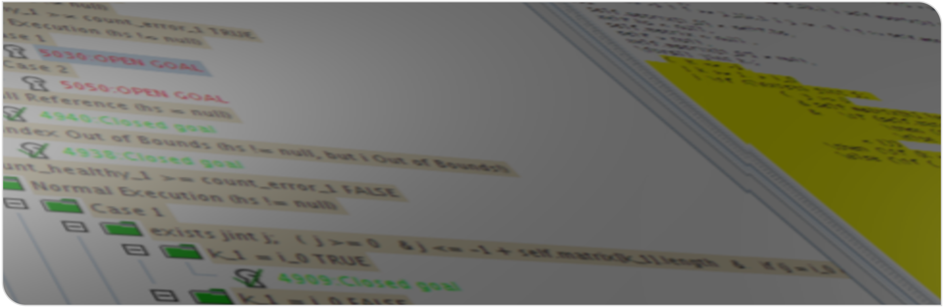
Karlsruhe Institute of Technology

Interactive Verification of Java Programs with JML and KeY

Wolfram Pfeifer | February 9, 2023



KeY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

Dynamic Logic (JavaDL), sequent calculus

Automatic and interactive application of rules

Symbolic Execution

Dynamic Frames

Counterexample generation

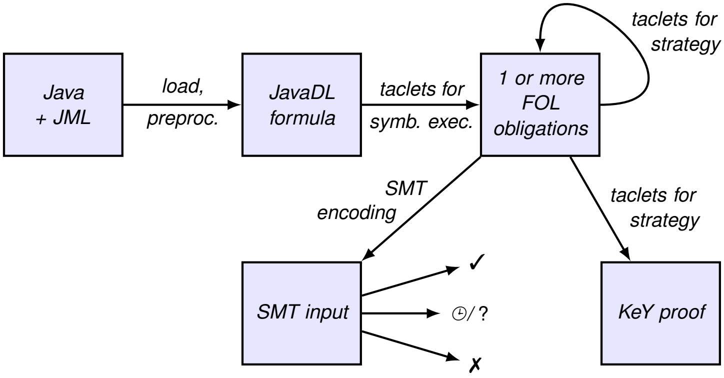
Information flow proofs

Testcase generation

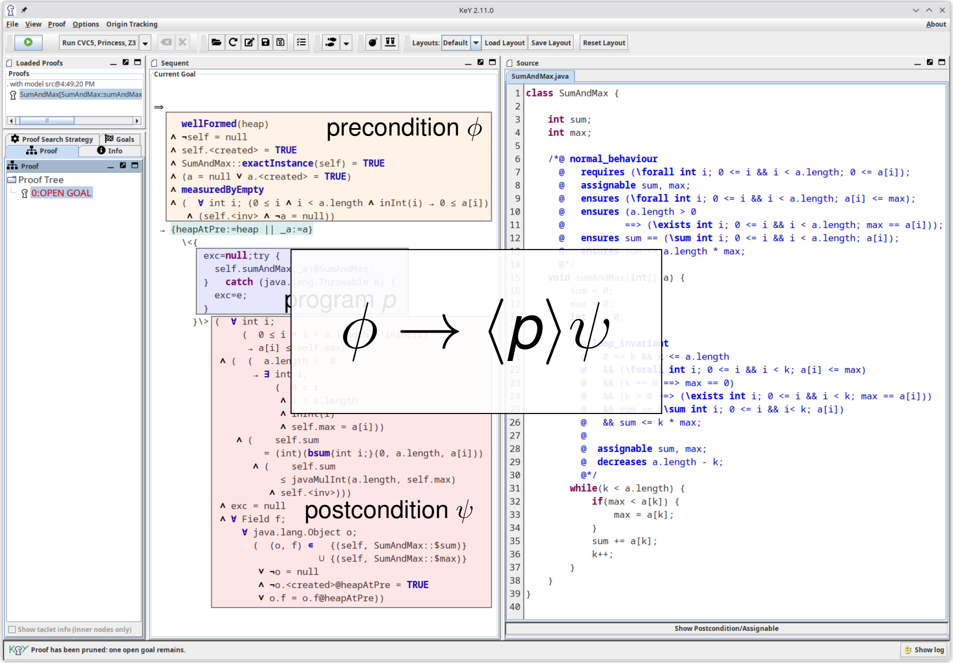
. . .

