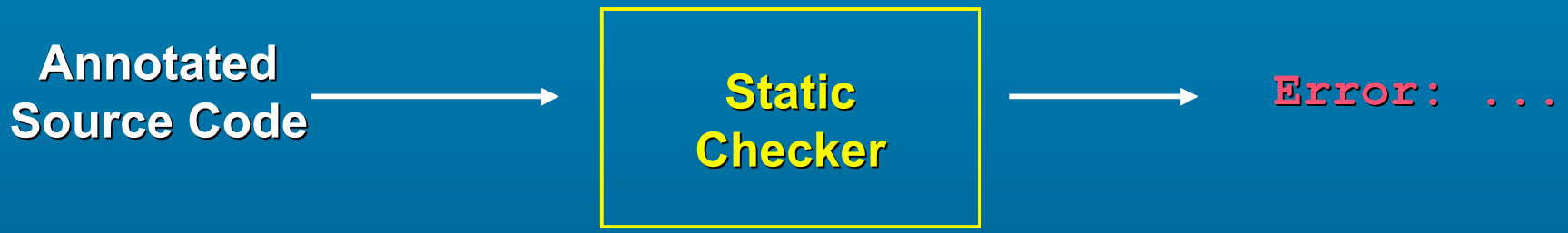
Extended Static Checking for Java

Cormac Flanagan

Joint work with: Rustan Leino, Mark Lillibridge, Greg Nelson, Jim Saxe, and Raymie Stata

Compaq Systems Research Center

What is “Static Checking”?



- type systems

```
Error: wrong number of arguments in method call
```

- lint

```
Error: unreachable code
```

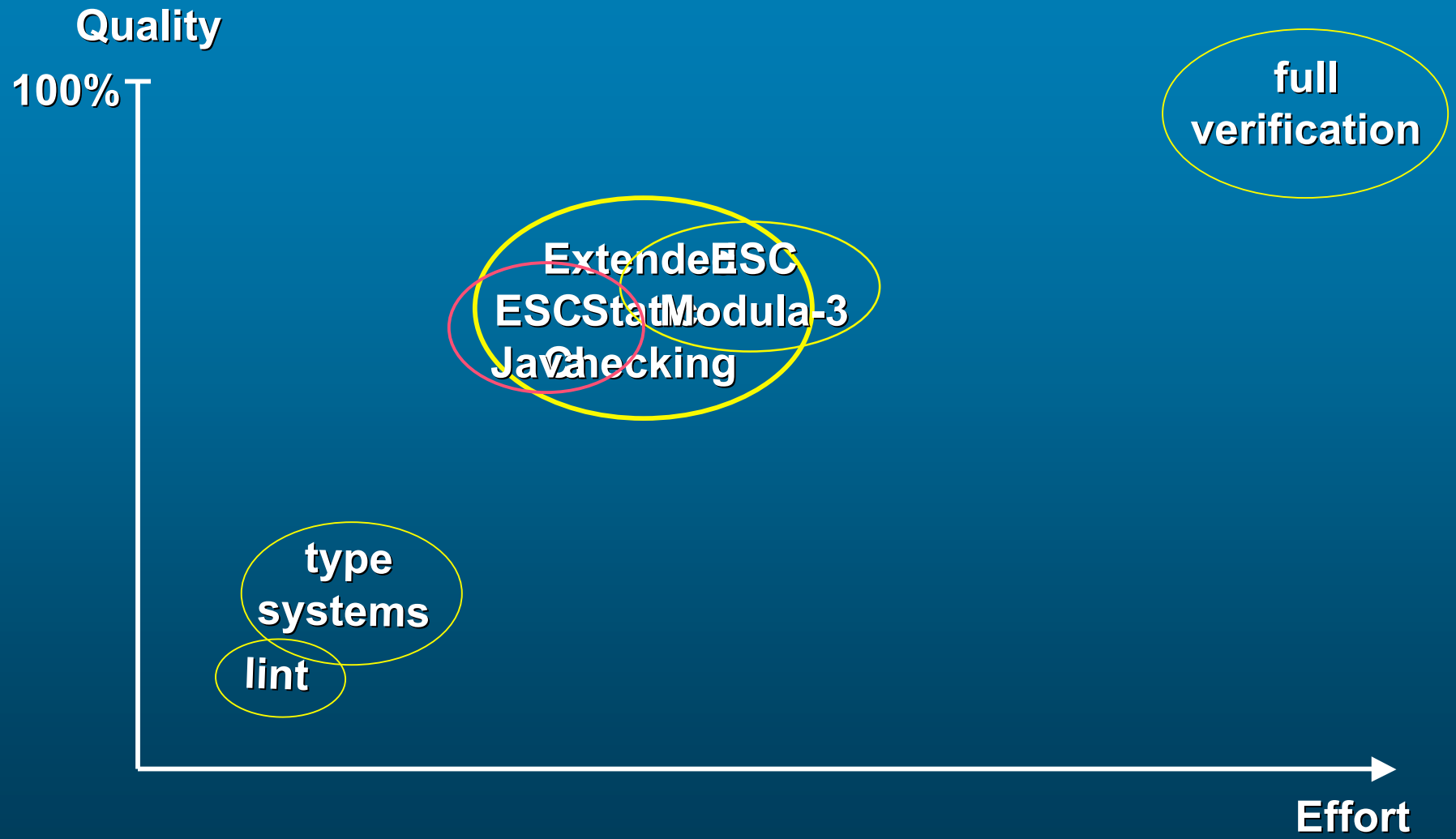
- full program verification

```
Error: qsort does not yield a sorted array
```

Why not just use testing?

- Testing essential but
  - Expensive
  - Finds errors late
  - Misses errors
- Static checking and testing complementary

