COMPAQ

Extended Static Checking for Java

- ESC/Java finds common errors in Java programs:

  null dereferences, array index bounds errors, type cast errors, race conditions, violations of annotations (e.g., preconditions, object invariants), etc.

- Annotation language is a subset of JML
- Powered by program verification technology
- Has been applied to 10,000's of lines of Java and has found real errors

http://research.compaq.com/SRC/esc/

COMPAQ

A simple example input: Output:

```
1: class Bag {
2:   int[ ] a;  //@ invariant a != null;
3:   int n;  //@ invariant 0 <= n && n <= a.length;
     …
215:  int min( ) {
216:    int m = Integer.MAX_VALUE;
217:    for (int i = 0; i <= n; i++) {
218:      if (a[i] < m) {
219:        m = a[i];
220:    } }
221:    return m;
222:  }
223: }
```

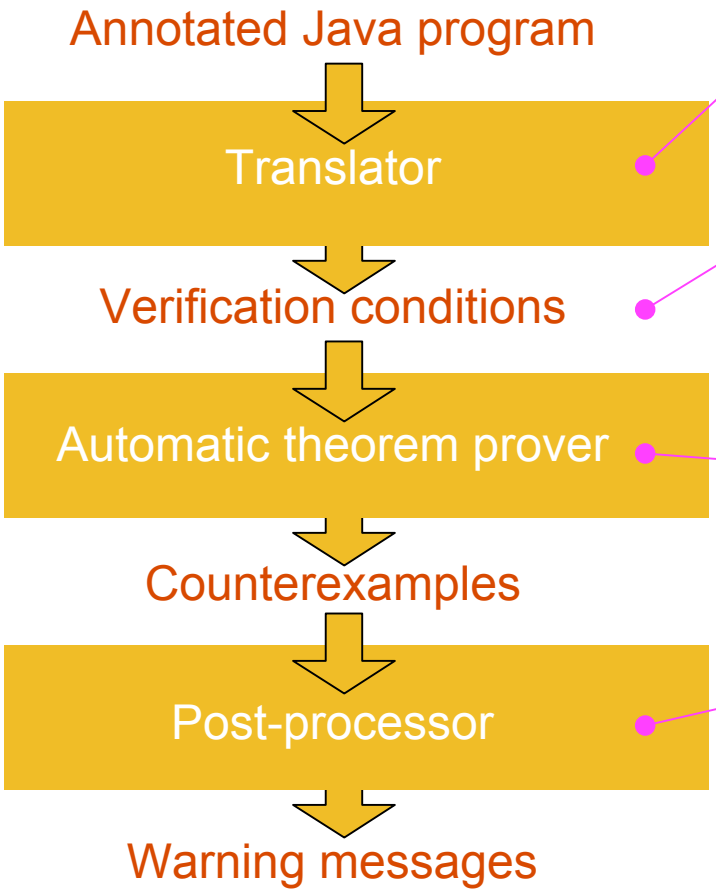
```
Bag.java:218: Warning: Array index possibly too large
    if (a[i] < m) {
        ^
Execution trace information:
  Reached top of loop after 0 iterations, line 217, col 4.
```

ESC/Java annotations are given in Java comments

This error may lead to an array index bounds error here, as detected by ESC/Java

COMPAQ

ESC/Java tool architecture



The translator “understands” the semantics of Java.

A *verification condition* is a logical formula that, ideally, is valid if and only if the program is free of the kinds of error under consideration.

The automatic theorem prover is invisible to users of ESC/Java.

Any counterexample that the theorem prover finds to a verification condition is turned into a precise warning message that ESC/Java outputs to the user.