`https://www.key-project.org`
`https://github.com/KeYProject/key`

Workflow



Java + JML
load, preproc.
JavaDL formula
taclets for symb. exec.
1 or more FOL obligations
taclets for strategy
SMT encoding
taclets for strategy
SMT input
✓
⌚/ ?
✗
KeY proof



```
⇒
  wellFormed(heap)
∧ ¬self = null
∧ self.<created> = TRUE
∧ SumAndMax::exactInstance(self) = TRUE
∧ (a = null ∨ a.<created> = TRUE)
∧ measuredByEmpty
∧ (  ∀ int i; (0 ≤ i ∧ i < a.length ∧ inInt(i) → 0 ≤ a[i])
   ∧ (self.<inv> ∧ ¬a = null))
→ {heapAtPre:=heap || _a:=a}
    \<{
       exc=null;try {
         self.sumAndMax(_a)@SumAndMax;
       }   catch (java.lang.Throwable e) {
         exc=e;
       }
     }\> (  ∀ int i;
              (  0 ≤ i ∧ i < a.length ∧ inInt(i)
               → a[i] ≤ self.max)
          ∧ (  (  a.length > 0
                → ∃ int i;
                    (  0 ≤ i
                     ∧ i < a.length
                     ∧ inInt(i)
                     ∧ self.max = a[i]))
             ∧ (     self.sum
                  = (int)(bsum{int i;}(0, a.length, a[i]))
                ∧ (     self.sum
                     ≤ javaMulInt(a.length, self.max)
                   ∧ self.<inv>)))
        ∧ exc = null
        ∧ ∀ Field f;
            ∀ java.lang.Object o;
              (  (o, f) ∈   {(self, SumAndMax::$sum)}
                          ∪ {(self, SumAndMax::$max)}
               ∨ ¬o = null
               ∧ ¬o.<created>@heapAtPre = TRUE
               ∨ o.f = o.f@heapAtPre))
```

