Slide Index
NPIC & HMIT (May 8, 1996):

12 slides
April 11, 1996

UNIT 1

Unravel

Unravel

USING A PROGRAM SLICING CASE TOOL FOR EVALUATING HIGH INTEGRITY SOFTWARE SYSTEMS

Leo Beltracchi LXB@NRC.GOV
James R. Lyle JLYLE@NIST.GOV
Dolores R. Wallace DWALLACE@NIST.GOV

Nuclear Plant Instrumentation, Control, and
Human-Machine Interface Technologies
May 6-9, 1996 at The Nittany Lion Inn,
The Pennsylvania State University

Version 1.5

Talk Outline

- Problem Introduction
- Program Slicing
- Unravel Design
- Results Using Unravel
- Unravel Demonstration
- Functional Analysis
- Testing the Slice
- Conclusions

Problems in Software Safety Evaluation

- Need to find software faults (bugs) quickly.
- Need to identify existence of common code.
- Need to identify impact of a code change.
- Need to connect inputs (e.g. a sensor signal) to outputs (e.g. trip signal).
- Need to unravel code to confirm that different algorithms achieve the same function.
- All of the above are variations on the theme:

Need to understand an existing program.

Program Slicing

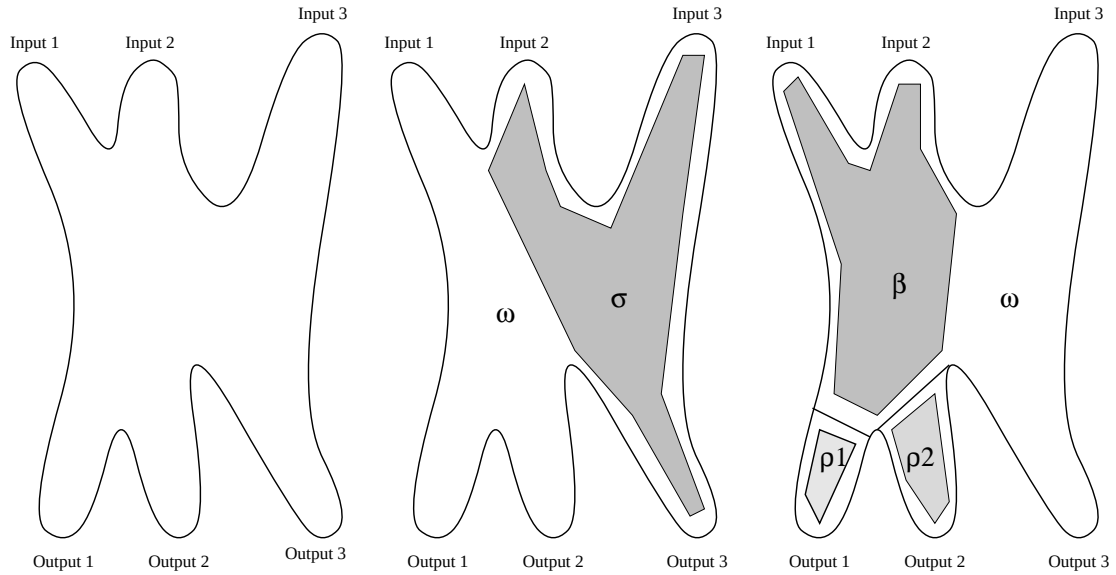
- **Program Slice:** All program statements relevant to a given computation.
- **Slicing Criterion:** Specifies the slice (computation) for a variable, **v**, at a statement, n.
- Program slices for a given slicing criterion are obtained from a given program, **P**, by deleting zero or more statements from **P**, but still computing the same value for **v** at statement n.
- Data–flow analysis is used to identify statements that may be deleted without affecting the computation.

Combining Slices

Program slices can be combined to find or exclude common code.

- The intersection of two slices (**backbone slice**) is the set of statements common to the two computations.
- A slice minus a backbone slice gives the statements unique to a computation (**program dice**).