Comparison of Static Checkers



Note: Graph is not to scale

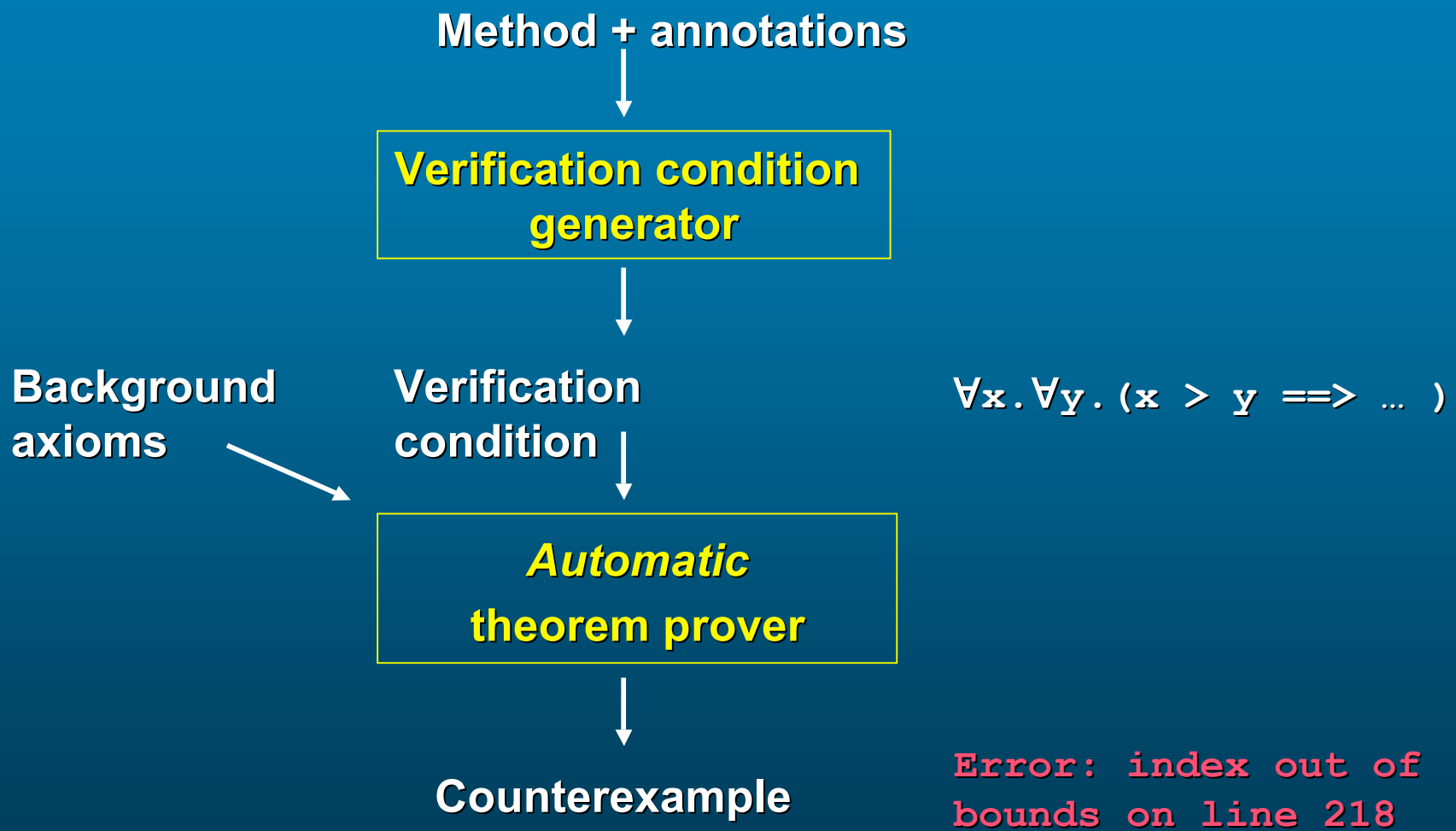
Goals of ESC/Java

- Practical static checking
- Detect common run-time errors
  - null dereferences
  - array bounds
  - type casts
  - race conditions
  - deadlocks
  - ...
- Modular checking

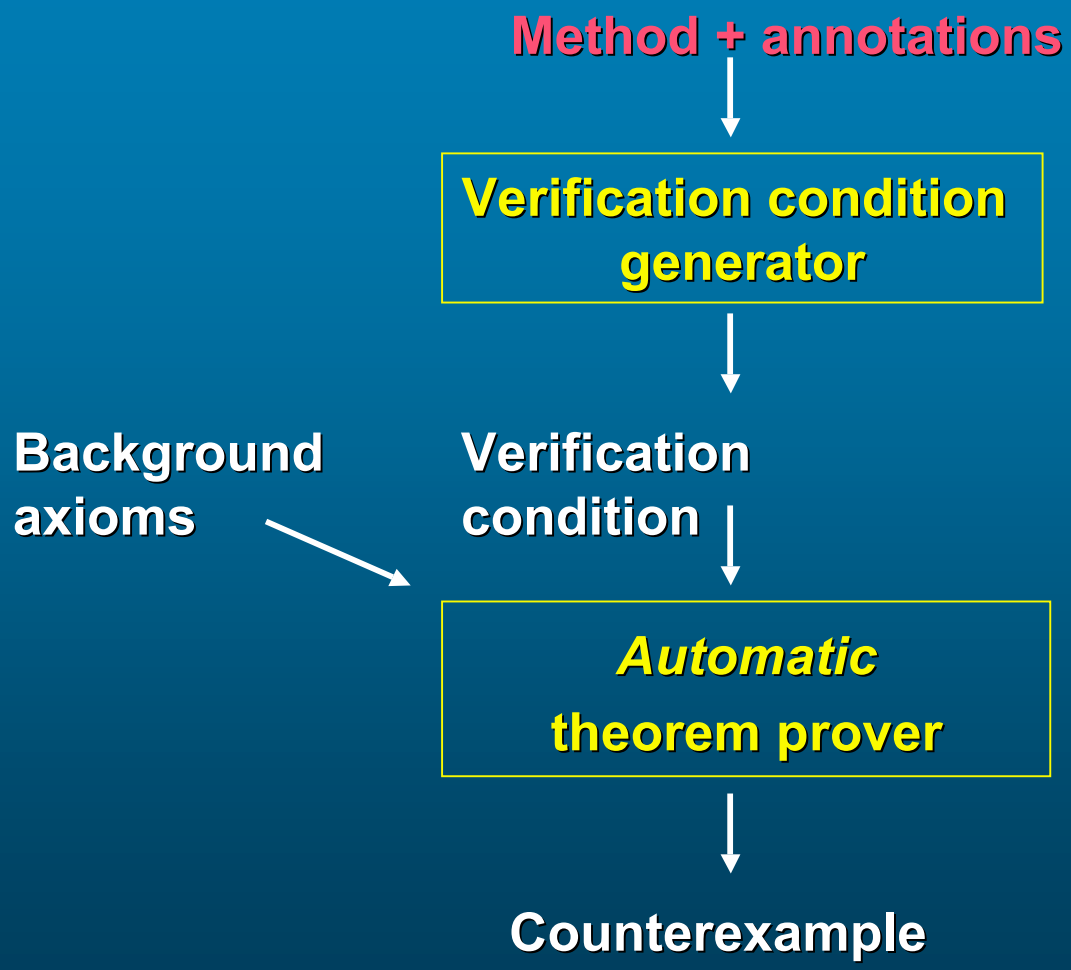
Non-goals of ESC/Java

- Complete functional verification
- Completeness
  - May not pass all programs
- Soundness
  - May fail to detect errors
  - Error-resistant, not error-proof