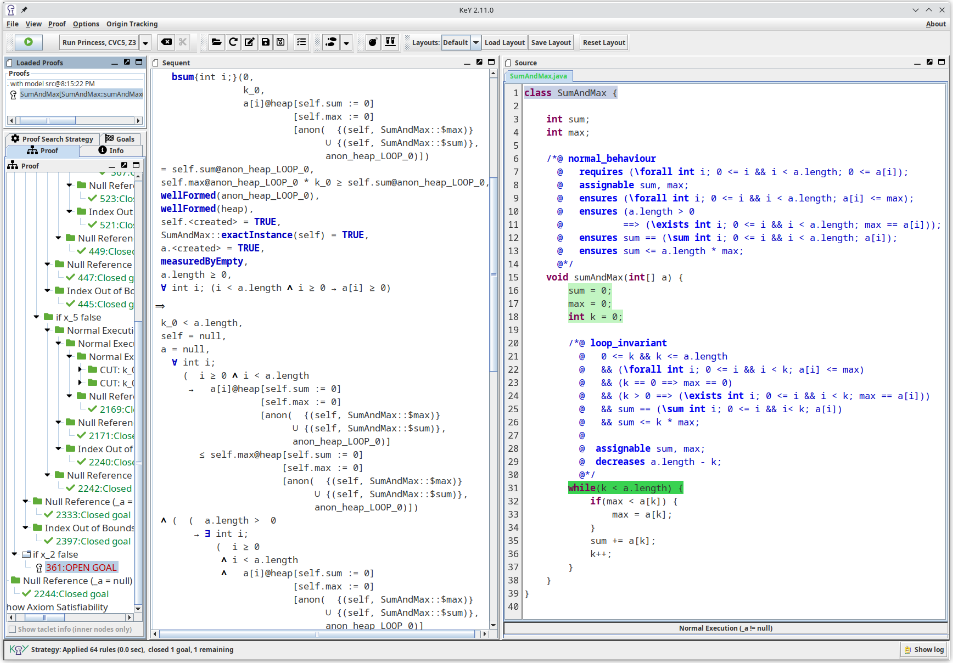
```
class SumAndMax {

    int sum;
    int max;

    /*@ normal_behaviour
      @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
      @   assignable sum, max;
      @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
      @   ensures (a.length > 0
      @         ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
      @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
      @   ensures sum <= a.length * max;
      @*/
    void sumAndMax(int[] a) {
        sum = 0;
        max = 0;
        int k = 0;

        /*@ loop_invariant
          @   0 <= k && k <= a.length
          @   && (\forall int i; 0 <= i && i < k; a[i] <= max)
          @   && (k == 0 ==> max == 0)
          @   && (k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]))
          @   && sum == (\sum int i; 0 <= i && i< k; a[i])
          @   && sum <= k * max;
          @
          @ assignable sum, max;
          @ decreases a.length - k;
          @*/
        while(k < a.length) {
            if(max < a[k]) {
                max = a[k];
            }
            sum += a[k];
            k++;
        }
    }
}
```

Interactive Java Verification with KeY

Live Demo!

Case studies

- TimSort (930 LOC, 460 Spec.) 🐞 [Gouw et al. 2015]
- DualPivotQuickSort (340 LOC) ✓ [Beckert et al. 2017]
- IdentityHashMap (140 LOC, 350 Spec.) ✓ [Boer et al. 2022]
- Super-Scalar Sample Sort (900 LOC) ✓
- . . .


KeY 2.11.0
File View Proof Options Origin Tracking About
Run Princess, CVC5, Z3
Layouts: Default Load Layout Save Layout Reset Layout
Loaded Proofs
Proofs
with model src@8:15:22 PM
SumAndMax[SumAndMax::sumAndMax
Proof Search Strategy Goals
Proof Info
Proof
Sequent
Source
SumAndMax.java
Show taclet info (inner nodes only)
Show log

