CMSC 631 – Program Analysis and Understanding
Fall 2003

Tools Based on Data Flow Analysis

Administrivia

- Office hour change:
  - TuTh 10-11, or by appointment
  - (Friday office hour conflicted with SysChat)
- Typo on HW1 fixed
  - Ignore "induction variables" part
  - Copy statement is of form x := y
- Papers for next time
  - Don't get caught up in unfamiliar terms in survey
  - Second "paper" is really for marketing purposes

Class So Far

- Too hard/easy?
- Amount of reading?
- Background assumptions?

Terminology

- *Inter*procedural
  - Multiple procedures at a time
- *Intra*procedural
  - One procedure at a time

Sound and Complete

- All program analyses must be approximate
  - see paper for Thursday
- A program checking system is *sound* if
  - Every dynamic error that could occur is detected
  - But the static checking system may complain about things that can't actually happen
- A program checking system is *complete* if
  - Everything that is detected is an actual error
  - But some actual errors may not be caught

The Tools

- Is LCLint
  - Inter- or intraprocedural?
  - Sound? Complete?
- Is ESP
  - Inter- or intraprocedural?
  - Sound? Complete?

No (Useful) Tool for C is Sound

```
char *s, *t;
intptr_t i, j;

j=0;
for (i =0; i<s; i++) j++;
t = (char *) j;
```

- So what?
  - At least we know what we're sound up to
  - It's not any better in theory in other languages
    - Proof: Write a C interpreter in Java

Big Questions

- What are the tradeoffs between testing and static analysis?
  - Other techniques for detecting errors?
-
- What does it mean for a program to be correct?
  - Annotations?
  - Redundancy

Questions About the Tools

- Is the analysis scalable?
  - Full vs partial correctness
- What kinds of annotations are necessary?
  - Will programmers actually use them?
- What problems can be solved?
  - How flexible are the tools?

LCLint

- Tries to detect three classes of errors:
  - Null pointer dereferences
  - Uninitialized memory
  - Allocation/deallocation problems
  - (Plus more stuff not in this paper)
- Technique: Data flow analysis

Null Pointer Errors

- Three "nullness" states possible:
  - null = may be NULL
  - notnull = definitely not NULL
  - relnull = don't care
- Example from paper:

```
extern char *gname;
void setName(/*@null@*/ char *pname) {
    gname = pname;
}
```


Discussion

- Annotations
  - Syntax?
  - Location? (with code/somewhere else)
  - Default assumptions?
- How do we merge states?

Allocation of Memory

- Six allocation states
  - only = unshared, must deallocate or pass on
  - keep = "like only, except that the caller may safely use the reference after the call"
  - temp = can't deallocate or make external ref
  - owned/dependent = shared ref with one owner
  - shared = garbage collected
- Example:

```
null [out] only void *malloc(size_t size);
void free(null [out] only void *ptr);
```


Discussion

- Are these properties orthogonal?
- How does aliasing play in to this?
- Issues that arise in practice?

- see restrict, Vault, unique pointers in Cyclone

Analysis Algorithm

- Treat CFG as acyclic graph
  - No back edges – loops?
- Assume preconditions initially true
- Propagate facts forward
  - Handling of function calls?
    - Soundness? Main?
  - What objects are the facts about?
  - How does this choice interact with modeling of loops?
- Check that postcondition is met at exit

Experience

- Database example
  - 1000 lines source, 300 lines interface spec (?)
  - 15 annotations needed
  - Found some errors
  -
- LCLint example
  - > 100,000 lines of code
  - A few days of work; # annotations?
    - 75 places comments used to suppress warnings
  - Found a few more bugs after passed LCLint

ESP

- Path-sensitive program checker
  - Programmer specifies automaton
    - Errors states = potential bugs in program
  - Programmer specifies automaton transitions
    - E.g., fclose(f) puts f in the *closed* state
  - ESP simulates automaton for a particular object (e.g., a file) along all program paths
    - If the automaton enters error state, signal an error

Example Automaton