Karlsruhe Institute of Technology

KASTEL

Advancements in User Interface and Usability of KeY

Wolfram Pfeifer | September 12, 2023

KeY

Deductive verifier for (sequential) Java

KeY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

KeY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

KeY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

Dynamic Logic (JavaDL), sequent calculus

KeY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

Dynamic Logic (JavaDL), sequent calculus

Symbolic Execution

KeY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

Dynamic Logic (JavaDL), sequent calculus

Symbolic Execution

Dynamic Frames

KeY

Karlsruhe Institute of Technology

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

Dynamic Logic (JavaDL), sequent calculus

Symbolic Execution

Dynamic Frames

Automatic and interactive application of rules

KeY

Deductive verifier for (sequential) Java

Java Modeling Language (JML)

Modular specification/verification

Dynamic Logic (JavaDL), sequent calculus

Symbolic Execution

Dynamic Frames

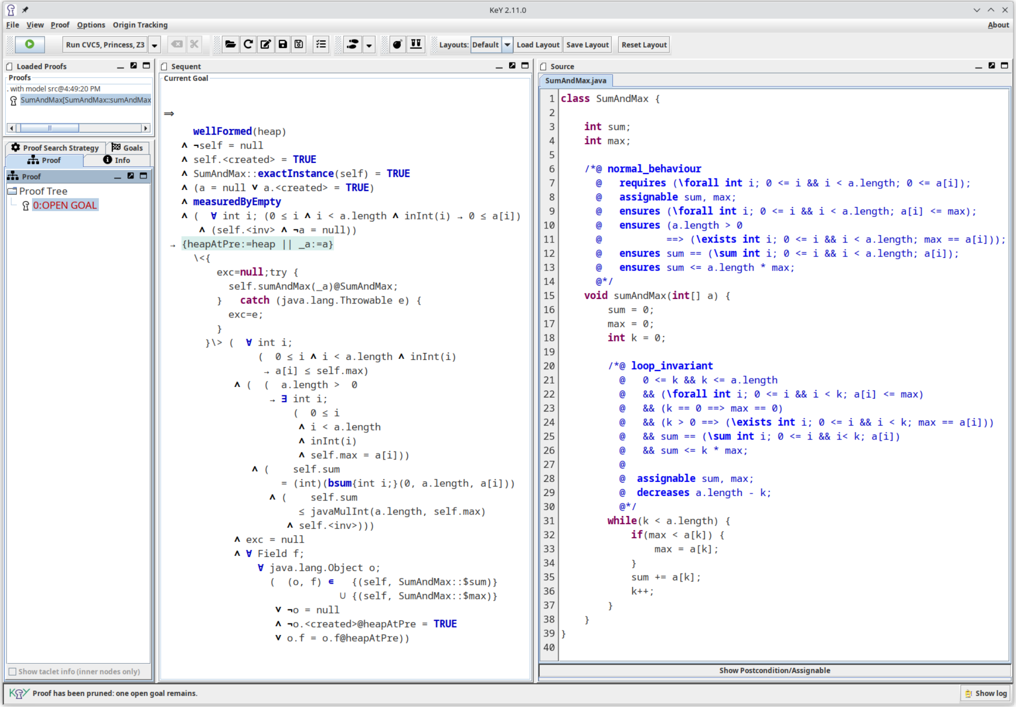
Automatic and interactive application of rules

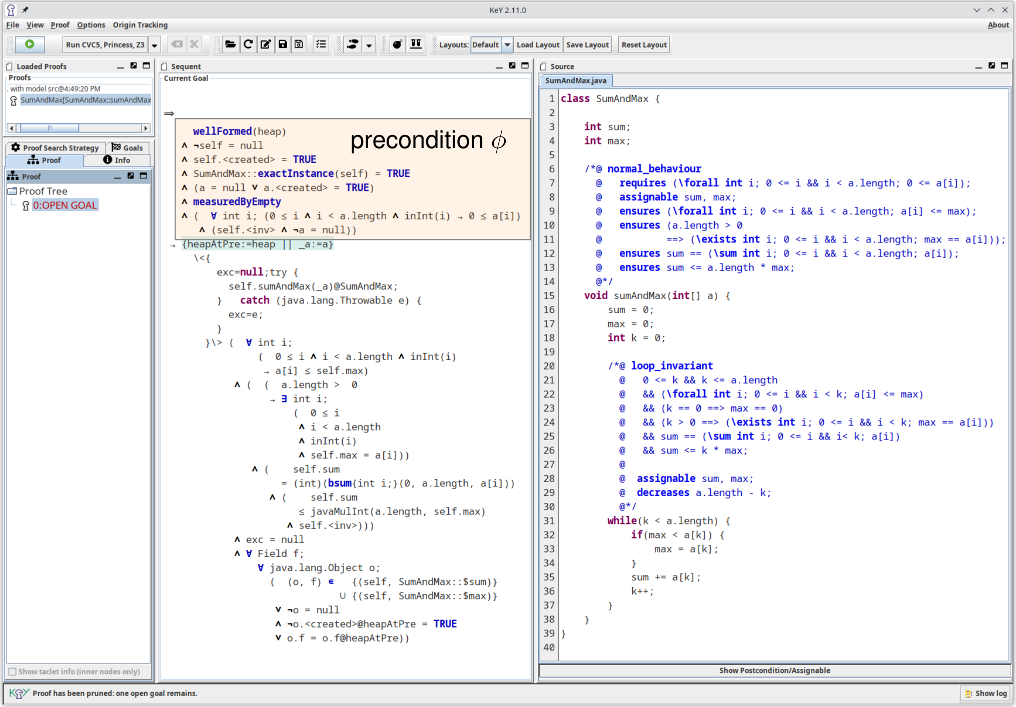
Optional translation to SMT-LIB

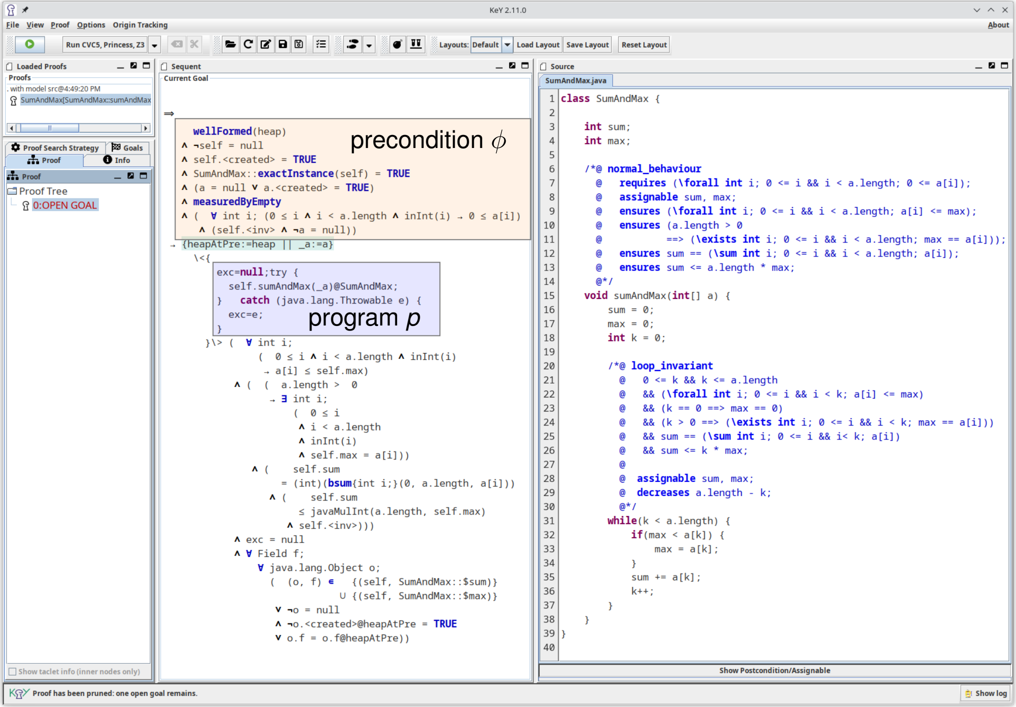
KeY

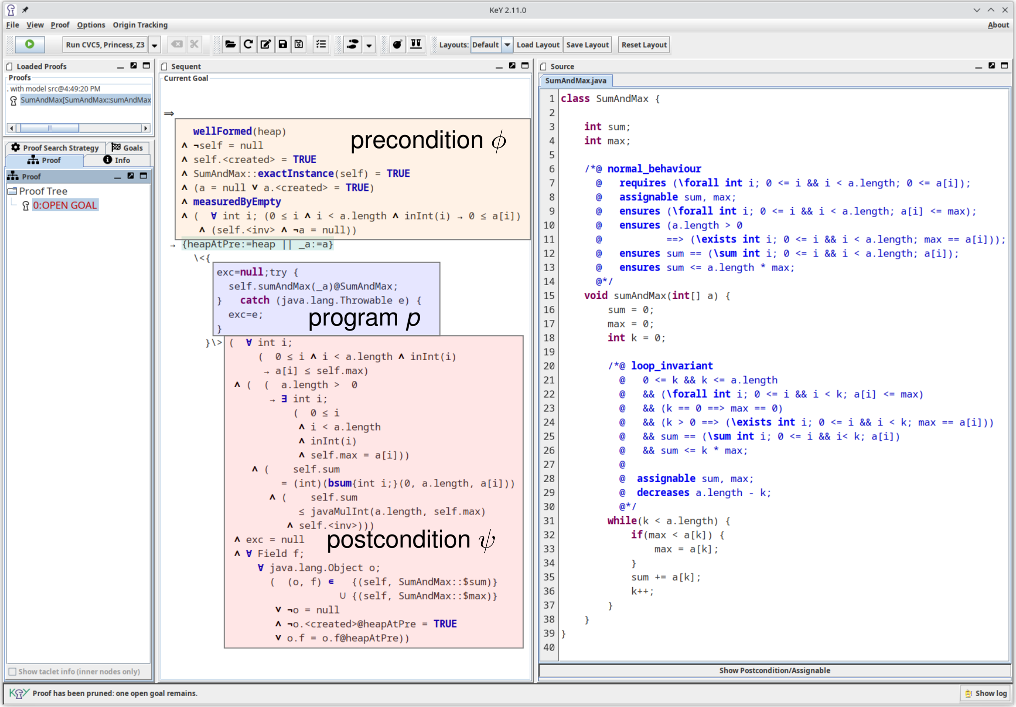
Karlsruhe Institute of Technology

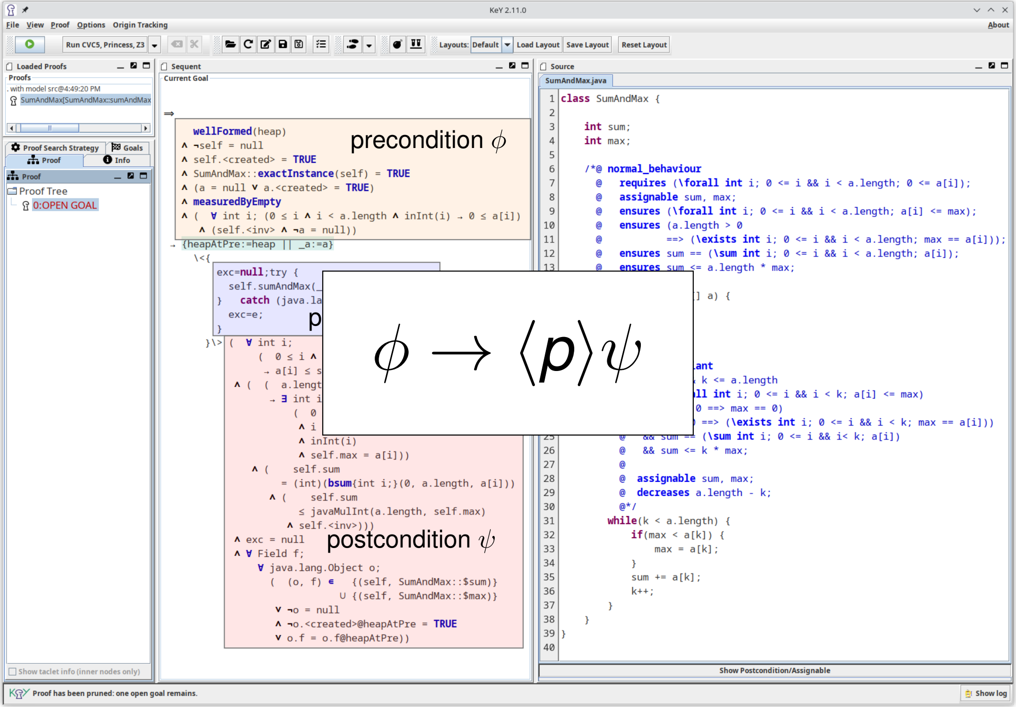
- Deductive verifier for (sequential) Java
- Java Modeling Language (JML)
- Modular specification/verification
- Dynamic Logic (JavaDL), sequent calculus
- Symbolic Execution
- Dynamic Frames
- Automatic and interactive application of rules
- Optional translation to SMT-LIB
- Counterexample generation
- Information flow proofs
- Testcase generation
- . . .