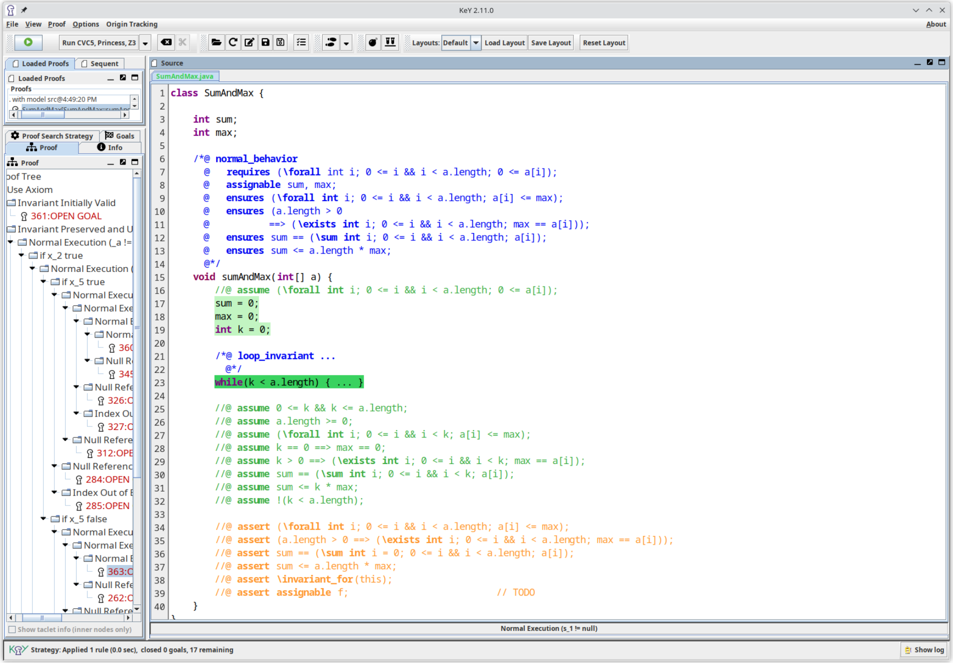
```
307:C
Null Refer
  523:Clo
Index Out
  521:Clo
Null Referen
  449:Closed
Null Reference
  447:Closed g
Index Out of Bo
  445:Closed g
if x_5 false
Normal Executi
Normal Exec
Normal Ex
CUT: k_(
CUT: k_(
Null Refer
  2169:Cl
Null Referen
  2171:Close
Index Out of
  2240:Close
Null Reference
  2242:Closed
Null Reference (_a =
  2333:Closed goal
Index Out of Bounds
  2397:Closed goal
if x_2 false
  361:OPEN GOAL
Null Reference (_a = null)
  2244:Closed goal
how Axiom Satisfiability
```

```
  bsum{int i;}(0,
               k_0,
               a[i]@heap[self.sum := 0]
                        [self.max := 0]
                        [anon(  {(self, SumAndMax::$max)}
                              ∪ {(self, SumAndMax::$sum)},
                              anon_heap_LOOP_0)])
= self.sum@anon_heap_LOOP_0,
self.max@anon_heap_LOOP_0 * k_0 ≥ self.sum@anon_heap_LOOP_0,
wellFormed(anon_heap_LOOP_0),
wellFormed(heap),
self.<created> = TRUE,
SumAndMax::exactInstance(self) = TRUE,
a.<created> = TRUE,
measuredByEmpty,
a.length ≥ 0,
∀ int i; (i < a.length ∧ i ≥ 0 → a[i] ≥ 0)

⇒

k_0 < a.length,
self = null,
a = null,
  ∀ int i;
    (  i ≥ 0 ∧ i < a.length
     →   a[i]@heap[self.sum := 0]
                  [self.max := 0]
                  [anon(  {(self, SumAndMax::$max)}
                        ∪ {(self, SumAndMax::$sum)},
                        anon_heap_LOOP_0)]
       ≤ self.max@heap[self.sum := 0]
                      [self.max := 0]
                      [anon(  {(self, SumAndMax::$max)}
                            ∪ {(self, SumAndMax::$sum)},
                            anon_heap_LOOP_0)])
∧ (  (  a.length >  0
      → ∃ int i;
          (  i ≥ 0
           ∧ i < a.length
           ∧   a[i]@heap[self.sum := 0]
                        [self.max := 0]
                        [anon(  {(self, SumAndMax::$max)}
                              ∪ {(self, SumAndMax::$sum)},
                              anon_heap_LOOP_0)]
```

```
class SumAndMax {

    int sum;
    int max;

    /*@ normal_behaviour
      @  requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
      @  assignable sum, max;
      @  ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
      @  ensures (a.length > 0
      @          ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
      @  ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
      @  ensures sum <= a.length * max;
      @*/
    void sumAndMax(int[] a) {
        sum = 0;
        max = 0;
        int k = 0;

        /*@ loop_invariant
          @   0 <= k && k <= a.length
          @   && (\forall int i; 0 <= i && i < k; a[i] <= max)
          @   && (k == 0 ==> max == 0)
          @   && (k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]))
          @   && sum == (\sum int i; 0 <= i && i< k; a[i])
          @   && sum <= k * max;
          @
          @  assignable sum, max;
          @  decreases a.length - k;
          @*/
        while(k < a.length) {
            if(max < a[k]) {
                max = a[k];
            }
            sum += a[k];
            k++;
        }
    }
}
```

