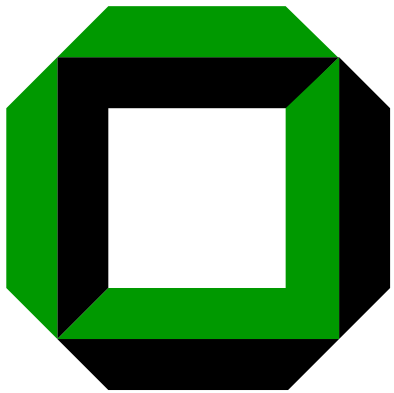
Formale Entwicklung objektorientierter Software

Praktikum im Wintersemester 2007/2008

Prof. P. H. Schmitt, Dr. T. Käufl, C. Engel, B. Weiß

Institut für Theoretische Informatik
Universität Karlsruhe

21. November 2007



ESC/Java2

Extended Static Checking (ESC)

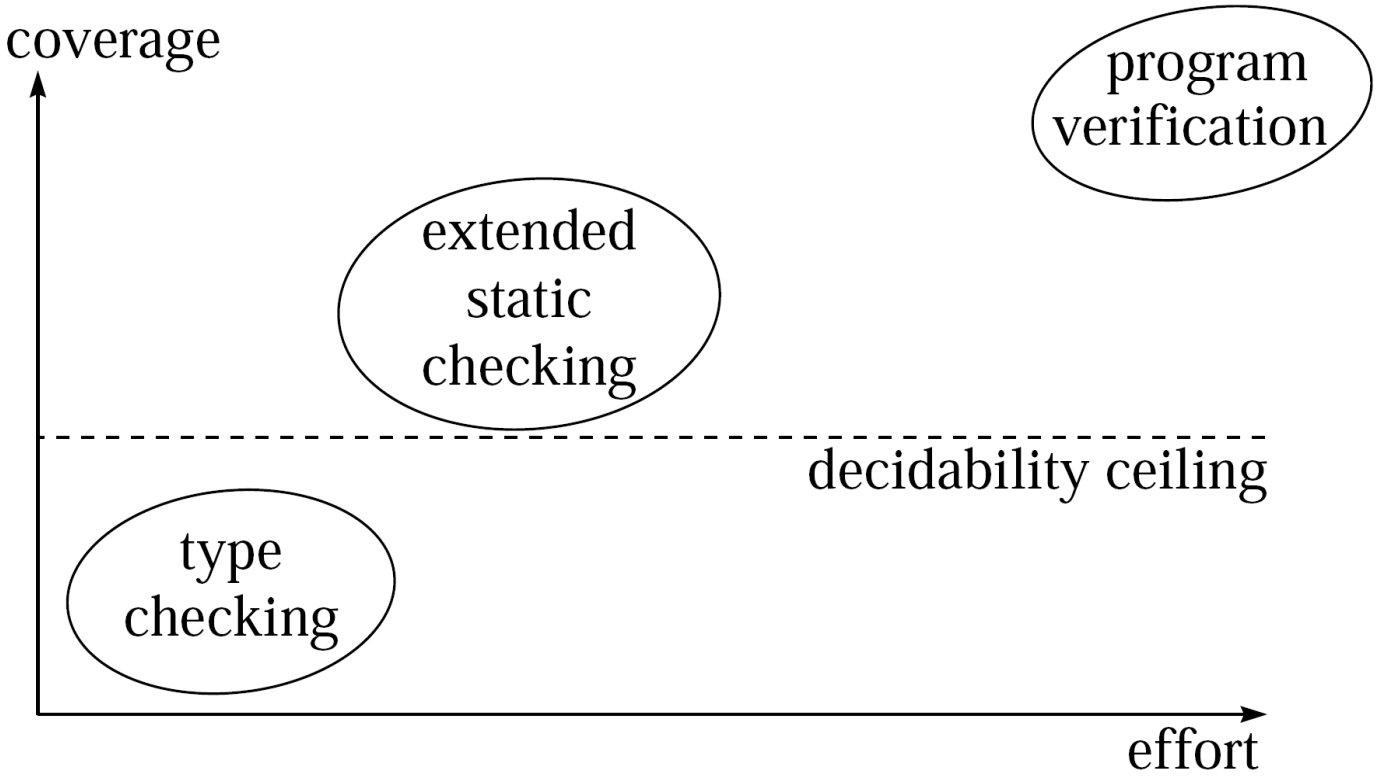
- Automated compile-time checking ("static")

Extended Static Checking (ESC)

- Automated compile-time checking (“static”)
- More powerful than what is done by compilers (“extended”)

