow \text{true}$ | <pre>{ E.nome = temp(); gera(E.nome = 1); }</pre> |
| $E \rightarrow \text{false}$ | <pre>{ E.nome = temp(); gera(E.nome = 0); }</pre> |

Expressões Booleanas – Numérica Exemplo

- Considerando a expressão

`x = a < b or c < d and e < f`

- Tradução em TAC pelo esquema

| | | | |
|-----|-------------------|-----|-------------------|
| 100 | if a < b goto 103 | 107 | t2 = 1 |
| 101 | t1 = 0 | 108 | if e < f goto 111 |
| 102 | goto 104 | 109 | t3 = 0 |
| 103 | t1 = 1 | 110 | goto 112 |
| 104 | if c < d goto 107 | 111 | t3 = 1 |
| 105 | t2 = 0 | 112 | t4 = t2 and t3 |
| 106 | goto 108 | 113 | t5 = t1 or t4 |

Expressões Booleanas – Fluxo de Controle

- ▶ Avaliação por fluxo de controle
 - ▶ Tradução em TAC sob a forma de desvios
 - ▶ **Curto-circuito**

- ▶ Exemplo

```
if (x < 100 || x > 200 && x != y) x = 0;
```

- ▶ Tradução em TAC (com curto-circuito)

```
    if x < 100 goto L2
    ifFalse x > 200 goto L1
    ifFalse x != y goto L1
L2:  x = 0
L1:
```

Expressões Booleanas – Fluxo de Controle

- ▶ Funções auxiliares
 - ▶ `gera()`
 - ▶ `rot()` – cria um novo rótulo simbólico
- ▶ Atributos herdados para cada expressão booleana B
 - ▶ `B.t` – contém o rótulo alvo caso a expressão for verdade
 - ▶ `B.f` – contém o rótulo alvo caso a expressão for falsa

Expressões Booleanas – Fluxo de Controle

$B \rightarrow \{ B_1.t=B.t; B_1.f=rot(); \} B_1 \text{ or } \{ B_2.t=B.t; B_2.f=B.f; \} B_2$
 $\{ B.code=B_1.code \parallel label(B_1.f) \parallel B_2.code \}$

$B \rightarrow \{ B_1.t=rot(); B_1.f=B.f; \} B_1 \text{ and } \{ B_2.t=B.t; B_2.f=B.f; \} B_2$
 $\{ B.code=B_1.code \parallel label(B_1.t) \parallel B_2.code \}$

$B \rightarrow \text{not } \{ B_1.t=B.f; B_1.f=B.t; \} B_1 \{ B.code=B_1.code; \}$

$B \rightarrow (B_1) \{ B.code=B_1.code; B.t=B_1.t; B.f=B_1.f; \}$

$B \rightarrow \text{true } \{ B.code=gera(\text{goto } B.t); \}$

$B \rightarrow \text{false } \{ B.code=gera(\text{goto } B.f); \}$

$B \rightarrow E_1 \text{ relop } E_2 \{ B.code=E_1.code \parallel E_2.code \parallel$
 $\text{gera}(\text{if } E_1.local \text{ relop.lexval } E_2.local \text{ goto } B.t) \parallel$
 $\text{gera}(\text{goto } B.f); \}$

Expressões Booleanas – Fluxo de Controle

- ▶ Exemplo

```
if (x < 100 || x > 200 && x != y) x = 0;
```

- ▶ Tradução em TAC

```
...  
if x < 100 goto VERDADE  
goto X1  
X1:    if x > 200 goto X2  
      goto FALSO  
X2:    if x != y goto VERDADE  
      goto FALSO  
VERDADE: x = 0  
FALSO:  ...
```

Controle de Fluxo

Controle de Fluxo

- ▶ Controlar o fluxo de execução
 - ▶ Gerar código de controle
 - ▶ Utiliza rótulos e desvios
- ▶ Estudaremos três situações (if, if else, while)
- ▶ Gramática (B é uma expressão booleana)

$$S \rightarrow \text{if (B) } S_1$$
$$S \rightarrow \text{if (B) } S_1 \text{ else } S_2$$
$$S \rightarrow \text{while (B) } S_1$$

Controle de Fluxo – if

```
S  →  if { B.t=rot(); B.f=S.next; }  
      (B) { S1.next=S.next; }  
      S1 { S.code=B.code || gera(B.t:) || S1.code }
```

Fluxo de Execução – if else

```
S  →  if  { B.t=rot(); B.f=rot(); }  
      (B) { S1.next=S.next; }  
      S1 else { S2.next=S.next; }  
      S2 { S.code=B.code || gera(B.t:) || S1.code || gera(goto S.next) ||  
          gera(B.f:); || S2.code }
```

Controle de Fluxo – while

```
S  →  while  { B.f=S.next; B.t=rot(); }  
      (B)  { S.begin=rot(); S1.next=S.begin; }  
      S1  { S.code=gera(S.begin:) || B.code ||  
           gera(B.t:) || S1.code || gera(goto S.begin) }
```

Gramática para Exercício

```
S → attr { S.code=gera(attr.lexval) || gera(goto S.next) }
S → if { B.t=rot(); B.f=rot(); }
    (B) { S1.next=S.next; }
    S1 else { S2.next=S.next; }
    S2 { S.code=B.code || gera(B.t:) || S1.code ||
        gera(B.f:); || S2.code }
S → while { B.f=S.next; B.t=rot(); }
    (B) { S.begin=rot(); S1.next=S.begin; }
    S1 { S.code=gera(S.begin:) || B.code ||
        gera(B.t:) || S1.code || gera(goto S.begin) }
B → { B1.t=rot(); B1.f=B.f; } B1 and { B2.t=B.t; B2.f=B.f; } B2
    { B.code=B1.code || label(B1.t) || B2.code }
B → E1 relop E2 { B.code=E1.code || E2.code ||
    gera(if E1.local relop.lexval E2.local goto B.t) ||
    gera(goto B.f); }
```

Controle de Fluxo – Exercício

- Gere o TAC para o código seguinte

```
while (a < b && e != f) {  
    if (c < d){  
        x = y + z;  
    }else{  
        x = x - z;  
    }  
}
```

Atributos herdados com Bison

- ▶ É possível!

```
%union { int val; }
```

```
%type<val> X Y
```

```
%%
```

```
S:  { $<val>$=20; } X '+' { $<val>$=10; } Y;
```

```
X: 'x' { printf ("%d\n", $<val>0);};
```

```
Y: 'y' { printf ("%d\n", $<val>0);};
```

```
%%
```

- ▶ Veja um exemplo útil em

<http://user.it.uu.se/~matkin/programming/yacc+lex/>

<http://epaperpress.com/lexandyacc/attr.html>

Conclusão

- ▶ Leituras Recomendadas
 - ▶ Livro do Dragão
 - ▶ Seções 6.4.3 e 6.4.4,
 - ▶ Série Didática
 - ▶ Seção 5.3.2

- ▶ Próxima Aula
Geração de Código