

A static analyzer for finding dynamic programming errors

2002. 11. 19

Traditional Check's Problems

- for code generation, not for error-check
- Inter-procedural (interactions between function are not detected)
- reporting false errors. (non-achievable path)
- Require substantial additional work by the user
- requiring test cases

Goal

- Develop a source code analyzer that could find Purify-like errors with Purify's ease of use, but without needing test cases.

Requirement

- C and C++ program should be checked effectively.
- Information should be derived from the program text rather than acquired through user annotations.
- Analysis should be limited to achievable paths
- The information produced from the analysis should be enough to allow a user to characterize the underlying defects easily.

PREFIX

- Simulation: Use Virtual Machine => Memory Test & restrict *achievable* path.
- The behavior of a function : Model
- Model is automatically generated by information extracted from programs.
- Bottom-Up: can apply an entire program, or subset of a program.

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 char *(int size)
5 {
6     char *result;
7
8     if (size > 0)
9         result = (char *)malloc(size);
10    if (size == 1)
11        return NULL;
12    result[0] = 0;
13    return result;
14 }
```

Figure 1. An example function.

```
example1.c(11) : warning 14 : leaking memory
problem occurs in function '!'
The call stack when memory is allocated is:
example1.c(9) : !
Problem occurs when the following conditions are true:
example1.c(8) : when 'size > 0' here
example1.c(10) : when 'size == 1' here
Path includes 4 statements on the following lines: 8 9 10 11
example1.c(9) : used system model 'malloc' for function call:
'malloc(size)'
function returns a new memory block
memory allocated
```

Figure 2. First error message.

The Advantage of Simulation

- Easily get achievable path
- Memory: exact values and predicates
 - Memory Layout
 - Bit Operation
 - Dereference
- End-of-path analysis
 - Any unreachable memory or resource that has been allocated, but not freed, is reported.

(*disadvantage*)

achievable #paths is often quite large -> samples of achievable paths to simulate.
required a user config

Pseudo-code for function simulation

```
while (there are more paths to simulate){
    initialize memory state

    simulate the path, identifying inconsistencies and updating the
    memory state

    perform end-of-path analysis using the final memory state,

    identifying inconsistencies and creating per-path summary
}

combine per-path summaries into a model for the function
```

Conditions, assumptions and choice points

- ```

4 char *f(int size)
5 {
6 char *result;
7
8 if (size > 0)
9 result = (char *)malloc(size);
10 if (size == 1)
11 return NULL;
12 result[0] = 0;
13 return result;
14 }

```
- We don't know the value of "size".
  - Assumption: `size > 0`
  - Condition: `8: if (size > 0) => false`
  - Choice points: `10: if (size == 1) ( need to make assumption : size == 1, size != 1 )`
  - Assumption: `size <= 0`
  - Condition: `8: if (size > 0) => false`
  - Choice points: `10: if (size == 1) => false`

## Models

- **MODEL: The behavior of a function: model consists of exclusive outcomes.**

## OUTCOME

- *Guards* -> an element of test set (input dependent)
- *Constraint* -> precondition
- *Results* -> postcondition

## Ex. Model

```

1 int deref(int *p) (deref
2 { (param p)
3 if (p==NULL) (alternate return_0
4 return NULL; (guard peq p NULL)
5 return *p; (constraint memory_initialized p)
6 } (result peq return NULL)
)
 (alternate return_X
 (guard pne p NULL)
 (constraint memory_initialized p)
 (constraint memory_valid_pointer p)
 (constraint memory_initialized *p)
 (result peq return *p)
)
)

```

Figure 8. The model of the dereferencing function.

## Model Generation

- System lib's models are provided by PREFIX.
- **automatic** generation from user program
- **Merging states(outcomes)** at the end of function simulation is not necessary semantically but performance.

## If source is not available?

- Third party components: source code or models are not available => cannot make models.
- The general philosophy :  
**avoid generating incorrect messages** even at the potential expense of failing to identify real defects.

## Result(1)

Table III. Relationships between available Models, coverage, execution time and defects reported.

| Model set     | Execution time (minutes) | Statement coverage | Branch coverage | Predicate coverage | Total warning count | Using unit memory | NULL pointer deref | Memory leak |
|---------------|--------------------------|--------------------|-----------------|--------------------|---------------------|-------------------|--------------------|-------------|
| None          | 12                       | 90.1%              | 87.8%           | 83.9%              | 15                  | 2                 | 11                 | 0           |
| System        | 13                       | 88.9%              | 86.3%           | 82.1%              | 25                  | 6                 | 12                 | 7           |
| System & auto | 23                       | 73.1%              | 73.1%           | 68.6%              | 248                 | 110               | 24                 | 124         |

## Result(2)

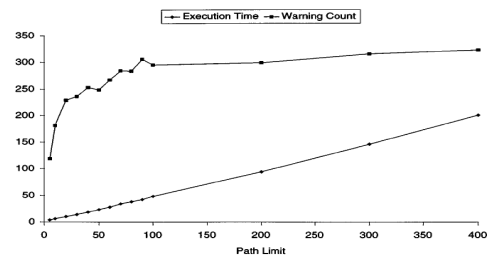


Figure 9. Relationships between path limits, execution time and defects reported.

## Continuing work

- improvement of the presentation, sorting and filtering tools
- additional classes of defects, such as race and deadlock conditions in multi-threaded code
- applying the technology to additional languages.

## Limitations

- The source string in strcpy should be NULL terminated.
- The destination string in strcpy will be NULL terminated.
- The lengths of the source and destination strings will be the same length.
- The memory for the source and destination strings should not overlap.
- The length of the source and destination strings combined should not be greater than the size of available memory.
- These characteristics are typical of those missing from the modeling language.
- There are obvious limitations of expressiveness in the language as it now stands.