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
⟹
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))
precondition ϕ
→ {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
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++;
}
}
}
Show Postcondition/Assignable
Proof has been pruned: one open goal remains.
Show log

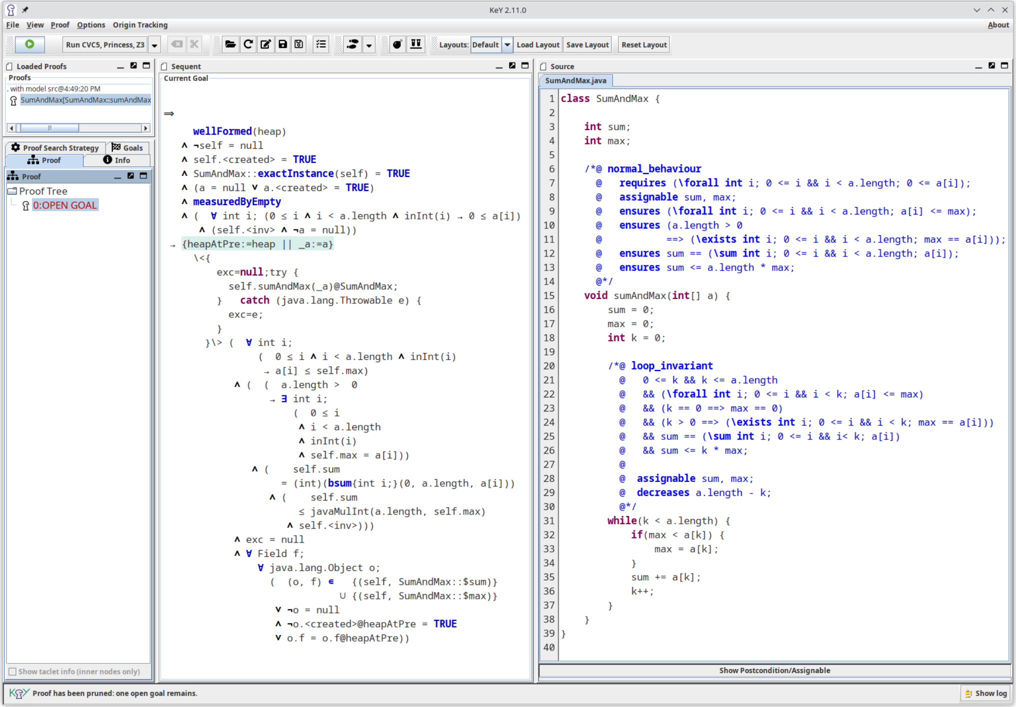

KeY 2.11.0
File View Proof Options Origin Tracking
About
Run CVC5, Princess, Z3
Layouts: Default
Load Layout
Save Layout
Reset Layout
Loaded Proofs
Proofs
. with model src@4:49:20 PM
SumAndMax[SumAndMax::sumAndMax
Proof Search Strategy
Goals
Proof
Info
Proof Tree
0:OPEN GOAL
Show taclet info (inner nodes 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))
precondition ϕ
→ {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))
Source
SumAndMax.java
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++;
}
}
}
Show Postcondition/Assignable
Proof has been pruned: one open goal remains.
Show log

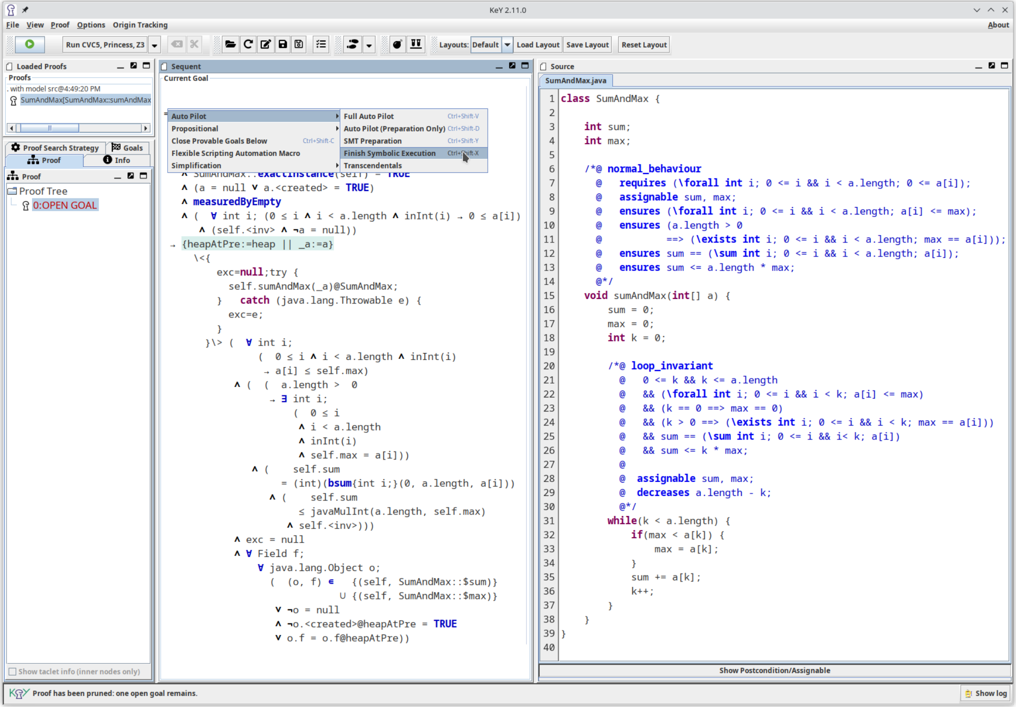

KeY 2.11.0
File View Proof Options Origin Tracking About
Run CVC5, Princess, Z3
Layouts: Default Load Layout Save Layout Reset Layout
Loaded Proofs
Proofs
. with model src@4:49:20 PM
SumAndMax[SumAndMax::sumAndMax
Proof Search Strategy Goals
Proof Info
Proof
Proof Tree
0:OPEN GOAL
Show taclet info (inner nodes only)
Sequent
Current Goal
⟹
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))
precondition ϕ
→ {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
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++;
}
}
}
Show Postcondition/Assignable
Proof has been pruned: one open goal remains.
Show log

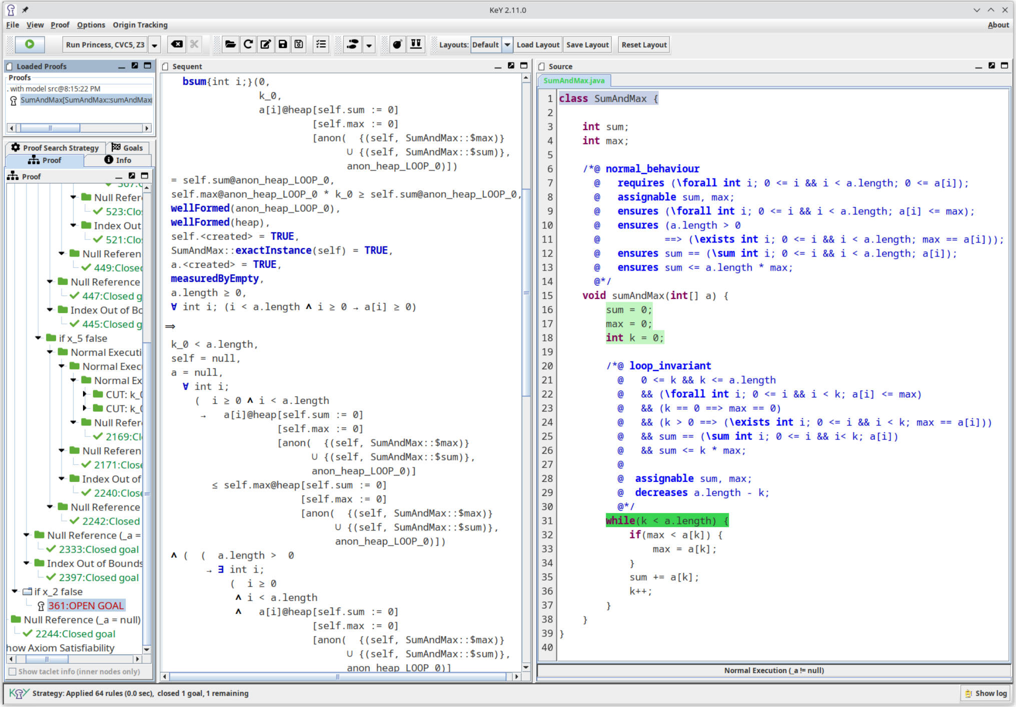


KeY 2.11.0
File View Proof Options Origin Tracking
About
Run CVC5, Princess, Z3
Layouts: Default Load Layout Save Layout Reset Layout
Loaded Proofs
Proofs
. with model src@4:49:20 PM
SumAndMax[SumAndMax::sumAndMax
Proof Search Strategy
Goals
Proof
Info
Proof Tree
0:OPEN GOAL
Show taclet info (inner nodes 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))
precondition ϕ
{heapAtPre:=heap || _a:=a}
exc=null;try {
self.sumAndMax(_
} catch (java.la
exc=e;
}
ϕ → ⟨p⟩ψ
∧ 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
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;
@ && 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++;
}
}
}
Show Postcondition/Assignable
Proof has been pruned: one open goal remains.
Show log

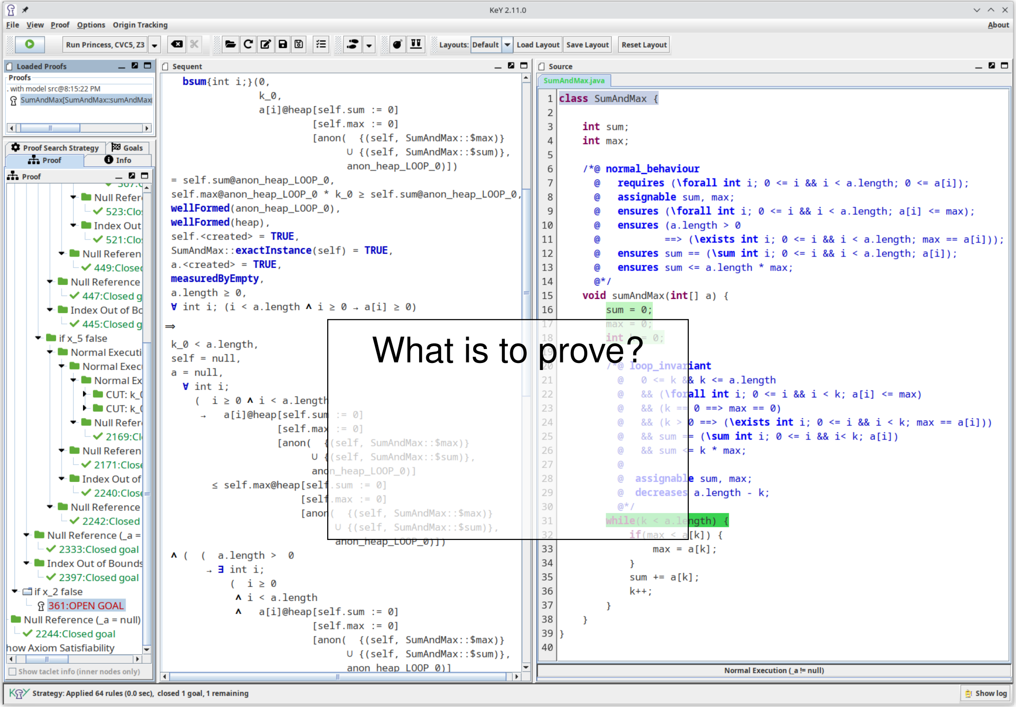
Part 1: Interaction on Source Code Level

