Foundations of Data-Flow Analysis and Constant Propagation

VYPe

Jan Chaloupka (xchalo08), David Chaloupka (xchalo09)

December 4, 2012

Content

- Data-Flow Analysis
- Constant Propagation

What is Data-Flow Analysis?

- Data flow analysis (DFA) is a special form of static analysis
- General steps:
  1. Transform program into a *Control flow graph (CFG)*
  2. Choose a property to inspect, eg. live variables, available expressions, constants.
     (property defines *flow functions* as in "flow of information")
  3. Repeatedly apply flow functions to the CFG until a solution is found (maximum fix-point)

Basic Blocks & Control flow graph

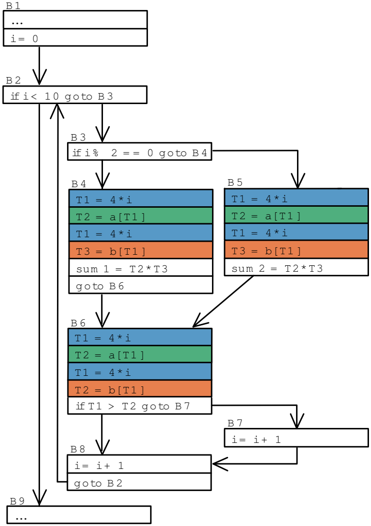
- Basic block (BB) is a *sequence* of statements that is executed *as a whole (atomically)*
  - Always entered via first statement (noone jumps inside the BB)
  - The whole BB is executed (contains no jump instruction)
  - Atomic execution of the BB ends by last statement (may be a jump, label etc.)
- Control flow graph (CFG) is a directed graph
  - Equivalent to the original program
  - nodes $\approx$ BBs
  - edges $\approx$ transfers of control (eg. jumps) between BBs

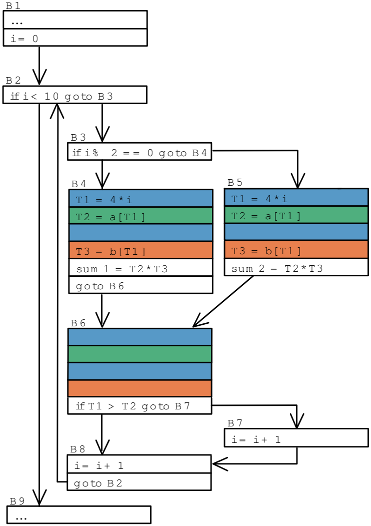
Sample program (spoiler: available expressions)

```
int a[10];
int b[10];
...
for(int i = 0 ; i < 10 ; i++) {
    if (i % 2 == 0)
        sum1 += a[i] * b[i];    // indexed access
    else
        sum2 += a[i] + b[i];    // indexed access

    if (a[i] > b[i])            // indexed access, again
        i++;
}
```

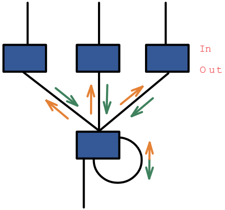
Available expressions: redundancies elimination





How to find available expressions (or other things)?

- A generic mathematical framework exists (magic with lattices)
- How it works
  - Pick a property (eg. available expressions, live variables)
  - Define flow functions
    (how to combine information from adjacent BBs)
  - Attach an input and output set In_b, Out_b to every basic block
    (set of available expressions, set of live variables etc.)
  - Repeatedly recompute In_b, Out_b sets of all BBs



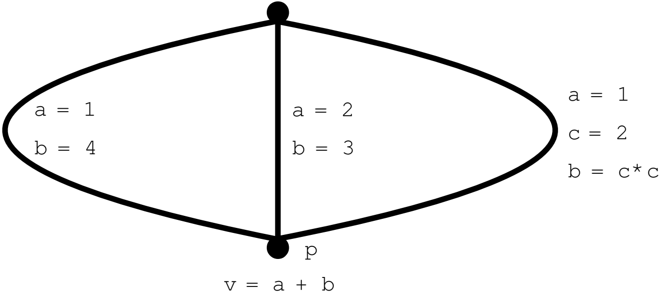
  - Fix-point is found, we are done

Content

- Data-Flow Analysis
- Constant Propagation

Constant propagation

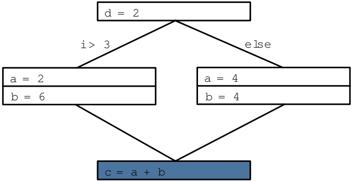
If on every path leading to the point p the expression ends with the same value, we can replace that value with a constant.

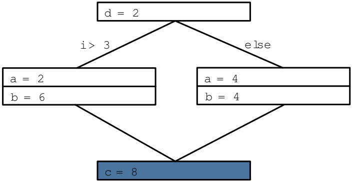


Constant propagation (example)

```
if (i > 3) {
    a = 2;
    b = 6;
} else {
    a = 4;
    b = 4;
}
c = b + a;
```

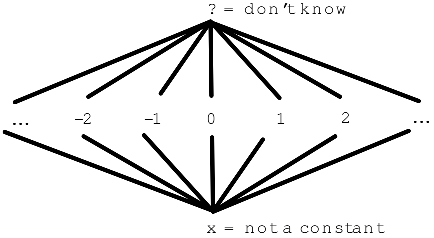
```
if (i > 3) {
    a = 2;
    b = 6;
} else {
    a = 4;
    b = 4;
}
c = 8;
```



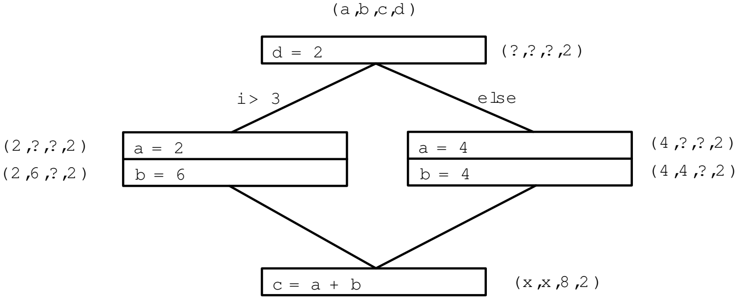


Constant Propagation

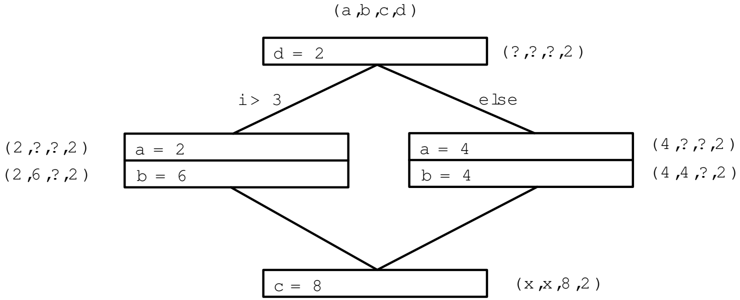
- Need for generalization of flow functions (change of variables not known in advance, e.g. user input)
- Special lattice
- If given variable can have more values ⇒ join ⇒ go down in lattice



Constant propagation example



Constant propagation example



The best solution

- To get the best solution we need to compute flow functions for all paths in the program (MOP = meet over paths)
- In case of loop there are infinitely many paths $\Rightarrow$ not computable
- Instead we compute with edges between BBs (MFP = maximum fixpoint)

References

T. Vojnar, L. Holik *Formal Analysis and Verification, lecture 10*. 2011/2012.