Extended Static Checking (ESC)

- Automated compile-time checking ("static")
- More powerful than what is done by compilers ("extended")



(source: Flanagan et al., "Extended Static Checking for Java")

History of ESC

- ESC/Modula-3
  - DEC / Compaq Research, -1996

History of ESC

- ESC/Modula-3
  - DEC / Compaq Research, -1996
- ESC/Java
  - successor to ESC/Modula-3 targeting Java instead of Modula-3
  - Compaq Research, -2000

History of ESC

- ESC/Modula-3
  - DEC / Compaq Research, -1996
- ESC/Java
  - successor to ESC/Modula-3 targeting Java instead of Modula-3
  - Compaq Research, -2000
- ESC/Java2
  - successor to ESC/Java using JML as specification language
  - University of Nijmegen (-2005), UCD Dublin (2005-)

History of ESC

- ESC/Modula-3
  - DEC / Compaq Research, -1996
- ESC/Java
  - successor to ESC/Modula-3 targeting Java instead of Modula-3
  - Compaq Research, -2000
- ESC/Java2
  - successor to ESC/Java using JML as specification language
  - University of Nijmegen (-2005), UCD Dublin (2005-)
- Spec#, Boogie
  - "ESC for C#"
  - Microsoft Research, 2002-

Example

```
/*@ ensures
  @  (\forall int x; 0<=x && x<a.length; a[x]<=\result);
  @*/
int getMaximum(int[] a) {
    int max = 0;
    for(int i = 0; i <= a.length; i++) {
        if(a[i] > max) max = a[i];
    }
    return 0;
}
```

Example

```
/*@ ensures
  @  (\forall int x; 0<=x && x<a.length; a[x]<=\result);
  @*/
int getMaximum(int[] a) {
    int max = 0;
    for(int i = 0; i <= a.length; i++) {
        if(a[i] > max) max = a[i];
    }
    return 0;
}
```

Warning: Possible null dereference

Example

```
/*@ ensures
  @  (\forall int x; 0<=x && x<a.length; a[x]<=\result);
  @*/
int getMaximum(/*@ non_null @*/ int[] a) {
    int max = 0;
    for(int i = 0; i <= a.length; i++) {
        if(a[i] > max) max = a[i];
    }
    return 0;
}
```

Warning: Possible null dereference

Example

```
/*@ ensures
  @  (\forall int x; 0<=x && x<a.length; a[x]<=\result);
  @*/
int getMaximum(/*@ non_null @*/ int[] a) {
    int max = 0;
    for(int i = 0; i <= a.length; i++) {
        if(a[i] > max) max = a[i];
    }
    return 0;
}
```

Warning: Array index possibly too large

Example

```
/*@ ensures
  @  (\forall int x; 0<=x && x<a.length; a[x]<=\result);
  @*/
int getMaximum(/*@ non_null @*/ int[] a) {
    int max = 0;
    for(int i = 0; i < a.length; i++) {
        if(a[i] > max) max = a[i];
    }
    return 0;
}
```

Warning: Array index possibly too large

Example

```
/*@ ensures
  @  (\forall int x; 0<=x && x<a.length; a[x]<=\result);
  @*/
int getMaximum(/*@ non_null @*/ int[] a) {
    int max = 0;
    for(int i = 0; i < a.length; i++) {
        if(a[i] > max) max = a[i];
    }
    return 0;
}
```

Warning: Postcondition possibly not established

Example

```
/*@ ensures
  @  (\forall int x; 0<=x && x<a.length; a[x]<=\result);
  @*/
int getMaximum(/*@ non_null @*/ int[] a) {
    int max = 0;
    for(int i = 0; i < a.length; i++) {
        if(a[i] > max) max = a[i];
    }
    return max;
}
```

Warning: Postcondition possibly not established

Example

```
/*@ ensures
  @  (\forall int x; 0<=x && x<a.length; a[x]<=\result);
  @*/
int getMaximum(/*@ non_null @*/ int[] a) {
    int max = 0;
    for(int i = 0; i < a.length; i++) {
        if(a[i] > max) max = a[i];
    }
    return max;
}
```

[0.162 s 10333000 bytes] passed

Warnings

Possible runtime exceptions:

- `NullPointerException`

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `ClassCastException`

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `ClassCastException`
- `ArrayStoreException`

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `ClassCastException`
- `ArrayStoreException`
- `ArithmeticException`

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `ClassCastException`
- `ArrayStoreException`
- `ArithmeticException`
- ...

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `ClassCastException`
- `ArrayStoreException`
- `ArithmeticException`
- ...