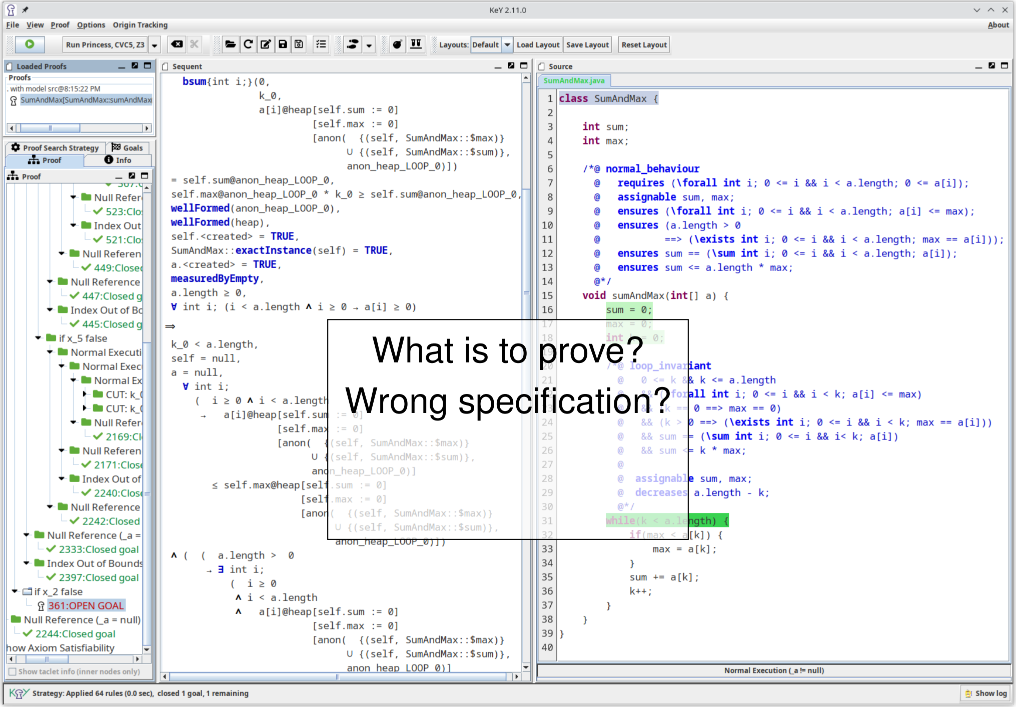

KeY 2.11.0
File View Proof Options Origin Tracking
About
Run CVC5, Princess, Z3
Layouts: Default
Load Layout
Save Layout
Reset Layout
Loaded Proofs
Proofs
. with model src@4:49:20 PM
SumAndMax[SumAndMax::sumAndMax
Proof Search Strategy
Goals
Proof
Info
Proof
Proof Tree
0:OPEN GOAL
Show taclet info (inner 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
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++;
}
}
}
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
Auto Pilot
Propositional
Close Provable Goals Below Ctrl+Shift-C
Flexible Scripting Automation Macro
Simplification
Full Auto Pilot Ctrl+Shift-V
Auto Pilot (Preparation Only) Ctrl+Shift-D
SMT Preparation Ctrl+Shift-Y
Finish Symbolic Execution Ctrl+Shift-X
Transcendentals
∧ (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
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++;
}
}
}
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 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
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
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
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
Show log

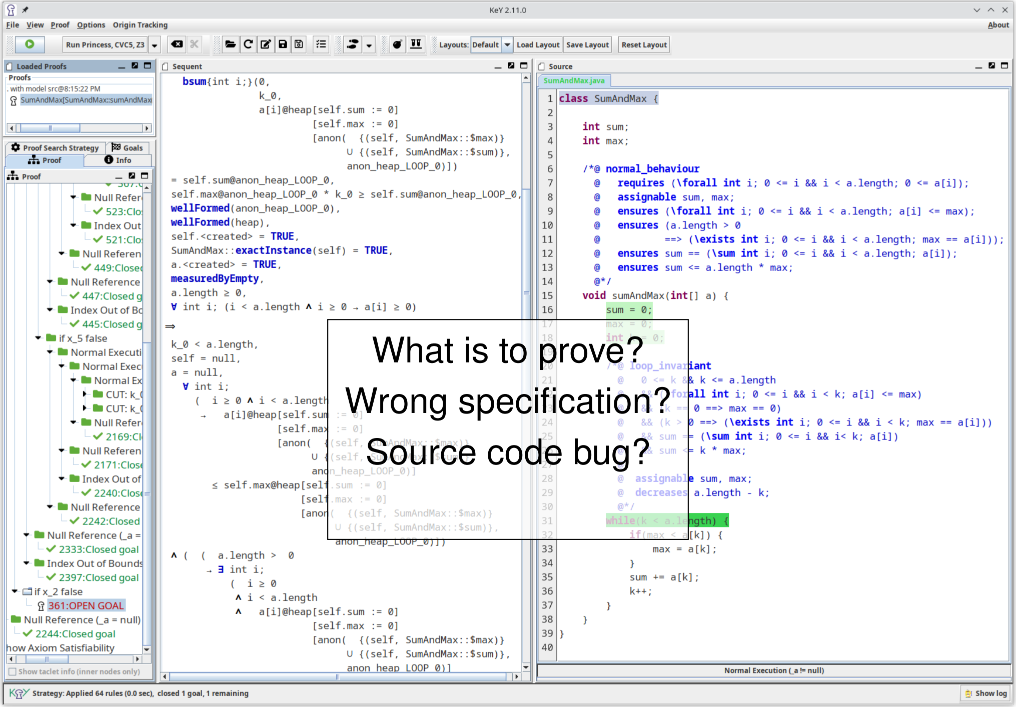
What is to prove?

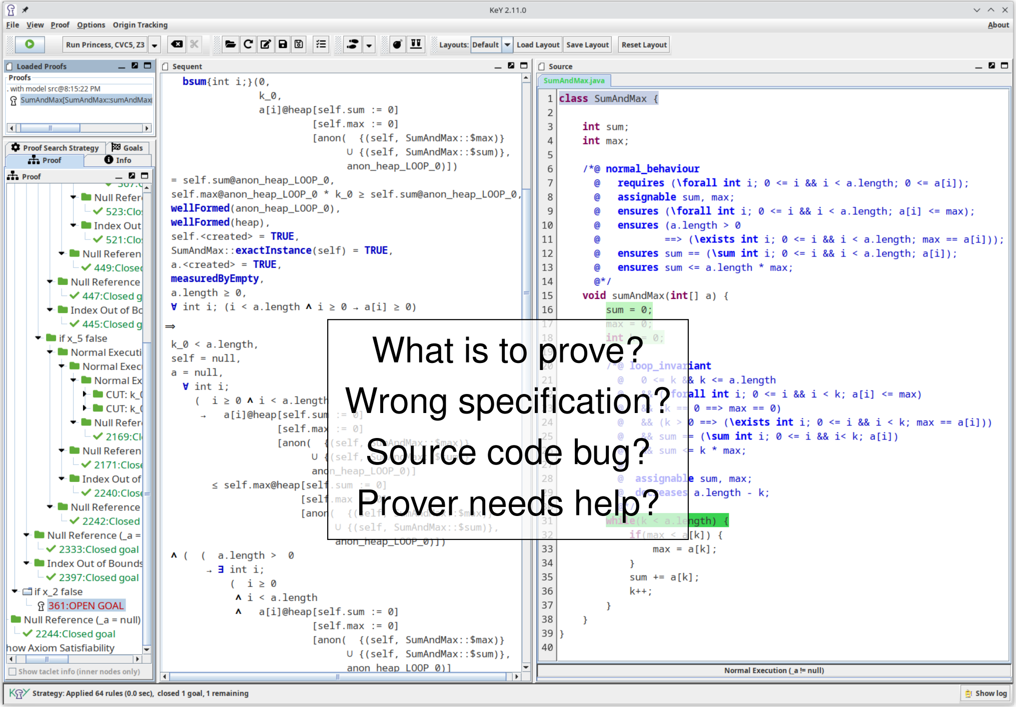




What is to prove?

Wrong specification?


KeY 2.11.0
What is to prove?
Wrong specification?
Source code bug?
361:OPEN GOAL
SumAndMax.java
Strategy: Applied 64 rules (0.0 sec), closed 1 goal, 1 remaining

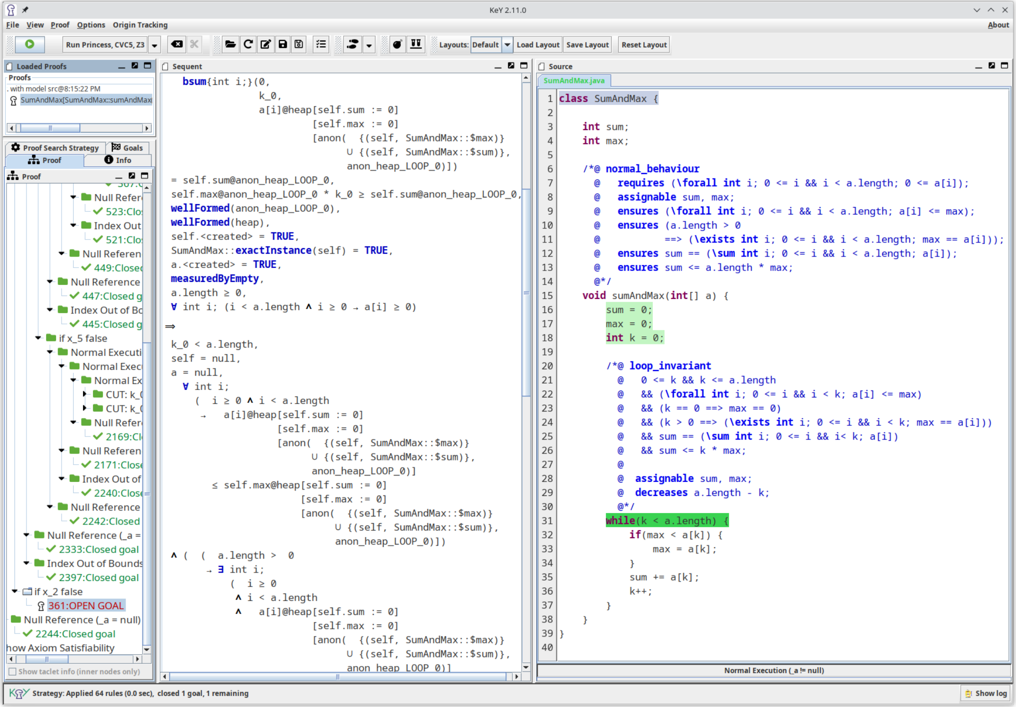


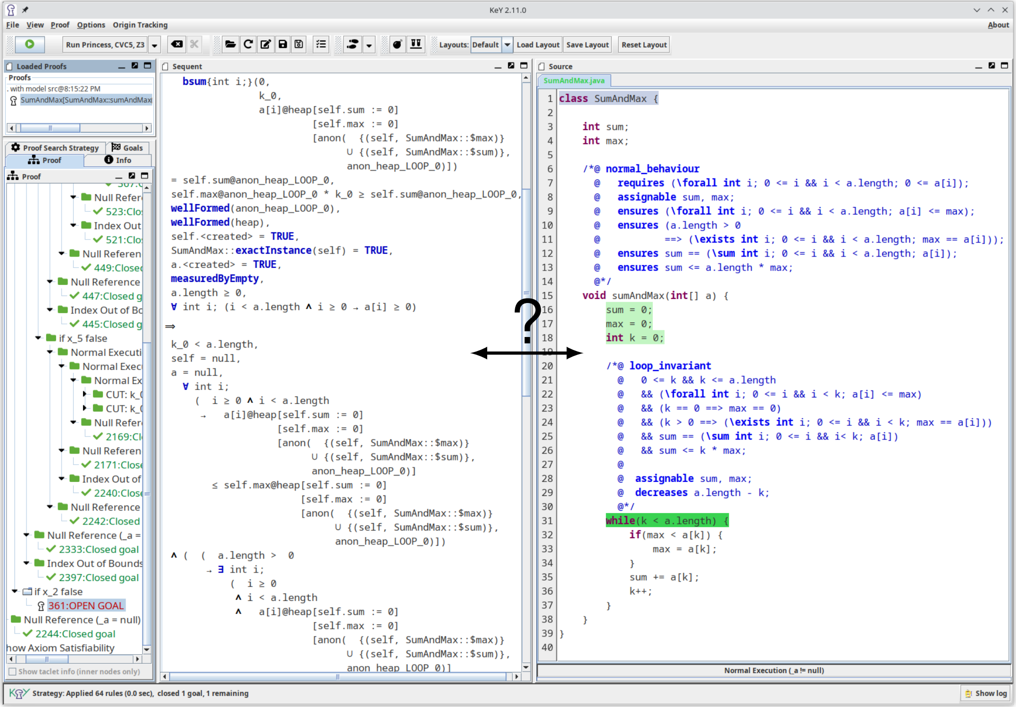
What is to prove?

