STM Project Proposal

Hitarth
Bhishmaraj

MSc CS Students, CMI

October 24, 2019

1 Introduction

Srivas: Your motivation for problem should be made as follows:

1. Solver-aided programming languages and tools, such as Rosette (give ref to PLDI paper) help exploit power of SAT/SMT solvers in designing solutions to domain-specific constraint solving problems.

2. Most of the existing efforts have used SAT solvers have restricted to arithmetic and bit-vector theories

3. In this project we propose to use Rosette-Racket for analysis (verification and synthesis) of array manipulating programs.

4. Arrays pose a challenge because they are, in general, undecidable and most interesting properties about arrays require quantifier instantiation.

5. We simplify our problem by restricting ourselves to decidable fragments of the arrays theory (give ref of the paper I had uploaded) and the class of programs for analysis.

To makes it easier to automate the process of program verification, debugging and synthesis, we usually focus on a (small) set of programming constructs. This makes it possible to automatically encode the program in logical statements such that it's verification etc can be done.

For this project we have decided to focus on the simple programs which involve arrays. We will allow the basic operations like accessing and storing in an array.

2 Problem: Verification, Debugging and Synthesis in straight line array programs

Srivas: Please use the following array initializing program as he running example to define and illustrate problem statement. You can make suitable changes to the program to inject a bug and synthesize the bug fixes. Your program has hardly any meat in it.

Note: Although the example shows a loop, it doesn't mean you are supporting loop verification; implicit assumption here is that the user provides a loop invariantas illustrated in the next picture.

```
a: array 0..99 of integer;
i: integer;

for i:=0 to 99 do
        assert($\forall x \in \mathbb{N}_0 .\ x < i \implies a[x] = 0$);
        a[i]:=0;
        assert($\forall x \in \mathbb{N}_0 .\ x \leq i \implies a[x] = 0$);
done;
assert($\forall x \in \mathbb{N}_0 .\ x \leq 99 \implies a[x] = 0$);
```

$$\begin{aligned} & (\forall x \in \mathbb{N}_0 .\ x < i \implies \mathsf{a}[x] = 0) \\ & \wedge\ \mathsf{a}' = \mathsf{a}\{i \leftarrow 0\} \\ \implies & (\forall x \in \mathbb{N}_0 .\ x \leq i \implies \mathsf{a}'[x] = 0)\ . \end{aligned}$$

Consider a simple program as given below:

```
/*Program to set a value at a given index
if it is larger than existing value*/

void update_value(array, index, value){
    if array[index] < value {
        array[index] = value;
    }
}
```

In this program we are doing a simple operation. We know that the final postcondition would be: $(a[i] < v \implies a_1[i] == v \wedge (\forall j.j \neq i.a[j] = a_1[j])) \wedge (a[i] >= v \implies a_1 = a)$, where a and a_1 represent the array before and after the execution of the program respectively. Given the pre/post conditions, we can ask for the correctness of the program.

Usually while writing the programs involving arrays, we might make an error while accessing the arrays. This is exactly what we will be focussing on while synthesizing. We shall assume that the error is only on the accessing operation, and with the help of Rosette we will try to fix that.

Our main goals in this project are as follows:

Srivas: Organize the goals as follows:

1. Methodology: Develop a method of converting the problem of performing the following analysis of a restricted class of array manipulating programs into instances of SMT array logic.
    - Verifying an array manipulating program against its specification given as pre-post conditions and loop invariants (for program with loops).
    - Localizing the location of a bug when verification fails
    - Synthesizing a fix for the bug when there exists a fix within a restricted class of possible fixes
2. Implementation of our method within the Rosette-Racket solver-aided programming tool/language framework.
3. Experiment our implementation on a targeted class of benchmark examples.

1. To take a program and the pre and post conditions, and verify the correctness of the program using Rosette.
2. Given two programs, where one is assumed to be correct, verify the correctness of the other suspected program.
3. If the program is incorrect as per the either the conditions or the second program which is assumed to be correct, find the parts of the program which might be the cause of the error.
4. If the program is incorrect, then try to synthesize the correct program by only focussing on the bugs due to **array access operations**.

3 Approach

We will firstly define a basic language which only deals with arrays of integers and provides basic functionalities like accessing and storing in an array. Then we will embed this language with basic programming constructs for arrays in Racket/Rosette.

Rosette provides us with various tool to verify, debug and synthesize once we write the interpreter in it. We will then explore how we can automate this process and auto fix the bugs related to array accessing.

We will show with examples how we can use this for debugging and synthesis in such programs.

We also plan to create a seperate GUI application which can provide an easy to use interface to the user to get these functionalities.

4 Expected Results

We expect to have a system which can take simple array programs and help the user with verification, debugging and synthesis provided they give either pre and post conditions or another program which has the correct behaviour.

Almost all the programs use arrays in one way or another, but most of the useful programs involving arrays also have loops. At least for now, our plan is not to indulge with loops. We might consider the loops if we are successful in the first phase of our project.

References

1. Rosette Language guide - https://docs.racket-lang.org/rosette-guide/index.html