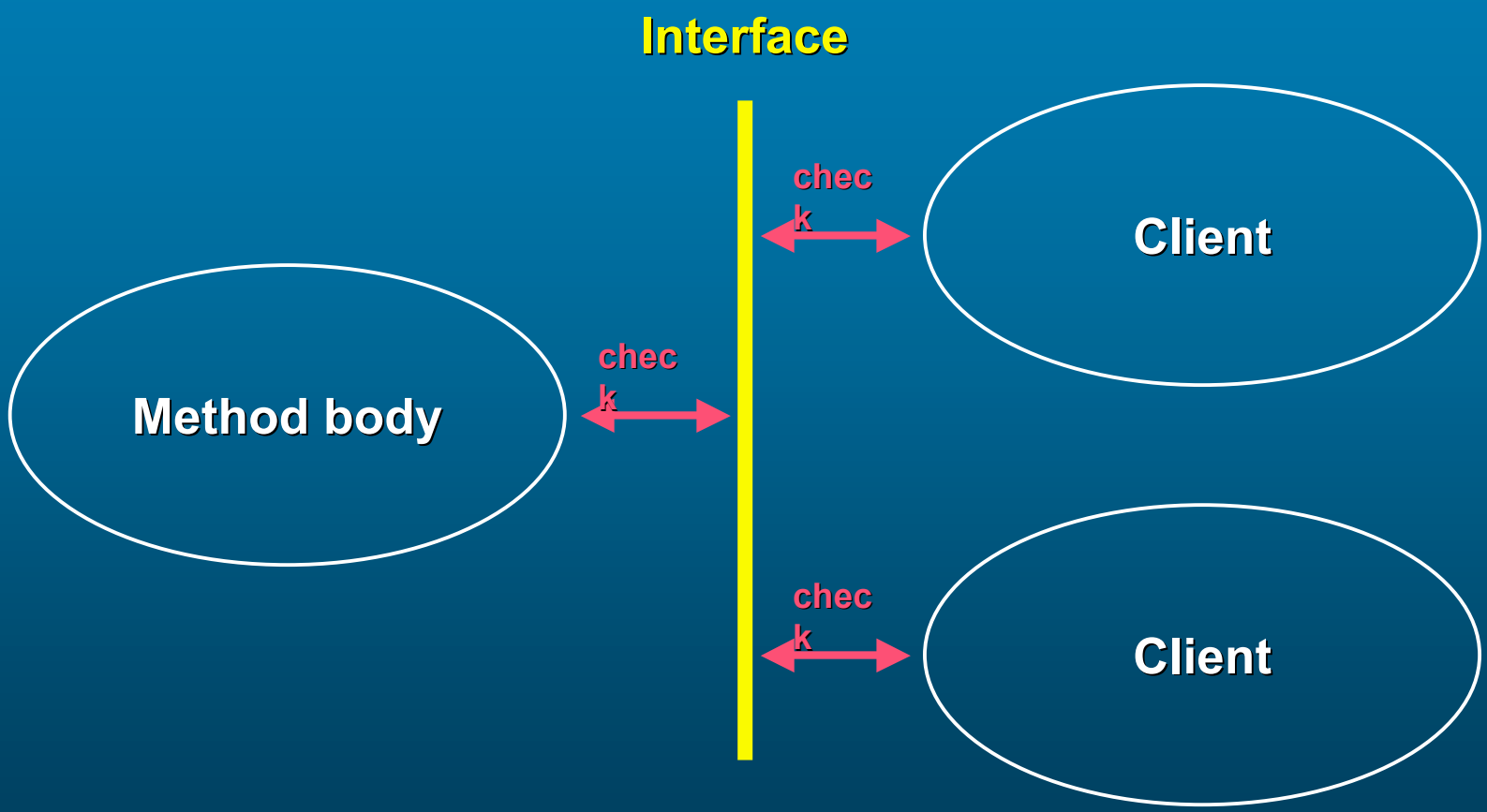
Architecture of ESC/Java



Input to ESC/Java



Modular checking



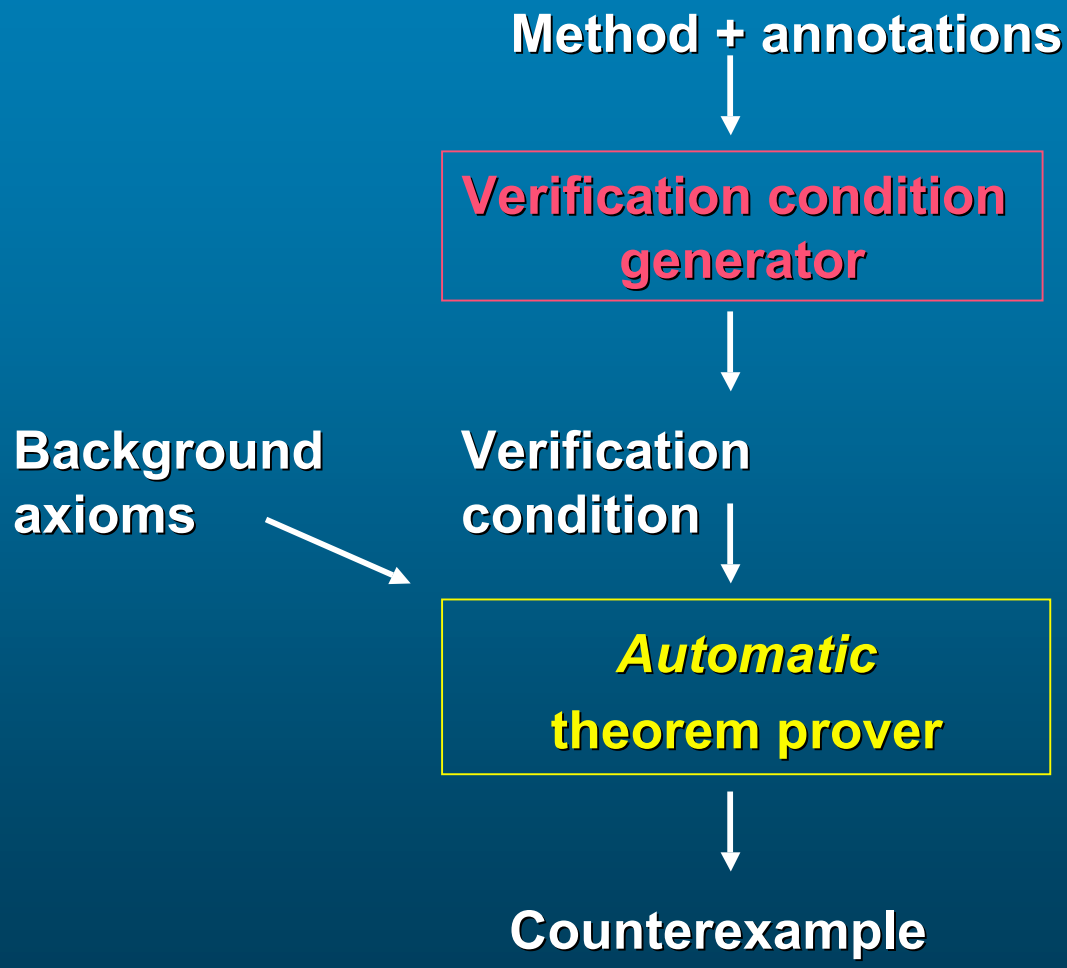
Describing interfaces

```
public class Vector {

    Object[] a;
    //@ invariant a != null
    int size;
    //@ invariant size <= a.length

    public Object elementAt(int i)
    //@ requires 0 <= i && i < size
    { ... }

    public Object[] copyToArray()
    //@ ensures RES != null && RES.length == size
    //@ modifies size, a[0], a[*]
    { ... }
}
```

Input to ESC/Java's "checking engine"

- Method implementation
- Interface annotations
  - `requires`
  - `ensures`
  - `modifies`
  - `invariants`