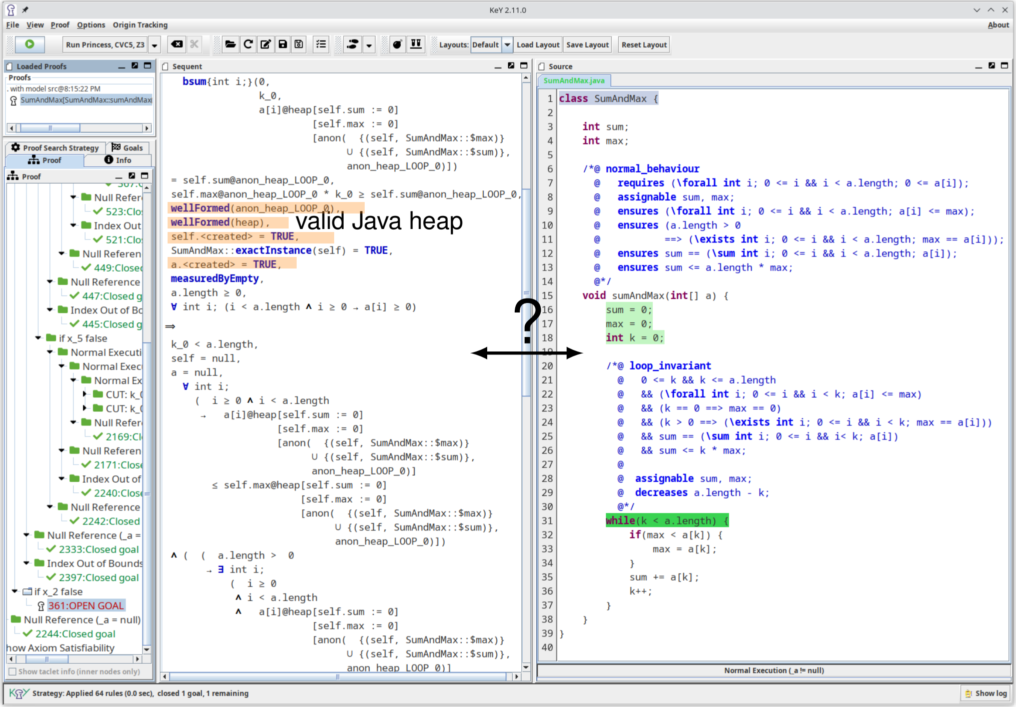
Wrong specification?

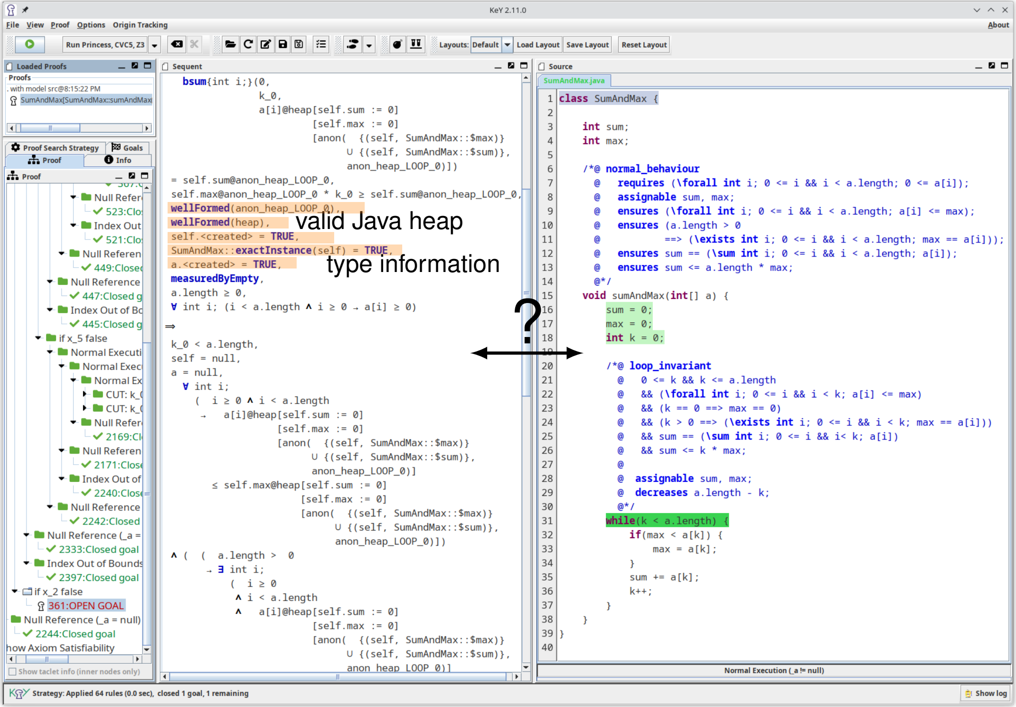
Source code bug?

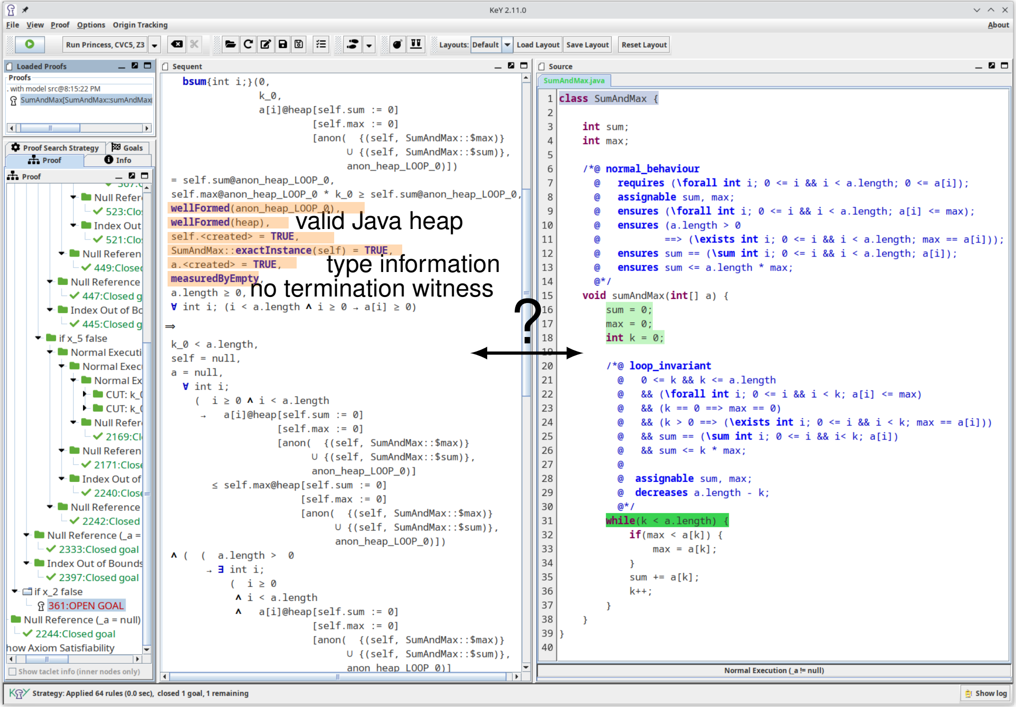
Prover needs help?

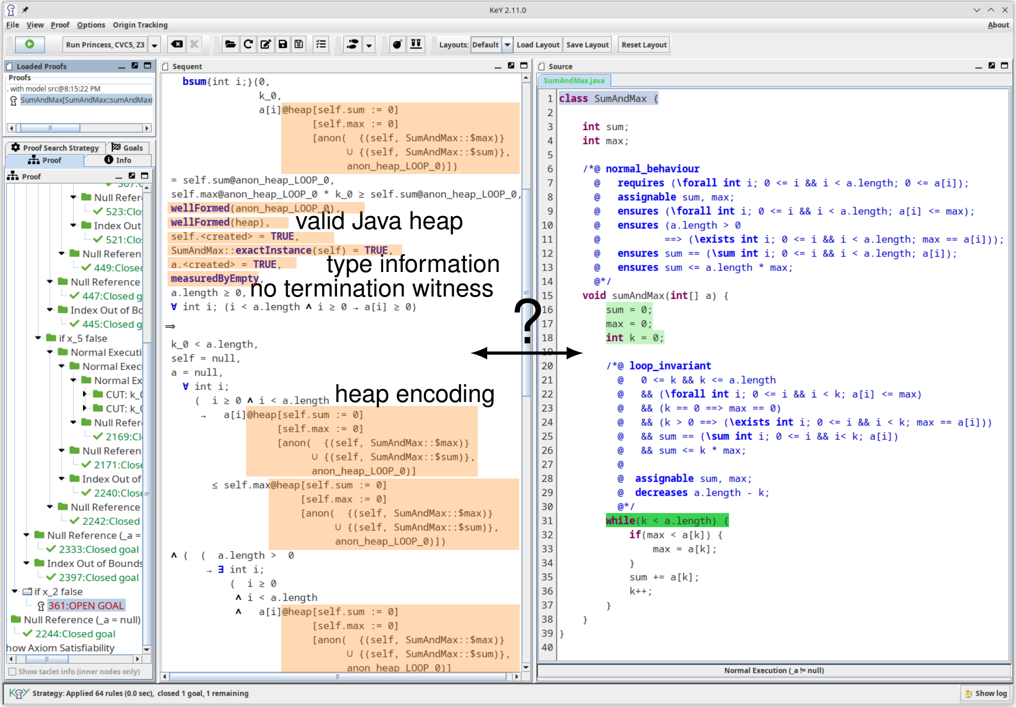

KeY 2.11.0
File View Proof Options Origin Tracking
About
Run Princess, CVC5, Z3
Layouts: Default
Load Layout
Save Layout
Reset Layout
Loaded Proofs
Proofs
. with model src@8:15:22 PM
SumAndMax[SumAndMax::sumAndMax
Proof Search Strategy
Goals
Proof
Info
Proof
Null Refer
523:Clo
Index Out
521:Clo
Null 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
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
Show log


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: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
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
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
Show log


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
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
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),
valid Java 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
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
Show log


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
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
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),
valid Java heap
self.<created> = TRUE,
SumAndMax::exactInstance(self) = TRUE,
a.<created> = TRUE,
type information
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 Princess, CVC5, Z3
Layouts: Default Load Layout Save Layout Reset Layout
Loaded Proofs
Sequent
Source
SumAndMax.java
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,
valid Java heap
type information
no termination witness
?
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)
361:OPEN GOAL
Strategy: Applied 64 rules (0.0 sec), closed 1 goal, 1 remaining
Show log


KeY 2.11.0
File View Proof Options Origin Tracking
Run Princess, CVC5, Z3
Layouts: Default Load Layout Save Layout Reset Layout
Loaded Proofs
Sequent
Source
SumAndMax.java
valid Java heap
type information
no termination witness
heap encoding
?
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,
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)
361:OPEN GOAL
Strategy: Applied 64 rules (0.0 sec), closed 1 goal, 1 remaining
Show log

Progress so far

"Integrating Source Code, Specification and Proof State into a Single Interactive View for the Deductive Verification Tool KeY" (Master's Thesis, Mike Schwörer)

Idea: Represent a goal (sequent) of the proof as JML.

