Extended Static Checking for Java

K. Rustan M. Leino
DIGITAL SRC

Joint work with: Cormac Flanagan, Mark Lillibridge, Greg Nelson, James B. Saxe, Raymie Stata

Hopper et al.'s visit, WRL, 17 Mar. 1998

Testing vs. Verification

- Testing has proven more cost-effective than formal verification
- Many errors are not found by testing
- Those that are, are found late
- Test cases vs. specifications

Extended Static Checking

- Statically detect certain common run-time errors
- Use formal methods
  - for limited checking

What is Extended Static Checking?

$$\begin{tikzcd}
\text{Annotated Java program} \arrow[r] & \boxed{\text{ESC/Java}} \arrow[r] & \text{“null-dereference error on line 218”}
\end{tikzcd}$$

Checks for:

- null-dereference errors
- array bounds errors
- type cast errors
- race conditions
- dead locks
- ...

How ESC/Java works

$$\begin{tikzcd}
\text{Annotated Java} \arrow[d] \\
\boxed{\text{Java-to-V.C. compiler}} \arrow[d] \\
\text{Verification condition (logical formula)} \arrow[d] \\
\boxed{\text{Theorem prover}} \arrow[d] \\
\text{Error message}
\end{tikzcd}$$

ESC/Java Goal

Deploy ESC technology in checker that lay programmers are eager to use.

```
class C {
  int[]  a;
  int  n;

  C( int[]  input)
  {
     n = input.length ;
     a = new  int[n] ;
     System.arraycopy(input,0,a,0,n);
  }
```

```
class C {
  int[]  a;
  int   n;

  C( int[]  input)
  {
     n = input.length ;  null-dereference
     a = new  int[n] ;
     System.arraycopy(input,0,a,0,n);
  }
```

```
class C {
  int[]  a;
  int   n;

  C( int[]  input) /*@ requires input != null */
  {
      n = input.length;
      a = new  int[n] ;
      System.arraycopy(input,0,a,0,n);
  }
```

```
int extractMin()   {
   int m = Integer.MAX_VALUE;
   int mi = 0;
   for (int i = 0;   i < n;   i++) {
         if ( a[i] < m)  {
              mi = i;
              m = a[i];
   }     }
   if  (n != 0) {
         n--;
         a[mi] = a[n];
   }
   return m;
}
```

```
int extractMin()   {
   int m = Integer.MAX_VALUE;
   int mi = 0;
   for (int i = 0;   i < n;   i++) {
         if ( a[i] < m)  {  null-dereference
               mi = i ;
               m = a[i];
   }     }
   if  (n != 0) {
         n--;
         a[mi] = a[n];
   }
   return m;
}
```

```
class C {
   int[]  a; /*@ invariant  a != null ;*/
   int  n;

   C( int[]  input)
   {
       n = input.length ;
       a = new  int[n] ;
       System.arraycopy(input,0,a,0,n);
   }
```

```
int extractMin()   {
   int m = Integer.MAX_VALUE;
   int mi = 0;
   for (int i = 0;   i < n;   i++) {
         if ( a[i] < m) {  array index
              mi = i;       out of bounds
              m = a[i];
   }     }
   if (n != 0) {
         n--;
         a[mi] = a[n];  array index
   }                   out of bounds
   return m;
}
```

```
class C {
  int[]  a; /*@ invariant a != null; */
  int  n;   /*@ invariant 0 <= n && n <= a.length; */

  C( int[]  input)
  {
      n = input.length;
      a = new  int[n];
      System.arraycopy(input,0,a,0,n);
  }
```

Modular Checking

- Run checker in context of one class
- No global program information

Soundness and Completeness

A matter of degree

Complicated and expensive, because

- modular checking
- properties of arithmetic and floats
- complicated invariants and data structures
- ...

Our decisions based on

- ESC/Modula-3 experience
- guessing

and may change over time.

Summary and Status

- ESC/Modula-3 experience encouraging
- ESC/Java focus on usability
  - catch bugs
  - give up soundness & completeness
- Internal ESC/Java by summer

http://www.research.digital.com/SRC/esc/Esc.html