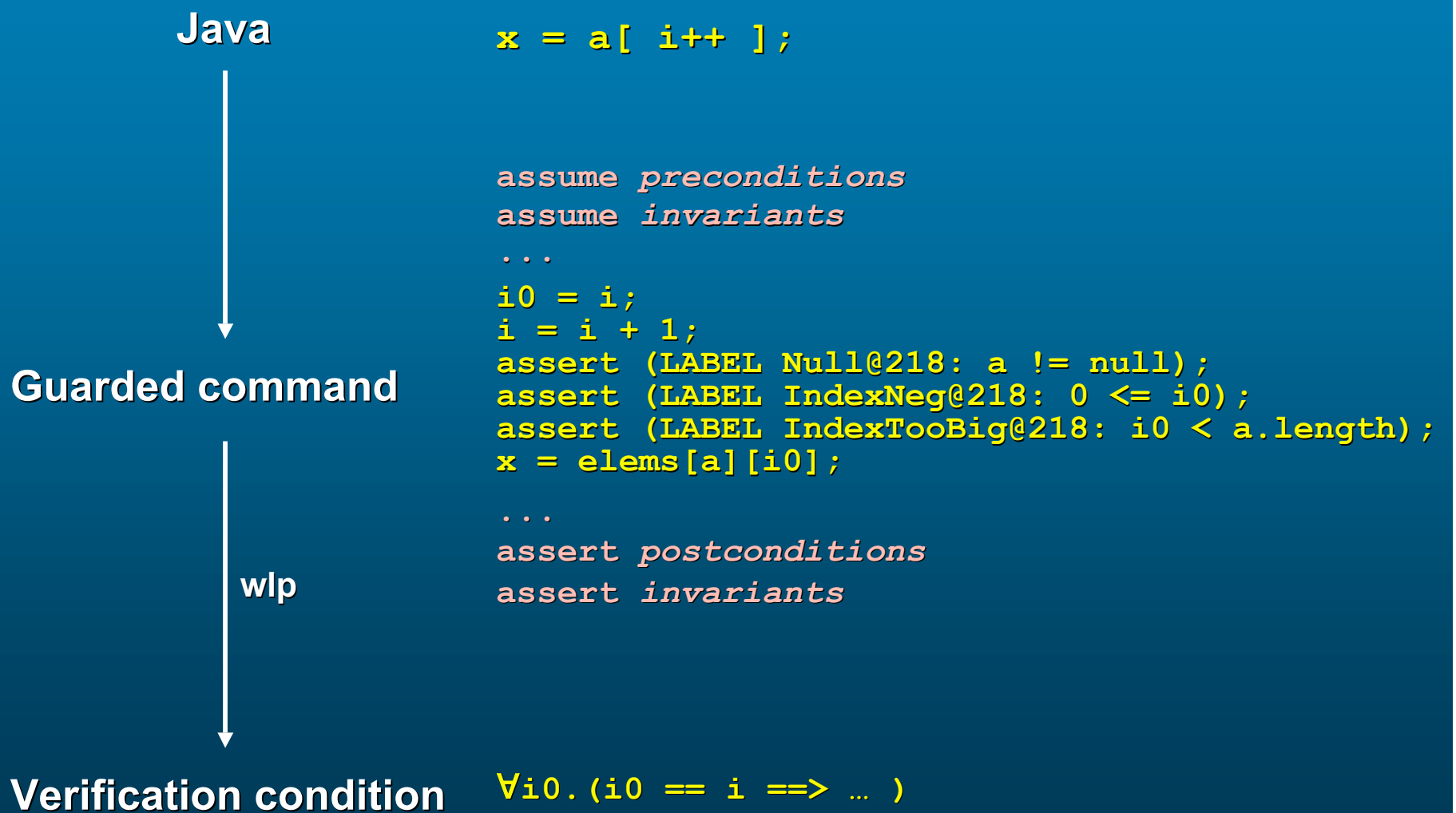
Verification condition generation



Verification condition generation

- Easy for small languages [Dijkstra]
- Much harder for real languages
  - Object-oriented
  - Typed
  - Dynamic allocation
  - Exceptions
  - Aliasing
  - Threads

Verification conditions for real programs



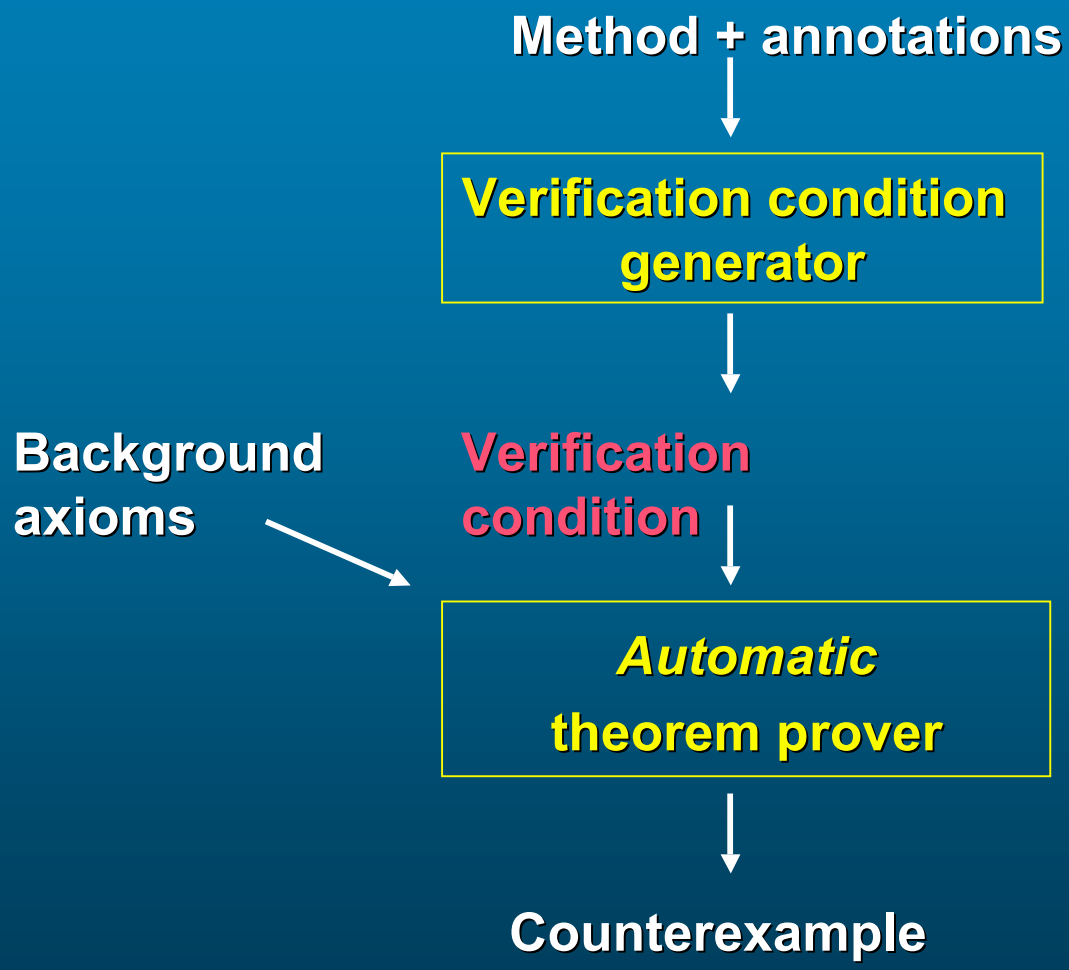
Exceptions

- Java has exceptions
- Add exceptions (`raise` and `catch`) to guarded command language
- Calculate wlp of GC statement with respect to normal and exceptional postconditions

Method overriding

- Method in subclass can override method in superclass
  - Must respect interface of overridden method
  - Weaker requires clause
  - Stronger ensures clause

Verification condition



Verification condition

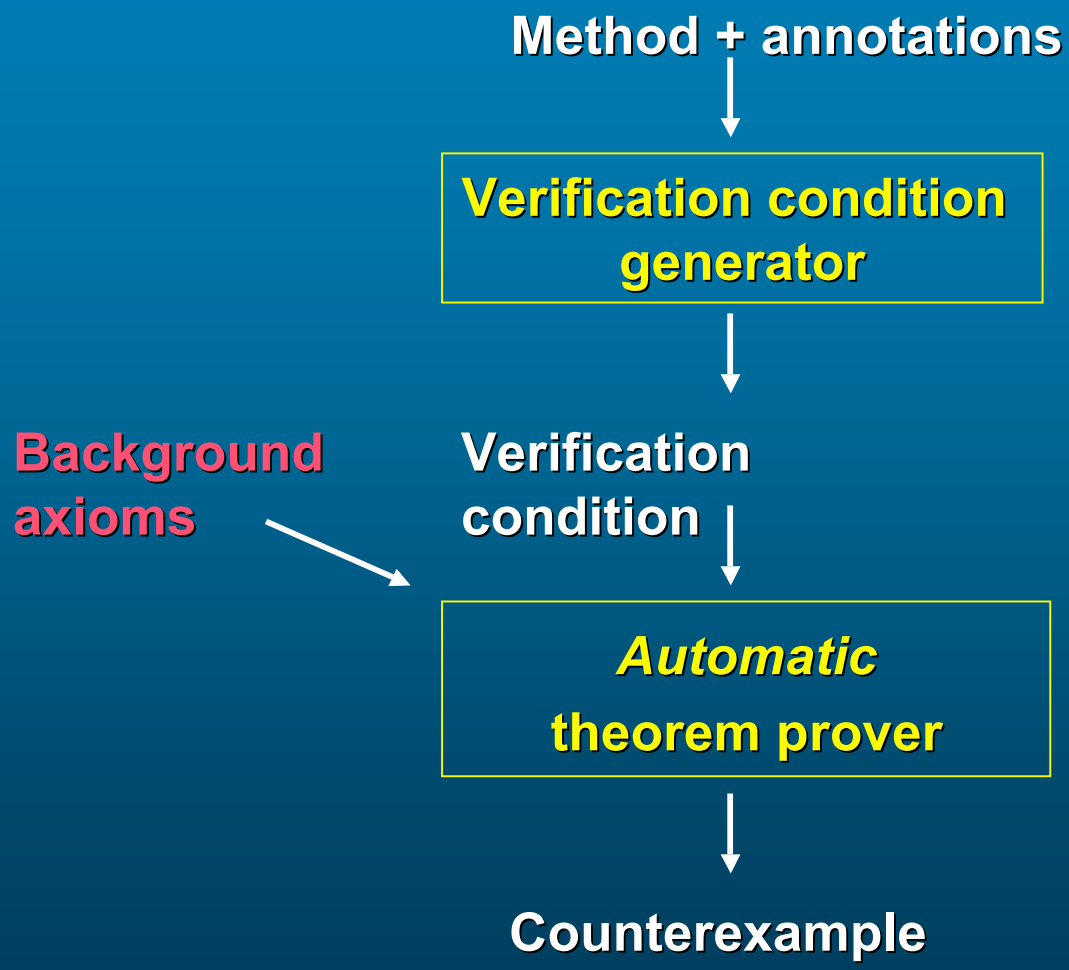
- Formula in untyped, first-order predicate calculus
  - equality and function symbols
  - quantifiers
  - arithmetic operations
  - select and store operations
  - Eg. $\forall x.\forall y.(x > y \Longrightarrow \ldots)$

