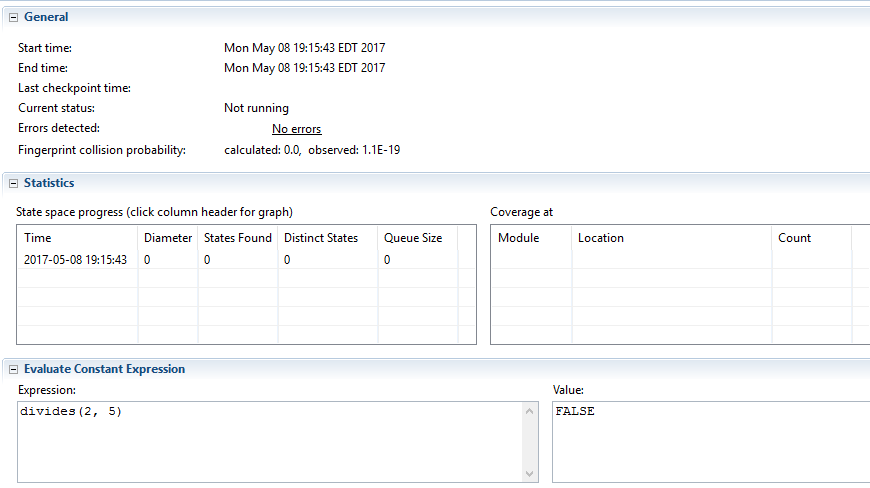
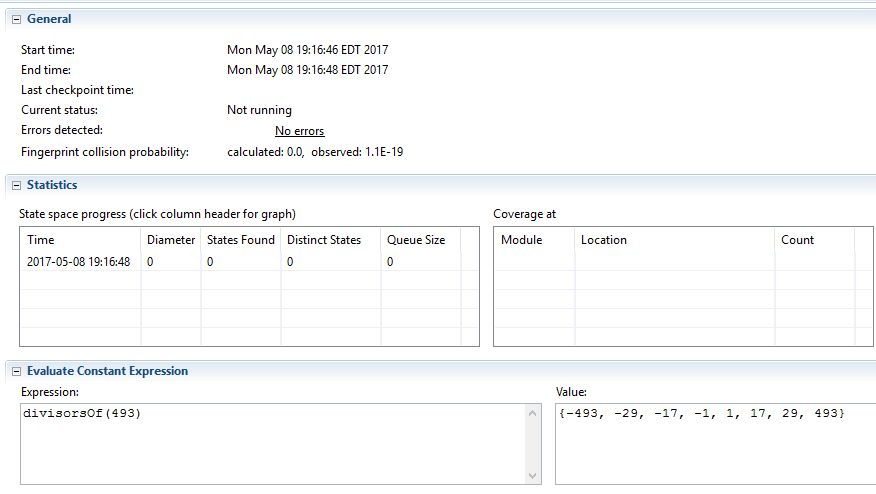


