On the Concept of Variable Roles and its Use in Software Analysis

Florian Zuleger, TU Vienna
FMCAD, Portland, 23.10.2013

Joint work with
Yulia Demyanova, Helmut Veith, TU Vienna
Variable Roles

Intuitively, **variable roles** are **patterns of how variables are used by programmers**

**Ex. 1**
```
int i = 0;
while (i < n) {
    a[i] = 0;
    i++;
}
```
- *i* is a loop iterator
- *i* is an array index

**Ex. 2**
```
int x = 2 * y;
```
- *x, y* are linear variables

**Ex. 3**
```
int x = y << 1;
```
- *x, y* are bitvectors

**Ex. 4**
```
int i = getchar();
```
- *i* is a character

**Ex. 5**
```
int i = open(path, flags);
```
- *i* is a file descriptor
Outline

1. Choice and Formalisation

2. Experimental Validation

3. Discussion: Uses of Variable Roles
<table>
<thead>
<tr>
<th>Variable Role</th>
<th>Informal Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNT CONST</td>
<td>not assigned any value in the program</td>
</tr>
<tr>
<td>CONST ASSIGN</td>
<td>assigned only numeric literals or CONST ASSIGN variables</td>
</tr>
<tr>
<td>COUNTER</td>
<td>only incremented/decremented or assigned zero</td>
</tr>
<tr>
<td>LINEAR</td>
<td>assigned only linear combinations of LINEAR variables</td>
</tr>
<tr>
<td>BOOL</td>
<td>assigned only zero, one, BOOL variables or boolean expressions</td>
</tr>
<tr>
<td>INPUT</td>
<td>variable is passed to a function by reference at least one</td>
</tr>
<tr>
<td>BRANCH COND</td>
<td>occurs in the condition of if statement at least once</td>
</tr>
<tr>
<td>BITVECTOR</td>
<td>occurs in a bitwise operation or assigned the result of a bitwise operation at least once</td>
</tr>
<tr>
<td>UNRESOLVED</td>
<td>assigned the value of a pointer dereference</td>
</tr>
<tr>
<td>CHAR</td>
<td>assigned only character literals, CHAR variables or initialised in a specific library function (e.g. getchar)</td>
</tr>
<tr>
<td>LOOP ITERATOR</td>
<td>occurs in the condition of the loop iterator and must be assigned in the loop body</td>
</tr>
</tbody>
</table>
Choice and Formalisation

• Roles were chosen studying 5.2 KLOC code from Cbench benchmark (standard C programs):
  – Goal: find the smallest set of roles to classify every occurring variable
  – Restriction to the types int, float, and char

• Standard dataflow analysis serves as
  1) definition and
  2) algorithm to compute variable roles.
Role Definition: Example

int n=0;
int y=x;
while(x){
    n=n+1;
    x=x&(x-1);
}

LINEAR: greatest fixed point
Iterations:
0:{x,y,n}  1:{y,n}  2:{n}

BITVECTOR: one pass
“all variables in bitvector operations”: {x}
Implementation

• Prototype built on top of **clang**
• Flow-insensitive analysis  
  (analysis requires only the AST)
• Trade-off between cost and precision:
  – Interprocedural analysis
  – No pointer analysis implemented

• Systematic study of (syntactic) usage patterns of variables
Outline

1. Choice and Formalisation

2. Experimental Validation

3. Discussion: Uses of Variable Roles
Experiment

How to validate that our definition of variable roles is useful?

Opportunity:
• SVCOMP (Competition on Software Verification) contains files in different categories
• Files classified by human expert

Experiment: Can the relative frequencies of the variable roles replace the human expert in the classification of the files into competition categories?
Experiment: Results

• Multiclass vector support machine
• Output: probability of membership in category
• Random selection of training set

<table>
<thead>
<tr>
<th>Training set (% of all files)</th>
<th>Correct classification (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>1. probability</td>
</tr>
<tr>
<td></td>
<td>84.06</td>
</tr>
<tr>
<td>80</td>
<td>85.19</td>
</tr>
<tr>
<td>70</td>
<td>83.80</td>
</tr>
<tr>
<td>60</td>
<td>80.23</td>
</tr>
<tr>
<td>50</td>
<td>81.40</td>
</tr>
</tbody>
</table>
Outline

1. Choice and Formalisation

2. Experimental Validation

3. Discussion: Uses of Variable Roles
Variable Roles in Program Analysis

Reviewer: „How can variable roles help to avoid plane crashes?“

Many program analysis tools treat a program as a formula and program analysis as constraint solving → tools work the same for obfuscated code??

Our vision: variable roles enable a systematic study of heuristics in program analysis and help to understand the strength of program analysis tools
Envisioned Uses of Variable Roles

• Program analysis tools: selection of predicates or abstract domains *guided by variable roles* (e.g. in ASTREÉ)

• *Quantitative characteristics* on software verification benchmarks
  → Explaining the results

• Building a *portfolio-solver*
Conclusion

Variable Roles have predictive power.

Work in progress, your feedback is very welcome!

Future Work:
• Extract roles from variable names / comments
• Explore connection to types