Example verification condition

- Verification condition large but “dumb”


```
(IMPLIES (DISTINCT |ecReturn| |L_14.4|) (IMPLIES (AND (EQ |a@pre:2.8| |a:2.8|) (EQ |a:2.8| (asField |a:2.8| (array |T_int|))) (<
(fClosedTime |a:2.8|) alloc) (EQ |n@pre:3.6| |n:3.6|) (EQ |n:3.6| (asField |n:3.6| |T_int|)) (EQ |MAX_VALUE@pre:3.4.26|
|MAX_VALUE:3.4.26|) (EQ |@true| (is |MAX_VALUE:3.4.26| |T_int|)) (EQ |elems@pre| elems) (EQ elems (asElems elems)) (< (eClosedTime
elems) alloc) (EQ LS (asLockSet LS)) (EQ |alloc@pre| alloc) (EQ |@true| (is |this<1>| |T_Bag|)) (EQ |@true| (isAllocated |this<1>|
alloc)) (NEQ |this<1>| null)) (FORALL (tmp1 |tmp2:21.4| |tmp3:21.6| |m:12.8| |mindex:13.8| |i:14.13| |tmp0:14.28|) (AND (IMPLIES (<= 1
(select |n:3.6| |this<1>|)) (AND (LBLNEG |Null@15.10~15.10| (NEQ (select |a:2.8| |this<1>|) null)) (LBLNEG |IndexNegative@15.10~15.11|
(<= 0 1)) (LBLNEG |IndexTooBig@15.10~15.11| (< 1 (arrayLength (select |a:2.8| |this<1>|))))) (IMPLIES (< (select (select elems (select
|a:2.8| |this<1>|)) 1) |MAX_VALUE:3.4.26|) (AND (LBLNEG |Null@17.12~17.12| (NEQ (select |a:2.8| |this<1>|) null)) (LBLNEG
|IndexNegative@17.12~17.13| (<= 0 1)) (LBLNEG |IndexTooBig@17.12~17.13| (< 1 (arrayLength (select |a:2.8| |this<1>|))))) (FORALL
(|m:17.8|) (IMPLIES (EQ |m:17.8| (select (select elems (select |a:2.8| |this<1>|)) 1)) (FORALL (|i:14.28|) (IMPLIES (AND (EQ |i:14.28|
(+ 1 1)) (EQ |@true| |bool$false|)) (FORALL (|tmp2:21.4<1>|) (IMPLIES (EQ |tmp2:21.4<1>| (select |a:2.8| |this<1>|)) (AND (LBLNEG
|Null@21.16~21.16| (NEQ (select |a:2.8| |this<1>|) null)) (LBLNEG |IndexNegative@21.16~21.17| (<= 0 (select (store |n:3.6| |this<1>| (-
(select |n:3.6| |this<1>|) 1)) |this<1>|))) (LBLNEG |IndexTooBig@21.16~21.17| (< (select (store |n:3.6| |this<1>| (- (select |n:3.6|
|this<1>|) 1)) |this<1>|) (arrayLength (select |a:2.8| |this<1>|))))) (LBLNEG |Null@21.4~21.4| (NEQ |tmp2:21.4<1>| null)) (LBLNEG
|IndexNegative@21.4~21.5| (<= 0 1)) (LBLNEG |IndexTooBig@21.4~21.5| (< 1 (arrayLength |tmp2:21.4<1>|))) (LBLNEG
|Exception:11.6~11.6@11.2~11.2| (EQ |ecReturn| |ecReturn|))))))))))) (IMPLIES (NOT (< (select (select elems (select |a:2.8| |this<1>|))
1) |MAX_VALUE:3.4.26|)) (FORALL (|i:14.28|) (IMPLIES (AND (EQ |i:14.28| (+ 1 1)) (EQ |@true| |bool$false|)) (FORALL (|tmp2:21.4<1>|)
(IMPLIES (EQ |tmp2:21.4<1>| (select |a:2.8| |this<1>|)) (AND (LBLNEG |Null@21.16~21.16| (NEQ (select |a:2.8| |this<1>|) null)) (LBLNEG
|IndexNegative@21.16~21.17| (<= 0 (select (store |n:3.6| |this<1>| (- (select |n:3.6| |this<1>|) 1)) |this<1>|))) (LBLNEG
|IndexTooBig@21.16~21.17| (< (select (store |n:3.6| |this<1>| (- (select |n:3.6| |this<1>|) 1)) |this<1>|) (arrayLength (select |a:2.8|
|this<1>|))))) (LBLNEG |Null@21.4~21.4| (NEQ |tmp2:21.4<1>| null)) (LBLNEG |IndexNegative@21.4~21.5| (<= 0 0)) (LBLNEG
|IndexTooBig@21.4~21.5| (< 0 (arrayLength |tmp2:21.4<1>|))) (LBLNEG |Exception:11.6~11.6@11.2~11.2| (EQ |ecReturn| |ecReturn|)))))))))
(IMPLIES (NOT (<= 1 (select |n:3.6| |this<1>|))) (AND (IMPLIES (EQ |L_14.4| |L_14.4|) (FORALL (|tmp2:21.4<1>|) (IMPLIES (EQ
|tmp2:21.4<1>| (select |a:2.8| |this<1>|)) (AND (LBLNEG |Null@21.16~21.16| (NEQ (select |a:2.8| |this<1>|) null)) (LBLNEG
|IndexNegative@21.16~21.17| (<= 0 (select (store |n:3.6| |this<1>| (- (select |n:3.6| |this<1>|) 1)) |this<1>|))) (LBLNEG
|IndexTooBig@21.16~21.17| (< (select (store |n:3.6| |this<1>| (- (select |n:3.6| |this<1>|) 1)) |this<1>|) (arrayLength (select |a:2.8|
|this<1>|))))) (LBLNEG |Null@21.4~21.4| (NEQ |tmp2:21.4<1>| null)) (LBLNEG |IndexNegative@21.4~21.5| (<= 0 0)) (LBLNEG
|IndexTooBig@21.4~21.5| (< 0 (arrayLength |tmp2:21.4<1>|))) (LBLNEG |Exception:11.6~11.6@11.2~11.2| (EQ |ecReturn| |ecReturn|))))))
(IMPLIES (NOT (EQ |L_14.4| |L_14.4|)) (AND (LBLNEG |Exception:11.6~11.6@11.2~11.2| (EQ |L_14.4| |ecReturn|)))))))))))
```