Normal Execution (_a != null)

Strategy: Applied 64 rules (0.0 sec), closed 1 goal, 1 remaining



```
. with model src@4:49:20 PM
SumAndMax[SumAndMax::sumAnd
```

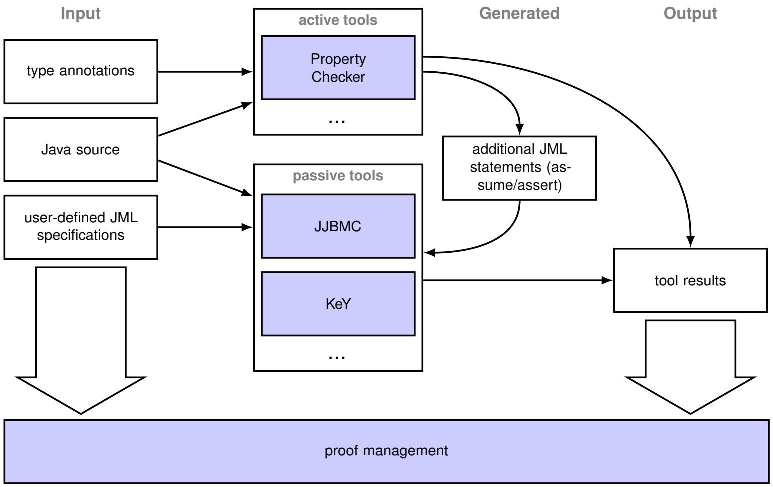
```
oof Tree
Use Axiom
Invariant Initially Valid
  361:OPEN GOAL
Invariant Preserved and U
Normal Execution (_a !=
  if x_2 true
    Normal Execution (
      if x_5 true
        Normal Execu
          Normal Exe
            Normal E
              Norm
                360
              Null R
                345
            Null Refe
              326:O
            Index Ou
              327:O
          Null Refere
            312:OPE
        Null Referenc
          284:OPEN
        Index Out of I
          285:OPEN
      if x_5 false
        Normal Execu
          Normal Exe
            Normal E
              363:O
            Null Refe
              262:O
          Null Refere
```

```java
class SumAndMax {

    int sum;
    int max;

    /*@ normal_behavior
      @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
      @   assignable sum, max;
      @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
      @   ensures (a.length > 0
      @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
      @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
      @   ensures sum <= a.length * max;
      @*/
    void sumAndMax(int[] a) {
        //@ assume (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
        sum = 0;
        max = 0;
        int k = 0;

        /*@ loop_invariant ...
          @*/
        while(k < a.length) { ... }

        //@ assume 0 <= k && k <= a.length;
        //@ assume a.length >= 0;
        //@ assume (\forall int i; 0 <= i && i < k; a[i] <= max);
        //@ assume k == 0 ==> max == 0;
        //@ assume k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]);
        //@ assume sum == (\sum int i; 0 <= i && i < k; a[i]);
        //@ assume sum <= k * max;
        //@ assume !(k < a.length);

        //@ assert (\forall int i; 0 <= i && i < a.length; a[i] <= max);
        //@ assert (a.length > 0 ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
        //@ assert sum == (\sum int i = 0; 0 <= i && i < a.length; a[i]);
        //@ assert sum <= a.length * max;
        //@ assert \invariant_for(this);
        //@ assert assignable f;                        // TODO
    }
```

Normal Execution (s_1 != null)

Strategy: Applied 1 rule (0.0 sec), closed 0 goals, 17 remaining

Tool cooperation



Conclusion

Feel free to try out KeY:

`https://www.key-project.org/download/`

My current work/research

- Interaction concepts
- Tool cooperation (type systems, model checkers, SMT solvers, interactive theorem provers, . . .)
- Engineering: Useful APIs for the community