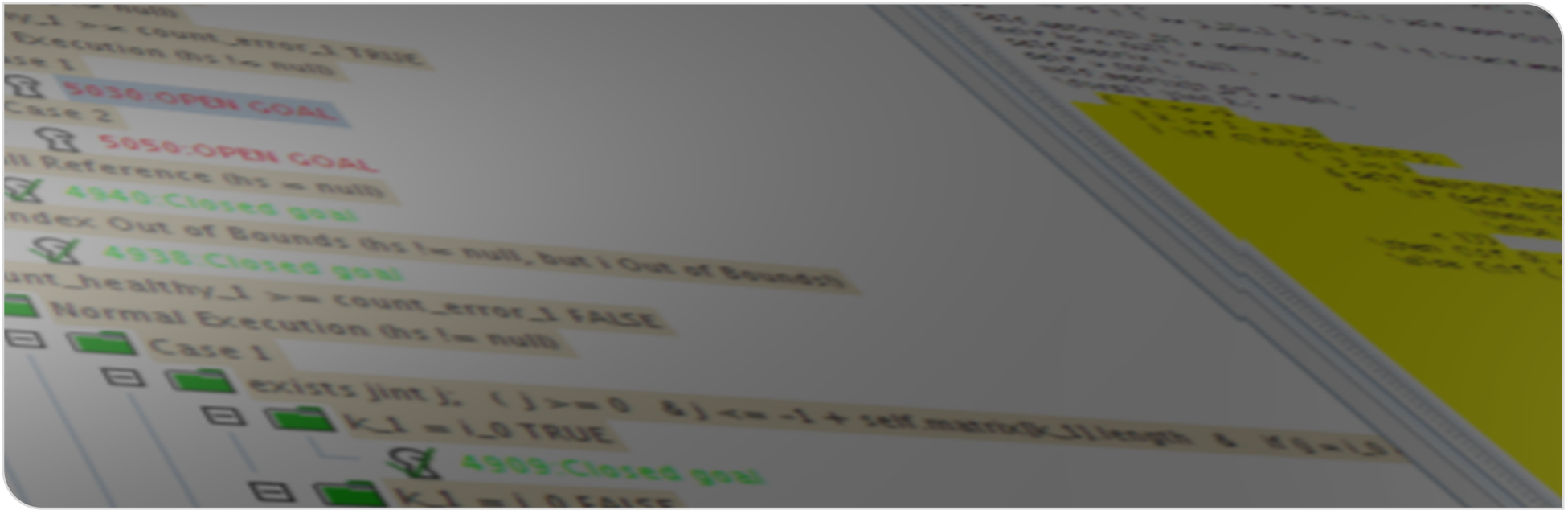
Karlsruhe Institute of Technology

KASTEL

Interactive Verification of Java Programs with JML and KeY

Wolfram Pfeifer | February 9, 2023



KeY

Karlsruhe Institute of Technology

TECHNISCHE UNIVERSITÄT DARMSTADT

CHALMERS UNIVERSITY OF TECHNOLOGY

Deductive verifier for (sequential) Java

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

KeY

Karlsruhe Institute of Technology

TECHNISCHE UNIVERSITÄT DARMSTADT

CHALMERS UNIVERSITY OF TECHNOLOGY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

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

KeY

Karlsruhe Institute of Technology

TECHNISCHE UNIVERSITÄT DARMSTADT

CHALMERS UNIVERSITY OF TECHNOLOGY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

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

KeY

Karlsruhe Institute of Technology

TECHNISCHE UNIVERSITÄT DARMSTADT

CHALMERS UNIVERSITY OF TECHNOLOGY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

Dynamic Logic (JavaDL), sequent calculus

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

KeY

Karlsruhe Institute of Technology

TECHNISCHE UNIVERSITÄT DARMSTADT

CHALMERS UNIVERSITY OF TECHNOLOGY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

