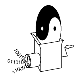


CS 412
Introduction to Compilers

Andrew Myers

Cornell University

Lecture 28: Dataflow analysis frameworks

9 Apr 01

Administration

- Homework 4 due Friday the 13th
- Prelim April 17, 7:30PM-9:30PM
  - static semantics, IR and assembly code generation, object-oriented languages, data-flow analysis, optimization

Available expressions

- Idea: want to perform common subexpression elimination

```
a = x+1          a = x+1
  ...      ⇒       ...
b = x+1          b = a
```

- Transformation is safe if original x+1 is an *available expression* (still computes same value)

Dataflow values

- Let $in[n]$, $out[n]$ be sets of nodes whose computed expression is available at n

n	$gen[n]$	$kill[n]$
a=b *OP* c	$\{n\} - kill[n]$	$uses$(a)
a=[b]	$\{n\} - kill[n]$	$uses$(a)
[a]=b	{}	$uses$([x]) (for all x that may be equal to a)
a=f(b_1,...b_n)	{}	$uses$([x]) (for all x)
other	{}	{}

Constraints

$out[n] \supseteq gen[n]$

"An expression made available by n at least reaches n's output"

$in[n'] \subseteq out[n]$ (if n' is succ. of n)

"An expression is available at n' only if it is available at *every* predecessor n"

$out[n] \cup kill[n] \supseteq in[n]$

"An expression available on input is either available on output or killed"

Dataflow equations

$out[n] \supseteq gen[n]$

$in[n'] \subseteq out[n]$ (if n' is succ. of n)

$out[n] \cup kill[n] \supseteq in[n]$

Equations for iterative solution:

$out[n] = gen[n] \cup (in[n] - kill[n])$

$in[n'] = \bigcap_{n \in \mathrm{pred}[n']} out[n]$

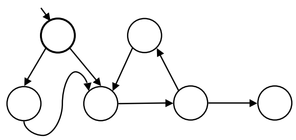
$\sqcap = \cap$ Starting condition:

$in[n]$ is set of *all* nodes

$in[start] = \emptyset$

Dataflow analysis

- Propagates information about program through flowgraph. Dataflow values make up space L
- Solution: $in[n]$, $out[n] \in L$ for every node n
- Live variable analysis: set of live variables
- Available expressions: set of available exprs



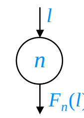
Dataflow analysis framework

Dataflow analysis characterized by:

1. Space of values L
2. Flow function F_n for every node n

$out[n] = F_n(in[n])$

$F_n : L \rightarrow L$



"If $l \in L$ is true before executing node n, $F_n(l)$ is true afterward"

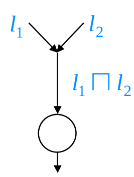
All analyses so far: $F_n(l) = gen[n] \cup (l - kill[n])$

Live vars: $F_n(l) = use[n] \cup (l - def[n])$
($gen = use$, $kill = def$)

Combining operator

1. Space of values L
2. Flow function F_n for every node n
3. Combining operator $\sqcap$

"If we know either l_1 or l_2 holds on entry to n, we know at most $l_1 \sqcap l_2$"



$in[n] = \bigsqcap_{n' \in pred[n]} out[n']$

live vars: $\sqcap = \cup$ avail exprs: $\sqcap = \cap$

Iterative analysis

4. maximum information $\top \in L$

```
for all n, in[n] = out[n] = ⊤
  repeat until no change
      for all n
          in[n] = ⊓n' ∈ pred[n] out[n']
          out[n] = Fn(in[n])
      end
  end
```


Questions

Will iterative analysis

- produce a solution when it terminates?
- produce the best solution possible?
- terminate?