Progress so far

“Integrating Source Code, Specification and Proof State into a Single Interactive View for the Deductive Verification Tool KeY” (Master’s Thesis, Mike Schwörer)

Idea: Represent a goal (sequent) of the proof as JML.

1. Take initial PO and assign origins/categories to the terms
2. Transform correctly under rule applications
3. Render the new view:
   Input: Sequent with origin/category tags, Java/JML
   Output: Source code with additional JML assume/assert statements placed

Progress so far

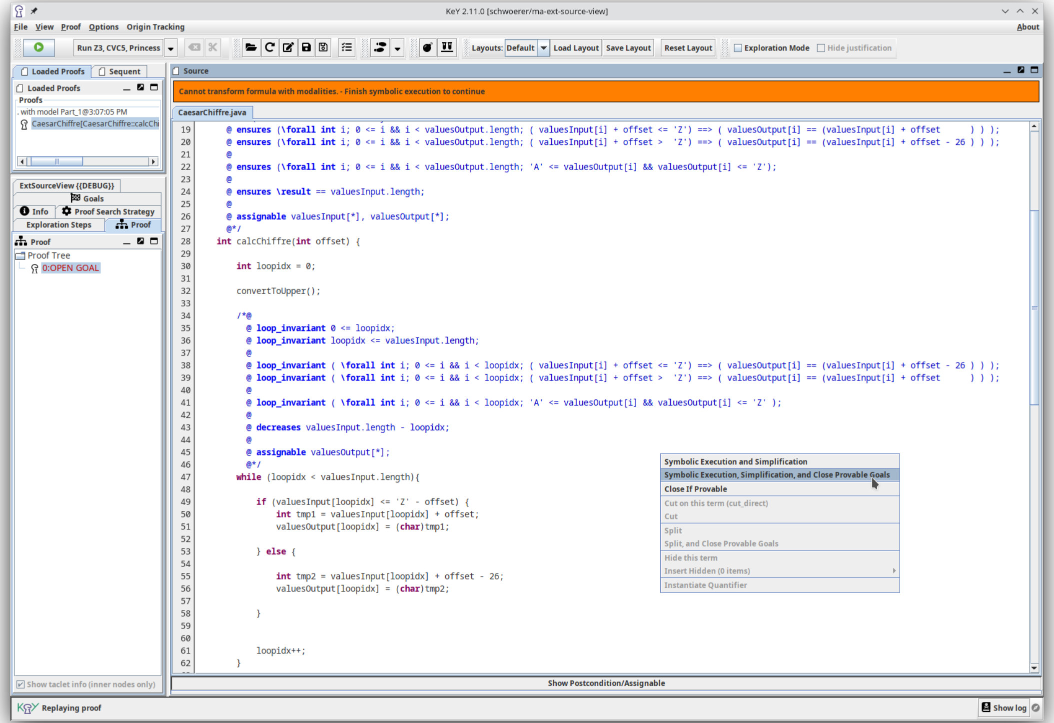
"Integrating Source Code, Specification and Proof State into a Single Interactive View for the Deductive Verification Tool KeY" (Master's Thesis, Mike Schwörer)

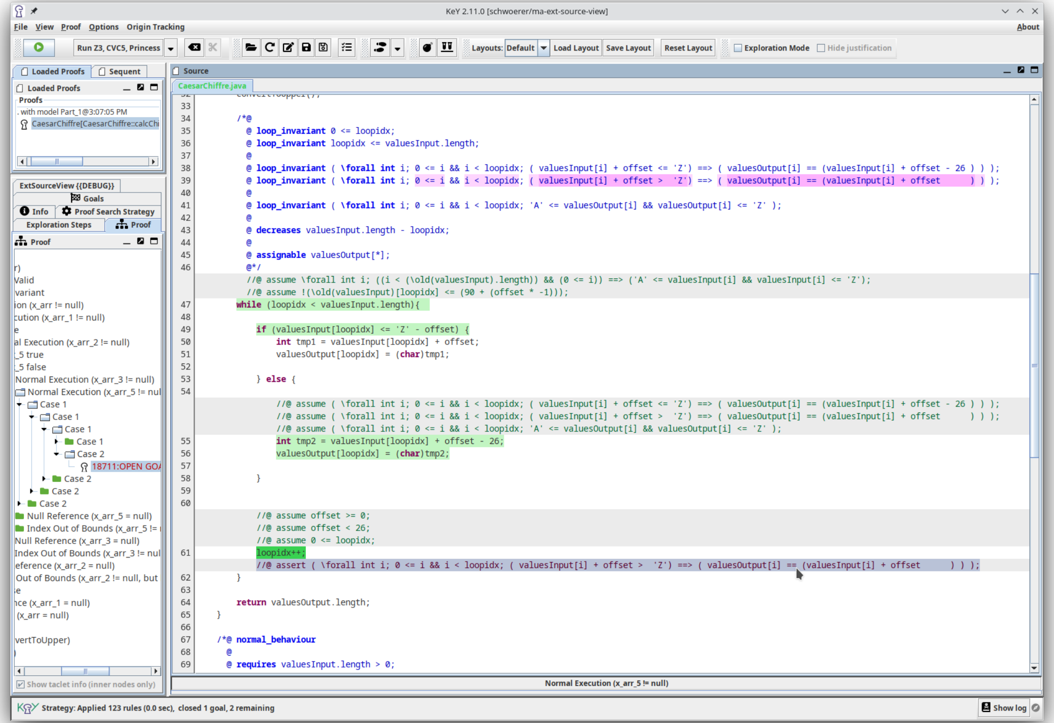
Idea: Represent a goal (sequent) of the proof as JML.

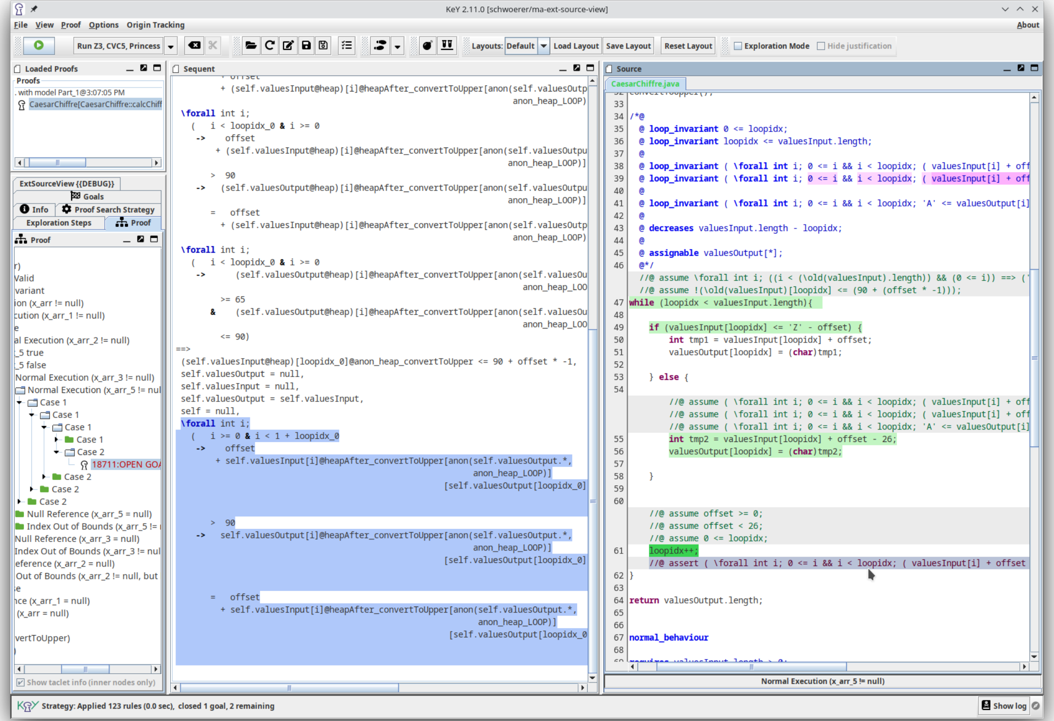
1. Take initial PO and assign origins/categories to the terms
2. Transform correctly under rule applications
3. Render the new view:
   Input: Sequent with origin/category tags, Java/JML
   Output: Source code with additional JML assume/assert statements placed

Assumptions

- Symbolic execution has finished (no modalities).
- All updates are applied.
- Restrictions to allowed programs (e.g., no `for` loops, only `return` + variable, ...).


KeY 2.11.0 [schwoerer/ma-ext-source-view]
File View Proof Options Origin Tracking
About
Run Z3, CVC5, Princess
Layouts: Default
Load Layout
Save Layout
Reset Layout
Exploration Mode
Hide justification
Loaded Proofs
Sequent
Proofs
. with model Part_1@3:07:05 PM
CaesarChiffre[CaesarChiffre::calcChi
ExtSourceView {{DEBUG}}
Goals
Info
Proof Search Strategy
Exploration Steps
Proof
Proof Tree
0:OPEN GOAL
Show taclet info (inner nodes only)
Source
Cannot transform formula with modalities. - Finish symbolic execution to continue
CaesarChiffre.java
@ ensures (\forall int i; 0 <= i && i < valuesOutput.length; (valuesInput[i] + offset <= 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset)));
@ ensures (\forall int i; 0 <= i && i < valuesOutput.length; (valuesInput[i] + offset > 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset - 26)));
@
@ ensures (\forall int i; 0 <= i && i < valuesOutput.length; 'A' <= valuesOutput[i] && valuesOutput[i] <= 'Z');
@
@ ensures \result == valuesInput.length;
@
@ assignable valuesInput[*], valuesOutput[*];
@*/
int calcChiffre(int offset) {
int loopidx = 0;
convertToUpper();
/*@
@ loop_invariant 0 <= loopidx;
@ loop_invariant loopidx <= valuesInput.length;
@
@ loop_invariant (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + offset <= 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset - 26)));
@ loop_invariant (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + offset > 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset)));
@
@ loop_invariant (\forall int i; 0 <= i && i < loopidx; 'A' <= valuesOutput[i] && valuesOutput[i] <= 'Z');
@
@ decreases valuesInput.length - loopidx;
@
@ assignable valuesOutput[*];
@*/
while (loopidx < valuesInput.length){
if (valuesInput[loopidx] <= 'Z' - offset) {
int tmp1 = valuesInput[loopidx] + offset;
valuesOutput[loopidx] = (char)tmp1;
} else {
int tmp2 = valuesInput[loopidx] + offset - 26;
valuesOutput[loopidx] = (char)tmp2;
}
loopidx++;
}
Symbolic Execution and Simplification
Symbolic Execution, Simplification, and Close Provable Goals
Close If Provable
Cut on this term (cut_direct)
Cut
Split
Split, and Close Provable Goals
Hide this term
Insert Hidden (0 items)
Instantiate Quantifier
Show Postcondition/Assignable
Replaying proof
Show log