Violations of JML specifications:

- Pre- and postconditions, assignable-clauses

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `ClassCastException`
- `ArrayStoreException`
- `ArithmeticException`
- ...

Violations of JML specifications:

- Pre- and postconditions, assignable-clauses
- Assertions

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `ClassCastException`
- `ArrayStoreException`
- `ArithmeticException`
- ...

Violations of JML specifications:

- Pre- and postconditions, assignable-clauses
- Assertions
- Class invariants

Warnings

Possible runtime exceptions:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `ClassCastException`
- `ArrayStoreException`
- `ArithmeticException`
- ...

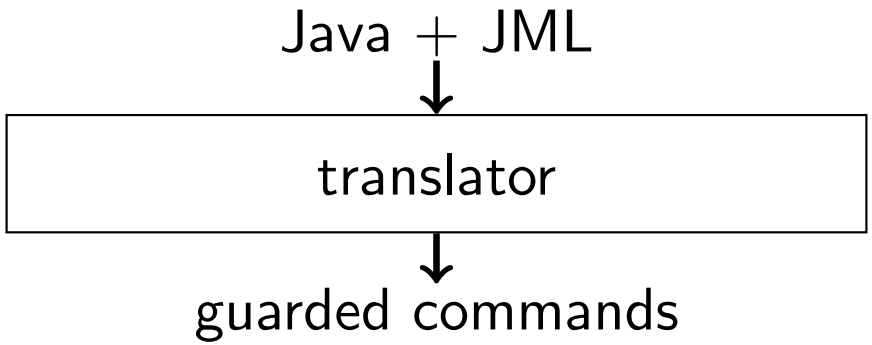
Violations of JML specifications:

- Pre- and postconditions, assignable-clauses
- Assertions
- Class invariants
- Loop invariants