Dynamic Logic (JavaDL), sequent calculus

Automatic and interactive application of rules

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

KeY

Karlsruhe Institute of Technology

TECHNISCHE UNIVERSITÄT DARMSTADT

CHALMERS UNIVERSITY OF TECHNOLOGY

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

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

KeY

Karlsruhe Institute of Technology

TECHNISCHE UNIVERSITÄT DARMSTADT

CHALMERS UNIVERSITY OF TECHNOLOGY

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

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

KeY

Karlsruhe Institute of Technology

TECHNISCHE UNIVERSITÄT DARMSTADT

CHALMERS UNIVERSITY OF TECHNOLOGY

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

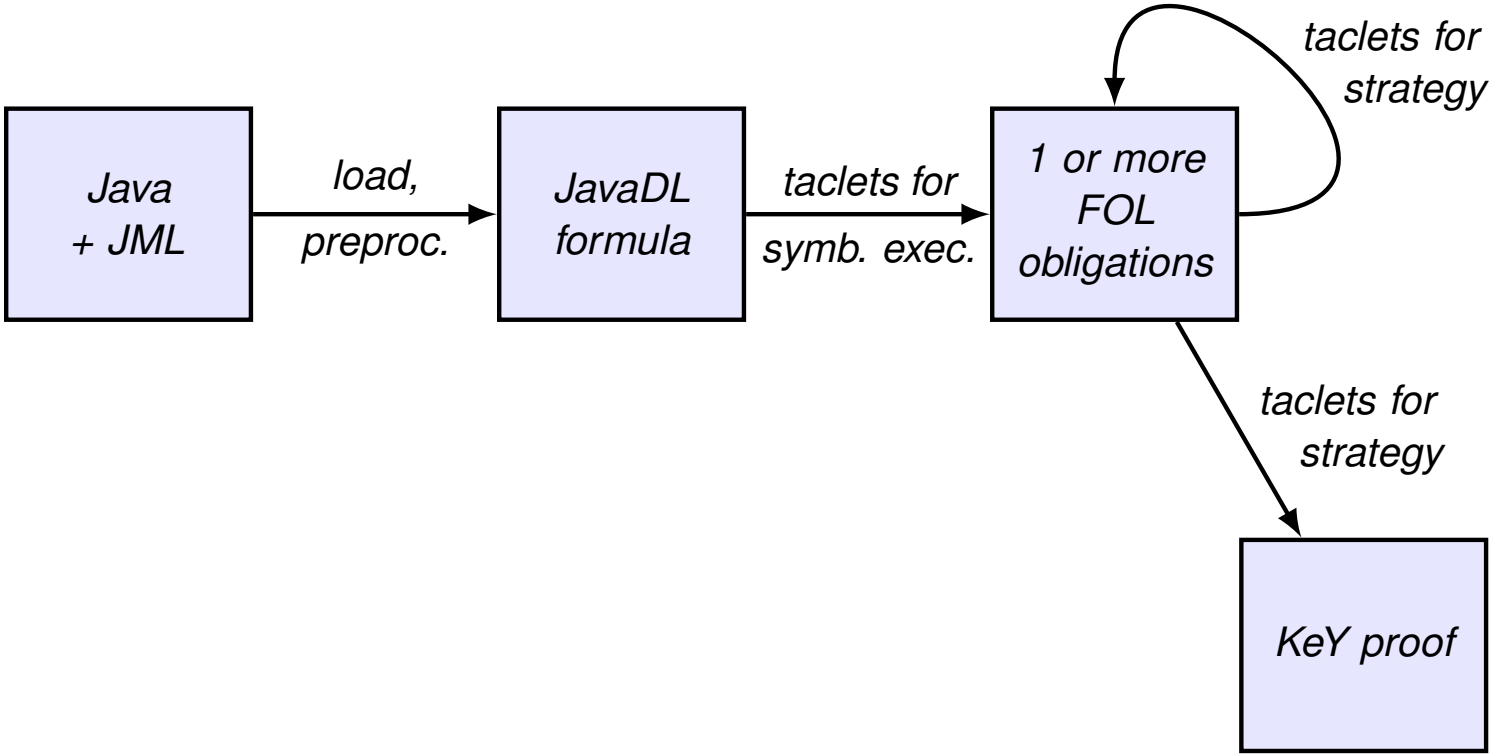
...