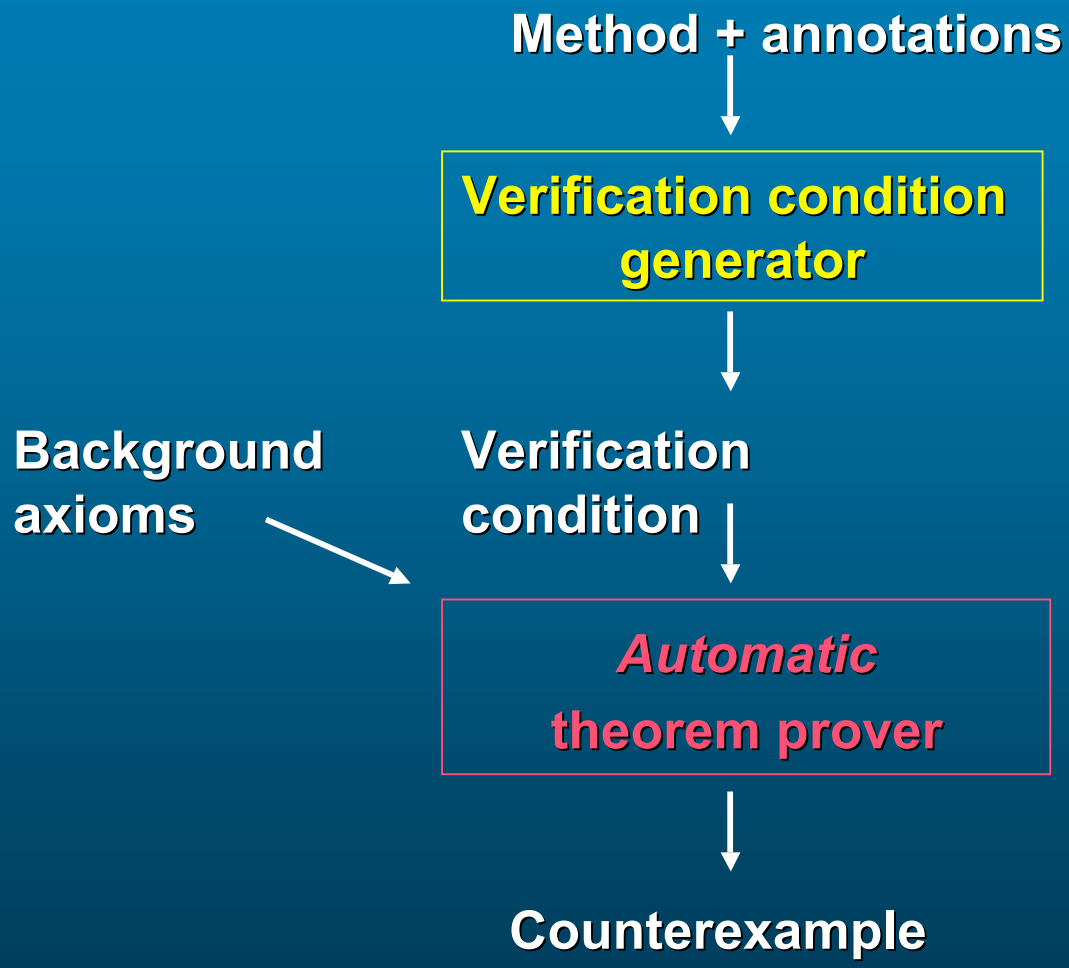
Background axioms



Background axioms

- Additional properties of Java that the theorem prover needs to know
- A variable of type T always holds a value whose type is a subtype of T
- The subtyping relation is reflexive, anti-symmetric, and transitive
- `new` returns an object that is distinct from all existing objects
- ... lots more ...
- `java.lang.Object` has no supertype

