A crash course on program analysis

Ivan Yang

SJTUG

March 26, 2016

Outline

1. Basics
   - Halting problem
   - Problem
   - Conclusion

Outline

Outline

Halting problem

Definition

Whether the execution of a specific program for a given input will terminate.

- It's undecidable!
- Because of the input!
- If no input, we can know what's the output is.
- Interpreter

Prediction of program behavior

- Living variables
- Memory leak
- Out of bounds
- Expression reuse
- Dead code

Problem facing

Example

Suppose, for example, you want to determine whether a program might have a out-of-bounds for array indexing at the following expreesion:a[i]

- Change all the exit entry in the program into infinite loops.
- Now the program is guaranteed to non-terminate
- Change the index expression into

Example

((i ¿= 0 && i ¡ a.length) ? a[i] : exit())

- If we could solve halting problem, we can use it to check the out-out-bounds errors!

Consider the other direction:
If we have a program that determine the presence of out-of-bounds errors in a specific program.
Perform the same transformation from above on indexing expression.
This process eliminates all the out-of-bounds errors from the program by transforming them into termination.
Then transform all the exit points into a explicit out-of-bound array reference, e.g. a[a.length + 10]

Conclusion

- If the array-bounds error-checker finds an out-of-bounds error, then it has determined that the original program halts.
- If the array-boudns error-checker claims that the program is free of out-of-bounds errors, then it sovles halting problem.

Compromise

Both of the followings are possible:

- an algorithm that works on some programs for some inputs; and
- an algorithm that works on some programs for all inputs.

If the algorithm can say “Yes” or “No” or “I dont know”, the halting problem is solvable.
Of course, it could always say “I dont know.”

Sign Example

Example

$$\mathbb{S} = \{-, +, 0\}$$

$$\widehat{\mathbb{Z}} = \mathbb{P}(S)$$

$$\alpha(x) = \begin{cases} \{-\} & \text{if } x < 0 \\ \{0\} & \text{if } x = 0 \\ \{+\} & \text{if} x > 0 \end{cases}$$

Addition and multiplication

Example

$$\{+, 0\} \oplus \{-\} = \{-\}$$

$$\{-\} \oplus \{+\} = \{-, +, 0\}$$

$$\{-\} \otimes \{0, +\} = \{-, 0\}$$

Thus, to analyze 4×-3, we convert it into $\alpha(4) \otimes \alpha(-3)$, we get $\{-\}$

Grammar

```
<prog> ::= <stmt> ...

<stmt> ::= <label> :
         |  goto <label> ;
         |  <var> := <exp> ;
         |  if <exp> goto <label> ;

<exp> ::= <exp> + <exp>
        |  <exp> * <exp>
        |  <exp> = <exp>
        |  <int>
        |  <var>
```

Example

```
        a := 1 ;
top:    if n = 0 goto done ;
        a := a * n ;
        n := n + -1 ;
        goto top ;
done:
```

S-expression grammar

```
<prog> ::= <stmt> ...

<stmt> ::= (label <label>)
        |  (goto <label>)
        |  (:= <var> <exp>)
        |  (if <exp> goto <label>)

<exp> ::= (+ <exp> <exp>)
       |  (* <exp> <exp>)
       |  (= <exp> <exp>)
       |  <int>
       |  <var>
```

workflow

- Interpreter
- Abstract semantics
- Abstract analyzer

Popular tools

- Facebook Infer
- Facebook Flow
- Shellcheck
- OCLint