P. Thiemann, S. Wehr — Wintersemester 2006/2007

Compilerbau

`http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2006ws/`

Übungsblatt 12

7.2.2006

Aufgabe 1 (Liveness Analyse)
Betrachten sie folgendes Pseudo–Assembler Programm:

```
    x := 2
    y := 4
    x := 1
    if y > x GOTO label
    z := y * y
    GOTO exit
label:
    z := y
exit:
    x := z
```

Führen sie eine Liveness Analyse des Programms durch. Gehen sie dazu wie folgt vor:

1. Zeichnen sie den Kontrollflussgraphen (CFG) des Programms.
2. Berechnenen sie für jeden Knoten des CFGs die Mengen *def, use, live-in* sowie *live-out.*
3. Erstellen sie den Register–Interferenz–Graphen.

Nehmen sie dabei an, dass lediglich die Variable `x` am Ende des Programms lebendig ist.

Aufgabe 2 (Extended Basic Blocks)
Um Programmanalysen effizienter zu machen, verwendet man meistens nicht einzelne Instruktionen als Knoten des CFGs, sondern fasst mehrere Instruktionen zu sogenannten *Extended Basic Blocks* zusammen. Dabei ist ein Extended Basic Block eine Sequenz von Knoten $N_1, \ldots, N_k$ des ursprünglichen CFGs, so dass für $1 \leq i < k$ gilt: N_i ist der einzige Vorgänger von N_{i+1}. Außerdem muss diese Sequenz möglichst lang sein.

(i) Um eine Liveness Analyse auf Extended Basic Blocks zu definieren, muss man die Definition der Mengen *def* (enthält die in einem Block definierten Variablen) und *use* (enthält die in einem Block benutzten Variablen) entsprechend anpassen. Führen sie diese Anpassung durch. Definieren sie dazu für einen Knoten NM, welcher durch Verschmelzen der Knoten N und M entsteht, die Mengen $\mathit{def}(NM)$ und $\mathit{use}(NM)$ unter Benutzung von $\mathit{def}(N)$, $\mathit{def}(M)$, $\mathit{use}(N)$ und $\mathit{use}(M)$.

(ii) Zeichnen sie für folgendes Pseudoassembler–Programm den aus Extended Basic Blocks bestehenden CFG und berechnen sie für jeden Knoten die Mengen *def* und *use.*

```
    m := 1
    v := 0
label_0:
    if v >= n GOTO exit
    r := v
    s := 0
label_1:
    if r < n then GOTO label_2
    v := v + 1
    GOTO label_0
label_2:
    x := MEM[r]
    s := s + x
    if s <= m GOTO label_3
    m := s
label_3:
    r := r + 1
    GOTO label_1
exit:
    return m
```

Aufgabe 3 (Reaching Definitions Analyse)
Die Analyse *Reaching Definitions* soll für jeden Punkt im Programm bestimmen, welche Zuweisungen den Punkt möglicherweise erreichen, ohne überschrieben zu werden. Dabei wird eine Zuweisung durch ein Paar (x, n) beschrieben, wobei x die zugewiesene Variable und n die Zeilennummer der Zuweisung ist. Außerdem gibt es eine spezielle Zeilennumer ?, welche verwendet wird, um die initiale Zuweisungsstelle aller Variablen des Programms zu markieren. Sei als Beispiel folgendes Programm gegeben.

```
1  x := 1
2  if (some_complex_condition)  {
3    x := 5
4  } else {
5    x := y
6    z := x + y
7  }
8  print(x + z)
```

Dann erreichen die Zuweisungen $(\mathtt{x}, 3)$, $(\mathtt{x}, 5)$, $(\mathtt{z}, 6)$ sowie $(\mathtt{y}, ?)$ und $(\mathtt{z}, ?)$ den Programmpunkt in Zeile 8. Beachten sie, dass dies für die Zuweisung $(\mathtt{x}, 1)$ *nicht* gilt, da sie in beiden Zweigen des `if`–Statements überschrieben wird.

Für die Reaching Definitions Analyse definiert man zwei Funktionen *kill* und *gen*. Die Funktion *kill* liefert für einen Knoten im CFG die Menge der Zuweisungen, die durch den Knoten überschrieben werden; *gen* liefert für einen Knoten im CFG die Menge der Zuweisungen, die durch den Knoten erzeugt werden.

Betrachten sie nun folgendes Programm:

```
x := 5
y := 1
while (x > 1) {
  y := x * x
  x := x - 1
}
print(y)
```

(i) Zeichnen sie den CFG für das Programm und überlegen sie sich für jeden Knoten N, wie die Mengen $\mathit{kill}(N)$ und $\mathit{gen}(N)$ aussehen müssen.

(ii) Nehmen sie an, dass ein Knoten im CFG genau ein Statement *stmt* enthält, wobei *stmt* wie folgt definiert ist:

$$\mathit{stmt} ::= x := e \mid \texttt{while}(e) \mid \texttt{print}(e)$$

Dabei bezeichnet x eine Variable und e einen arithmetischen Ausdruck. Geben sie nun eine Definition für die Funktionen *kill* und *gen* an.

(iii) Analog zur Liveness Analyse wird für jeden Knoten N des CFGs die Mengen $\mathit{in}_{\mathrm{rd}}(N)$ und $\mathit{out}_{\mathrm{rd}}(N)$ berechnet. Die Menge $\mathit{in}_{\mathrm{rd}}(N)$ soll dabei die Menge der Zuweisungen enthalten, welche den Eingang des Knotens N erreichen; die Menge $\mathit{out}_{\mathrm{rd}}(N)$ soll die Menge der Zuweisung enthalten, die den Ausgang des Knotens N erreichen. Anders als bei der Liveness Analyse wird diese Information allerdings vorwärts (und nicht rückwärts) weiterpropagiert. Stellen sie die Gleichungen auf, welche $\mathit{in}_{\mathrm{rd}}(N)$ und $\mathit{out}_{\mathrm{rd}}(N)$ definieren.

(iv) Bestimmen sie für jeden Knoten N des in (i) erstellten CFGs die Mengen $\mathit{in}_{\mathrm{rd}}(N)$ und $\mathit{out}_{\mathrm{rd}}(N)$.

Abgabe: 14.2.2007
Die Abgabe erfolgt bis zu Beginn der Übungsstunde. Für jede Aufgabe werden vier Punkte vergeben; für Plagiate werden keine Punkte vergeben.