Programmer's View After a Slice



Initial View

After Slice on Output 3

Relations Between Two Slices

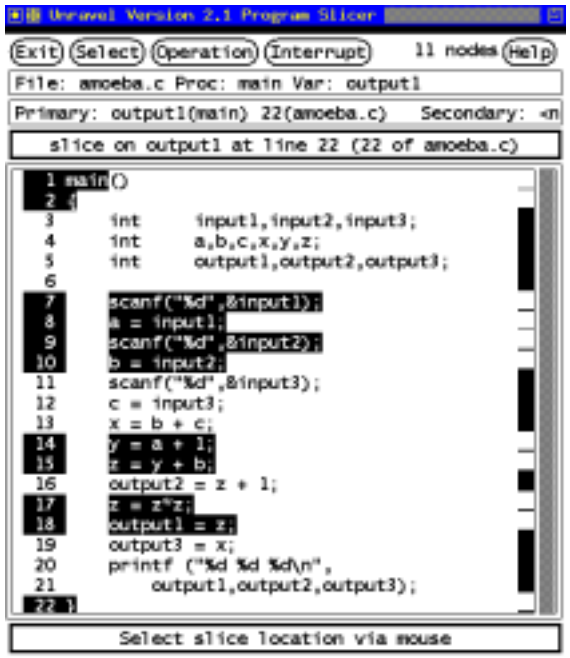
Computing a Program Slice

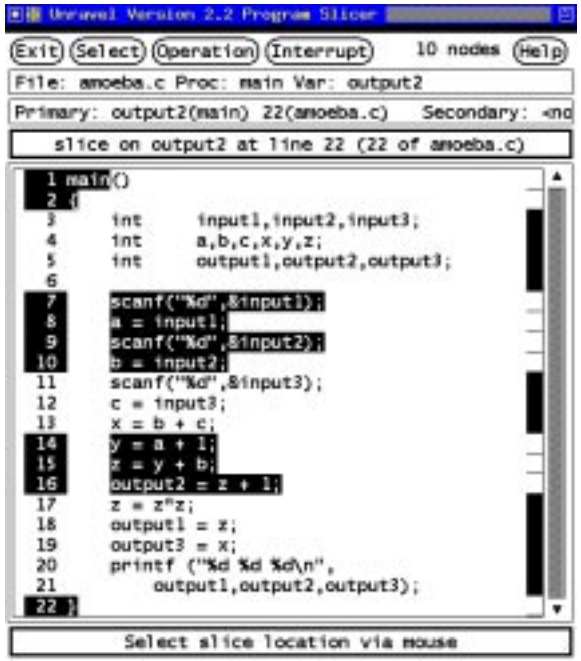
To locate the statements that influence variable v just before execution reaches statement m we would compute a program slice for the criterion $< m, v >$.

1. Start with program flow-graph
2. Annotate with variables referenced and assigned to
3. the defs(n) set and the slicing criterion determine inclusion
4. the refs(n) set gives new criterion

$$S_{<m,v>} = \begin{cases} S_{<n,v>} & \text{if } v \notin \text{defs(n)} \\ \{n\} \cup S_{<n,x>} \forall x \in \text{refs(n)} & \text{otherwise} \end{cases}$$

Slices on Output 1 and Output 2





Combined Slices on Output 1 and Output 2

Using Unravel on Sample Code

Unravel was tried on sample safety system code, generated by an NRC staff person.

- Allowed auditor to extract a computation for manual examination
- Found questionable sharing of code

Unravel Performance

(1) **Unravel** significantly enhances ability to analyze code.

(2) **Unravel** is easy to operate.

(3) **Unravel** can disclose subtle relationships in code that would require a C expert to discover.

(4) The majority of the slices were less than 25 percent of the size of the original program (90 percent for the safety system example).

(5) Requested slices were computed is less than one minute.