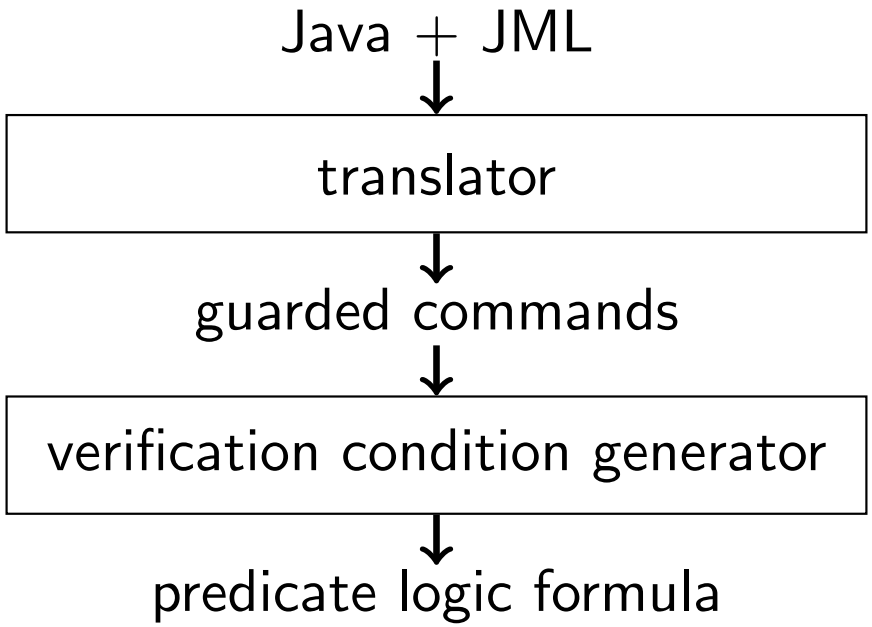
Inner Workings

Java + JML

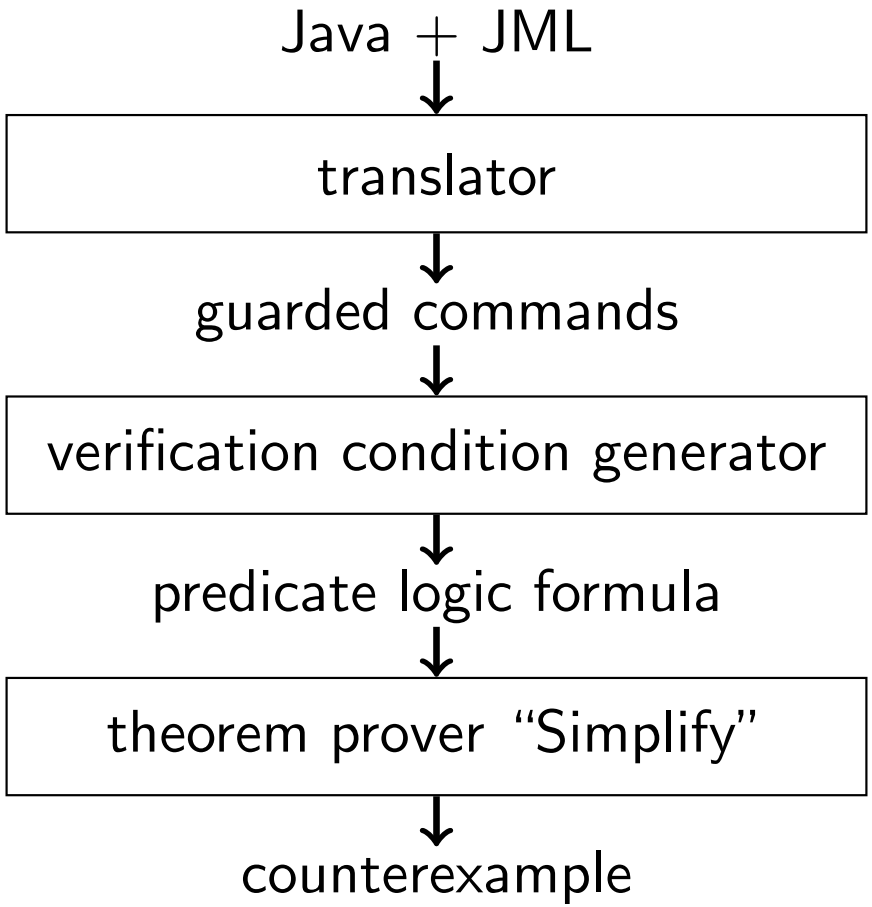
Inner Workings



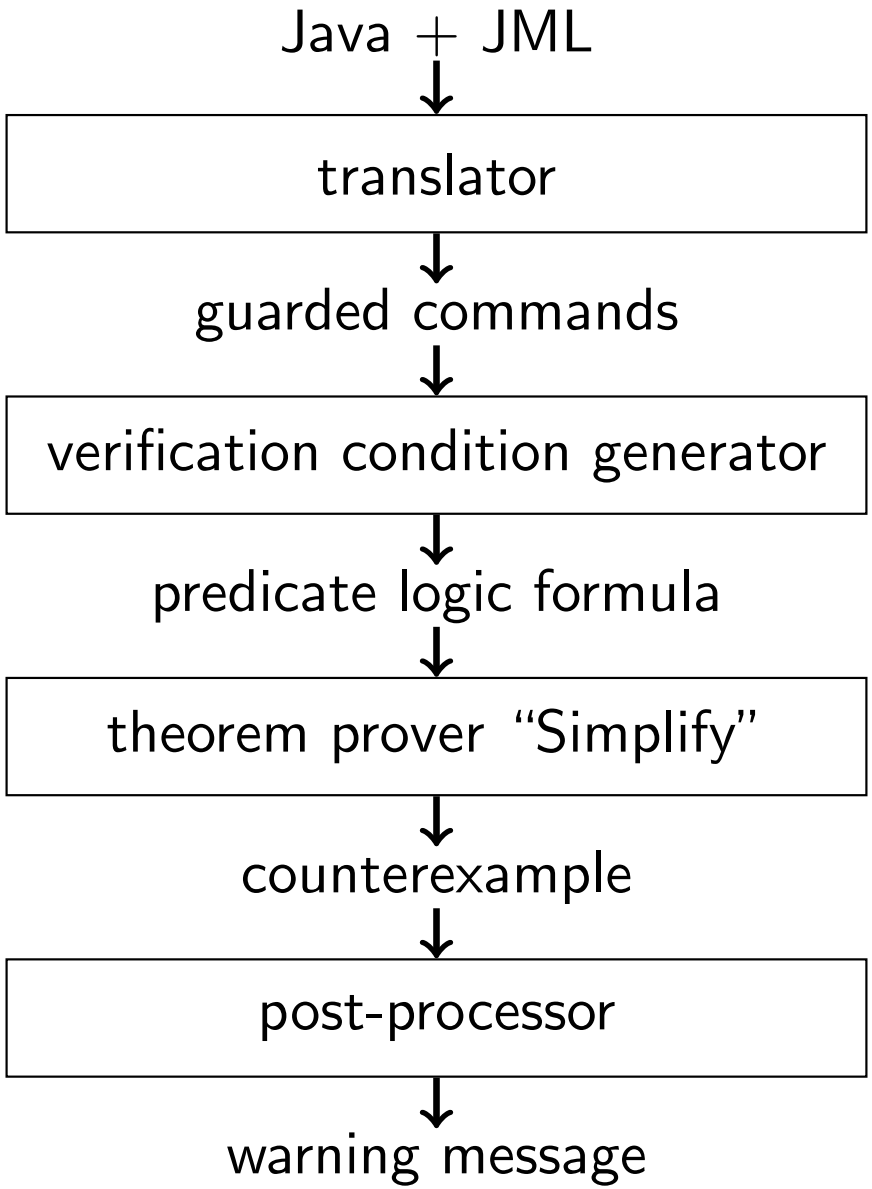
Inner Workings



Inner Workings



Inner Workings



Translator

Translates Java into a simpler intermediate language based on Dijkstra's guarded commands.

Translator

Translates Java into a simpler intermediate language based on Dijkstra's guarded commands.

```
if(i > j) max = i;
else max = j;
//@ assert max >= i && max >= j;
```

Translator

Translates Java into a simpler intermediate language based on Dijkstra's guarded commands.

```
if(i > j) max = i;
else max = j;
//@ assert max >= i && max >= j;
```

```
{  ASSUME integralGT(i:2.15, j:2.18);
   ASSUME (\lblpos trace.Then^0,5.11 true);
   max:2.21 = i:2.15;
   state = state:6.9
 []
   ASSUME boolNot(integralGT(i:2.15, j:2.18));
   ASSUME (\lblpos trace.Else^1,7.8 true);
   max:2.21 = j:2.18;
   state = state:8.9
};
ASSERT (\lblneg Assert@10.12 boolAnd(integralGE(max:2.21, i:2.15),
                                     integralGE(max:2.21, j:2.18)));
```

Verification Condition Generator

- Turns guarded commands into predicate logic “verification condition”

Verification Condition Generator

- Turns guarded commands into predicate logic “verification condition”
- Based on computing weakest preconditions / strongest postconditions

Verification Condition Generator

- Turns guarded commands into predicate logic "verification condition"
- Based on computing weakest preconditions / strongest postconditions
- Ideally (but not always):
  Verification condition valid $\Leftrightarrow$ program correct

Verification Condition Generator

- Turns guarded commands into predicate logic "verification condition"
- Based on computing weakest preconditions / strongest postconditions
- Ideally (but not always):
  Verification condition valid $\Leftrightarrow$ program correct

```
(EXPLIES
  (LBLNEG |vc.MyClass.max.4.4|
    (IMPLIES
      (AND
        (EQ |max@pre:2.21| |max:2.21|)
        (EQ |@true|
