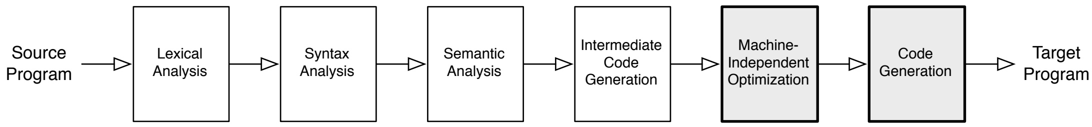
HW 11: More Dataflow Fun and Tiling

CSCI 434T
Spring, 2019

Overview



Time to wrap up dataflow analsis and at least briefly examine other code generation schemes beyond translating to TAC and then x86.

Readings

- From last week: Dragon 9.3 – 9.5.2
- Dragon 8.9 – 8.9.2

Exercises

1. Start thinking about PA 4:
   - How will you represent the dataflow facts for each analysis in your compiler?
   - How will you use them to perform each optimization on a TAC list?
2. We want to design a dataflow analysis to compute ranges for integer variables in the program. For this, we extend the set N of integer numbers with plus and minus infinity:

   $$\mathbb{N}^* = \mathbb{N} \cup \{+\infty, -\infty\}$$

   such that $-\infty < n$ and $n < +\infty$ for any integer number n. We then use a lattice over the set

   $$L = \{[l, u] \mid l, u \in \mathbb{N}^* \text{ and } l \leq u\} \cup \{\top\}$$

   (a) Explain what the element $\top$ represents and why we need it.
   (b) Define the partial order and the meet operator $\wedge$ for elements in this lattice (including $\top$).
   (c) Sketch the structure of the resulting lattice.
   (d) Using this lattice to compute ranges of variables will fail. Explain why.
   (e) To solve the problems from the last part, we define a lattice

   $$L' = \{[l, u] \mid l, u \in \{-\infty, -1, 0, 1, +\infty\} \text{ and } l \leq u\} \cup \{\top\}$$

   (with the same partial order as before) and build a dataflow analysis that computes ranges in L'. Show the transfer functions for assignments of constants `x = n` and arithmetic operations `x = y + z` and `x = y * z`, where `x`, `y`, and `z` are program variables and `n` is an integer constant.
   (f) Using the revised lattice and your transfer function, show how the dataflow analysis works for the following program:

```
x = 0;
while (...) {
  y = x;
  if (...) {
    x = x+1;
  } else {
    y = y-1;
  }
}
```

(g) Assuming programs consist only of the three kinds of statements shown above, does this analysis always yield the same result as the Meet-Over-Paths solution? If yes, show why. If not, show a counter-example program and indicate the MOP and dataflow solutions.

(h) This problem is an example of an analysis where it makes sense to propagate different information along different out edges from a node. Suppose we extend our lattice as follows:

$$L' = \{[l, u] \mid l, u \in \{-\infty, -k, \ldots, -2, -1, 0, 1, 2, \ldots, k, +\infty\} \text{ and } l \leq u\} \cup \{\top\}$$

for some integer k. Give a suitable transfer function for each of the out edges of a branch conditional on the comparison `x < n`, and show that with this analysis we can derive bound `[0, 10]` for variable i in the following example (assuming $k \geq 10$):

```
i = 0;
while (i < 10) {
    i = i + 1;
}
```

Why are such bounds useful?

3. Dragon 8.9.1.

   - For (a) and (c), assume all variables are stored at global memory locations. For (b), assume `x`, `y`, and `z` are at global memory locations, and `i`, `j`, and `k` are locals stored relative to the stack pointer, as in the example from the book.
   - You may wish to add an additional tile form to dereference an address stored in a register.

   Reflect on the relative merits of Tiling vs. TAC. What do you see as the strengths/weaknesses of each?