**CS 412/413**

Introduction to
Compilers and Translators
Andrew Myers
Cornell University

Lecture 27: Dataflow analysis theory
5 April 00

---

### Dataflow analysis

- Abstractly: propagates dataflow values representing information about program through flowgraph. Space of values: \( L \)
- Solution: \( \text{in}[n], \text{out}[n] \in L \) 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 \( n \)
3. \( \text{out}[n] = F_n(\text{in}[n]) \)
4. \( F_n : L \rightarrow L \)

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

Live vars: \( F_n(l) = \text{use}[n] \cup (l - \text{def}[n]) \)

---

### Iterative analysis

for all \( n \), \( \text{in}[n] = \text{out}[n] = L \)

repeat until no change

for all \( n \)

\[
\begin{align*}
\text{in}[n] &= \bigcap_{n' \in \text{pred}(n)} \text{out}[n'] \\
\text{out}[n] &= F_n(\text{in}[n])
\end{align*}
\]

end
Questions
Will iterative analysis
- produce a solution when it terminates?
- produce the best solution possible?
- terminate?

• Depends on properties of \( L, F, \sqcap \)

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: \( l_i \sqsubseteq l_j \) if \( l_i \) has more information than \( l_j \)

• Live variables: \( l_i \sqsubseteq l_j \) \( \iff \) \( l_i \supseteq l_j \)
• Available expressions: \( l_i \sqsubseteq l_j \) \( \iff \) \( l_i \subseteq l_j \)

Partial orders
• \( L \) is a partial order defined by ordering operator \( \sqsubseteq \)
• Some elements are incomparable
• Properties of a partial order
  \( x \sqsubseteq x \) (reflexive)
  \( x \sqsubseteq y \) \& \( y \sqsubseteq z \) \( \Rightarrow \) \( x \sqsubseteq z \) (transitive)
  \( x \sqsubseteq y \) \& \( y \sqsubseteq x \) \( \Rightarrow \) \( x = y \) (anti-symmetry)
• Examples: integers ordered by \( \leq \), types ordered by \( <: \), sets ordered by \( \subseteq \) or \( \supseteq \).

Example: subsets of \{a, b, c\}

Greatest lower bound
• Combining operator \( l_i \sqcap l_j \) gives element \( l \) such that \( l \sqsubseteq l_i, l \sqsubseteq l_j \)
• \( l \) is a lower bound for \( l_i, l_j \)
• 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 semi-lattice

Meet-over-paths solution
• Consider a traversal of flowgraph visiting nodes \( a, b, c, \ldots, n \)
• Assume \( l_0 \) is initial information
• Information known is \( F_0, \ldots, F_n(F_n(F_0(l_0))) \)
• Best possible solution is \( l \) such that \( l \sqsubseteq F_0, \ldots, F_n(F_n(F_0(l_0))) \) for all paths \( a, b, c, \ldots, n \)
• MOP soln: \( \sqcap \text{ all paths } p F_p(F_p(F_p(\ldots))) \)
Data-flow equations

- Algorithm repeatedly recomputes each \( \text{out}[n] \) as
  \[
  F_n(\bigcap_{n' \in \text{pred}(n)} \text{out}[n'])
  \]
- Let \( x_1...x_n \) be \( \text{out}[1]...\text{out}[n] \). Algorithm:
  \[
  x_i = F_i(\bigcap_{j \in \text{pred}(i)} x_j)
  \]
- Solution is point in \( L^n \): \( X = (x_1,...x_n) \)
- Total set of equations is \( X = F(X) \) where
  \[
  F(x_1,...x_n) = (F_1(\bigcap_{j \in \text{pred}(1)} x_j), F_2(...),...)
  \]

Fixed points

- Iterative analysis: initialize all \( x_i \) with top of lattice (\( X_0 = (\top, \top, \top,...) \)), apply \( F(X) \) until fixed point is reached: \( F^k(X_0) = F^{k+1}(X_0) \)
- \( F^k(X_0) \) is a fixed point of \( F \): a value that \( F \) maps to itself
- Wanted: maximal fixed point (we know that minimum-information solution \( \bot \) works)

