編譯器期末報告

Machine Independent Optimization Phase

B062040027 鄭乃心

與機器無關的最佳化：

將由 Parser 所輸出的 Matrix 或 Syntax Tree 進行最佳化，所輸出最佳的 Matrix(Reduces Syntax Tree)，以減少儲存空間及執行時間。

最佳化處理技巧：

一、 消除共同的表達式

X := a + (m * n);　　Y := b + (m * n);

⇨ Temp := (m * n);　X := a + Temp;　Y := b + Temp;

⇨ 在最佳化前，編譯器必須計算兩次 m*n，最佳化後只需要計算次。

二、 減少編譯器的計算時間

A = (3 * 5 + 1 * 9) + B;

⇨ A = 24+ B;

⇨ 若是很簡單能自己算出來的，就不需要喪編譯器計算。

三、 最佳化布林表達式

If C1 or C2 Then S1

⇨ If C1 Then S1　　Else If C2 then S1

⇨ 在最佳化前，每次都要判斷 C1 or C2，最佳化後判斷更快速。

四、 最佳化迴圈

```
Bound := 10;                          Bound := 10;
While( I <= Bound - 2) do             t := Bound  –  2;
    While( I <= 10 )do                While( I <= t) do
    Begin                    =>       Begin
        X := 1;                           X := 1;
        Y := X + Z;                       While( I <= 10 )do
    End;                                      Begin
                                                  Y := X + Z;
                                              End;
                                      End;
```

⇨ 最佳化後，能夠減少 X := 1;這條程式碼的執行次數。

五、 邏輯順序變換

```
for I in range(1,5):                  if a == b:
  if a == b:                   =>         for I in range(1,5):
      print I;                              print I;
```

⇨ 最佳化前一定得執行迴圈，須經多次計算，判斷是否印出 I。最佳化後，先判斷判斷式是否成立，成立再進入回圈，減少計算次數。

六、 迴圈合併

```
for i in range(1, 10):
    phones.a(data[i].phone)
for j in range(1, 10):
    address.a(data[j].address)
```

⇨

```
for i in range(1, 10):
    phones.a(data[i].phone)
    address.a(data[j].address)
```

⇨ 最佳化前需要經過兩個迴圈運算(共 18 次)。

⇨ 最佳化後只需要一個（共 9 次)。

七、 刪除不必要的指令

```
c = a * b;                    c = a * b;
x = a;                =>      d = a* b + 4;
d = x * b + 4;
```

⇨ 若 x 後續用不到的話，就將它刪除

八、 刪除不必要的暫存變數

```
L1 : t1 := i * 4                          L1 : t1 := i * 4
     t2 := a[t1]                               t2 := a[t1]
     t3 := i * 4                               t4 := b[t1]
     t4 := b[t3]                               t5 := t2 * t4
     t5 := t2 * t4            =>               prod :=  prod + t5
     t6 := prod + 5                            i := i + 1
     prod := t6                                if i <= 20 goto L1
     t7 := i + 1
     i := t7
     if i <= 20 goto L1
```

九、 Variable Propagation

```
c = a * b;                          c = a * b;
x = a;                    =>        x = a;
d = x * b + 4;                      d = a* b + 4;
```

⇨ 把 x 換成 a 之後，不影響結果，並且可以像範例一一樣做消除共同表達式的動作。

十、 Induction Variable and Strength Reduction

```
I = 1;                          I = 1;
While( I < 10){                 t = 4;
    Y = I * 4;                  While(t < 40){
}                                       y = t;
                                        t = t + 4;
                                }
```

⇨ 利用低強度的運算符(+)代替高強度的運算符(*)。