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

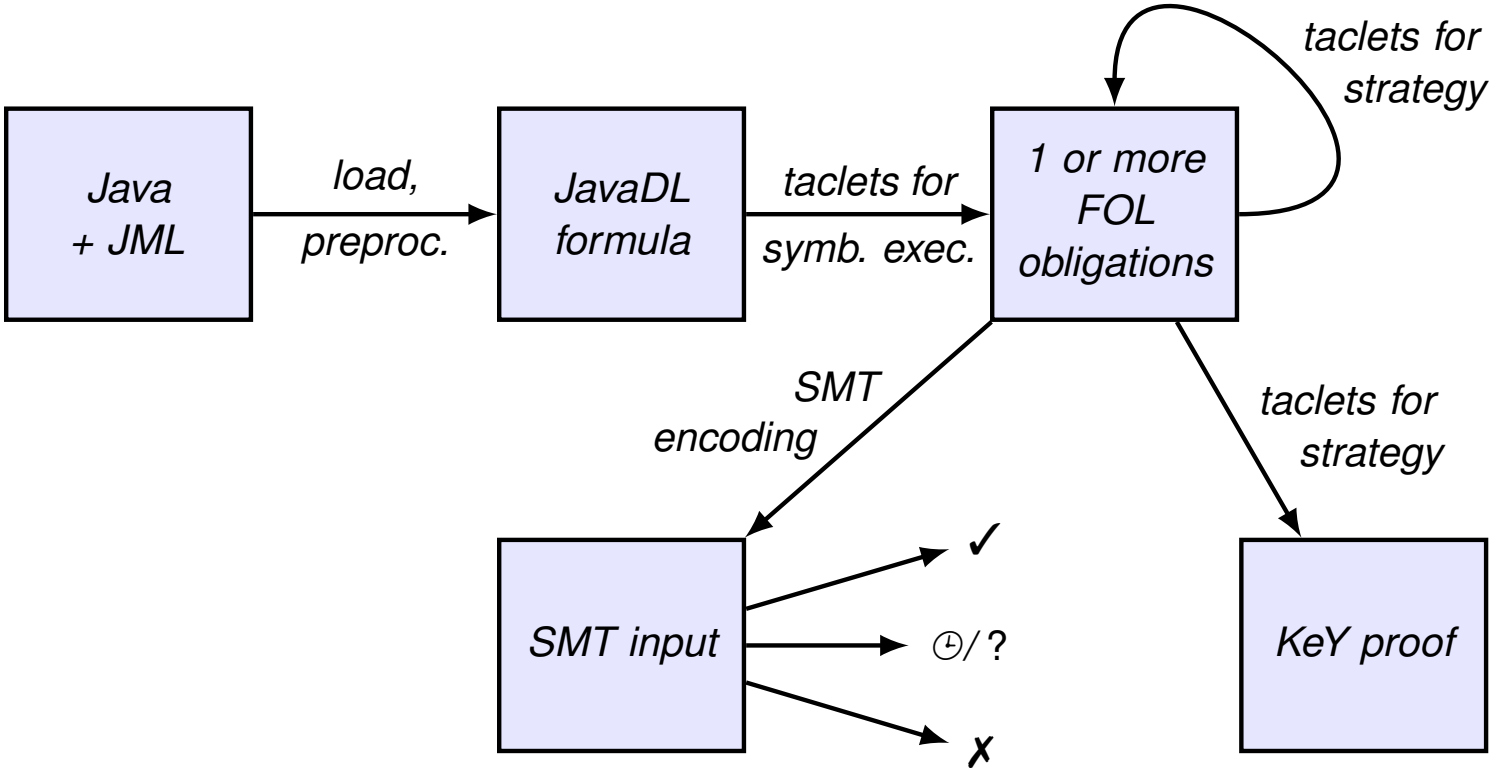
Workflow


Java
+ JML
load,
preproc.
JavaDL
formula
taclets for
symb. exec.
1 or more
FOL
obligations