- Depends on properties of L, F_n, $\sqcap$

L as partial order

- Best solution has as much information as possible – allows most optimization
  - Live variables: smallest possible set
  - Available expressions: largest possible set
- Some dataflow values contain more information: $l_1 \sqsubseteq l_2$ if l_2 has at least as much information as l_1

- Live variables: $l_1 \sqsubseteq l_2 \Leftrightarrow l_1 \supseteq l_2$
- Available expressions: $l_1 \sqsubseteq l_2 \Leftrightarrow l_1 \subseteq l_2$

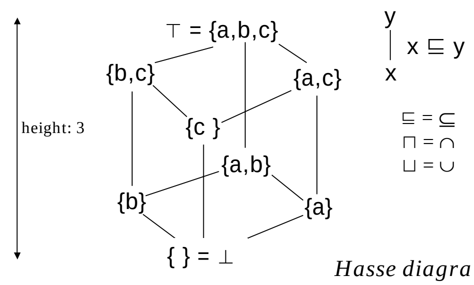
Partial orders

- L is a *partial order* defined by ordering relation $\sqsubseteq$
- Some elements are incomparable
- Properties of a partial order

$$x \sqsubseteq x \qquad \textit{(reflexive)}$$
$$x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z \qquad \textit{(transitive)}$$
$$x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y \qquad \textit{(anti-symmetry)}$$

- Examples: integers ordered by $\leq$, types ordered by **<:**, sets ordered by $\subseteq$ or $\supseteq$.

Example: subsets of {a,b,c}



Hasse diagram

Greatest lower bound

- Combining operator $l_1 \sqcap l_2$ gives element l such that $l \sqsubseteq l_1$, $l \sqsubseteq l_2$
- l is a *lower bound* for l_1, l_2 (consistent with both)
- Want *greatest* such element (most info): *greatest lower bound* (GLB)
- Partial order with GLB/meet ($\sqcap$) and LUB/join ($\sqcup$) is a *lattice*
- With only GLB, a *lower semilattice*

Meet-over-paths solution

- Consider traversal of flowgraph visiting nodes $a, b, c, \ldots, n$
- Assume l_0 is initial information
- Knowable information is
  $F_n(\ldots(F_c(F_b(F_a(l_0)))))$
- Best possible solution is l such that
  $l \sqsubseteq F_n(\ldots(F_c(F_b(F_a(l_0)))))$
  for *all* paths $a, b, c, \ldots, n$
- MOP (meet-over-paths) soln:
  $\prod_{all\ paths\ p} F_{p_1}(F_{p_2}(F_{p_3}(\ldots)))$

Data-flow equations

- Algorithm repeatedly recomputes each *out*[*n*] as
  $$F_n(\textstyle\prod_{n' \in pred[n]} out[n'])$$
- Let $x_1 \ldots x_n$ be *out*[1]...*out*[*n*]. Algorithm:
  $$x_i = F_i(\textstyle\prod_{j \in pred[i]} x_j)$$
- Solution is point in L^n : $\mathbf{X} = (x_1,\ldots x_n)$
- Total set of equations is $\mathbf{X} = \mathbf{F}(\mathbf{X})$ where
  $\mathbf{F}(\mathbf{X}) = (F_1(\prod_{j \in pred[1]} x_j), F_2(\prod_{j \in pred[2]} x_j), \ldots)$
- Any solution to constraints is *fixed point* of **F**

Fixed points

- Iterative analysis: initialize all x_i with top of lattice ($\mathbf{X}_0 = (\top, \top, \top, \ldots)$), apply $\mathbf{F}(\mathbf{X})$ until fixed point is reached:
  $$\mathbf{F}^k(\mathbf{X}_0) = \mathbf{F}^{k+1}(\mathbf{X}_0)$$
- $\mathbf{F}^k(\mathbf{X}_0)$ is a fixed point of F: a value that F maps to itself
- Wanted: maximal fixed point

Monotonicity

- Flow functions map lattice values to other lattice values; must be *monotonic*
- Monotonicity:

$$l_1 \sqsubseteq l_2 \Rightarrow F(l_1) \sqsubseteq F(l_2)$$

"If you have more information entering a node, you have at least as much leaving"