KeY 2.11.0 [schwoerer/ma-ext-source-view]
File View Proof Options Origin Tracking About
Run Z3, CVC5, Princess
Layouts: Default Load Layout Save Layout Reset Layout Exploration Mode Hide justification
Loaded Proofs Sequent
Proofs
. with model Part_1@3:07:05 PM
CaesarChiffre[CaesarChiffre::calcChi
ExtSourceView {{DEBUG}}
Goals
Info Proof Search Strategy
Exploration Steps Proof
Proof
Case 1
Case 2
18711:OPEN GOA
Null Reference (x_arr_5 = null)
Index Out of Bounds (x_arr_5 != null
Null Reference (x_arr_3 = null)
Show taclet info (inner nodes only)
Source
CaesarChiffre.java
/*@
@ loop_invariant 0 <= loopidx;
@ loop_invariant loopidx <= valuesInput.length;
@
@ loop_invariant (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + offset <= 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset - 26)));
@ loop_invariant (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + offset > 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset)));
@
@ loop_invariant (\forall int i; 0 <= i && i < loopidx; 'A' <= valuesOutput[i] && valuesOutput[i] <= 'Z');
@
@ decreases valuesInput.length - loopidx;
@
@ assignable valuesOutput[*];
@*/
//@ assume \forall int i; ((i < (\old(valuesInput).length)) && (0 <= i)) ==> ('A' <= valuesInput[i] && valuesInput[i] <= 'Z');
//@ assume !(\old(valuesInput)[loopidx] <= (90 + (offset * -1)));
while (loopidx < valuesInput.length){
if (valuesInput[loopidx] <= 'Z' - offset) {
int tmp1 = valuesInput[loopidx] + offset;
valuesOutput[loopidx] = (char)tmp1;
} else {
//@ assume (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + offset <= 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset - 26)));
//@ assume (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + offset > 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset)));
//@ assume (\forall int i; 0 <= i && i < loopidx; 'A' <= valuesOutput[i] && valuesOutput[i] <= 'Z');
int tmp2 = valuesInput[loopidx] + offset - 26;
valuesOutput[loopidx] = (char)tmp2;
}
//@ assume offset >= 0;
//@ assume offset < 26;
//@ assume 0 <= loopidx;
loopidx++;
//@ assert (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + offset > 'Z') ==> (valuesOutput[i] == (valuesInput[i] + offset)));
}
return valuesOutput.length;
}
/*@ normal_behaviour
@
@ requires valuesInput.length > 0;
Normal Execution (x_arr_5 != null)
Strategy: Applied 123 rules (0.0 sec), closed 1 goal, 2 remaining
Show log


KeY 2.11.0 [schwoerer/ma-ext-source-view]
File View Proof Options Origin Tracking
About
Run Z3, CVC5, Princess
Layouts: Default
Load Layout
Save Layout
Reset Layout
Exploration Mode
Hide justification
Loaded Proofs
Proofs
. with model Part_1@3:07:05 PM
CaesarChiffre[CaesarChiffre::calcChiff
ExtSourceView {{DEBUG}}
Goals
Info
Proof Search Strategy
Exploration Steps
Proof
Proof
r)
Valid
variant
ion (x_arr != null)
cution (x_arr_1 != null)
e
al Execution (x_arr_2 != null)
_5 true
_5 false
Normal Execution (x_arr_3 != null)
Normal Execution (x_arr_5 != nul
Case 1
Case 1
Case 1
Case 1
Case 2
18711:OPEN GOA
Case 2
Case 2
Case 2
Null Reference (x_arr_5 = null)
Index Out of Bounds (x_arr_5 != i
Null Reference (x_arr_3 = null)
Index Out of Bounds (x_arr_3 != nul
eference (x_arr_2 = null)
Out of Bounds (x_arr_2 != null, but
e
nce (x_arr_1 = null)
(x_arr = null)
vertToUpper)
Show taclet info (inner nodes only)
Sequent
+ (self.valuesInput@heap)[i]@heapAfter_convertToUpper[anon(self.valuesOutp
anon_heap_LOOP)
\forall int i;
(i < loopidx_0 & i >= 0
-> offset
+ (self.valuesInput@heap)[i]@heapAfter_convertToUpper[anon(self.valuesOutpu
anon_heap_LOOP)]
> 90
-> (self.valuesOutput@heap)[i]@heapAfter_convertToUpper[anon(self.valuesOutpu
anon_heap_LOOP)]
= offset
+ (self.valuesInput@heap)[i]@heapAfter_convertToUpper[anon(self.valuesOutp
anon_heap_LOOP)
\forall int i;
(i < loopidx_0 & i >= 0
-> (self.valuesOutput@heap)[i]@heapAfter_convertToUpper[anon(self.valuesOu
anon_heap_LOO
>= 65
& (self.valuesOutput@heap)[i]@heapAfter_convertToUpper[anon(self.valuesOu
anon_heap_LOO
<= 90)
==>
(self.valuesInput@heap)[loopidx_0]@anon_heap_convertToUpper <= 90 + offset * -1,
self.valuesOutput = null,
self.valuesInput = null,
self.valuesOutput = self.valuesInput,
self = null,
\forall int i;
(i >= 0 & i < 1 + loopidx_0
-> offset
+ self.valuesInput[i]@heapAfter_convertToUpper[anon(self.valuesOutput.*,
anon_heap_LOOP)]
[self.valuesOutput[loopidx_0]
> 90
-> self.valuesOutput[i]@heapAfter_convertToUpper[anon(self.valuesOutput.*,
anon_heap_LOOP)]
[self.valuesOutput[loopidx_0]
= offset
+ self.valuesInput[i]@heapAfter_convertToUpper[anon(self.valuesOutput.*,
anon_heap_LOOP)]
[self.valuesOutput[loopidx_0
Source
CaesarChiffre.java
33
34 /*@
35 @ loop_invariant 0 <= loopidx;
36 @ loop_invariant loopidx <= valuesInput.length;
37 @
38 @ loop_invariant (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + off
39 @ loop_invariant (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + off
40 @
41 @ loop_invariant (\forall int i; 0 <= i && i < loopidx; 'A' <= valuesOutput[i]
42 @
43 @ decreases valuesInput.length - loopidx;
44 @
45 @ assignable valuesOutput[*];
46 @*/
//@ assume \forall int i; ((i < (\old(valuesInput).length)) && (0 <= i)) ==> ('
//@ assume !(\old(valuesInput)[loopidx] <= (90 + (offset * -1)));
47 while (loopidx < valuesInput.length){
48
49 if (valuesInput[loopidx] <= 'Z' - offset) {
50 int tmp1 = valuesInput[loopidx] + offset;
51 valuesOutput[loopidx] = (char)tmp1;
52
53 } else {
54
//@ assume (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + off
//@ assume (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + off
//@ assume (\forall int i; 0 <= i && i < loopidx; 'A' <= valuesOutput[i]
55 int tmp2 = valuesInput[loopidx] + offset - 26;
56 valuesOutput[loopidx] = (char)tmp2;
57
58 }
59
60
//@ assume offset >= 0;
//@ assume offset < 26;
//@ assume 0 <= loopidx;
61 loopidx++;
//@ assert (\forall int i; 0 <= i && i < loopidx; (valuesInput[i] + offset
62 }
63
64 return valuesOutput.length;
65
66
67 normal_behaviour
68
Normal Execution (x_arr_5 != null)
Strategy: Applied 123 rules (0.0 sec), closed 1 goal, 2 remaining
Show log

Part 2: Proof Slicing

Proof Slicing

Sequent calculus: A sequent $\phi_0, \ldots, \phi_n \implies \psi_0, \ldots, \psi_m$ is valid iff

$\phi_0 \land \ldots \land \phi_n \rightarrow \psi_0 \lor \ldots \lor \psi_m$ is valid.

Proof Slicing

Sequent calculus: A sequent $\phi_0, ..., \phi_n \Longrightarrow \psi_0, ..., \psi_m$ is valid iff

$\phi_0 \wedge ... \wedge \phi_n \rightarrow \psi_0 \vee ... \vee \psi_m$ is valid.

Example proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{p \Longrightarrow q, p}\ \mathit{close}}{p, \neg q \Longrightarrow p}\ \text{notLeft}}{p \wedge \neg q \Longrightarrow p}\ \mathit{andLeft}}{\Longrightarrow (p \wedge \neg q) \rightarrow p}\ \mathit{impRight}$$

Proof Slicing

Sequent calculus: A sequent $\phi_0, \dots, \phi_n \Longrightarrow \psi_0, \dots, \psi_m$ is valid iff

$\phi_0 \land \dots \land \phi_n \rightarrow \psi_0 \lor \dots \lor \psi_m$ is valid.

Example proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{p \Longrightarrow q, p}\;\textit{close}}{p, \neg q \Longrightarrow p}\;\text{notLeft}}{p \land \neg q \Longrightarrow p}\;\textit{andLeft}}{\Longrightarrow (p \land \neg q) \rightarrow p}\;\textit{impRight}$$

Proof Slicing

Karlsruhe Institute of Technology

Sequent calculus: A sequent $\phi_0, \dots, \phi_n \implies \psi_0, \dots, \psi_m$ is valid iff

$\phi_0 \land \dots \land \phi_n \rightarrow \psi_0 \lor \dots \lor \psi_m$ is valid.

Example proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{p \implies q, p}\ \textit{close}}{p, \neg q \implies p}\ \text{notLeft}}{p \land \neg q \implies p}\ \textit{andLeft}}{\implies (p \land \neg q) \rightarrow p}\ \textit{impRight}$$

Proof Slicing

Sequent calculus: A sequent $\phi_0, \ldots, \phi_n \Longrightarrow \psi_0, \ldots, \psi_m$ is valid iff

$\phi_0 \land \ldots \land \phi_n \rightarrow \psi_0 \lor \ldots \lor \psi_m$ is valid.

Example proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{p \Longrightarrow q, p}\ \textit{close}}{p, \neg q \Longrightarrow p}\ \text{notLeft}}{p \land \neg q \Longrightarrow p}\ \textit{andLeft}}{\Longrightarrow (p \land \neg q) \rightarrow p}\ \textit{impRight}$$

Proof Slicing

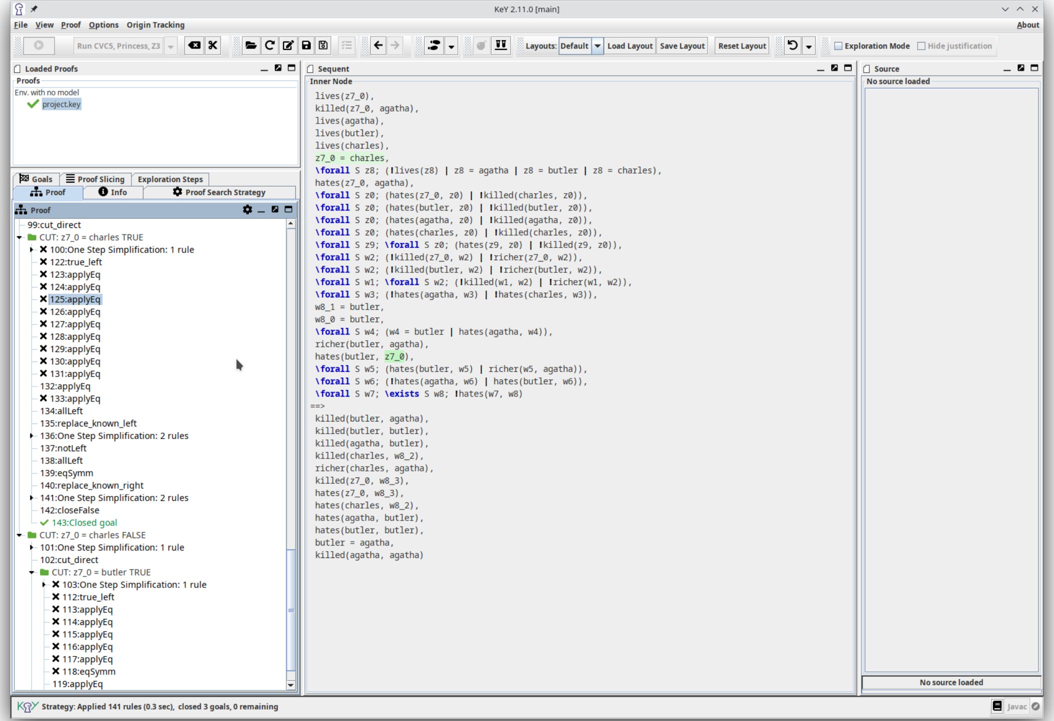
Sequent calculus: A sequent $\phi_0, \ldots, \phi_n \Longrightarrow \psi_0, \ldots, \psi_m$ is valid iff

