Slicing Object-Oriented Software

L. Larsen and M. J. Harrold
Clemson, Ohio State, IBM
ICSE’96
Overview

• Slicing for imperative languages
• Slicing of object-oriented languages
• Questions
Traditional Slicing

- Slice: the parts of the program that potentially affect certain values at some program point
  - Input: \((n, V); \, n \in \text{CFG}, \, V \subseteq \text{Vars}\)
  - Output: executable program obtained by deleting statements from the original program
  - The values of \(v \in V\) at point \(n\) are the same
  - Goal: help debugging
Traditional Slicing (cont)

• Current definition
  – Input: node \( n \); \( V \) is the set of all vars used/defined at \( n \)
  – Output: set of statements and control predicates
  – Based on reachability in a \textit{dependence graph}

• Applications
  – Debugging
  – Software maintenance
  – Testing
  – Reverse engineering

• Survey paper by F. Tip (J. Prog. Lang., 1995)
System Dependence Graph

• Nodes
  – Statements
  – Control predicates
  – Formal/actual vertices (incl. globals)

• Edges
  – Data dependencies (e.g., reaching definitions)
  – Control dependencies (based on post-dominance)
  – Parameter passing
Slicing with SDG

• Horwitz-Reps-Binkley, 1990
  – Input: a node $n$ in the SDG
  – Output: a set of SDG nodes that can reach $n$ along realizable paths

• First compute summary edges

• Pass 1: Backward reachability from $n$
  – Does not descent into called procedures

• Pass 2: Backward reachability
  – Descends into called procedures

• Example: slice w.r.t. C9-A1out
Class Dependence Graph

• Structure: similar to SDG
  – A procedure dependence graph for each method
  – Represents dependencies that can be determined without knowledge of the calling environment

• Additional nodes and edges
  – Class entry node and class member edges
    • For convenience; not used for slicing
  – Formal nodes for referenced instance variables
  – Matching actual nodes for instance variables
Slicing with ClassDG

• Complete programs
  – Same as the traditional slicing
  – Example: C20-A1out

• Incomplete programs
  – Use a worst-case driver
  – Correctness issue: unknown derived classes
Questions

• How do you determine formal/actual nodes
  – What is used/modified directly or indirectly?
• Dealing with pointers
• Dependencies due to instance variables
  – Values preserved across calls
  – How do you add actual-actual edges?
• Very high-level, a lot of open questions
• No empirical results
• Is slicing any good?