Technische Universität München
Fakultät für Informatik
Prof. Dr. H. Seidl

Winter Semester 10/11
Vesal Vojdani
Aleksandr Karbyshev

Program Optimization

Exercise Sheet 1

Deadline: November 2, in 02.07.053, before 15:00.

Exercises marked with (P) are discussed and solved together at the lab sessions. Exercises marked with (H) are to be solved independently as homework.

Exercise 1: *(P) VoTUM: Available expressions* — *No Points*

Go to http://www2.in.tum.de/projects/votum, install VoTUM 0.7.5, and make yourself familiar with the system.

Now consider the following sequence of C statements:

```
int a,b,c;
c = (a + b);
c = (a - b);
while (a + b < 5)
    b = a + b;
```

Use VoTUM to compute available expressions and perform the optimization to eliminate redundant computations.

Exercise 2: *(P) Semantics of Pre- and Post-increments* — *No Points*

Assume we have a language $\mathcal{L}_1$ given by the following grammar:

e	$::=$	x	Variable
	$\mid$	c	Constant
	$\mid$	$x{+\!\!+}$	Post-increment of a variable
	$\mid$	${+\!\!+}x$	Pre-increment of a variable
	$\mid$	$e_1 + e_2$	Addition of two expressions
	$\mid$	$e_1 - e_2$	Subtraction of two expressions

The pre- and post-increments should be familiar from Java (where $x{+\!\!+}$ is written `x++`). In the lecture, expressions had no side-effects, but these increments update the value of a given variable. Therefore, an expression now will both return a value and update the state $\rho \in \mathit{Env} = \mathit{Var} \to \mathbb{Z}$, so the semantics of an expression has the following type:

$$[\![e]\!]_1 \colon \mathit{Env} \to \mathbb{Z} \times \mathit{Env}\,.$$

Define for each kind of expression $e \in \mathcal{L}_1$, the semantic function $[\![e]\!]_1$. For this, and in the lecture, we use the update operation $\oplus$, which is defined as follows:

$$(\rho \oplus \{x \mapsto a\})(y) = \begin{cases} a & \text{if } y = x \\ \rho(y) & \text{otherwise.} \end{cases}$$

Exercise 3: *(H) Translating Pre- and Post-increments* *10 Points*

Let $\mathcal{L}_2$ be the language defined by the following grammar:

$$l ::= e \mid x := e \mid l_1;\ l_2$$
$$e ::= x \mid c \mid e_1 + e_2 \mid e_1 - e_2$$

Words generated by the first rule are essentially sequences of expressions and variable assignments. Right-hand sides of assignments are pure, i.e., they do not modify a state.

a) Give the semantics for the language $\mathcal{L}_2$. For that, define by structural induction a function of the type

$$[\![e]\!]_2 \colon \mathit{Env} \to \mathbb{Z} \times \mathit{Env}\,.$$

The value of a sequence l is the value of the last expression in the sequence. For example,

$$[\![x := 5;\ x + 3]\!]_2\,\rho = \langle 8, \rho \oplus \{x \mapsto 5\}\rangle, \quad [\![1;\ y := 3]\!]_2\,\rho = \langle 3, \rho \oplus \{y \mapsto 3\}\rangle\,.$$

b) Define a function $\mathcal{T} \colon \mathcal{L}_1 \to \mathcal{L}_2$ which translates an expression of the language $\mathcal{L}_1$ to an expression of the language $\mathcal{L}_2$ that produces the same value and preserves the effect on variables occurring in the original expression (the translation may introduce temporary variables), i.e.,

$$\forall e \in \mathcal{L}_1, \forall \rho, \rho'_1, \rho'_2 \in \mathit{Env},\ \forall v_1, v_2 \in \mathbb{Z}\colon$$
$$\langle v_1, \rho'_1\rangle = [\![e]\!]_1\rho \text{ and } \langle v_2, \rho'_2\rangle = [\![\mathcal{T}(e)]\!]_2\rho \text{ implies that}$$
$$v_1 = v_2 \text{ and } \forall x \in \mathsf{Vars}(e),\ \rho'_1\,x = \rho'_2\,x\,.$$

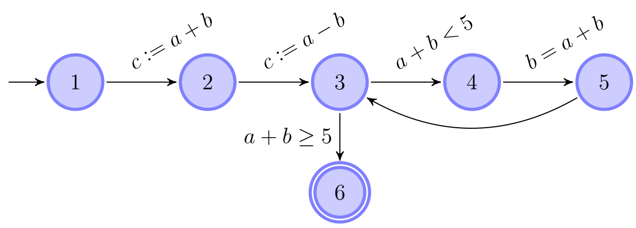
For this, you should define the auxiliary function $\mathcal{T}' \colon \mathcal{L}_1 \times \mathbb{Z} \to \mathcal{L}_2$. The integer parameter i indicates the next free temporary variable t_i, in which the result of the computation will be stored. For example,

$$\mathcal{T}'(x\text{++} + (y - \text{++}x), 5) =$$
$$t_5 := x;\ x := x + 1;\ t_6 := y;\ x := x + 1;\ t_7 := x;\ t_6 := t_6 - t_7;\ t_5 := t_5 + t_6$$

We can then define $\mathcal{T}(e) = \mathcal{T}'(e, 1); t_1$. You do not *have to* give a proof of correctness, unless you really want to.

Exercise 4: *(H) Computing Available Expressions* *5 Points*

The following is a control flow graph of the program from exercise 1.



Write down the constraint system and compute for each node u the set $A[u]$ of available expressions for this program.