Abstract

*CodeSurfer* is a powerful source code analysis and navigation tool for a range of languages, including C/C++ and x86 machine code. The *Path Inspector* is an add-on to CodeSurfer that allows a user to reason about paths through the program, and which can be used to find programming flaws.

1. CodeSurfer

CodeSurfer is a whole-program static-analysis tool. Given a program, it creates a range of intermediate representations for the program:
- Abstract Syntax Tree (AST).
- Call Graph.
- Interprocedural Control-Flow Graph (CFG).
- Points-to Graph.
- The set of variables used and modified for each procedure (GMOD).
- Control Dependence Graph.
- Data Dependence Graph.

Because CodeSurfer does a whole-program analysis and because it understands pointers, these representations are much more complete than for most existing program analysis tools.

CodeSurfer then provides a variety of tools for viewing and querying these representations. Features important for software maintenance include the following:

- **Navigation**
  - from occurrences to and from declarations for macros, types, variables, and procedures.
  - from the definitions of variables to and from places where those definitions may be used.
    This includes indirects through pointers.
  - from procedure call sites to the called procedure. This also includes indirect calls through pointers.
- **Pointer Analysis**
  - See statically what pointers may point to.
- **Metrics**
  - including complexity and Halstead.
- **Finder**
  - Find variables by occurrence, use, define, or points-to information.
  - Find functions by call, indirect call, or definition.
- **Program Analysis**
  - slicing, chopping, predecessor, successor.
  These are useful for impact analysis.

All of these features are based on well-founded techniques for program analysis.

2. Path Inspector

The Path Inspector is an add-on to CodeSurfer that uses model-checking techniques to allow the user to reason about paths through the program. The user poses a query in terms of a state machine that specifies a behavior, usually something that should not happen. If the system finds a counter-example, the user can browse the path in a manner similar to a debugger. A number of standard templated queries are provided, such as “Every path that goes through A also goes through B”, or “All paths from A to B must also go through C”, where the metavariables A, B, and C correspond to user-specifiable sets of program points.

Several "canned" queries are provided that allow the user to find bugs in a program. These include queries to find the following flaws:

- Null pointer dereferences.
- Use of dynamic storage after a free.
- Double free of dynamic storage.
- Memory leaks.
- Misuse of POSIX API functions.

Users can choose to author their own queries using a special-purpose state-machine specification language (named PIQL).

3. Extensibility

CodeSurfer and the Path Inspector are open and extensible. Scheme is used as the scripting language. All of the underlying representations are accessible through the scripting language with a well-defined API.

4. Availability

CodeSurfer and the Path Inspector are available at no cost to educational institutions for research purposes.