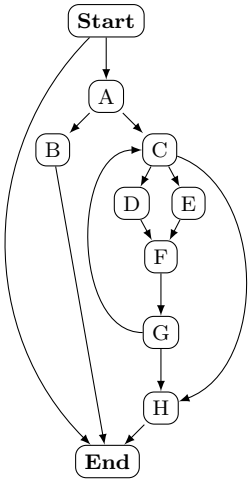
Karlsruher Institut für Technologie
Lehrstuhl für Programmierparadigmen
Sprachtechnologie und Compiler WS 2010/2011
Dozent: Prof. Dr.-Ing. G. Snelting
Übungsleiter: Matthias Braun `matthias.braun@kit.edu`

Übungsblatt 12 | Ausgabe: 9.7.2012 | Besprechung: 16.7.2012

Aufgabe 1: *Dominanz*

Gegeben sei folgender Kontrollflussgraph:



Berechnen Sie den Dominatorbaum mittels Fixpunktiteration.

Aufgabe 2: *Datenflussanalyse*

2.1 *Konstantenfaltung*

Gegeben sei folgendes Programm:

```
    int main(void) {
1       int x = 1;
2       int i = 0;

3       while (i < 10) {
4           x = 2 - x;
5           i++;
        }

6       return x;
    }
```

Führen Sie eine Datenflussanalyse zur Konstantenfaltung durch:

1. Wie sieht der zugehörige Verband aus?
2. Stellen Sie die Datenflussgleichungen auf.
3. Berechnen Sie den Fixpunkt.
4. Falls die Analyse zu einem beliebigen Zeitpunkt abgebrochen wird, können die Zwischenergebnisse verwendet werden?

2.2 *Lebendige Variablen (Zusatzaufgabe)*

Gegeben sei folgendes Programm:

```
0    void func(int y) {
1        int a = 1
2        int z = a+y
3        while (y < z) {
4            t = z-1
5            if (z > 4) {
6                z = t;
             }
7            a = z+1
         }
8        a = a;
9        return;
     }
```

Führen Sie eine Datenflussanalyse durch um Lebendige Variablen zu bestimmen.

1. Wie sieht der zugehörige Verband aus?
2. Stellen Sie die Datenflussgleichungen auf.

   Idee: durchlaufe Programm rückwärts, ab einer Benutzung ist eine Variable lebendig, ab einer Definition tot. Benutze gen-kill-Schema. gen enthält Variablen die Benutzt werden, kill Variablen die Definitert werden.
3. Berechnen Sie den Fixpunkt.

Aufgabe 3: *SSA Darstellung*

Gegeben sind die folgenden zwei Programme in einer C-artigen Pseudosprache.

Programm `A`:

```
t = read_int();
a = t;
t = read_int();
b = t;
// let a be the bigger value
if (a < b) {
    t = a;
    a = b;
    b = t;
}
printf("Distance: %d\n", a-b);
```

Programm `B`:

```
list_get_last(first) {
    element = first;
    do {
        prev = element;
        element = element.next;
    while (element != null);
    return prev;
}
```

3.1 *SSA Aufbau*

Bringen Sie die Programme `A` und `B` in SSA-Form!

3.2 *Optimierungen*

Wenden Sie Kopienfortschaltung auf die beiden Programmen in SSA-Form an!

3.3 *SSA Abbau*

Bringen Sie die Programme nach den Transformationen der letzten Aufgabe in eine nicht-SSA Form – d.h. eine Darstellung ohne ϕ-Funktionen!