Progess Report

Presentor: Xie Li

August 3, 2021

Overview

- A simple example for symbolic execution.
- Modifying SMACK: symbolic execution for `malloc` and `bitcast` from pointer to pointer(with assumption).
- Proposed a framework based on problems encountered.

The Simplest Example

Here is the simplest example in C related to memory operations.

```c
int main(){
    int *i;
    i = (int*)malloc(sizeof(int));
    free(i);
}
```

And its corresponding LLVM IR:

```
define dso_local i32 @main() {
    %1 = call noalias i8* @malloc(i64 4)
    %2 = bitcast i8* %1 to i32*
    %3 = bitcast i32* %2 to i8*
    call void @free(i8* %3)
    ret i32 0
}
```

Symbolic Execution Rules

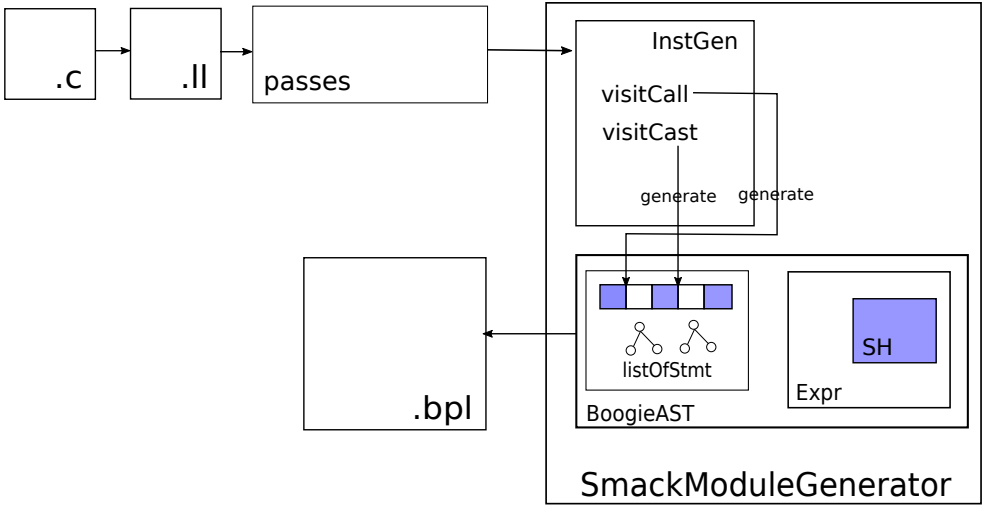
Symbolic Heap: $Q \equiv \Pi \mid \Sigma$

$$\Pi \mid \Sigma \Rightarrow_{\{x=\texttt{malloc}(e)\}} \Pi[x'/x] \mid \Sigma[x'/x] * x \mapsto e[x'/x] * \texttt{blk}(x+1, x+1+e[x'/x])$$

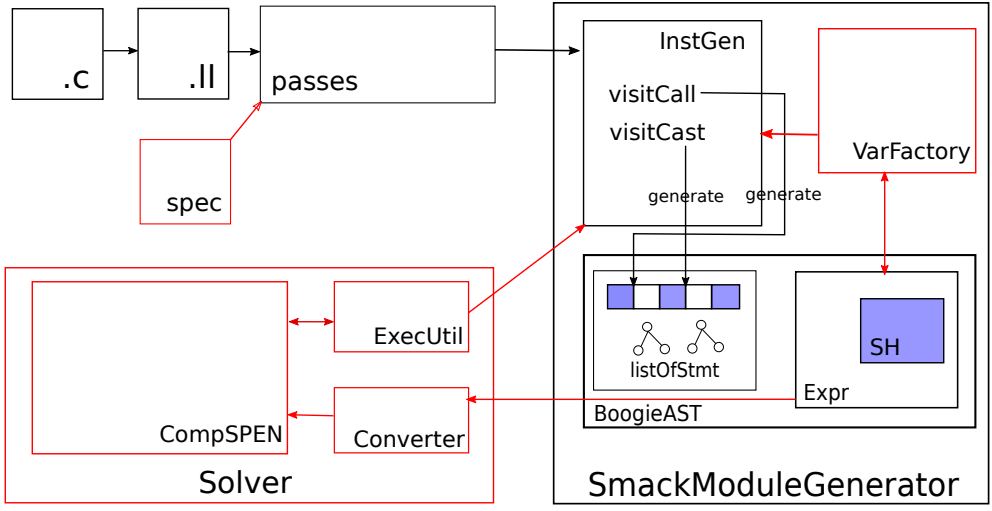
$$\Pi \mid \Sigma_0 * x \mapsto e * \texttt{blk}(x+1, x+1+e) * \Sigma_2 \Rightarrow_{\{\texttt{free}(x)\}} \Pi \mid \Sigma_0 * \Sigma_2$$

```
procedure {:entrypoint} main()
  returns ($r: i32)
{
  var $p0: ref;
  var $p1: ref;
  var $p2: ref;
$bb0:
  SymbHeap(true|emp)
  call {:cexpr "smack:entry:main"} boogie_si_record_ref(main);
  call $p0 := malloc(4);
  SymbHeap(true|emp # $p0 >--> 4 # Blk(($p0 + 1), (($p0 + 1) + 4)))
  $p1 := $bitcast.ref.ref($p0);
  SymbHeap(true|emp # $p0 >--> 4 # Blk(($p0 + 1), (($p0 + 1) + 4)))
  $p2 := $bitcast.ref.ref($p1);
  call free_($p2);
  $r := 0;
  return;
}
```

Framework


.c
.ll
passes
InstGen
visitCall
visitCast
generate
generate
.bpl
listOfStmt
SH
Expr
BoogieAST
SmackModuleGenerator

Framework


.c
.ll
passes
spec
InstGen
visitCall
visitCast
generate
generate
VarFactory
ExecUtil
CompSPEN
Converter
Solver
listOfStmt
SH
Expr
BoogieAST
SmackModuleGenerator

TODOs

- Translation: Incrementally investigate the instruction for execution. Thoroughly went through `InstGen` and find data structure available for the symbolic execution.
- AST: Implement a `VarFactory` module for better syntactical variable manipulation, extend `BoogieAST` to get more information and unify the basic memory unit.
- VCGen: Add `Solver` component to assist symbolic execution and together with `spec` for verification condition generation.

Current Problems and Needs

- Symbolic Execution: Directly using Z3/CompSPEN or use `BoogieAST`? Concentrate more on syntax or semantic? The problem of `free`.
- The conversion between different types of basic data structure: intermediate language unifying the basic unit of memory.
- Feasibility of current framework.