Workflow


Java
+ JML
load,
preproc.
JavaDL
formula
taclets for
symb. exec.
1 or more
FOL
obligations
taclets for
strategy
taclets for
strategy
KeY proof

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

KeY 2.11.0

File View Proof Options Origin Tracking About

Run CVC5, Princess, Z3 | Layouts: Default | Load Layout | Save Layout | Reset Layout

Loaded Proofs

Proofs

. with model src@4:49:20 PM

SumAndMax[SumAndMax::sumAndMax

Proof Search Strategy | Goals | Proof | Info

Proof

Proof Tree

0:OPEN GOAL

Show taclet info (inner nodes only)

Sequent

Current Goal

```
⇒
    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))
```

Source

SumAndMax.java

```
1  class SumAndMax {
2
3      int sum;
4      int max;
5
6      /*@ normal_behaviour
7        @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
8        @   assignable sum, max;
9        @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
10       @   ensures (a.length > 0
11       @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
12       @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
13       @   ensures sum <= a.length * max;
14       @*/
15     void sumAndMax(int[] a) {
16         sum = 0;
17         max = 0;
18         int k = 0;
19
20         /*@ loop_invariant
21           @   0 <= k && k <= a.length
22           @   && (\forall int i; 0 <= i && i < k; a[i] <= max)
23           @   && (k == 0 ==> max == 0)
24           @   && (k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]))
25           @   && sum == (\sum int i; 0 <= i && i< k; a[i])
26           @   && sum <= k * max;
27           @
28           @  assignable sum, max;
29           @  decreases a.length - k;
30           @*/
31         while(k < a.length) {
32             if(max < a[k]) {
33                 max = a[k];
34             }
35             sum += a[k];
36             k++;
37         }
38     }
39 }
40
```

Show Postcondition/Assignable

Proof has been pruned: one open goal remains.

Show log

KeY 2.11.0

File View Proof Options Origin Tracking About

Run CVC5, Princess, Z3 | Layouts: Default | Load Layout | Save Layout | Reset Layout

Loaded Proofs

Proofs

. with model src@4:49:20 PM

SumAndMax[SumAndMax::sumAndMax

Proof Search Strategy | Goals | Proof | Info

Proof

Proof Tree

0:OPEN GOAL

Show taclet info (inner nodes only)

Sequent

Current Goal

precondition ϕ

```
⇒
  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))
```

Source

SumAndMax.java

```
1  class SumAndMax {
2
3      int sum;
4      int max;
5
6      /*@ normal_behaviour
7        @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
8        @   assignable sum, max;
9        @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
10       @   ensures (a.length > 0
11       @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
12       @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
13       @   ensures sum <= a.length * max;
14       @*/
15     void sumAndMax(int[] a) {
16         sum = 0;
17         max = 0;
18         int k = 0;
19
20         /*@ loop_invariant
21           @   0 <= k && k <= a.length
22           @   && (\forall int i; 0 <= i && i < k; a[i] <= max)
23           @   && (k == 0 ==> max == 0)
24           @   && (k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]))
25           @   && sum == (\sum int i; 0 <= i && i< k; a[i])
26           @   && sum <= k * max;
27           @
28           @  assignable sum, max;
29           @  decreases a.length - k;
30           @*/
31         while(k < a.length) {
32             if(max < a[k]) {
33                 max = a[k];
34             }
35             sum += a[k];
36             k++;
37         }
38     }
39 }
40
```

Show Postcondition/Assignable

Proof has been pruned: one open goal remains.