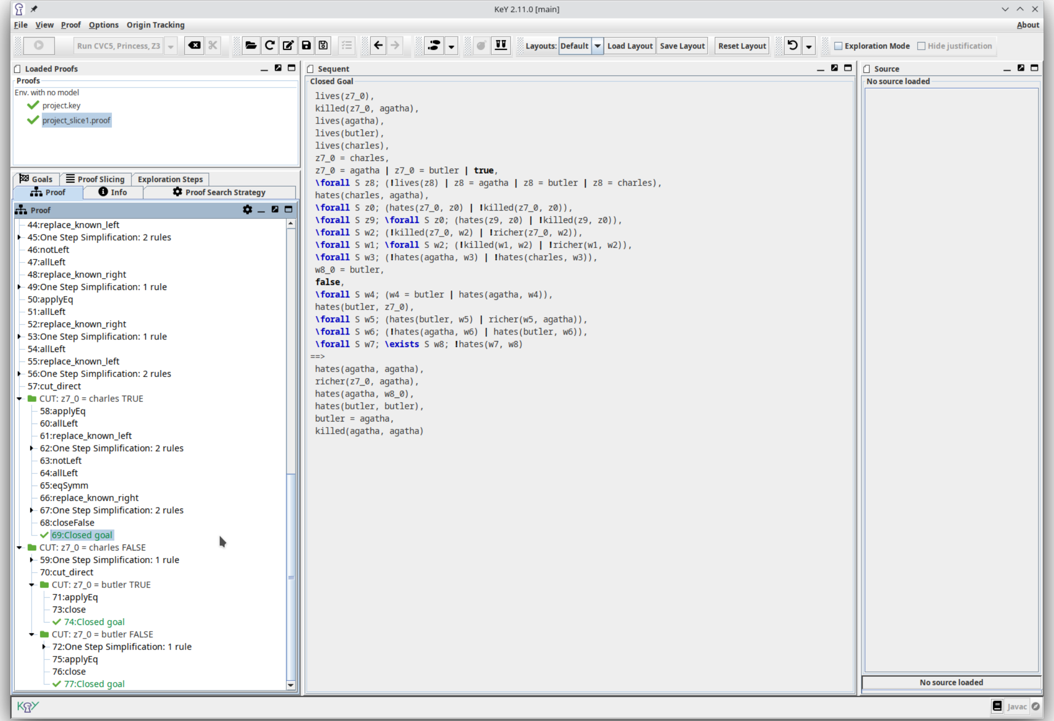
$$\phi_0 \land \ldots \land \phi_n \rightarrow \psi_0 \lor \ldots \lor \psi_m$$

is valid.

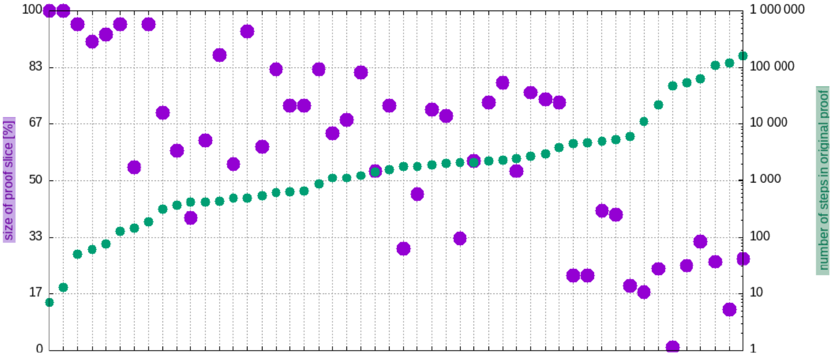
Example proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{p \Longrightarrow q, p}\ \textit{close}}{p, \neg q \Longrightarrow p}\ \text{notLeft}}{p \land \neg q \Longrightarrow p}\ \textit{andLeft}}{\Longrightarrow (p \land \neg q) \rightarrow p}\ \textit{impRight}$$


KeY 2.11.0 [main]
File View Proof Options Origin Tracking
About
Run CVC5, Princess, Z3
Layouts: Default Load Layout Save Layout Reset Layout
Exploration Mode Hide justification
Loaded Proofs
Proofs
Env. with no model
project.key
Goals Proof Slicing Exploration Steps
Proof Info Proof Search Strategy
Proof
99:cut_direct
CUT: z7_0 = charles TRUE
100:One Step Simplification: 1 rule
122:true_left
123:applyEq
124:applyEq
125:applyEq
126:applyEq
127:applyEq
128:applyEq
129:applyEq
130:applyEq
131:applyEq
132:applyEq
133:applyEq
134:allLeft
135:replace_known_left
136:One Step Simplification: 2 rules
137:notLeft
138:allLeft
139:eqSymm
140:replace_known_right
141:One Step Simplification: 2 rules
142:closeFalse
143:Closed goal
CUT: z7_0 = charles FALSE
101:One Step Simplification: 1 rule
102:cut_direct
CUT: z7_0 = butler TRUE
103:One Step Simplification: 1 rule
112:true_left
113:applyEq
114:applyEq
115:applyEq
116:applyEq
117:applyEq
118:eqSymm
119:applyEq
Sequent
Inner Node
lives(z7_0),
killed(z7_0, agatha),
lives(agatha),
lives(butler),
lives(charles),
z7_0 = charles,
\forall S z8; (!lives(z8) | z8 = agatha | z8 = butler | z8 = charles),
hates(z7_0, agatha),
\forall S z0; (hates(z7_0, z0) | !killed(charles, z0)),
\forall S z0; (hates(butler, z0) | !killed(butler, z0)),
\forall S z0; (hates(agatha, z0) | !killed(agatha, z0)),
\forall S z0; (hates(charles, z0) | !killed(charles, z0)),
\forall S z9; \forall S z0; (hates(z9, z0) | !killed(z9, z0)),
\forall S w2; (!killed(z7_0, w2) | !richer(z7_0, w2)),
\forall S w2; (!killed(butler, w2) | !richer(butler, w2)),
\forall S w1; \forall S w2; (!killed(w1, w2) | !richer(w1, w2)),
\forall S w3; (!hates(agatha, w3) | !hates(charles, w3)),
w8_1 = butler,
w8_0 = butler,
\forall S w4; (w4 = butler | hates(agatha, w4)),
richer(butler, agatha),
hates(butler, z7_0),
\forall S w5; (hates(butler, w5) | richer(w5, agatha)),
\forall S w6; (!hates(agatha, w6) | hates(butler, w6)),
\forall S w7; \exists S w8; !hates(w7, w8)
==>
killed(butler, agatha),
killed(butler, butler),
killed(agatha, butler),
killed(charles, w8_2),
richer(charles, agatha),
killed(z7_0, w8_3),
hates(z7_0, w8_3),
hates(charles, w8_2),
hates(agatha, butler),
hates(butler, butler),
butler = agatha,
killed(agatha, agatha)
Source
No source loaded
No source loaded
Strategy: Applied 141 rules (0.3 sec), closed 3 goals, 0 remaining
Javac


KeY 2.11.0 [main]
File View Proof Options Origin Tracking
About
Run CVC5, Princess, Z3
Layouts: Default
Load Layout
Save Layout
Reset Layout
Exploration Mode
Hide justification
Loaded Proofs
Proofs
Env. with no model
project.key
project_slice1.proof
Goals
Proof Slicing
Exploration Steps
Proof
Info
Proof Search Strategy
Proof
44:replace_known_left
45:One Step Simplification: 2 rules
46:notLeft
47:allLeft
48:replace_known_right
49:One Step Simplification: 1 rule
50:applyEq
51:allLeft
52:replace_known_right
53:One Step Simplification: 1 rule
54:allLeft
55:replace_known_left
56:One Step Simplification: 2 rules
57:cut_direct
CUT: z7_0 = charles TRUE
58:applyEq
60:allLeft
61:replace_known_left
62:One Step Simplification: 2 rules
63:notLeft
64:allLeft
65:eqSymm
66:replace_known_right
67:One Step Simplification: 2 rules
68:closeFalse
69:Closed goal
CUT: z7_0 = charles FALSE
59:One Step Simplification: 1 rule
70:cut_direct
CUT: z7_0 = butler TRUE
71:applyEq
73:close
74:Closed goal
CUT: z7_0 = butler FALSE
72:One Step Simplification: 1 rule
75:applyEq
76:close
77:Closed goal
Sequent
Closed Goal
lives(z7_0),
killed(z7_0, agatha),
lives(agatha),
lives(butler),
lives(charles),
z7_0 = charles,
z7_0 = agatha | z7_0 = butler | true,
\forall S z8; (!lives(z8) | z8 = agatha | z8 = butler | z8 = charles),
hates(charles, agatha),
\forall S z0; (hates(z7_0, z0) | !killed(z7_0, z0)),
\forall S z9; \forall S z0; (hates(z9, z0) | !killed(z9, z0)),
\forall S w2; (!killed(z7_0, w2) | !richer(z7_0, w2)),
\forall S w1; \forall S w2; (!killed(w1, w2) | !richer(w1, w2)),
\forall S w3; (!hates(agatha, w3) | !hates(charles, w3)),
w8_0 = butler,
false,
\forall S w4; (w4 = butler | hates(agatha, w4)),
hates(butler, z7_0),
\forall S w5; (hates(butler, w5) | richer(w5, agatha)),
\forall S w6; (!hates(agatha, w6) | hates(butler, w6)),
\forall S w7; \exists S w8; !hates(w7, w8)
==>
hates(agatha, agatha),
richer(z7_0, agatha),
hates(agatha, w8_0),
hates(butler, butler),
butler = agatha,
killed(agatha, agatha)
Source
No source loaded
No source loaded
Javac

Evaluation



- Often, very large parts of proofs could be removed.
- Trend: The larger the proof, the larger the percentage.
- Most of the removed steps are normalizations of formulas which are never used later on.

Part 3: Proof Caching

Proof Caching

Motivation: Finding the correct and provable specification is often an iterative process.

Proof Caching

Motivation: Finding the correct and provable specification is often an iterative process.

Observation: If $\Gamma \Longrightarrow \Delta$ is valid, then $\Gamma, E \Longrightarrow \Delta, Z$ is also valid (*).

Proof Caching

Motivation: Finding the correct and provable specification is often an iterative process.

Observation: If $\Gamma \Longrightarrow \Delta$ is valid, then $\Gamma, E \Longrightarrow \Delta, Z$ is also valid (*).

(*) Under some restrictions:

- The Java code referred to must be the same.
- Both must use the same prover settings for semantics.
- The same added rules must be present.

Proof Caching

Motivation: Finding the correct and provable specification is often an iterative process.

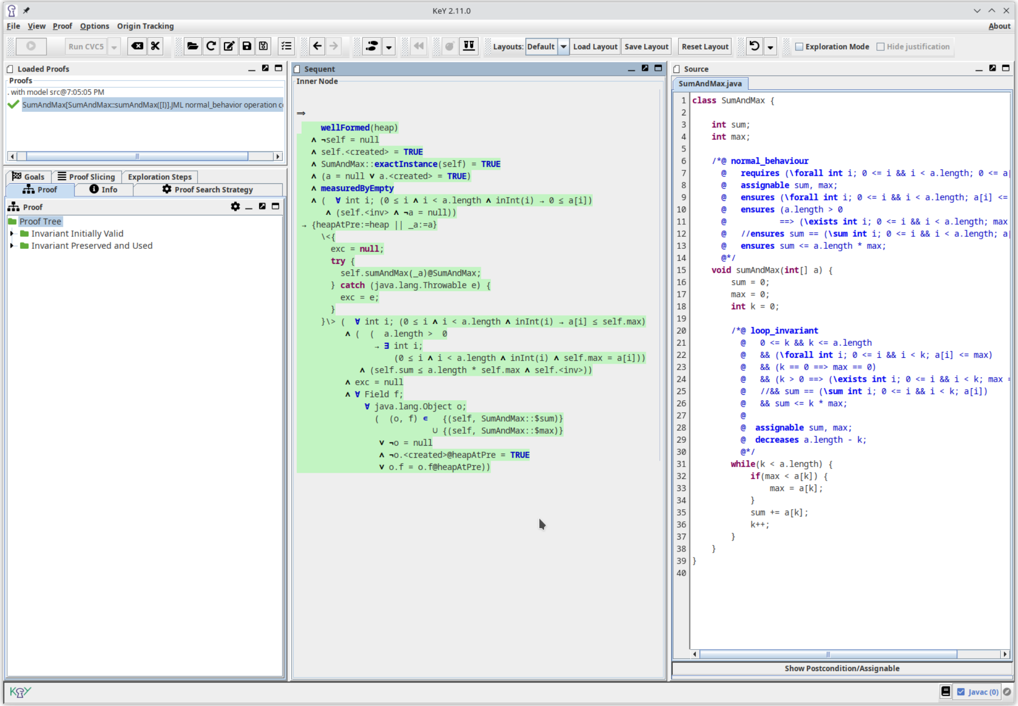
Observation: If $\Gamma \Longrightarrow \Delta$ is valid, then $\Gamma, E \Longrightarrow \Delta, Z$ is also valid (*).

