1 Intermediate Representation

2 Three Address Code

3 Global Optimizations

(Prepared by Aditya Senthilnathan)

Global optimisations go beyond the scope of a single basic block. It needs reasoning of the whole control flow graph which may be within the body of a method.

3.1 Global Constant Propagation

- This is similar to local constant propagation but now we are reasoning in a global level (across several basic blocks within the body of a method)
- In this optimisation, we basically ask the question "When can we replace the use of a variable **x** with a constant **k**

  Ans: If on **every path** to the use of x, if the **last assignment** to x is **x:=k**, then we can do this transformation/optimisation.

The algorithm for global constant propagation requires a class of algorithms which are called "Global Dataflow Analysis". This kind of reasoning (with paths, etc.) is required for many different types of optimisation problems in compilers. And so, it helps to create a common framework in which we can implement the same logic but with different kinds of properties so that there is reuse of the same idea.
Let's identify some traits that global optimisations tasks share:

- Optimisation depends on knowing property X at a certain program point P

  Eg. In global constant propagation, we were interested in knowing if x:=k is the last assignment on all possible paths, at the program point of the last basic block (where x is used) in our example graph.
- Proving X at P requires knowledge of entire program

  Eg. In our global constant propagation example, to be able to say x:=k is the last assignment to x on all possible paths that can reach program point P, we need to know the characteristics of the entire program. As opposed to local optimistaion where you only need info about the concerned basic block. Here, even though transformation is only made in a basic block, it relies on a property that requires knowledge of the entire program.
- It's OK to be conservative

  We can either say,

  X is definitely true

  or say,

We don't know if X is true

We want to get "X is definitely true" as much as possible because only then can we apply the transformation but if we get the conservative answer "We don't know if X is true", then we don't apply the transformation to preserve program correctness. Knowing if X is definitely not true is useless because in this case we still don't apply the transformation and so this use case is clubbed with "We don't know if X is true".

4 DFA Values

Recall that for global constant propagation, we need the following:

Optimisation: Replace a use of x by a constant k
Constraint: On every path to the use of x, the last assignment to x should be x:=k

We need to do this for every variable in the program. But first let's do this for a single variable x and then we can generalise it to more variables.

4.1 Dataflow Analysis Values

These are abstract values that are supposed to capture the set or category of concrete values that x (a variable) may take.

At each program point, we associate one of the following values with X (a variable),

Value	Explanation
$\top$ (top)	This value represents either that • This statement never executes • or we have not executed or seen it yet
$\bot$ (bottom)	This value represents either that • We can't say that X is a constant • X is definitely not a constant
C (constant)	This value represents that X is a constant C

Question: What are we trying to do here? What do these values mean?

Answer: We are trying to figure out what are the possible values of X (a variable) at every program point. The values we are talking about essentially tell us if the program reached here what is the value of X or all the possible values of X at this program point.
Now we will discuss each of the values in a little more detail,

- $\top$ (top) - This value represents that this statement never executes i.e the program/control never reaches here. This might be because this part of the program is never reachable from the beginning of the program. So in this case we just say that the value is $\top$ here because the statement never executes and therefore it is not meaningful to say what this value is.
- C (constant) - It could be any constant. Eg. 0,1,2,-2,etc. This value says that whenever this program point is reached, irrespective of the path taken, we are sure that X will be equal to constant value C at this point.
- $\bot$ (bottom) - This value essentially says that we don't know if X is a constant or not at this point. We are not sure of it's value. This might be happening because it's possible that on different paths X could take different values or on some path X is not a constant, etc. Even if we know for sure that X is not a constant, we still assign $\bot$ to X. This is a conservative assignment. Technically, even if we (DFA algorithm) assign $\bot$ to X at every program point conservatively, then this solution is also correct but it is trivially true and is not of much use to us.

Note: If we're still in the middle of execution of the DFA algorithm and at a particular point, we have value $\top$ for one of the variables in the program then it could also mean that the DFA algorithm hasn't seen this point yet or hasn't reached this point yet. The above discussion and interpretation of $\top$ only applies if a variable has $\top$ value after the DFA algorithm has finished executing

4.2 Using DFA values for transformation

Given global constant information ($\top$, $\bot$, C)

- Inspect x = ? (either $\top$ or $\bot$ or C) at the program point that just precedes the statement which you want to examine (the statement that uses x)
- If x = C, then replace the use of x with C

For some examples of how to assign DFA values see the lecture video

5 Dataflow Analysis Algorithm (DFA)