CS412/CS413

Introduction to Compilers

Tim Teitelbaum

Lecture 27: More Dataflow Analysis

31 Mar 08

Lattices

- Lattice:
  - Set augmented with a partial order relation $\sqsubseteq$
  - Each subset has a LUB and a GLB
  - Can define: meet $\sqcap$, join $\sqcup$, top $\top$, bottom $\bot$
- Use lattice to express information about a point in a program, where S1 $\sqsubseteq$ S2 means “S1 is less or equally precise as S2”
- To compute information: build constraints that describe how the lattice information changes
  - Effect of instructions: transfer functions
  - Effect of control flow: meet operation

Transfer Functions

- Let L = dataflow information lattice
- Transfer function $F_I : L \rightarrow L$ for each instruction I
  - Describes how I modifies the information in the lattice
  - If in[I] is info before I and out[I] is info after I, then

    Forward analysis: $out[I] = F_I(in[I])$

    Backward analysis: $in[I] = F_I(out[I])$
- Transfer function $F_B : L \rightarrow L$ for each basic block B
  - Is composition of transfer functions of instructions in B
  - If in[B] is info before B and out[B] is info after B, then

    Forward analysis: $out[B] = F_B(in[B])$

    Backward analysis: $in[B] = F_B(out[B])$

Control Flow

- Meet operation models how to combine information at split/join points in the control flow
  - If in[B] is info before B and out[B] is info after B, then:

    Forward analysis: $in[B] = \sqcap \{out[B'] \mid B' \in pred(B)\}$
    Backward analysis: $out[B] = \sqcap \{in[B'] \mid B' \in succ(B)\}$

- Can alternatively use join operation $\sqcup$ (equivalent to using the meet operation $\sqcap$ in the reversed lattice)

Treatment as $F: L^n \rightarrow L^n$

- For a data flow analysis problem
  - With lattice L
  - Basic blocks $B_1, ..., B_n$
  - Transfer functions $F_1, ..., F_n$
- Treat as
  - Iteration of function $F: L^n \rightarrow L^n$

    $\top, F(\top), F(F(\top)), \ldots$
  - Where F summarizes effect of one sweep for all blocks B in a given order of either

    out[B] = ... and in[B] $=F_B$(out[B]) (for backward)

    in[B] = ... and out[B]$=F_B$(in[B]) (for forward)

Monotonicity

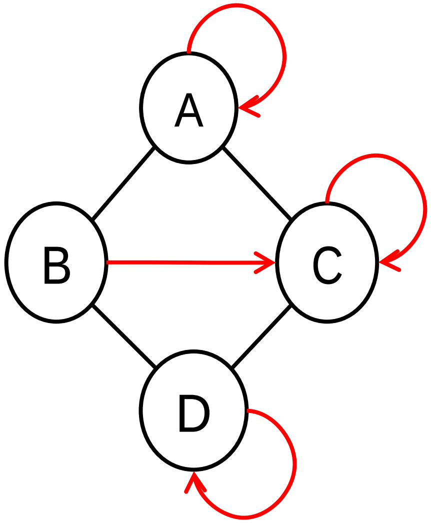
- Function F : L → L is monotonic if

$$x \sqsubseteq y \text{ implies } F(x) \sqsubseteq F(y)$$

- A monotonic function is “order preserving”
- Contrast with

$$\text{For all } x,\ F(x) \sqsubseteq x$$

- F is monotonic but $C = F(B) \not\sqsubseteq B$



Monotonicity of Meet

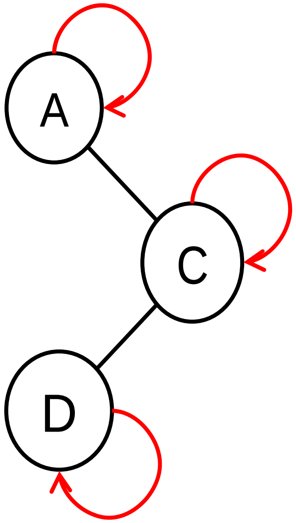
- Meet operation is monotonic over L x L, i.e.,

  $x_1 \sqsubseteq y_1$ and $x_2 \sqsubseteq y_2$ implies $(x_1 \sqcap x_2) \sqsubseteq (y_1 \sqcap y_2)$

- Proof:
  - any lower bound of $\{x_1,x_2\}$ is also a lower bound of $\{y_1,y_2\}$, because $x_1 \sqsubseteq y_1$ and $x_2 \sqsubseteq y_2$
  - $x_1 \sqcap x_2$ is a lower bound of $\{x_1,x_2\}$
  - So $x_1 \sqcap x_2$ is a lower bound of $\{y_1,y_2\}$
  - But $y_1 \sqcap y_2$ is the greatest lower bound of $\{y_1,y_2\}$
  - Hence $(x_1 \sqcap x_2) \sqsubseteq (y_1 \sqcap y_2)$

Fixed Points

- x in lattice L is a fixed point of function F iff x=F(x)
- Tarski-Knaster Fixed Point Theorem. The fixed points of a monotonic function on a complete lattice form a complete lattice. In particular, there is a maximal fixed point (MFP).