          (is |max:2.21| T_int)
          )
        (EQ |j@pre:2.18| |j:2.18|)
        ...
```

Simplify

- Automated theorem prover for predicate logic with linear integer arithmetic

Simplify

- Automated theorem prover for predicate logic with linear integer arithmetic
- Tries to prove validity of a formula φ by searching for model of $\neg\varphi$:
  - returns "valid" if no model is found, or
  - a counterexample which is believed to satisfy $\neg\varphi$

Simplify

- Automated theorem prover for predicate logic with linear integer arithmetic
- Tries to prove validity of a formula φ by searching for model of $\neg\varphi$:
  - returns "valid" if no model is found, or
  - a counterexample which is believed to satisfy $\neg\varphi$
- Sound: If answer is "valid", then φ is valid

Simplify

- Automated theorem prover for predicate logic with linear integer arithmetic
- Tries to prove validity of a formula φ by searching for model of $\neg\varphi$:
  - returns "valid" if no model is found, or
  - a counterexample which is believed to satisfy $\neg\varphi$
- Sound: If answer is "valid", then φ is valid
- Not complete: A reported counterexample may be wrong

Simplify

- Automated theorem prover for predicate logic with linear integer arithmetic
- Tries to prove validity of a formula φ by searching for model of $\neg\varphi$:
  - returns "valid" if no model is found, or
  - a counterexample which is believed to satisfy $\neg\varphi$
- Sound: If answer is "valid", then φ is valid
- Not complete: A reported counterexample may be wrong
- Written in Modula-3 for ESC/Modula-3

Simplify

- Automated theorem prover for predicate logic with linear integer arithmetic
- Tries to prove validity of a formula φ by searching for model of $\neg\varphi$:
  - returns "valid" if no model is found, or
  - a counterexample which is believed to satisfy $\neg\varphi$
- Sound: If answer is "valid", then φ is valid
- Not complete: A reported counterexample may be wrong
- Written in Modula-3 for ESC/Modula-3
- Also available as an external decision procedure in KeY

