Feb 17th Report

By Xiang Chen (Echo)

Feb 17, 2020

Contents

1 Fix initialized array variable bugs

I fix the small errors about initialized array variable.

Below is the example for Integer Arrays:

Type Checker:

```java
public void visitIntArrayVar(IntArrayVar a) {
    // uninitialized declaration
    // e.g. a : ARRAY[INTEGER]

    if (a.mode instanceof modes.UninitializedDecl) {
      // if this variable is declared for the first time, simply add it to the map
      if (!varMap.containsKey(a.name)) {
        varMap.put(a.name, new Pair<VarType, Verifier>(new IntArray(), null));
      }
      // if this variable is not declared for the first time, change its type to
         unknown type
      // and add the error message
      else {
        varMap.replace(a.name, new Pair<VarType, Verifier>(new UnknowType(), null));
        errormsg.add("Error: Type declaration of variable " + a.name + " is ambigous
            . "
            + "Please make sure each variable is declared exactly once.");
      }
    }
    // verification
    // verify a[1]
    else if (a.mode instanceof modes.Verification) {

      // type check this arithmetic variable's index first
      TypeChecker checker = new TypeChecker();
      a.index.accept(checker);

      InfixPrinter infixPrinter = new InfixPrinter();
      a.index.accept(infixPrinter);

      // e.g. a[1], also need to store its name and type to the varMap
      String arrayElement = a.name + "[" + infixPrinter.infixOutput + "]";

      // check if the type of the index is integer type
      // e.g. error: verify a[2.1 * 2]
      if (!(varMap.containsKey(infixPrinter.infixOutput))) {
        errormsg.add("Error: Cannot recognize " + infixPrinter.infixOutput + ".");
      }
      else if (!(varMap.get(infixPrinter.infixOutput).a instanceof types.IntType)) {
        errormsg.add(infixPrinter.infixOutput + " is not integer type, cannot use it
             as array index value.");
      }

      if (!varMap.containsKey(a.name)) {
        errormsg.add("Error: variable " + a.name + " has not been declared.");
      }
      // if it has unknown type
      else if (varMap.containsKey(a.name) && (varMap.get(a.name).a instanceof types.
         UnknowType)) {
        errormsg.add("Error: Type of variable " + a.name + " in this expression is
            ambigous. "
            + "Please make sure each variable is declared exactly once.");
      }
      // if it's not declared as integer array type
      else if (varMap.containsKey(a.name) && !(varMap.get(a.name).a instanceof types.IntArray)) {
        errormsg.add("Error: variable " + a.name + " is not declared as a integer array.");
      }


      // if there is no error, check the map first
      else if (checker.errormsg.isEmpty()) {
        if (!varMap.containsKey(arrayElement)) {
          varMap.put(arrayElement, new Pair<VarType, Verifier>(new IntType(), null));
        }
        else if (varMap.containsKey(a.name) && (varMap.get(a.name).a instanceof types.UnknowType)) {
          varMap.replace(a.name, new Pair<VarType, Verifier>(new UnknowType(), null));
          errormsg.add("Error: Type declaration of variable " + a.name + " is ambigous. "
              + "Please make sure each variable is declared exactly once.");
        }
      }else {
        errormsg.addAll(checker.errormsg);
      }
    }
    // initialized declaration
    // e.g. a : ARRAY[INTEGER] = << 1, 2, 6, 0 >>
    else if (a.mode instanceof modes.InitializedDecl) {
      // type check its elements first
      for (int i = 0; i < a.arrayValue.size(); i++) {
        TypeChecker checker = new TypeChecker();
        a.arrayValue.get(i).accept(checker);
        errormsg.addAll(checker.errormsg);

        InfixPrinter printer = new InfixPrinter();
        a.arrayValue.get(i).accept(printer);

        if (!(varMap.containsKey(printer.infixOutput))) {
          errormsg.add("Error: Cannot recognize " + printer.infixOutput + ".");
        }
        else if (varMap.get(printer.infixOutput).a instanceof types.RealType) {
          errormsg.add(printer.infixOutput + " is not integer type, cannot perform this assignment.");
        }
      }
      if (errormsg.isEmpty()) {
        // if this variable is declared for the first time, simply add it to the map
        if (!varMap.containsKey(a.name)) {
          varMap.put(a.name, new Pair<VarType, Verifier>(new IntArray(), null));
        }
        // if this variable is not declared for the first time, change its type to unknown type
        // and add the error message
        else {
          varMap.replace(a.name, new Pair<VarType, Verifier>(new UnknowType(), null));
          errormsg.add("Error: Type declaration of variable " + a.name + " is ambigous. "
              + "Please make sure each variable is declared exactly once.");
        }
      }
    }
  }
```

PrettyPrinter:

```
@Override
  public void visitIntArrayVar(IntArrayVar a) {
    // uninitialized declaration
    // e.g. a : ARRAY[INTEGER]
    if(a.mode instanceof modes.UninitializedDecl) {
      // use the PrefixPrinter to return the output
      PrefixPrinter p = new PrefixPrinter();
      a.accept(p);
    }
    // verification
    else if (a.mode instanceof modes.Verification) {
      printOtherExpr(a);
    }
    // initialized declaration
    // e.g. a : ARRAY[INTEGER] = <<1, 2, 6, 9>>
    else if (a.mode instanceof modes.InitializedDecl) {
      // use the PrefixPrinter to return the output
      PrefixPrinter p = new PrefixPrinter();
      a.accept(p);
    }
  }
```

2 Regression Tests

I also fix the problem that some of the regression tests do not pass after I added the REAL type.

The reason is because **HashMap** does not have the fixed order, when I try to access the Map, it is not guarantee that the order is the same as the order I put the elements into the Map.

I simply change it to **LinkedHashMap**, then the problem solved.

I also added 50 test cases for arrays.