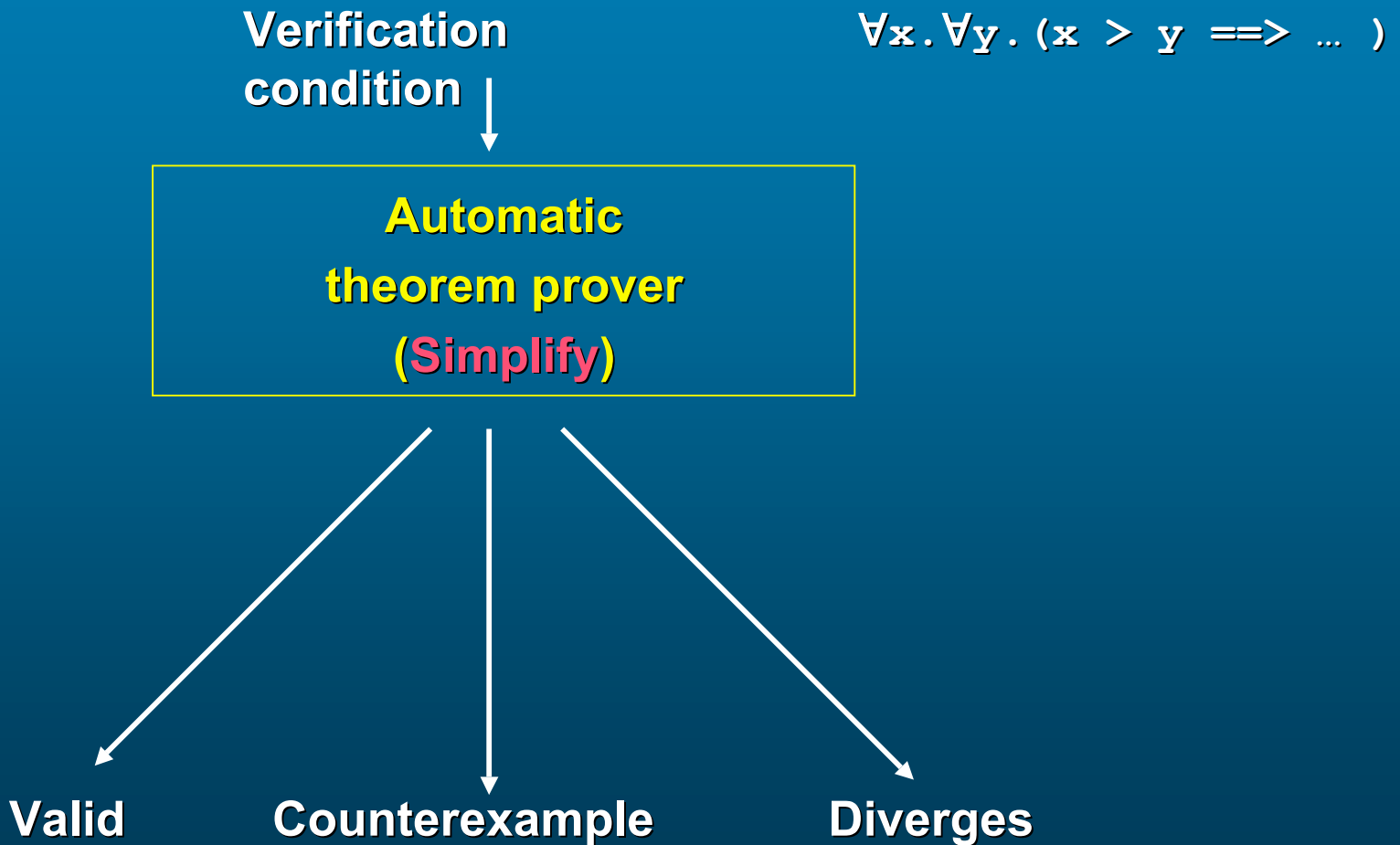
Automatic theorem proving



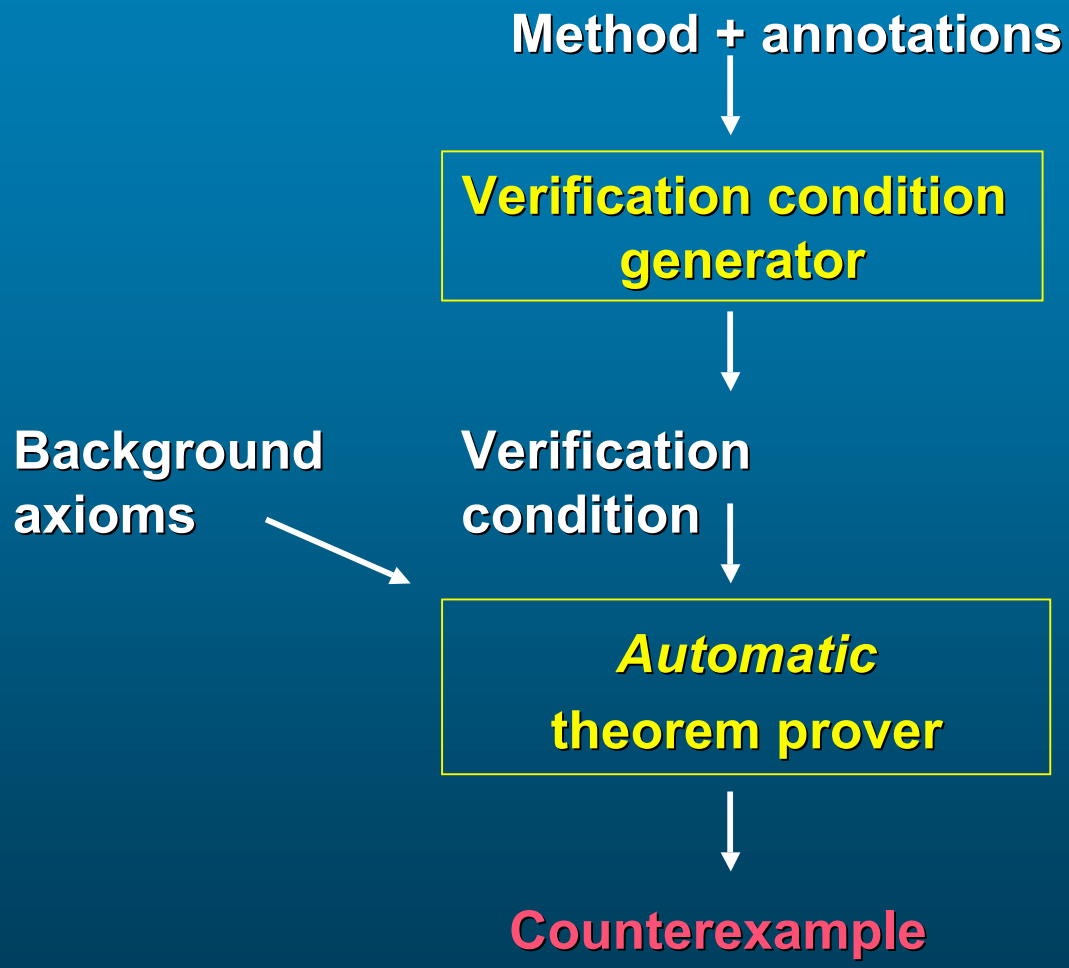
Automatic theorem proving

- Use *Simplify*
  - Theorem prover from ESC/Modula-3
  - Accepts formulae in untyped, first-order predicate calculus
  - Attempts to prove or refute

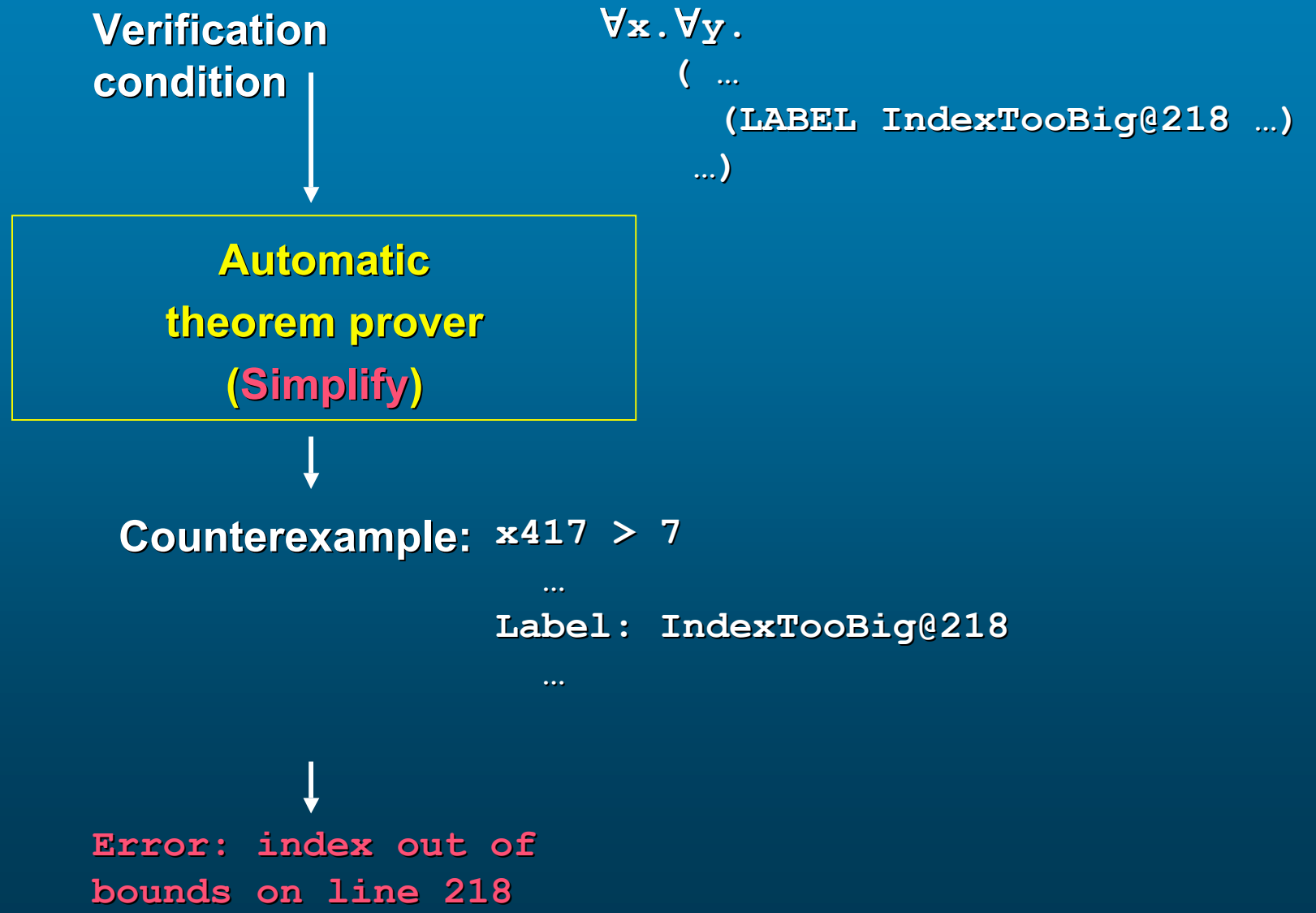
Automatic theorem proving



Handling counterexamples



Error message from counterexample



Initial experience

- First implementation is done
- Run on 30,000+ lines of code (mostly itself)
- Caught several errors
  - null dereference, array bounds
- Programmer can annotate and check about 300 lines per hour
- Looks promising ...

