

# Aufgabe 1: Dominator-Analyse

Optimierende Übersetzer WS 2014/2015

Abgabetermin: 5. November 2014, 16:00 Uhr

## 1 Intraprozedurale Dominator-Analyse

Dominator-Analyse ermittelt die Dominanzrelation zwischen Knoten in einem gerichteten Graphen. Diese Relation ist in einem gerichteten Graphen mit einem eindeutigen vorgängerlosen Startknoten  $s_0$  wie folgt definiert: Ein Knoten  $d$  dominiert einen Knoten  $n$  genau dann, wenn jeder Pfad von  $s_0$  nach  $n$  durch  $d$  verläuft. Trivialerweise dominiert sich jeder Knoten selbst.

In dieser Aufgabe soll eine Dominator-Analyse auf dem Kontrollflussgraphen (CFG) von Programmen definiert werden. Die Analyse ermittelt also, welche Anweisungen bestimmt vor welchen anderen ausgeführt werden müssen.

Im Folgenden ein Beispiel für intraprozedurale Dominator-Analyse für die Sprache WHILE:

<code>[a := 1]<sup>1</sup>;</code>	$\ell$	$DA_{\circ}(\ell)$	$DA_{\bullet}(\ell)$
<code>[b := a]<sup>2</sup>;</code>	1	{}	{1}
<code>while [a &lt; 10]<sup>3</sup> do (</code>	2	{1}	{1, 2}
<code>if [a &lt; b]<sup>4</sup> then</code>	3	{1, 2}	{1, 2, 3}
<code>[a := a + 1]<sup>5</sup>;</code>	4	{1, 2, 3}	{1, 2, 3, 4}
<code>else</code>	5	{1, 2, 3, 4}	{1, 2, 3, 4, 5}
<code>[b := b + 1]<sup>6</sup>;</code>	6	{1, 2, 3, 4}	{1, 2, 3, 4, 6}
<code>[c := a + b]<sup>7</sup>;</code>	7	{1, 2, 3, 4}	{1, 2, 3, 4, 7}
<code>)</code>			

Spezifizieren Sie die Dominator-Analyse für WHILE:

- Definieren Sie  $kill_{DA}(\ell)$  und  $gen_{DA}(\ell)$ .
- Definieren Sie die Gleichungen für  $DA_{\circ}(\ell), DA_{\bullet}(\ell): Lab_{\star} \rightarrow \mathcal{P}(Lab_{\star})$ .

## 2 Intraprozedurale Dominator-Analyse mit PAG

Spezifizieren Sie mit PAG eine Dominator-Analyse für die Sprache  $SL_1$ . Diese Sprache ist jene Teilmenge von C++, die der Sprache WHILE ohne Funktionsaufrufe entspricht. Gültige Anweisungen mit Entsprechung in WHILE

sind somit Zuweisungen, `if`-Verzweigungen und `while`-Schleifen. Es sind lediglich einfache Ausdrücke aus den üblichen Rechen- und Vergleichsoperatoren erlaubt.

Als Unterschied gegenüber `WHILE` muß jede Variable vor ihrer Verwendung deklariert werden, dabei sind die Typen `int` und `bool` zulässig. Es existiert eine einzige Funktion mit der Signatur `int main()`, deren letzte Anweisung eine `return`-Anweisung mit einem Wert vom Typ `int` ist.

Das folgende Programm in `SL1` (und somit `C++`) entspricht dem obigen `WHILE`-Beispiel:

```
int main() {
    int a, b, c;
    a = 1;
    b = a;
    while (a < 10) {
        if (a < b)
            a = a + 1;
        else
            b = b + 1;
        c = a + b;
    }
    return 0;
}
```

Ihre PAG-Analyse soll den Namen `da` tragen. Zur Ermittlung einer eindeutigen Kennung für Anweisungen stellt `SATIrE` das Attribut `label` bereit. Sie können dieses innerhalb von Transferfunktionen einfach wie eine vordefinierte Variable vom Typ `snum` verwenden.

Ihre Analyse soll die gesamte Sprache `SL1` abdecken. Das Verhalten der Analyse auf Programmen, die nicht in `SL1` liegen, ist Ihnen freigestellt. Sie müssen also keine Fehlerbehandlung für sonstige Sprachkonstrukte implementieren.

**Hinweis:** Das Dokument in `/usr/local/optub/doc/live_variables.pdf` auf der Übungsmaschine `g0.complang.tuwien.ac.at` bietet eine Einführung in die Verwendung von PAG und `SATIrE`.

### 3 Abgabe

Senden Sie ihre Lösungen bis **5. November 2014, 16:00 Uhr** per E-Mail an `gergo@complang.tuwien.ac.at`. Die Betreffzeile der E-Mail soll ‘`00: Aufgabe 1, Nachname`’ lauten. Hängen Sie die Antworten auf die Textfragen 1(a) und 1(b) als PostScript- oder PDF-Datei, die Analyse-spezifikation für Teilaufgabe 2 als `.opt1a`-Datei an die E-Mail an.