

Feb 17th Report

By Xiang Chen (Echo)

Feb 17, 2020

Contents

1	Fix initialized array variable bugs	2
2	Regression Tests	4

1 Fix initialized array variable bugs

I fix the small errors about initialized array variable.

Below is the example for Integer Arrays:

Type Checker:

```
1 public void visitIntArrayVar(IntArrayVar a) {
2     // uninitialized declaration
3     // e.g. a : ARRAY[INTEGER]
4
5     if (a.mode instanceof modes.UninitializedDecl) {
6         // if this variable is declared for the first time, simply add it to the map
7         if (!varMap.containsKey(a.name)) {
8             varMap.put(a.name, new Pair<VarType, Verifier>(new IntArray(), null));
9         }
10        // if this variable is not declared for the first time, change its type to
11        // unknown type
12        // and add the error message
13        else {
14            varMap.replace(a.name, new Pair<VarType, Verifier>(new UnknowType(), null));
15            errorMsg.add("Error: Type declaration of variable " + a.name + " is ambiguous
16                "
17                + "Please make sure each variable is declared exactly once.");
18        }
19    }
20    // verification
21    // verify a[1]
22    else if (a.mode instanceof modes.Verification) {
23        // type check this arithmetic variable's index first
24        TypeChecker checker = new TypeChecker();
25        a.index.accept(checker);
26
27        InfixPrinter infixPrinter = new InfixPrinter();
28        a.index.accept(infixPrinter);
29
30        // e.g. a[1], also need to store its name and type to the varMap
31        String arrayElement = a.name + "[" + infixPrinter.infixOutput + "]";
32
33        // check if the type of the index is integer type
34        // e.g. error: verify a[2.1 * 2]
35        if (!(varMap.containsKey(infixPrinter.infixOutput))) {
36            errorMsg.add("Error: Cannot recognize " + infixPrinter.infixOutput + ".");
37        }
38        else if (!(varMap.get(infixPrinter.infixOutput).a instanceof types.IntType)) {
39            errorMsg.add(infixPrinter.infixOutput + " is not integer type, cannot use it
40                as array index value.");
41        }
42
43        if (!varMap.containsKey(a.name)) {
44            errorMsg.add("Error: variable " + a.name + " has not been declared.");
45        }
46        // if it has unknown type
47        else if (varMap.containsKey(a.name) && (varMap.get(a.name).a instanceof types.
48            UnknowType)) {
49            errorMsg.add("Error: Type of variable " + a.name + " in this expression is
50                ambiguous. "
51                + "Please make sure each variable is declared exactly once.");
```

```

48     }
49     // if it's not declared as integer array type
50     else if (varMap.containsKey(a.name) && !(varMap.get(a.name).a instanceof types
51         .IntArray)) {
52         errorMsg.add("Error: variable " + a.name + " is not declared as a integer
53             array.");
54     }
55
56     // if there is no error, check the map first
57     else if (checker.errorMsg.isEmpty()) {
58         if (!varMap.containsKey(arrayElement)) {
59             varMap.put(arrayElement, new Pair<VarType, Verifier>(new IntType(), null))
60             ;
61         }
62         else if (varMap.containsKey(a.name) && (varMap.get(a.name).a instanceof
63             types.UnknowType)) {
64             varMap.replace(a.name, new Pair<VarType, Verifier>(new UnknowType(), null)
65             );
66             errorMsg.add("Error: Type declaration of variable " + a.name + " is
67                 ambiguous. "
68                 + "Please make sure each variable is declared exactly once.");
69         }
70     } else {
71         errorMsg.addAll(checker.errorMsg);
72     }
73 }
74 // initialized declaration
75 // e.g. a : ARRAY[INTEGER] = << 1, 2, 6, 0 >>
76 else if (a.mode instanceof modes.InitializedDecl) {
77     // type check its elements first
78     for (int i = 0; i < a.arrayValue.size(); i++) {
79         TypeChecker checker = new TypeChecker();
80         a.arrayValue.get(i).accept(checker);
81         errorMsg.addAll(checker.errorMsg);
82
83         InfixPrinter printer = new InfixPrinter();
84         a.arrayValue.get(i).accept(printer);
85
86         if (!(varMap.containsKey(printer.infixOutput))) {
87             errorMsg.add("Error: Cannot recognize " + printer.infixOutput + ".");
88         }
89         else if (varMap.get(printer.infixOutput).a instanceof types.RealType) {
90             errorMsg.add(printer.infixOutput + " is not integer type, cannot perform
91                 this assignment.");
92         }
93     }
94 }
95 if (errorMsg.isEmpty()) {
96     // if this variable is declared for the first time, simply add it to the map
97     if (!varMap.containsKey(a.name)) {
98         varMap.put(a.name, new Pair<VarType, Verifier>(new IntArray(), null));
99     }
100    // if this variable is not declared for the first time, change its type to
101    // unknown type
102    // and add the error message
103    else {
104        varMap.replace(a.name, new Pair<VarType, Verifier>(new UnknowType(), null)
105        );
106        errorMsg.add("Error: Type declaration of variable " + a.name + " is
107            ambiguous. "

```

```

98         + "Please make sure each variable is declared exactly once.");
99     }
100 }
101 }
102 }

```

PrettyPrinter:

```

1  @Override
2  public void visitIntArrayVar(IntArrayVar a) {
3      // uninitialized declaration
4      // e.g. a : ARRAY[INTEGER]
5      if(a.mode instanceof modes.UninitializedDecl) {
6          // use the PrefixPrinter to return the output
7          PrefixPrinter p = new PrefixPrinter();
8          a.accept(p);
9      }
10     // verification
11     else if (a.mode instanceof modes.Verification) {
12         printOtherExpr(a);
13     }
14     // initialized declaration
15     // e.g. a : ARRAY[INTEGER] = <<1, 2, 6, 9>>
16     else if (a.mode instanceof modes.InitializedDecl) {
17         // use the PrefixPrinter to return the output
18         PrefixPrinter p = new PrefixPrinter();
19         a.accept(p);
20     }
21 }

```

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.