- Example: *reaching definitions*. Lattice is all sets of defining nodes ordered by subset relation:

$$F_n(x) = gen[n] \sqcup (x - kill[n])$$

Termination

- First step either lowers some x_i or terminates
- Second step sees same x_i as first step ($\top$), or possibly lower: $\mathbf{F}(\mathbf{X}_0) \sqsubseteq \mathbf{X}_0$
- Monotonicity $l_1 \sqsubseteq l_2 \Rightarrow \mathbf{F}(l_1) \sqsubseteq \mathbf{F}(l_2)$

$\Rightarrow$ output of second step $\mathbf{F}^2(\mathbf{X}_0)$ is lower than first step (or it terminates): $\mathbf{F}^2(\mathbf{X}_0) \sqsubseteq \mathbf{F}^1(\mathbf{X}_0)$

- Induction: each iteration moves at least one node lower in lattice: $\mathbf{F}^{i+1}(\mathbf{X}_0) \sqsubseteq \mathbf{F}^i(\mathbf{X}_0)$
- # algorithm steps to fixed point is at most *height* of lattice H times number of nodes n: $k = O(nH)$

Solution quality

- MOP is best possible solution:

$$\sqcap_{\textit{all paths } p} F_{p1}(F_{p2}(F_{p3}(\ldots)))$$

- Does iterative analysis

$$x_i = F_i(\sqcap_{j \in pred[i]} x_j)$$

produce the MOP solution?

- Yes, if flow functions *distribute* over the meet operator:

$$\sqcap_i F_n(x_i) = F_n(\sqcap_i x_i)$$

- Not all analyses give MOP solution!

Reaching definitions

- L is all sets of defining nodes in call flow graph. Maximum information means *smallest possible* lists of reaching definitions, so:
- Top ($\top$) is the empty set { }, meet ($\sqcap$) is set union ($\cup$)

$x_n = out[n]$

$F_n(x) = gen[n] \cup (x - kill[n])$

$$x_i = F_i(\sqcap_{j \in pred[i]} x_j) \Rightarrow \begin{array}{l} in[n] = \cup_{n' \in prev[n]} out[n'] \\ out[n] = gen[n] \cup (in[n] - kill[n]) \end{array}$$

Monotonic?

$$F_n(x) = gen[n] \cup (x - kill[n])$$
$$x_1 \sqcap x_2 = x_1 \cup x_2$$
$$x \sqsubseteq y \Leftrightarrow x \supseteq y$$

- Is $F_n(x)$ monotonic?

$F_n(x) = gen[n] \cup (x - kill[n])$

$F_n(x \cup y) = gen[n] \cup ((x \cup y) - kill[n]) =$
$gen[n] \cup (x - kill[n]) \cup (y - kill[n]) =$
$F_n(x) \cup (y - kill[n])$

MOP?

$$F_n(x) = gen[n] \cup (x - kill[n])$$
$$x_1 \sqcap x_2 = x_1 \cup x_2$$

- Does $F_n(x)$ distribute over $\sqcap$?

$F_n(x \sqcap y) = F_n(x \cup y)$
$= gen[n] \cup ((x \cup y) - kill[n])$
$= (gen[n] \cup (x - kill[n]))$
$\quad \cup (gen[n] \cup (y - kill[n]))$
$= F_n(x) \cup F_n(y) = F_n(x) \sqcap F_n(y)$

$\therefore$ Iterative analysis always terminates, finds the best possible (meet-over-paths) solution to reaching definitions

Other analyses

- Live variables

$$F_n(l) = use[n] \cup (l - def[n])$$
$$\sqcap = \cup$$

- Available expressions

$$F_n(l) = gen[n] \cup (l - kill[n])$$
$$\sqcap = \cap$$

- Computes MOP solutions?

Summary

- Analyses for standard optimizations fit into dataflow analysis framework
- Iterative analysis finds solution if flow function monotonic in $\sqsubseteq$, combining function $\sqcap$ defines lower semilattice
- Solution is MOP if distribution condition $\bigsqcap_i F(x_i) = F(\bigsqcap_i x_i)$ holds