Post-processor

Analyses Simplify's counterexample to produce human readable warning, including

Post-processor

Analyses Simplify's counterexample to produce human readable warning, including

- Source code location

Post-processor

Analyses Simplify's counterexample to produce human readable warning, including

- Source code location
- Type of warning

Post-processor

Analyses Simplify's counterexample to produce human readable warning, including

- Source code location
- Type of warning
- Relevant specifications

Post-processor

Analyses Simplify's counterexample to produce human readable warning, including

- Source code location
- Type of warning
- Relevant specifications
- Execution trace

Post-processor

Analyses Simplify's counterexample to produce human readable warning, including

- Source code location
- Type of warning
- Relevant specifications
- Execution trace

```
MyClass.java:18: Warning: Possible violation of object invariant (Invariant)
    }
    ^
Associated declaration is "MyClass.java", line 12, col 26:
    static int i = 1; //@ invariant i > 0;
                          ^
Execution trace information:
    Executed then branch in "MyClass.java", line 15, col 18.
```

Counterexamples

```
o1.f = 1;
o2.f = 2;
//@ assert o1.f == 1;
//@ assert o2.f == 2;
```

Counterexamples

```
o1.f = 1;
o2.f = 2;
//@ assert o1.f == 1;
//@ assert o2.f == 2;
```

Warning: Possible assertion failure

Counterexamples

```
o1.f = 1;
o2.f = 2;
//@ assert o1.f == 1;
//@ assert o2.f == 2;
```

Warning: Possible assertion failure

excerpt of counterexample

```
(o1:2.37).(f:7.9) == 1
(o1:2.37).(f:8.9) == 2
typeof(o1:2.37) <: T_MyClass
o1:2.37 == o2:3.37
T_bigint == T_long
```

Counterexamples

```
o1.f = 1;
o2.f = 2;
//@ assert o1.f == 1;
//@ assert o2.f == 2;
```

Warning: Possible assertion failure

excerpt of counterexample

```
(o1:2.37).(f:7.9) == 1
(o1:2.37).(f:8.9) == 2
typeof(o1:2.37) <: T_MyClass
o1:2.37 == o2:3.37
T_bigint == T_long
```

Counterexamples

```
o1.f = 1;
o2.f = 2;
//@ assert o1 != o2 ==> o1.f == 1;
//@ assert o2.f == 2;
```

Warning: Possible assertion failure

excerpt of counterexample

```
(o1:2.37).(f:7.9) == 1
(o1:2.37).(f:8.9) == 2
typeof(o1:2.37) <: T_MyClass
o1:2.37 == o2:3.37
T_bigint == T_long
```

Counterexamples

```
o1.f = 1;
o2.f = 2;
//@ assert o1 != o2 ==> o1.f == 1;
//@ assert o2.f == 2;
```

[0.077 s 10045712 bytes] passed

Aliasing

```
int[] a = new int[10]; //@ invariant a!=null && a.length>0;
int x;                 //@ invariant x==a[0];

void m() {
    a[0] = 27;
    x = 27;
}
```

Aliasing

```
int[] a = new int[10]; //@ invariant a!=null && a.length>0;
int x;                 //@ invariant x==a[0];

void m() {
    a[0] = 27;
    x = 27;
}
```

Warning: Possible violation of object invariant

Aliasing

```
int[] a = new int[10]; //@ invariant a!=null && a.length>0;
int x;                 //@ invariant x==a[0];

void m() {
    a[0] = 27;
    x = 27;
}
```

Warning: Possible violation of object invariant

```
brokenObj != this
brokenObj.(a@pre:2.10) == tmp0!a:6.8
this.(a@pre:2.10) == tmp0!a:6.8
```

Aliasing

```
int[] a = new int[10]; //@ invariant a!=null && a.length>0;
int x;                 //@ invariant x==a[0];
                       //@ invariant a.owner==this;

void m() {
    a[0] = 27;
    x = 27;
}
```

Warning: Possible violation of object invariant

```
brokenObj != this
brokenObj.(a@pre:2.10) == tmp0!a:6.8
this.(a@pre:2.10) == tmp0!a:6.8
```

Aliasing

```
int[] a = new int[10]; //@ invariant a!=null && a.length>0;
int x;                 //@ invariant x==a[0];
                       //@ invariant a.owner==this;

void m() {
    a[0] = 27;
    x = 27;
}
MyClass() {
    //@ set a.owner = this;
}
```