Show log

KeY 2.11.0

File View Proof Options Origin Tracking About

Run CVC5, Princess, Z3 | Layouts: Default | Load Layout | Save Layout | Reset Layout

Loaded Proofs

Proofs

. with model src@4:49:20 PM

SumAndMax[SumAndMax::sumAndMax

Proof Search Strategy | Goals | Proof | Info

Proof

Proof Tree

0:OPEN GOAL

Show taclet info (inner nodes only)

Sequent

Current Goal

precondition ϕ

```
⇒
  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))
```

program p

Source

SumAndMax.java

```
1  class SumAndMax {
2
3      int sum;
4      int max;
5
6      /*@ normal_behaviour
7        @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
8        @   assignable sum, max;
9        @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
10       @   ensures (a.length > 0
11       @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
12       @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
13       @   ensures sum <= a.length * max;
14       @*/
15     void sumAndMax(int[] a) {
16         sum = 0;
17         max = 0;
18         int k = 0;
19
20         /*@ loop_invariant
21           @   0 <= k && k <= a.length
22           @   && (\forall int i; 0 <= i && i < k; a[i] <= max)
23           @   && (k == 0 ==> max == 0)
24           @   && (k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]))
25           @   && sum == (\sum int i; 0 <= i && i< k; a[i])
26           @   && sum <= k * max;
27           @
28           @  assignable sum, max;
29           @  decreases a.length - k;
30           @*/
31         while(k < a.length) {
32             if(max < a[k]) {
33                 max = a[k];
34             }
35             sum += a[k];
36             k++;
37         }
38     }
39 }
40
```

Show Postcondition/Assignable

Proof has been pruned: one open goal remains.

Show log

KeY 2.11.0

File View Proof Options Origin Tracking About

Run CVC5, Princess, Z3 | Layouts: Default | Load Layout | Save Layout | Reset Layout

Loaded Proofs

Proofs

. with model src@4:49:20 PM

SumAndMax[SumAndMax::sumAndMax

Proof Search Strategy | Goals | Proof | Info

Proof

Proof Tree

0:OPEN GOAL

Show taclet info (inner nodes only)

Sequent

Current Goal

precondition ϕ

```
⇒
  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;
      }
```

program p

```
    }\>(  ∀ 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))
```

postcondition ψ

Source

SumAndMax.java

```
1  class SumAndMax {
2
3      int sum;
4      int max;
5
6      /*@ normal_behaviour
7        @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
8        @   assignable sum, max;
9        @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
10       @   ensures (a.length > 0
11       @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
12       @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
13       @   ensures sum <= a.length * max;
14       @*/
15     void sumAndMax(int[] a) {
16         sum = 0;
17         max = 0;
18         int k = 0;
19
20         /*@ loop_invariant
21           @   0 <= k && k <= a.length
22           @   && (\forall int i; 0 <= i && i < k; a[i] <= max)
23           @   && (k == 0 ==> max == 0)
24           @   && (k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]))
25           @   && sum == (\sum int i; 0 <= i && i< k; a[i])
26           @   && sum <= k * max;
27           @
28           @  assignable sum, max;
29           @  decreases a.length - k;
30           @*/
31         while(k < a.length) {
32             if(max < a[k]) {
33                 max = a[k];
34             }
35             sum += a[k];
36             k++;
37         }
38     }
39 }
40
```

Show Postcondition/Assignable

Proof has been pruned: one open goal remains.

Show log

KeY 2.11.0

File View Proof Options Origin Tracking About

Run CVC5, Princess, Z3 | Layouts: Default | Load Layout | Save Layout | Reset Layout

Loaded Proofs

Proofs

. with model src@4:49:20 PM

SumAndMax[SumAndMax::sumAndMax

Proof Search Strategy | Goals | Proof | Info

Proof

Proof Tree

0:OPEN GOAL

Show taclet info (inner nodes only)

Sequent

Current Goal

precondition ϕ

```
⇒
  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
    }   catch (java.
      exc=e;
    }
  }\>(  ∀ int i;
        (  0 ≤ i
         → a[i] ≤
   ∧ (  (  a.len
         → ∃ int
            (
             ∧
             ∧ 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))
```

$$\phi \rightarrow \langle p \rangle \psi$$

postcondition ψ

Source

SumAndMax.java

```
1  class SumAndMax {
2
3      int sum;
4      int max;
5
6      /*@ normal_behaviour
7        @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
8        @   assignable sum, max;
9        @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
10       @   ensures (a.length > 0
11       @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
12       @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
                                    a.length * max;
                                    a) {
                                    t
                                    <= a.length
                                    int i; 0 <= i && i < k; a[i] <= max)
                                    ==> max == 0)
                                    => (\exists int i; 0 <= i && i < k; max == a[i]))
                                    \sum int i; 0 <= i && i< k; a[i])
26         @   && sum <= k * max;
27         @
28         @  assignable sum, max;
29         @  decreases a.length - k;
30         @*/
31       while(k < a.length) {
32           if(max < a[k]) {
33               max = a[k];
34           }
35           sum += a[k];
36           k++;
37       }
38   }
39 }
40
```