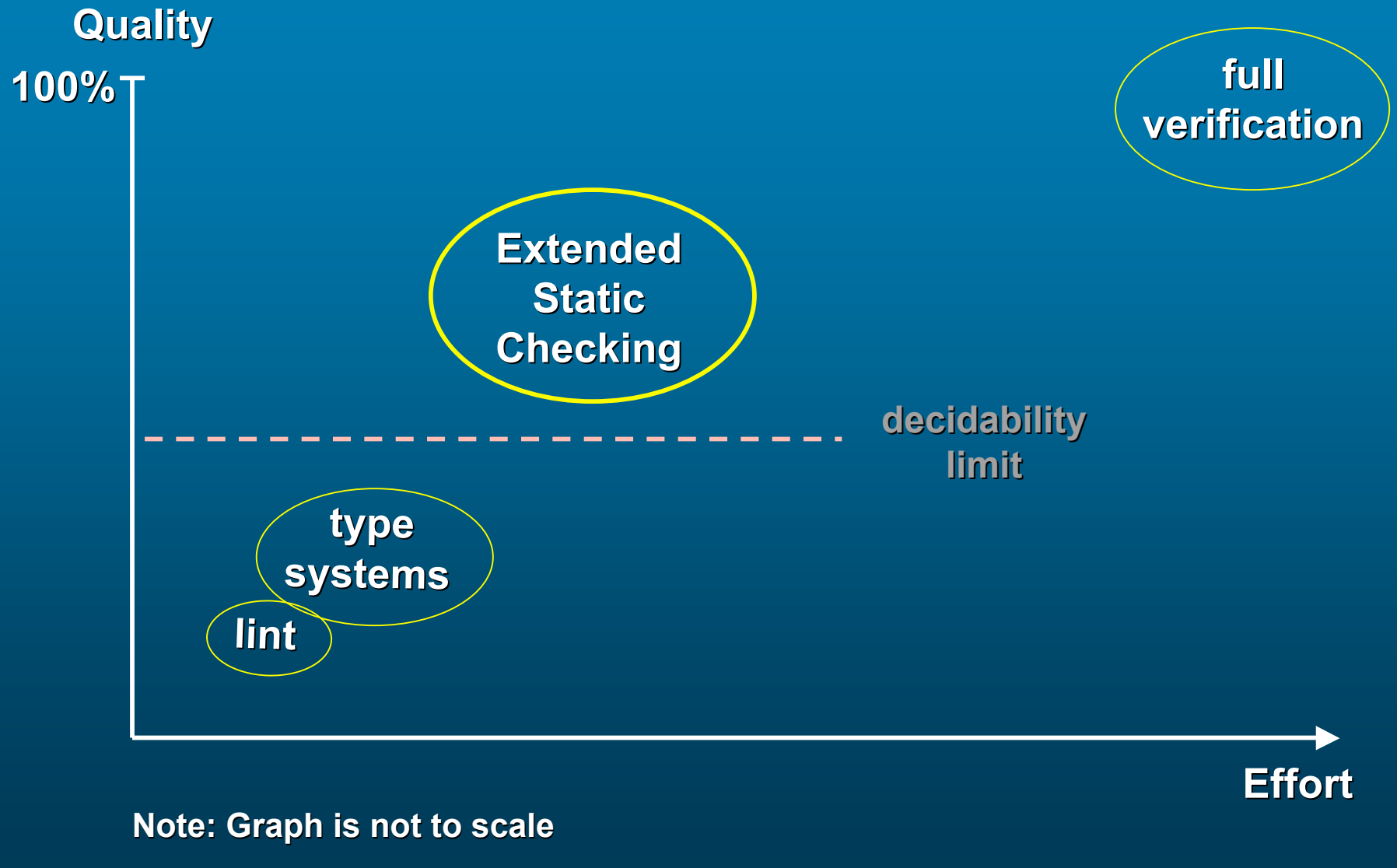
Demonstration

ESC/Java Summary

- Finds more errors than type checking
- Costs less than full verification
- Currently working; is being evaluated
- Potential as “software reliability metric”
- Practical checking based on automatic theorem proving may be possible

`www.research.digital.com/SRC/esc/Esc.html`

Comparison of Static Checkers



Metrics for Static Checkers

- **Cost**
  - of using the tool
- **Quality**
  - Does it miss errors?
  - Does it give spurious warnings?

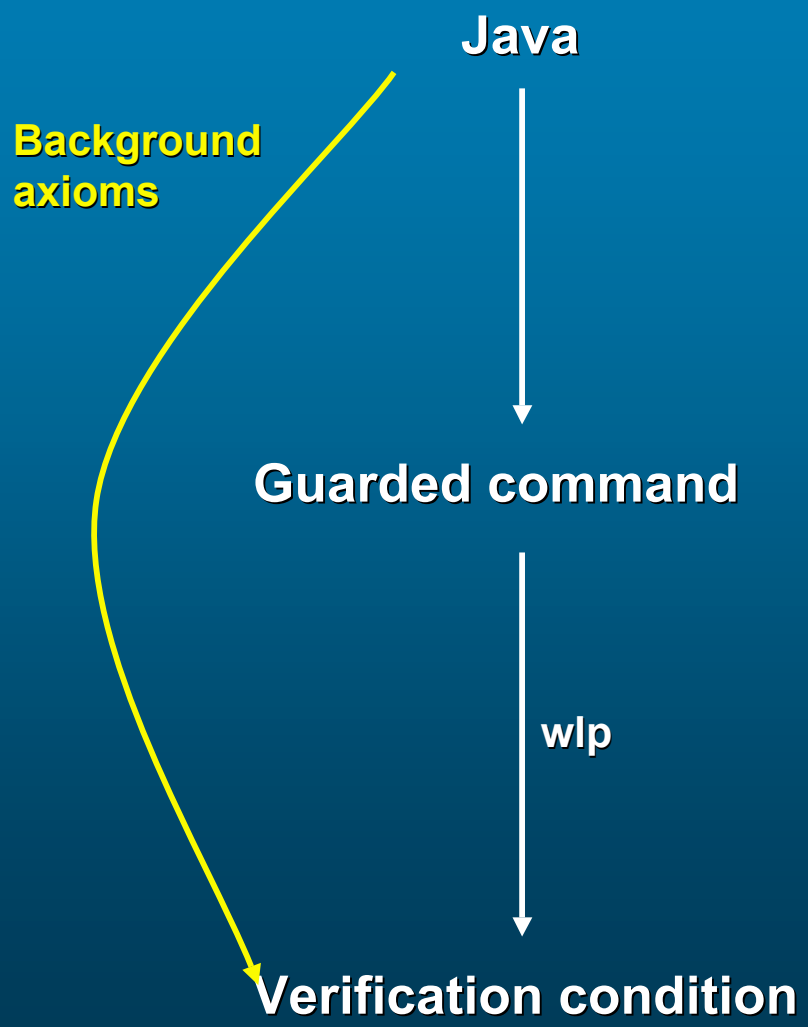
Challenges

- Automatic theorem proving
- Error messages from counterexample
- Verification conditions for real programs
  - Object-oriented
  - Typed
  - Dynamic allocation
  - Exceptions

ESC/Java vs. Testing

- Testing essential but
  - Expensive
  - Finds errors late
  - Misses errors
- ESC/Java ... ?

Background axioms



Additional annotations

```
//@ assert <exp>

//@ assume <exp>

//@ nowarn <error code>

//@ axiom <exp>
```

Describing interfaces

```
public Integer[] sum(Integer[] a, Integer[] b);
//@ requires a != null && b != null;
//@ requires a.length == b.length;
//@ ensures RES != null && RES.length == a.length;
//@ modifies a[0], b[*];
```