Monotonicity

- Flow functions map lattice values to other lattice values; must be monotonic
- Monotonicity:
  \[
  l_1 \subseteq l_2 \Rightarrow F(l_1) \subseteq F(l_2)
  \]
  “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:
  \[
  F_n(x) = \text{gen}[n] \cup (x - \text{kill}[n])
  \]

Solution quality

- MOP is best possible solution:
  \[
  \bigcap_{\text{all paths } p} P_{\text{pred}(F_p(F_{p'}(...)))}
  \]
- Does iterative analysis
  \[
  x_i = F_i(\bigcap_{j \in \text{pred}(i)} x_j)
  \]
  produce the MOP solution?
- Flow functions must distribute over the meet operator:
  \[
  \bigcap_{i} F(x_i) = F(\bigcap_{i} x_i)
  \]

Reaching definitions

- \( L \) is all sets of defining nodes in call flow graph. Maximum information means smallest possible lists of reaching definitions, so:
  \[
  \text{Top (}\top\text{)} \text{ is the empty set } \{\}, \text{ meet (}\cap\text{)} \text{ is set union (}\cup\text{)}
  \]
  \[
  x_n = \text{out}[n]
  \]
  \[
  F_n(x) = \text{gen}[n] \cup (x - \text{kill}[n])
  \]
  \[
  x_i = F_i(\bigcap_{j \in \text{pred}(i)} x_j) \subseteq \text{in}[n] = \bigcup_{n' \in \text{pred}(i)} \text{out}[n']
  \]
  \[
  \text{out}[n] = \text{gen}[n] \cup (\text{in}[n] - \text{kill}[n])
  \]
Monotonic?

\[ F_n(x) = \text{\textit{gen}}[n] \cup (x - \text{\textit{kill}}[n]) \]
\[ x_1 \cap x_2 = x_1 \cup x_2 \]

- Is \( F_n(x) \) monotonic?
  - If \( x \subseteq y \)
    - \( F_n(x) = \text{\textit{gen}}[n] \cup (x - \text{\textit{kill}}[n]) \)
    - \( \text{\textit{gen}}[n] \cup (x \cup y - \text{\textit{kill}}[n]) \)
    - \( \text{\textit{gen}}[n] \cup (x - \text{\textit{kill}}[n]) \cup (y - \text{\textit{kill}}[n]) \supseteq F_n(y) \)

MOP?

\[ F_n(x) = \text{\textit{gen}}[n] \cup (x - \text{\textit{kill}}[n]) \]
\[ x_1 \cap x_2 = x_1 \cup x_2 \]

- Does \( F_n(x) \) distribute over \( \cap \)?
  - \( F_n(x \cap y) = F_n(x \cup y) \)
  - \( = \text{\textit{gen}}[n] \cup (x \cup y - \text{\textit{kill}}[n]) \)
  - \( = (\text{\textit{gen}}[n] \cup (x - \text{\textit{kill}}[n])) \)
  - \( \cup (\text{\textit{gen}}[n] \cup (y - \text{\textit{kill}}[n])) \)
  - \( = F_n(x) \cup F_n(y) \)

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

Other analyses

- Live variables
  - \( F_n(l) = \text{\textit{use}}[n] \cup (l - \text{\textit{def}}[n]) \)
    - \( \cap = \cup \)

- Available expressions
  - \( F_n(l) = \text{\textit{gen}}[n] \cup (l - \text{\textit{kill}}[n]) \)
    - \( \cap = \cap \)

- Computes MOP solutions?

Summary

- Standard optimizations require data-flow analyses that fit into data-flow analysis framework
- Iterative analysis finds solution if flow function monotonic in \( \cap \), combining function \( \cap \) defines semi-lattice
- Solution is MOP if distribution condition \( \cap, F(x) = F(\cap, x) \) holds