Secure and Dependable System
Sagar Kumar
Bremen, 2017

Question 4.1

Factorial:

Pre-condition:

P(input): input >= 0

Post-condition:

Q(input,output): input <= output

Loop invariant condition:

Product == (factor-1)!

Termination ordering:

n-(factor-1)

Revert Immutable:

For this question I will use a function size() which determines the number of elements in the list

Pre-condition:

P(input[]): size(input) > 0

Post-condition:

Q(input[],rev[]): size(input) == size(rev)

Loop invariant:

(size(rev) + size(rest)) == size(x)

Termination ordering:

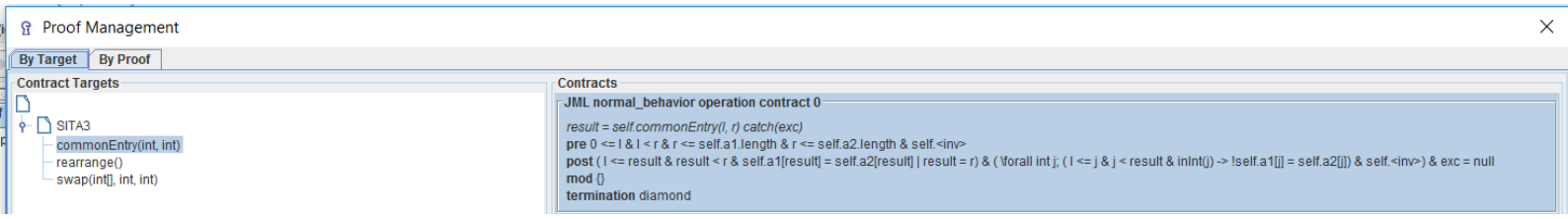
size(rest) == 0

Secure and Dependable System
Sagar Kumar
Bremen, 2017

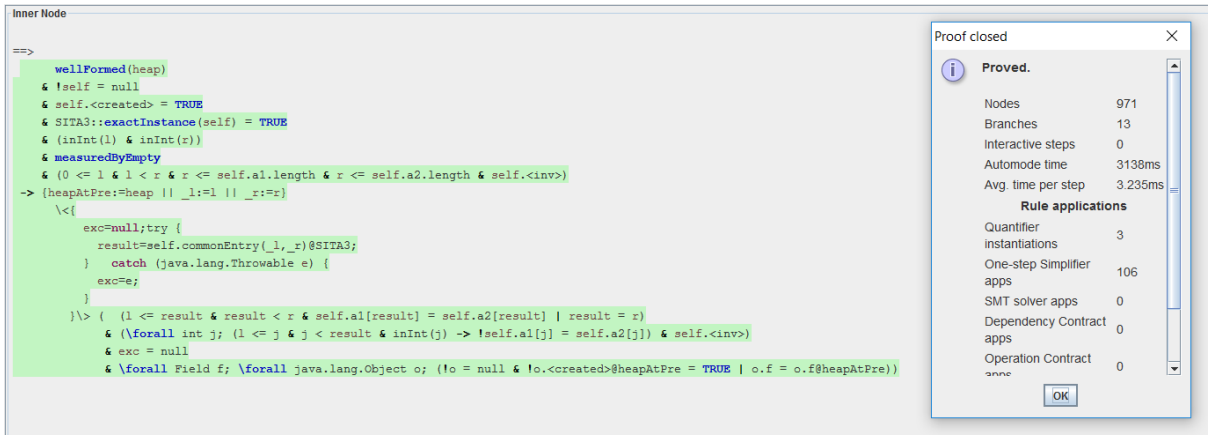
Question 4.2

I chose the function “search in two arrays”:

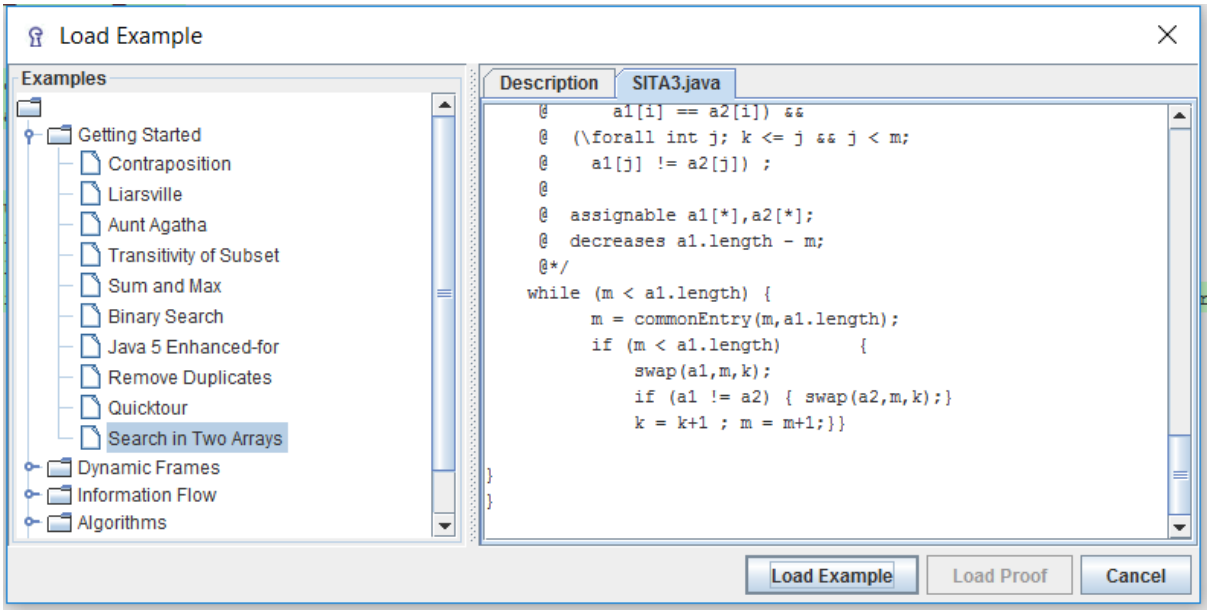
Pre/Post conditions:



Proving (w/ proof closed):



While loop:


Load Example
Examples
Getting Started
Contraposition
Liarsville
Aunt Agatha
Transitivity of Subset
Sum and Max
Binary Search
Java 5 Enhanced-for
Remove Duplicates
Quicktour
Search in Two Arrays
Dynamic Frames
Information Flow
Algorithms
Description
SITA3.java
@ a1[i] == a2[i]) &&
@ (\forall int j; k <= j && j < m;
@ a1[j] != a2[j]) ;
@
@ assignable a1[*],a2[*];
@ decreases a1.length - m;
@*/
while (m < a1.length) {
m = commonEntry(m,a1.length);
if (m < a1.length) {
swap(a1,m,k);
if (a1 != a2) { swap(a2,m,k);}
k = k+1 ; m = m+1;}}
}
}
Load Example
Load Proof
Cancel

Secure and Dependable System
Sagar Kumar
Bremen, 2017

Question 4.3

Take the if/else definition from 11.1.2. The lecture notes defines soundness as “a set of rules is sound if all provable formulas are theorems”. Two cases arise:

Case 1: C holds true and $[t]$ is executed to reach F. Then we know from 9.4 “Grammar and intuitions” that F will be true pre- and post- evaluation.

Case 2: The same goes for $\neg C$. If it holds true and $[t']$ is executed to reach F, then again from 9.4 we know that F will be true for pre- and post- evaluation.

If the statement is provable for both cases, which it is, we know that it is sound.

Take the while loop definition from 11.1.3. Much like the if/else statement, we again arrive at two distinct cases:

Case 1: $\mathrm{C\ holds}$ ≔ If C holds then we execute [t] by evaluating the loop invariant (which always holds true) and the condition C and reach F.

Case 2: $\neg\mathrm{C\ holds}$ ≔ If C does not hold then we evaluate the condition C and the loop invariant and simple do not execute [t] to reach F.

In either case we reach a definitive true state and no side effects can be seen. Hence the statement is sound.