

Righting Software

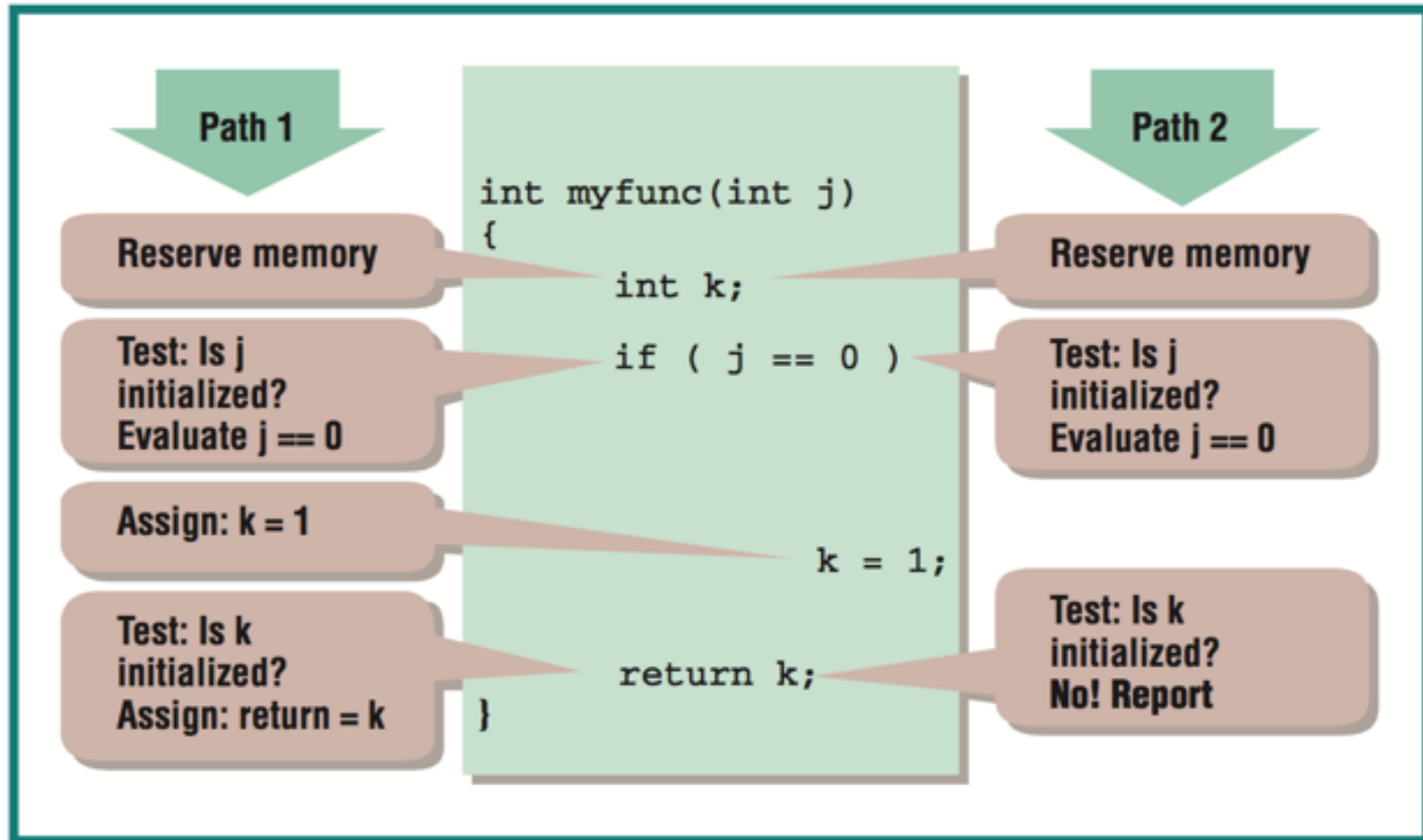
James R. Larus, Thomas Ball, Manuvir Das, Robert DeLine,
Manuel Fähndrich, Jon Pincus, Sriram K. Rajamani, and
Ramanathan Venkatapathy
IEEE Software, May-June 2004

Summary by Prof. Thomas LaToza
SWE 795, Spring 2017
Software Engineering Environments

Detecting Errors with Static Analysis

- Goal: Find important defects through static analysis.
 - Errors in using PL features such as referencing uninitialized variable
 - Errors in API usage such as closing a file descriptor twice
- NOT to determine that program definitely behaves as expected
 - Only to rule out programs that are obviously buggy

Key Idea: Path Exploration



Challenge: Path Feasibility

```
FILE* f;  
if (complex_calc1())  
    f = fopen(...);  
...  
if (complex_calc2())  
    fclose(f);
```

- Determining if path is feasible is hard
- Can't always succeed
- What happens when you don't?
 - Show an error that might be spurious (Sound)
 - Don't show an error that might be spurious (Complete)
 - Show an error if you're fairly confident (Heuristics)

(Some) MS Static Analysis Tools

- Prefix—traverse interprocedural paths, builds models of functions, limit # of paths & use heuristics
- Prefast—use local intraprocedural analysis to find idioms such as wrong param values
- Slam—find protocol violations by building & refining Boolean program abstractions
- ESP—less precise, more scalable interprocedural analysis by pruning irrelevant branches
- Valut—safe version of C that enables modular checking by adding annotations

History

- Tools initially built in late 90s and 2000s at Microsoft
- Used widely internally & in some cases externally (e.g., device drivers)
- Slam tool credited with helping reduce blue screens in MS Windows (other innovations helped too)

The screenshot shows the Microsoft Hardware Dev Center website. The top navigation bar includes the Microsoft logo, 'Technologies', 'Documentation', and 'Resources' menus, along with a search box and a 'Sign in' link. Below this is a dark navigation bar with 'Hardware Dev Center', 'Explore', 'Docs', 'Downloads', 'Samples', 'Support', 'Programs', and 'Dashboard' links. The main content area shows a breadcrumb trail: 'Windows Drivers > Device and Driver Technologies > Tools for Testing Drivers'. The page title is 'Static Driver Verifier', with a 'Last Updated: 2/2/2017' timestamp. A 'Table of contents' sidebar on the left lists 'Driver Development Tools' with sub-items like 'Index of Windows Driver Kit Tools', 'Tools for INF Files', 'Tools for Changing Boot Options for Driver Testing and Debugging', 'Tools for Building Drivers', 'Tools for Signing Drivers', and 'Tools for Testing Drivers'. The main content area has a 'Purpose' section that begins: 'Static Driver Verifier (also known as "StaticDV" or "SDV") is a static verification tool that systematically analyzes the source code of Windows kernel-mode drivers. SDV is a compile time tool that is capable of discovering defects and design issues in a driver. Based on a set of interface rules and a Static Analysis engine, defect by

Questions for discussion

- Overall reaction to the paper
- Would you use such a system for your everyday programming?
 - Why or why not?
- In what circumstances are these tools most useful?