Chains in Lattices

- A chain in a lattice L is a totally ordered subset S of L:

  $x \sqsubseteq y$ or $y \sqsubseteq x$ for any $x, y \in S$

- In other words:

  Elements in a totally ordered subset S can be indexed to form an ascending sequence:

  $x_1 \sqsubseteq x_2 \sqsubseteq x_3 \sqsubseteq \ldots$

  or they can be indexed to form a descending sequence:

  $x_1 \sqsupseteq x_2 \sqsupseteq x_3 \sqsupseteq \ldots$

- Height of a lattice = size of its largest chain
- Lattice with finite height: only has finite chains

Iterative Computation of Solution

- Let F be a monotonic function over lattice L
- $\top \sqsupseteq F(\top) \sqsupseteq F(F(\top) \sqsupseteq \ldots$ is a descending chain
- If L has finite height, the chain ends at the maximal fixed point of F (MFP)

Multiple Solutions

- Dataflow equations may have multiple solutions

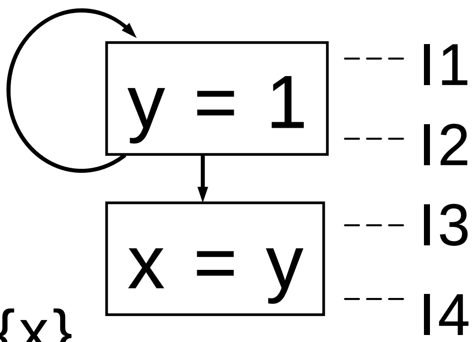
- Example: live variables

Equations: $I1 = I2 - \{y\}$

$I3 = (I4 - \{x\}) \cup \{y\}$

$I2 = I1 \cup I3$

$I4 = \{x\}$



Solution 1: $I1=\{\}$, $I2=\{y\}$, $I3=\{y\}$, $I4=\{x\}$

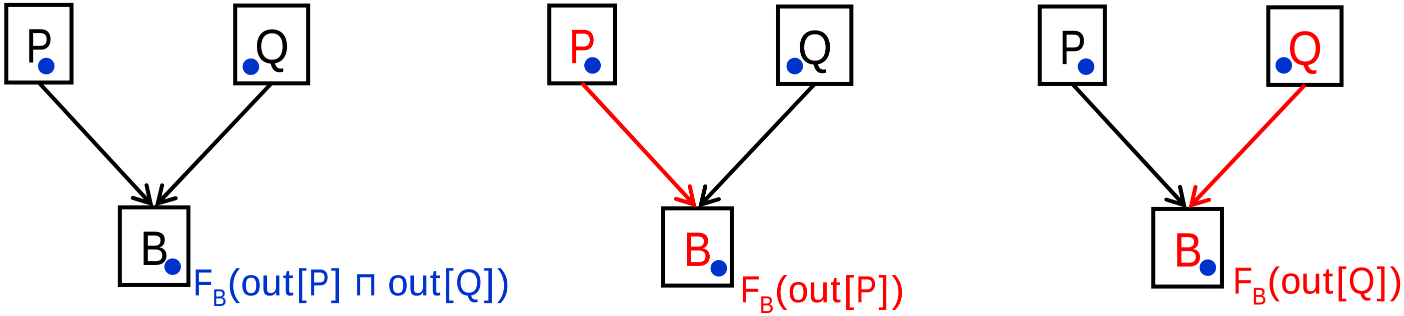
Solution 2: $I1=\{x\}$, $I2=\{x,y\}$, $I3=\{y\}$, $I4=\{x\}$

For any solution FP of the dataflow equations $FP \sqsubseteq MFP$

FP is said to be a conservative or safe solution

Meet Over Paths Solution (forward)

- Is MFP the best solution to an analysis problem?