Warning: Possible violation of object invariant

```
brokenObj != this
brokenObj.(a@pre:2.10) == tmp0!a:6.8
this.(a@pre:2.10) == tmp0!a:6.8
```

Aliasing

```
int[] a = new int[10]; //@ invariant a!=null && a.length>0;
int x;                 //@ invariant x==a[0];
                       //@ invariant a.owner==this;
void m() {
    a[0] = 27;
    x = 27;
}
MyClass() {
    //@ set a.owner = this;
}
```

[0.145 s 10129336 bytes] passed

ESC/Java2 is not sound

- `assume` is trusted blindly

ESC/Java2 is not sound

- `assume` is trusted blindly

```
i = 0;
//@ assume i == 1;
//@ assert i == 1;
```

ESC/Java2 is not sound

- `assume` is trusted blindly

```
i = 0;
//@ assume i == 1;
//@ assert i == 1;
```

[0.129 s 10099544 bytes] passed

ESC/Java2 is not sound

- `assume` is trusted blindly

```
i = 0;
//@ assume i == 1;
//@ assert i == 1;
```

```
[0.129 s 10099544 bytes] passed
```

- Overflows are ignored

ESC/Java2 is not sound

- `assume` is trusted blindly

```
i = 0;
//@ assume i == 1;
//@ assert i == 1;
```

```
[0.129 s 10099544 bytes] passed
```

- Overflows are ignored

```
i = j + 1;
//@ assert i > j;
```

ESC/Java2 is not sound

- `assume` is trusted blindly

```
i = 0;
//@ assume i == 1;
//@ assert i == 1;
```

```
[0.129 s 10099544 bytes] passed
```

- Overflows are ignored

```
i = j + 1;
//@ assert i > j;
```

```
[0.049 s 10406944 bytes] passed
```

ESC/Java2 is not sound (contd.)

- Loops are handled by unrolling a fixed number of times

ESC/Java2 is not sound (contd.)

- Loops are handled by unrolling a fixed number of times

```
for(i = 0; i < 2; i++) {}
//@ assert i == 567;
```

ESC/Java2 is not sound (contd.)

- Loops are handled by unrolling a fixed number of times

```
for(i = 0; i < 2; i++) {}
//@ assert i == 567;
```

[0.048 s 10407744 bytes] passed

ESC/Java2 is not sound (contd.)

- Loops are handled by unrolling a fixed number of times

```
for(i = 0; i < 2; i++) {}
//@ assert i == 567;
```

```
[0.048 s 10407744 bytes] passed
```

- Assignable-clauses are not always treated properly

ESC/Java2 is not sound (contd.)

- Loops are handled by unrolling a fixed number of times

```
for(i = 0; i < 2; i++) {}
//@ assert i == 567;
```

[0.048 s 10407744 bytes] passed

- Assignable-clauses are not always treated properly

```
//@ assignable \everything;
void setTo42() {i = 42;}
```

ESC/Java2 is not sound (contd.)

- Loops are handled by unrolling a fixed number of times

```
for(i = 0; i < 2; i++) {}
//@ assert i == 567;
```

[0.048 s 10407744 bytes] passed

- Assignable-clauses are not always treated properly

```
//@ assignable \everything;
void setTo42() {i = 42;}

i = 0;
setTo42();
//@ assert i == 0;
```

ESC/Java2 is not sound (contd.)

- Loops are handled by unrolling a fixed number of times

```
for(i = 0; i < 2; i++) {}
//@ assert i == 567;
```

[0.048 s 10407744 bytes] passed

- Assignable-clauses are not always treated properly

```
//@ assignable \everything;
void setTo42() {i = 42;}

i = 0;
setTo42();
//@ assert i == 0;
```

[0.024 s 10941408 bytes] passed

ESC/Java2 is not complete

- Runtime exceptions are always warned about

ESC/Java2 is not complete

- Runtime exceptions are always warned about

```
try {
    Object o = null;
    o.toString();
} catch(NullPointerException e) {}
```

ESC/Java2 is not complete

- Runtime exceptions are always warned about

```
try {
    Object o = null;
    o.toString();
} catch(NullPointerException e) {}
```

Warning: Possible null dereference

ESC/Java2 is not complete

- Runtime exceptions are always warned about

```
try {
    Object o = null;
    o.toString();
} catch(NullPointerException e) {}
```

Warning: Possible null dereference

- Reasoning is strictly modular

ESC/Java2 is not complete

- Runtime exceptions are always warned about

```
try {
    Object o = null;
    o.toString();
} catch(NullPointerException e) {}
```