Show Postcondition/Assignable

Proof has been pruned: one open goal remains.

Show log

Interactive Java Verification with KeY

Karlsruhe Institute of Technology

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

- Null Refer
  - 523:Clo
- Index Out
  - 521:Clo
- Null Referen
  - 449:Close
- 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

Show taclet info (inner nodes only)

Sequent

```
  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)]
```

Source

SumAndMax.java

```
1  class SumAndMax {
2
3      int sum;
4      int max;
5
6      /*@ normal_behaviour
7        @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
8        @   assignable sum, max;
9        @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
10       @   ensures (a.length > 0
11       @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
12       @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
13       @   ensures sum <= a.length * max;
14       @*/
15     void sumAndMax(int[] a) {
16         sum = 0;
17         max = 0;
18         int k = 0;
19
20         /*@ loop_invariant
21           @   0 <= k && k <= a.length
22           @   && (\forall int i; 0 <= i && i < k; a[i] <= max)
23           @   && (k == 0 ==> max == 0)
24           @   && (k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]))
25           @   && sum == (\sum int i; 0 <= i && i< k; a[i])
26           @   && sum <= k * max;
27           @
28           @  assignable sum, max;
29           @  decreases a.length - k;
30           @*/
31         while(k < a.length) {
32             if(max < a[k]) {
33                 max = a[k];
34             }
35             sum += a[k];
36             k++;
37         }
38     }
39 }
40
```

Normal Execution (_a != null)

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

Show log

KeY 2.11.0

File View Proof Options Origin Tracking About

Run CVC5, Princess, Z3 | Layouts: Default | Load Layout | Save Layout | Reset Layout

Loaded Proofs | Sequent

Loaded Proofs

Proofs

. with model src@4:49:20 PM

Proof Search Strategy | Goals | Proof | Info

Proof

pof 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
Norma
360
Null R
345
Null Refe
326:O
Index Ou
327:O
Null Refere
312:OPE
Null Referenc
284:OPEN
Index Out of E
285:OPEN
if x_5 false
Normal Execu
Normal Exe
Normal E
363:O
Null Refe
262:O
Null Refere

Show taclet info (inner nodes only)