- Alternative to MFP: a different way to compute solution
  - Let G be the control flow graph with start block B_0
  - For each path $p_n=[B_0, B_1, ..., B_n]$ from B_0 to block B_n define $F[p_n] = F_{Bn-1} \circ F_{B1} \circ ... \circ F_{B0}$
  - Compute solution as

    $in[B_n] = \sqcap \{ F[p_n](\text{start value}) \mid \text{all paths } p_n \text{ from } B_0 \text{ to } B_n\}$
- This solution is the Meet Over Paths (MOP) solution for block B_n

MFP versus MOP

- Precision: MOP solution is at least as precise as MFP

  $$\text{MFP} \sqsubseteq \text{MOP}$$

- Why not use MOP?
- MOP is intractable in practice

  1. Exponential number of paths: for a program consisting of a sequence of N if statement, there will 2^N paths in the control flow graph

  2. Infinite number of paths: for loops in the CFG

Distributivity

- Function $F : L \to L$ is distributive if

$$F(x \sqcap y) = F(x) \sqcap F(y)$$

- Property: F is monotonic iff $F(x \sqcap y) \sqsubseteq F(x) \sqcap F(y)$
  - any distributive function is monotonic!

Importance of Distributivity

- Property: if transfer functions are distributive, then the solution to the dataflow equations is identical to the meet-over-paths solution

  MFP = MOP

- For distributive transfer functions, can compute the intractable MOP solution using the iterative fixed-point algorithm

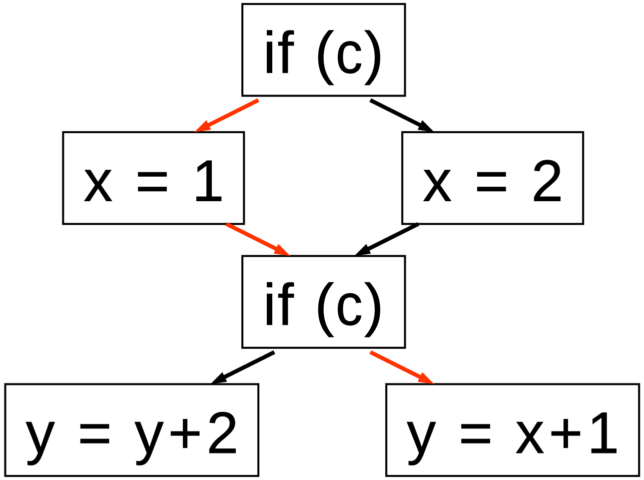
Better Than MOP?

- Is MOP the best solution to the analysis problem?

- MOP computes solution for all paths in the CFG
- There may be paths that will never occur in any execution
- So MOP is conservative

- IDEAL = solution that takes into account only paths that occur in some execution

- This is the best solution
- ... but it is undecidable



Dataflow Equations

- Solve equations: use an iterative algorithm
  - Initialize in[B_s] = start value
  - Initialize everything else to $\top$
  - Repeatedly apply rules
  - Stop when reach a fixed point

Kildall Algorithm (forward)

$in[B_S]$ = start value

$out[B] = \top$, for all B

repeat

  for each basic block $B \neq B_s$

    $in[B] = \sqcap \{out[B'] \mid B' \in pred(B)\}$

  for each basic block B

    $out[B] = F_B(in[B])$

until no change

Efficiency

- Algorithm is inefficient
  - Effects of basic blocks re-evaluated even if the input information has not changed
- Better: re-evaluate blocks only when necessary
- Use a worklist algorithm
  - Keep of list of blocks to evaluate
  - Initialize list to the set of all basic blocks
  - If out[B] changes after evaluating out[B] = F_B(in[B]), then add all successors of B to the list

Worklist Algorithm (forward)

$in[B_S]$ = start value

$out[B] = \top$, for all B

worklist = set of all basic blocks B

repeat

  remove a node B from the worklist

  $in[B] = \sqcap \{out[B'] \mid B' \in pred(B)\}$

  $out[B] = F_B(in[B])$

  if out[B] has changed **then**

    worklist = worklist ∪ succ(B)

until worklist = ∅

Correctness

- Initial algorithm is correct
  - If dataflow information does not change in the last iteration, then it satisfies the equations

- Worklist algorithm is correct
  - Maintains the invariant that

    $in[B] = \sqcap \{out[B'] \mid B' \in pred(B)\}$

    $out[B] = F_B(in[B])$

    for all the blocks B not in the worklist
  - At the end, worklist is empty

Summary

- Dataflow analysis
  - sets up system of equations
  - iteratively computes MFP
  - Terminates because transfer functions are monotonic and lattice has finite height

- Other possible solutions: FP, MOP, IDEAL
- All are safe solutions, but some are more precise:

  $\text{FP} \sqsubseteq \text{MFP} \sqsubseteq \text{MOP} \sqsubseteq \text{IDEAL}$

- MFP = MOP if distributive transfer functions

- MOP and IDEAL are intractable
- Compilers use dataflow analysis and MFP