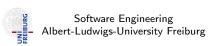
Formal Methods for Java

Lecture 9: How ESC works

Jochen Hoenicke



May 29, 2017

openjml (esc)

- Developed by the DEC Software Research Center (now HP Research),
- Extended by David Cok and Joe Kiniry (Kind Software)
- Rewritten in OpenJML by David Cok
- Proves correctness of specification,
- Is neither sound nor complete (but this will improve),
- Is useful to find bugs.

Assume and Assert

The basic specifications in ESC are assume and assert.

```
/*@ assume this.next != null; @*/
this.next.prev = this;
/*@ assert this.next.prev == this; @*/
```

- ESCJava proves that if the assumption holds in the pre-state, the assertion holds in the post-state.
- This is a Hoare triple.

Assume is Considered Harmful

Never assume something wrong. This enables ESC to prove everything:

```
Object[] a = new String[-5];
a[-3] = new Integer(2);
> escjava2 -q AssumeFalseTest.java
0 warnings
```

 $Object \ o = null;$

/*@ assume o != null; @*/

Requires and Ensures

```
The method specification is just translated into assume and assert:
    /*@ requires n > 0;
    @ ensures \result == (int) Math.sqrt(n);
    @*/
    int m() {
        ...
        return x;
    }
is treated as:
    /*@ assume n > 0; @*/
    ...
    /*@ assert x == (int) Math.sqrt(n); @*/
```

Calling Methods

. . .

And if m() is called the assumption and assertion is the other way round:

```
y = m(x);
...
is treated as
...
/*@ assert x > 0; @*/
y = m(x);
/*@ assume y == (int) Math.sqrt(x); @*/
...
```

Checking for Exceptions

```
To check for run-time exceptions ESC automatically inserts asserts: a[x] = "Hello"; is treated as: 

/*@ assert a != null & x >= 0 & x < a.length
@ && \typeof("Hello") <: \elemtype(\typeof(a));
@*/
a[x] = "Hello";
```

Loops in ESC

```
int a[] = new int[6];
for (int i = 0; i <= 6; i++) {
   a[i] = i;
}</pre>
```

Test.java:5: warning: The prover cannot establish an assertion (PossiblyNegativeIndex) in method test

Test.java:5: warning: The prover cannot establish an assertion (PossiblyTooLargeIndex) in method test

```
a[i] = i;
```

Adding Loop Invariant

```
int a[] = new int[6];
/*@ loop_invariant i >= 0; @*/
for (int i = 0; i <= 6; i++) {
    a[i] = i;
}</pre>
```

This is a bug in the code!

Checking Loop Invariant

```
int a[] = new int[6];
/*@ loop_invariant i >= 0; @*/
for (int i = 0; i <= 6; i++) {
    a[i] = i;
}</pre>
```

• Loop invariant holds initially:

```
int a[] = new int[6];
int i = 0;
/*@ assert i >= 0; @*/
```

• Loop invariant preserved by loop body:

```
/*@ assume i >= 0; @*/
if (i <= 6) {
   a[i] = i;
   i++;
   /*@ assert i >= 0; @*/
}
```

Checking Loop Invariant (2)

Internally, ESC checks this code.

```
/*@ assume precondition; @*/
int a[] = new int[6];
int i = 0:
/*@ assert i \geq 0; @*/ // check loop invariant initially
i = * // assign random values to all
a[*] = * // variables written in the loop
/*@ assume i \geq 0; @*/ // assume loop invariant
if (i \le 6) { // rewrite loop as if condition
 /*@ assert a != null && i >= 0 && i < a.length <math>@*/
 a[i] = i;
 i++:
 /*@ assert i >= 0; @*/ // check loop invariant after loop
 /*@ assume false; @*/ // don't check anything after this point
/*@ assert postcondition; @*/
```

ESC is Not Complete

```
ESC can only do limited reasoning:
  /*@ requires i == 5 & j== 3;
  @ ensures \result == 15;
  @*/
  int m(int i, int j) {
    return i*j;
  }
An error while executing a proof script for m:
(error "line 376 column 268: logic does not support nonlinear arithmetic")
```