Source

SumAndMax.java

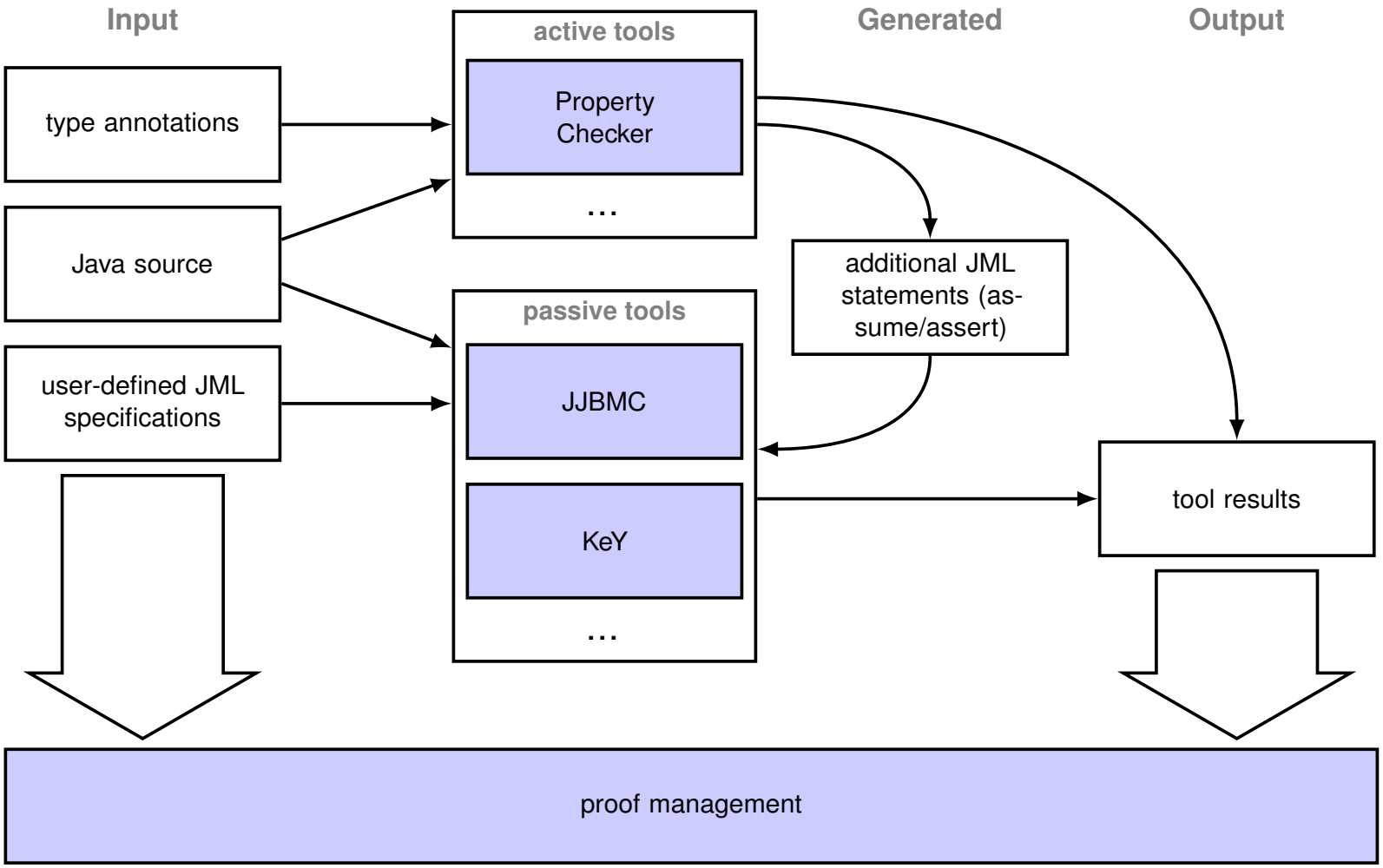
```
1  class SumAndMax {
2
3      int sum;
4      int max;
5
6      /*@ normal_behavior
7        @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
8        @   assignable sum, max;
9        @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
10       @   ensures (a.length > 0
11       @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
12       @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
13       @   ensures sum <= a.length * max;
14       @*/
15     void sumAndMax(int[] a) {
16         //@ assume (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
17         sum = 0;
18         max = 0;
19         int k = 0;
20
21         /*@ loop_invariant ...
22           @*/
23         while(k < a.length) { ... }
24
25         //@ assume 0 <= k && k <= a.length;
26         //@ assume a.length >= 0;
27         //@ assume (\forall int i; 0 <= i && i < k; a[i] <= max);
28         //@ assume k == 0 ==> max == 0;
29         //@ assume k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]);
30         //@ assume sum == (\sum int i; 0 <= i && i < k; a[i]);
31         //@ assume sum <= k * max;
32         //@ assume !(k < a.length);
33
34         //@ assert (\forall int i; 0 <= i && i < a.length; a[i] <= max);
35         //@ assert (a.length > 0 ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
36         //@ assert sum == (\sum int i = 0; 0 <= i && i < a.length; a[i]);
37         //@ assert sum <= a.length * max;
38         //@ assert \invariant_for(this);
39         //@ assert assignable f;                          // TODO
40     }
}
```

Normal Execution (s_1 != null)

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

Show log

Tool cooperation



Conclusion

Karlsruhe Institute of Technology

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