Warning: Possible null dereference

- Reasoning is strictly modular

```
int get27() {return 27;}
```

ESC/Java2 is not complete

- Runtime exceptions are always warned about

```
try {
    Object o = null;
    o.toString();
} catch(NullPointerException e) {}
```

Warning: Possible null dereference

- Reasoning is strictly modular

```
int get27() {return 27;}

//@ assert get27() == 27;
```

ESC/Java2 is not complete

- Runtime exceptions are always warned about

```
try {
    Object o = null;
    o.toString();
} catch(NullPointerException e) {}
```

Warning: Possible null dereference

- Reasoning is strictly modular

```
int get27() {return 27;}

//@ assert get27() == 27;
```

Warning: Possible assertion failure

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

```
//@ assert i*j == j*i;
```

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

```
//@ assert i*j == j*i;
```

Warning: Possible assertion failure

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

```
//@ assert i*j == j*i;
```

Warning: Possible assertion failure

- Floating point types are not well supported

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

```
//@ assert i*j == j*i;
```

Warning: Possible assertion failure

- Floating point types are not well supported

```
//@ assert 1.0 != 2.0;
```

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

```
//@ assert i*j == j*i;
```

Warning: Possible assertion failure

- Floating point types are not well supported

```
//@ assert 1.0 != 2.0;
```

Warning: Possible assertion failure

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

```
//@ assert i*j == j*i;
```

Warning: Possible assertion failure

- Floating point types are not well supported

```
//@ assert 1.0 != 2.0;
```

Warning: Possible assertion failure

- Strings are not well supported

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

```
//@ assert i*j == j*i;
```

Warning: Possible assertion failure

- Floating point types are not well supported

```
//@ assert 1.0 != 2.0;
```

Warning: Possible assertion failure

- Strings are not well supported

```
//@ assert "Hello_World".charAt(0) == 'H';
```

ESC/Java2 is not complete (contd.)

- Multiplication is not well supported

```
//@ assert i*j == j*i;
```

Warning: Possible assertion failure

- Floating point types are not well supported

```
//@ assert 1.0 != 2.0;
```

Warning: Possible assertion failure

- Strings are not well supported

```
//@ assert "Hello_World".charAt(0) == 'H';
```

Warning: Possible assertion failure

Conclusions

- Goal is finding many common programming errors quickly.

Conclusions

- Goal is finding many common programming errors quickly.
- Sacrifices soundness and completeness for automation and ease of use.

Conclusions

- Goal is finding many common programming errors quickly.
- Sacrifices soundness and completeness for automation and ease of use.
- Compared to type checking / runtime assertion checking:
  - Harder to use.
  - Provides stronger guarantees.

Conclusions

- Goal is finding many common programming errors quickly.
- Sacrifices soundness and completeness for automation and ease of use.
- Compared to type checking / runtime assertion checking:
  - Harder to use.
  - Provides stronger guarantees.
- Compared to KeY:
  - Easier to use.
  - Provides weaker guarantees.

Usage

`escjava2` [options] filename

Usage

`escjava2` [options] filename

Some useful command line options:

- `-help`: prints help message

Usage

`escjava2` [options] filename

Some useful command line options:

- `-help`: prints help message
- `-quiet`: suppresses informational messages

Usage

`escjava2` [options] filename

Some useful command line options:

- `-help`: prints help message
- `-quiet`: suppresses informational messages
- `-routine`: allows to check only a particular method

Usage

`escjava2` [options] filename

Some useful command line options:

- `-help`: prints help message
- `-quiet`: suppresses informational messages
- `-routine`: allows to check only a particular method
- `-suggest`: suggest specifications for eliminating warnings

Usage

`escjava2` [options] filename

Some useful command line options:

- `-help`: prints help message
- `-quiet`: suppresses informational messages
- `-routine`: allows to check only a particular method
- `-suggest`: suggest specifications for eliminating warnings
- `-counterexample`: show full counterexamples

Usage

`escjava2` [options] filename

Some useful command line options:

- `-help`: prints help message
- `-quiet`: suppresses informational messages
- `-routine`: allows to check only a particular method
- `-suggest`: suggest specifications for eliminating warnings
- `-counterexample`: show full counterexamples
- `-pgc`: print guarded commands

Usage

`escjava2` [options] filename

Some useful command line options:

- `-help`: prints help message
- `-quiet`: suppresses informational messages
- `-routine`: allows to check only a particular method
- `-suggest`: suggest specifications for eliminating warnings
- `-counterexample`: show full counterexamples
- `-pgc`: print guarded commands
- `-pvc`: print verification condition

THE

THE
END