Improving Precision of Program Analysis in the 2LS Framework

Author: Martin Smutny, xsmutn13@stud.fit.vutbr.cz
Supervisor: Ing. Viktor Malik

diffblue

Excel@FIT 2019

Motivation for This Work

2LS is a **verification framework** for analysing sequential **C programs**. Currenly, is usable for **analysis of numerical** and **dynamic variables**. Verification is based on **computing invariants** of source program by utilizing an **SMT solver**.

Due to **complexity** of computed invariants it is **hard** to identify parts of the original program that cause **undecidability of verification**.

We propose a solution, to **identify parts** of the original programs that cause problems to the verifier by **analysing computed loop invariants**.

Verification Example

Program written in C. Consists of two variables, one uinitialized and a simple loop.

Verification is inconclusive, user-specified *assertion* is *true* only if variable *y* is zero.

```
1  void main()
2  {
3      int x = 0;
4      int y;
5
6      while (y)
7      {
8          x++;
9      }
10
11     assert(x == 0);
12 }
```

Imprecise Variable Identification

So-called *inductive invariants* are computed in various *abstract domains* using *templates*. *Templates* reduce the invariant inference problem so it can be iteratively solved using *SMT solver*.

We are looking for template variables that have values representing the *top* value in their abstract domains.

Numerical variables: finite max. values of their types

Objects (static and dynamic): non-deterministic set of addresses

Interval domain

Analysis of numerical variables. Variable *x* and constant *d*:

$(x \leq d_1) \wedge (-x \leq d_2)$

Octagon domain

...

Heap domain

Analysis of objects on the heap, pointer *p*:

$p \rightarrow [\,obj_1 \mid obj_2\,]$

$p = \&obj_1 \vee p = \&obj_2$

Invariant Generator

SMT Solver

External solver, over CNF formula (translated SSA) using theory of bit-vectors.

Generated Invariant:

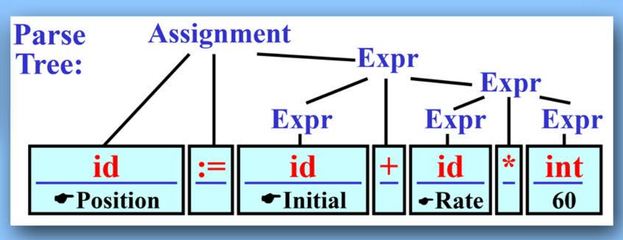
$(x \leq 2147483647) \wedge (-x \leq 2147483648)$

```
Variables:
1: x#lb5
---------------
1: Imprecise value of variable "x" at the end of the loop, that starts at line 6
```

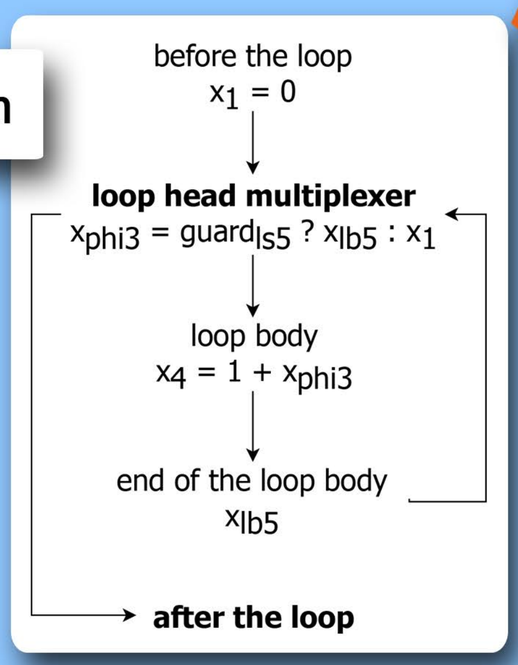
```
1  void main()
2  {
3      int x = 0;
4      int y;
5      assume(y == 0);
6
7      while (y)
8      {
9          x++;
10     }
11
12     assert(x == 0);
13 }
```

2LS Architecture

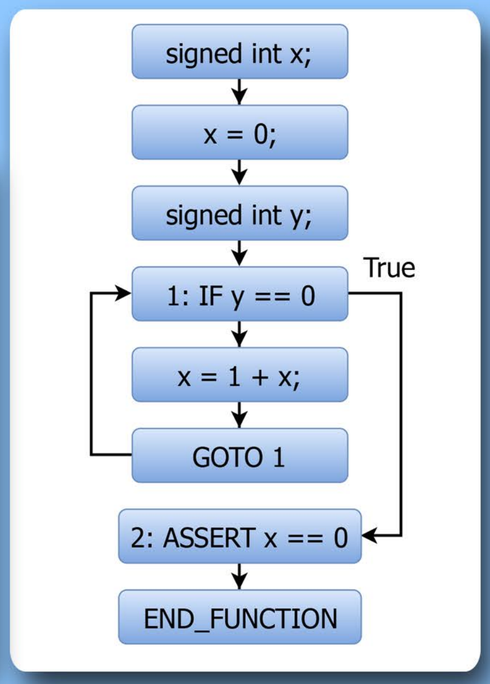
C Parser



SSA Form



GOTO Conversion



Intermediate program represenation using *GOTO programs*, which are control flow graphs. Various transformations, such as *function inlining* or *light-weight static analysis* to resolve function pointers, resulting in a *static call graph.*

Acyclic *single static assignment* form. Satisfies the property that each variable is assigned to only once. Cuts the loops at the end of the loop body and introduces *free variables*.

Property Checker

Failed | Unknown | Success

```
[main.assertion.1] assertion x == 0: OK

** 0 of 1 unknown
** 0 of 1 failed
VERIFICATION SUCCESSFUL
```