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

Winter Semester 10/11
Vesal Vojdani
Aleksandr Karbyshev

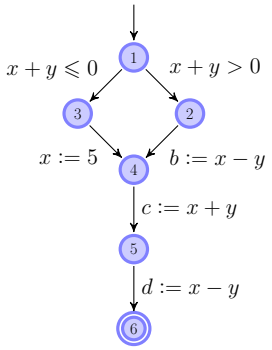
Program Optimization

Exercise Sheet 8

Deadline: January 24, at the lecture!

Exercise 1: *(H) Partial redundancy elimination* *12 Points*

Perform the partial redundancy elimination optimization on the following program.



You should perform all the steps in the algorithm systematically:

1. Compute the set of Very Busy Expressions at each program point.
2. Compute the set of Available Expressions at each program point.
3. Perform the hoisting transformation (T5.1), which stores the computation of a *safe* expression in a temporary variable. Explain which edges are inserted and why!
4. Replace expressions with their hoisted values (T5.2), and draw the CFG of the optimized program.

Exercise 2: *(H) Interprocedural analysis* *12 Points*

Consider the following program.

```
int x = 0;

int main() {
    int y = 5;
    x = 3; f();
    x = 7; f();
    return x;
}
```

```
void f() {
    int y = 7;
    if (*)
        y = x;
    x = y;
}
```

a) Using Copy-Constants analysis (pp. 558–561 of slides), compute the summary $M \in \mathbb{M}$ for the procedure `f`().

b) Recall definitions of the functions $enter^\sharp$ and $combine^\sharp$ for Constant Propagation analysis. For a call edge $k = (u, f(), v)$, the transfer function is $[\![k]\!]^\sharp\ D = combine^\sharp(D, [\![f]\!]^\sharp(enter^\sharp\ D))$. Compute the transfer function for the calls to the procedure `f`() in `main`() using your summary.

c) Having computed the transfer function for calls, you can write out a constraint system for Constant Propagation Analysis of the above program and solve it. Do it!

d) Now, instead of pre-computed summary M, use the tabulation approach of Sharir/Pnueli (p. 569). What are the values of a for the constraint variables $[\![\texttt{f}, a]\!]$ that you need to compute?

e) Using the call-string approach, analyse the program with $d = 0$ (here, it means empty call stack) and $d = 1$ (not more than one call), and give the result at the end of `main`() for each case. What about $d = 2$?