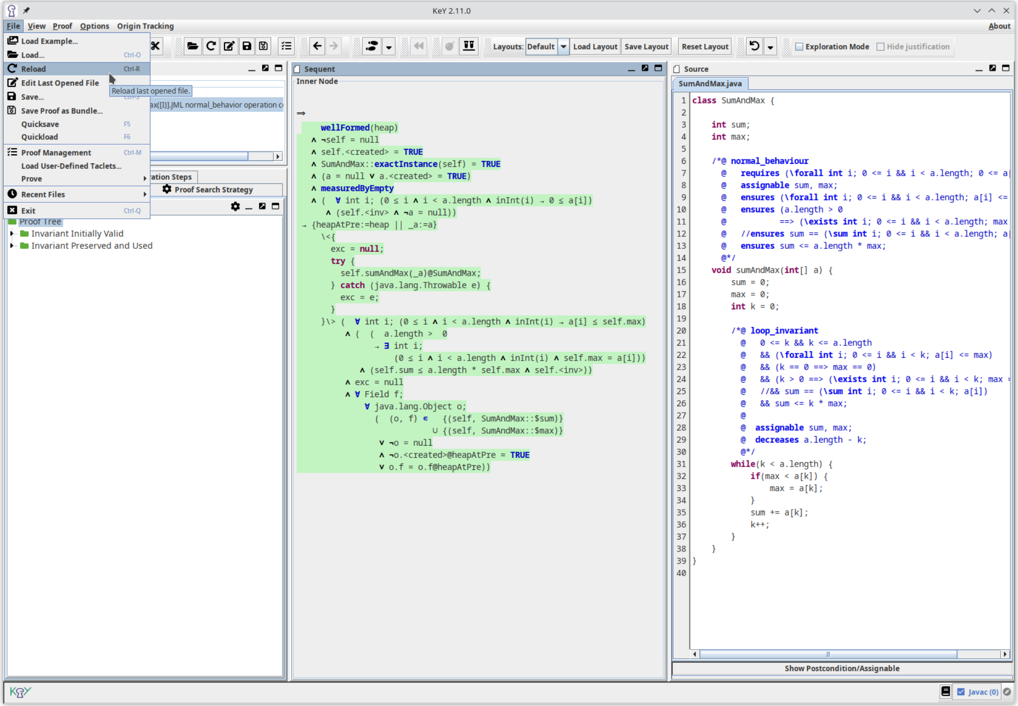
(*) Under some restrictions:

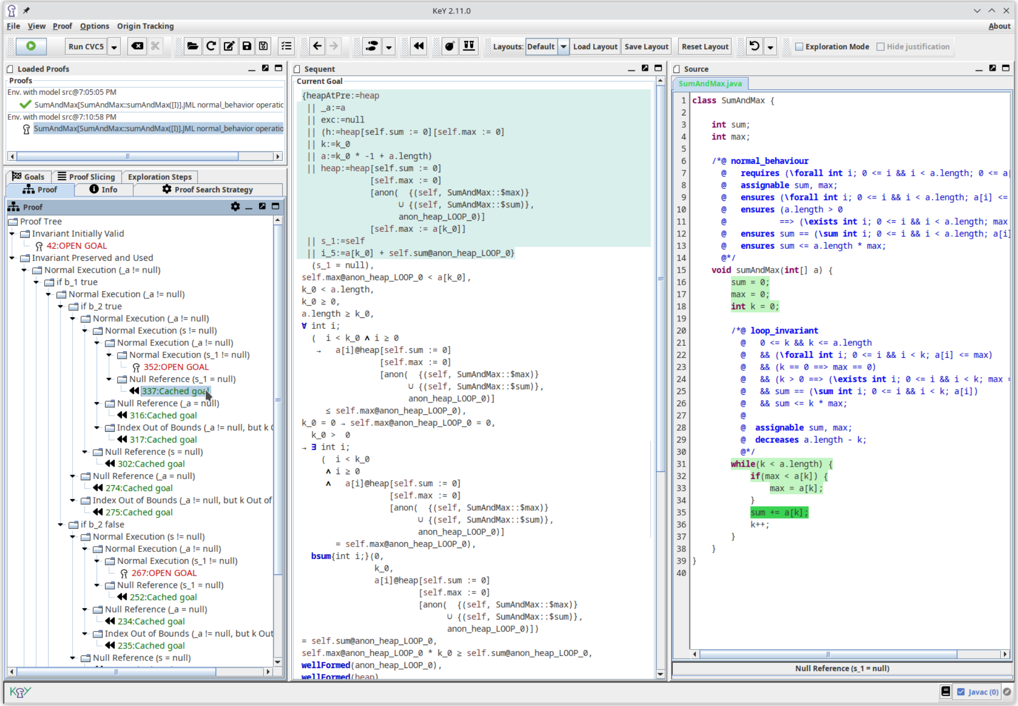
- The Java code referred to must be the same.
- Both must use the same prover settings for semantics.
- The same added rules must be present.

Ongoing work:

- Which sequents should be in the cache?
- Extend the caching beyond a single run of KeY.
- Relax the above conditions.


KeY 2.11.0
File View Proof Options Origin Tracking About
Run CVC5
Layouts: Default Load Layout Save Layout Reset Layout
Exploration Mode Hide justification
Loaded Proofs
Proofs
. with model src@7:05:05 PM
SumAndMax[SumAndMax::sumAndMax([I)].JML normal_behavior operation c
Goals Proof Slicing Exploration Steps
Proof Info Proof Search Strategy
Proof
Proof Tree
Invariant Initially Valid
Invariant Preserved and Used
Sequent
Inner Node
⟹
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 ≤ 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
class SumAndMax {
int sum;
int max;
/*@ normal_behaviour
@ requires (\forall int i; 0 <= i && i < a.length; 0 <= a
@ assignable sum, max;
@ ensures (\forall int i; 0 <= i && i < a.length; a[i] <=
@ ensures (a.length > 0
@ ==> (\exists int i; 0 <= i && i < a.length; max
@ //ensures sum == (\sum int i; 0 <= i && i < a.length; a
@ 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
@ //&& 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++;
}
}
}
Show Postcondition/Assignable
Javac (0)


KeY 2.11.0
File View Proof Options Origin Tracking
About
Load Example...
Load... Ctrl-O
Reload Ctrl-R
Edit Last Opened File
Reload last opened file.
Save...
Save Proof as Bundle...
Quicksave F5
Quickload F6
Proof Management Ctrl-M
Load User-Defined Taclets...
Prove
Recent Files
Exit Ctrl-Q
Layouts: Default
Load Layout
Save Layout
Reset Layout
Exploration Mode
Hide justification
ax([I]).JML normal_behavior operation c
ation Steps
Proof Search Strategy
Proof Tree
Invariant Initially Valid
Invariant Preserved and Used
Sequent
Inner Node
⇒
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 ≤ 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
class SumAndMax {
int sum;
int max;
/*@ normal_behaviour
@ requires (\forall int i; 0 <= i && i < a.length; 0 <= a
@ assignable sum, max;
@ ensures (\forall int i; 0 <= i && i < a.length; a[i] <=
@ ensures (a.length > 0
@ ==> (\exists int i; 0 <= i && i < a.length; max
@ //ensures sum == (\sum int i; 0 <= i && i < a.length; a
@ 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 =
@ //&& 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++;
}
}
}
Show Postcondition/Assignable
Javac (0)


KeY 2.11.0
File View Proof Options Origin Tracking About
Run CVC5
Layouts: Default Load Layout Save Layout Reset Layout
Exploration Mode Hide justification
Loaded Proofs
Proofs
Env. with model src@7:05:05 PM
SumAndMax[SumAndMax::sumAndMax([I)].JML normal_behavior operatic
Env. with model src@7:10:58 PM
SumAndMax[SumAndMax::sumAndMax([I)].JML normal_behavior operatic
Goals Proof Slicing Exploration Steps
Proof Info Proof Search Strategy
Proof
Proof Tree
Invariant Initially Valid
42:OPEN GOAL
Invariant Preserved and Used
Normal Execution (_a != null)
if b_1 true
Normal Execution (_a != null)
if b_2 true
Normal Execution (_a != null)
Normal Execution (s != null)
Normal Execution (_a != null)
Normal Execution (s_1 != null)
352:OPEN GOAL
Null Reference (s_1 = null)
337:Cached goal
Null Reference (_a = null)
316:Cached goal
Index Out of Bounds (_a != null, but k (
317:Cached goal
Null Reference (s = null)
302:Cached goal
Null Reference (_a = null)
274:Cached goal
Index Out of Bounds (_a != null, but k Out of
275:Cached goal
if b_2 false
Normal Execution (s != null)
Normal Execution (_a != null)
Normal Execution (s_1 != null)
267:OPEN GOAL
Null Reference (s_1 = null)
252:Cached goal
Null Reference (_a = null)
234:Cached goal
Index Out of Bounds (_a != null, but k Out
235:Cached goal
Null Reference (s = null)
Sequent
Current Goal
{heapAtPre:=heap
|| _a:=a
|| exc:=null
|| (h:=heap[self.sum := 0][self.max := 0]
|| k:=k_0
|| a:=k_0 * -1 + a.length)
|| heap:=heap[self.sum := 0]
[self.max := 0]
[anon({(self, SumAndMax::$max)}
∪ {(self, SumAndMax::$sum)},
anon_heap_LOOP_0)]
[self.max := a[k_0]]
|| s_1:=self
|| i_5:=a[k_0] + self.sum@anon_heap_LOOP_0}
(s_1 = null),
self.max@anon_heap_LOOP_0 < a[k_0],
k_0 < a.length,
k_0 ≥ 0,
a.length ≥ k_0,
∀ int i;
(i < k_0 ∧ i ≥ 0
→ a[i]@heap[self.sum := 0]
[self.max := 0]
[anon({(self, SumAndMax::$max)}
∪ {(self, SumAndMax::$sum)},
anon_heap_LOOP_0)]
≤ self.max@anon_heap_LOOP_0),
k_0 = 0 → self.max@anon_heap_LOOP_0 = 0,
k_0 > 0
→ ∃ int i;
(i < k_0
∧ i ≥ 0
∧ a[i]@heap[self.sum := 0]
[self.max := 0]
[anon({(self, SumAndMax::$max)}
∪ {(self, SumAndMax::$sum)},
anon_heap_LOOP_0)]
= self.max@anon_heap_LOOP_0),
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)
Source
SumAndMax.java
class SumAndMax {
int sum;
int max;
/*@ normal_behaviour
@ requires (\forall int i; 0 <= i && i < a.length; 0 <= a
@ assignable sum, max;
@ ensures (\forall int i; 0 <= i && i < a.length; a[i] <=
@ ensures (a.length > 0
@ ==> (\exists int i; 0 <= i && i < a.length; max
@ 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 =
@ && 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++;
}
}
}
Null Reference (s_1 = null)
Javac (0)

Summary

We have seen:

- a novel way for interaction with KeY on the source code level.
- “Proof Slicing” to minimize sequent calculus proofs.
- “Proof Caching” to avoid reproving the same formulas.

`https://github.com/KeYProject/key`