Eq-VIAP : Program Equivalence Checker

Pritom Rajkhowa

May 2019

1 Motivating Examples

1.1 Example 1

In this section, we illustrate our approach to program equivalence verification with a simple example. Consider the two different C code snippet(P) of Table- 3a and Table- 3b from benchmark of REVE [1] equivalence checker. Program P_1 has a set of variables $\vec{X} = \{x, g, i\}$ where and $\{x, g\}$ are input variables and it return x on its termination. Program P_2 also has same set of program variables $\vec{X} = \{x, g, i\}$ where and $\{x, g\}$ are input variables and similarly it return x on its termination.

To prove the equivalence between P_1 and P_2, VIAP first builds a sequential composition P' of P_1,P_2 as presented in Table- 4. All the variables in P_1 are renamed by adding $p1$ as prefix and variables in P_2 are renamed by adding $p2$ as prefix to avoid variable name conflict. VIAP then inserts the assumption and assertion that capture the partial equivalence of $P1, P2$: given the same inputs and same circumstance, they produce the same outputs.

Now if we prove that the composed program P' is safe, then it is proved that P_1 and P_2 are partially equivalent.

The translation module of VIAP implements Lin's [2] translation and generates the following set of axioms $\Pi^{\vec{X}}_{P'}$ for program P', where $\vec{X} = \{p1x, p1g, p1i$ $, p2x, p2g, p2i\}$:

`` int nested_while0(int x, int g) { int i = 0; while (i < x) { i = i + 1; g = g - 2; g = g + 1; while (x < i) { x = x + 2; x = x - 1; g = g + 1; } } return g; } ``` Table 1: Program P_1	``` int nested_while1(int x, int g) { int i = 0; while (i < x) { i = i + 1; g = g - 1; while (x < i) { x = x + 1; g = g + 1; } } return g; } ``` Table 2: Program P_2

Table 3: The program $P1, P2$ are taken from the benchmark [1]

```
int main()
{
     int p1x,p1g,p2x,p2g;
     int p1i,p1i
     p1i=0; p2i=0;

     assume(p1x==p2x && p1g==p2g)

     while (p1i < p1x) {

           p1i = p1i + 1;
           p1g = p1g - 2;
           p1g = p1g + 1;

           while (p1x < p1i) {

                  p1x = p1x + 2;
                  p1x = p1x - 1;
                  p1g = p1g + 1;

           }
     }

     while (p2i < p2x) {

         p2i = p2i + 1;
         p2g = p2g - 1;

         while (p2x < p2i) {


                p2x = p2x + 1;
                p2g = p2g + 1;

         }
    }

    assert(p1g==p2g)
}
```


Table 4: The program P' represents the sequential composition P_1, P_2

$$p2g_1 = p2g_{14}(N_4), p1g_1 = p1g_7(N_2), p1i_1 = (N_2 + 0), p2i_1 = (N_4 + 0)$$
$$p1x_1 = p1x_7(N_2), p2x_1 = p2x_{14}(N_4)$$
$$(N_1(n_2) \geq (n_2 + 1) - p1x_7(n_2))$$
$$(n_1 < N_1(n_2) \to (n_1 + p_1x_7(n_2) < n_2))$$
$$p1x_7(n_2 + 1) = (N_1(n_2) + p1x_7(n_2))$$
$$p1g_7(n_2 + 1) = (N_1(n_2) + (p1g_7(n_2) - 1)$$
$$p1x_7(0) = p1x, p1g_7(0) = p1g$$
$$(N_2 \geq p1x_7(N_2))$$
$$(n_2 < N_2 \to n_2 < p1x_7(n_2))$$
$$(N_3(n_4) \geq (n_4 + 1 - p2x_{14}(n_4)))$$
$$(n_3 < N_3(n_4) \to (n_3 + p2x_{14}(n_4) < (n_4 + 1))$$
$$p2g_{14}(n_4 + 1) = (N_3(n_4) + (p2g_{14}(n_4) - 1))$$
$$p2x_{14}(n_4 + 1) = (N_3(n_4) + p2x_{14}(n_4))$$
$$p2g_{14}(0) = p2g, p2x_{14}(0) = p2x$$
$$(N_4 \geq p2x_{14}(N_4))$$
$$(n_4 < N_4) \to (n_4 + 0 < p2x_{14}(n_4))$$

where $p1x_1$ denotes the output value of the program variable $p1x$ and $p1x_7(n_2)$ a temporary function denoting the value of $p1x$ after the n_2th iteration of the while loop. Similar conventions apply to $p1g, p1i, p2x, p2i, p2g$ and their subscripted versions. N_2, N_4 are a system-generated natural number constant denoting the number of iterations that the outer loop runs before exiting. N_1 is a system- generated natural number function denoting that for each `k`, $N_1(k)$ is the number of iterations that the inner loop runs during the kth iteration of the outer loop. Similar conventions apply to N_3. The variable n_1,n_2,n_3,n_4 ranges over natural numbers and is universally quantified.

The assertion to prove is as follows:

$$\alpha : p1g_7(N_2) = p2g_{14}(N_4) \tag{1}$$

The assumption is

$$\beta : p1x = p2x \wedge p1g = p2g \tag{2}$$

Eq-VIAP can then be made to successfully prove the assertion β with respect to $\Pi_{P'}^{\vec{X}} \wedge \alpha$.

References

[1] M. Kiefer, V. Klebanov, and M. Ulbrich, "Relational program reasoning using compiler ir," *Journal of Automated Reasoning*, vol. 60, no. 3, pp. 337–363, 2018.

[2] F. Lin, "A formalization of programs in first-order logic with a discrete linear order," *Artificial Intelligence*, vol. 235, pp. 1 – 25, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000437021630011X