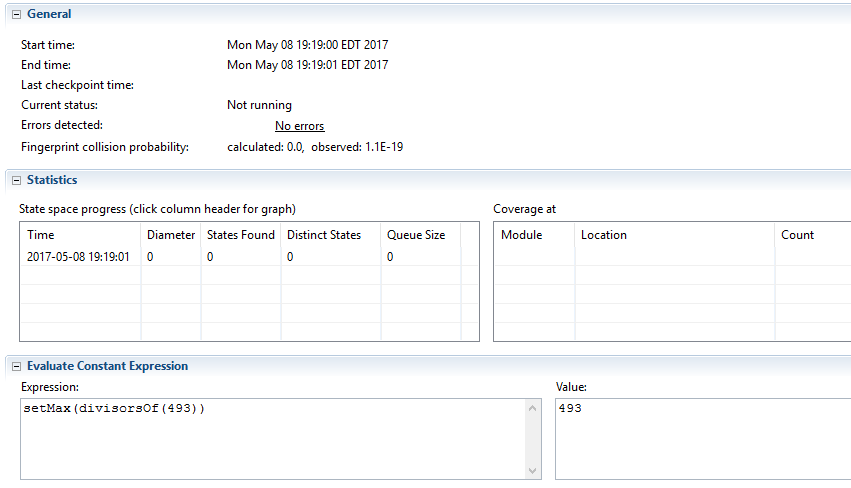
There is an integer q such that 2q = 4

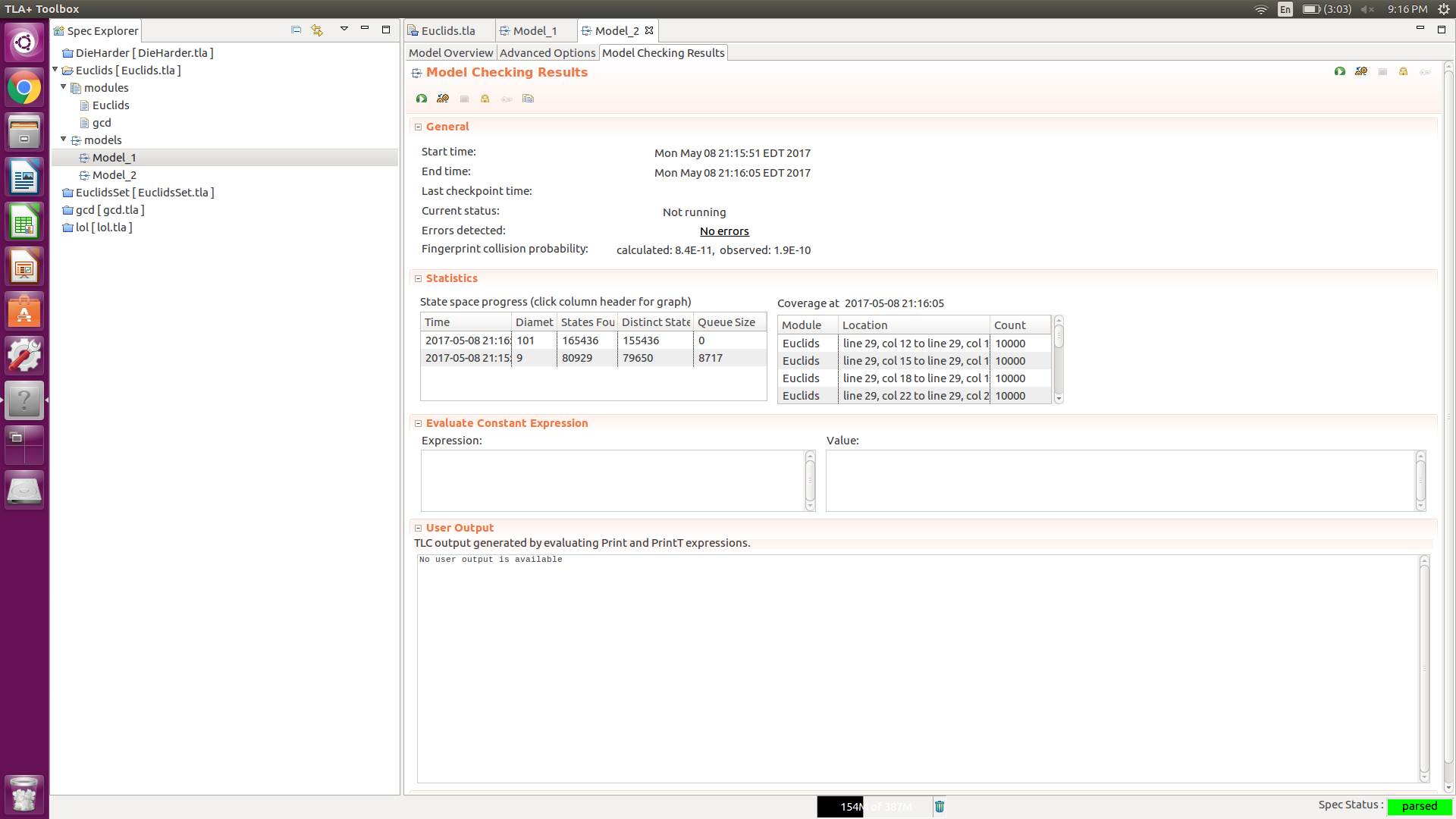


There is no integer q such that 2q = 5

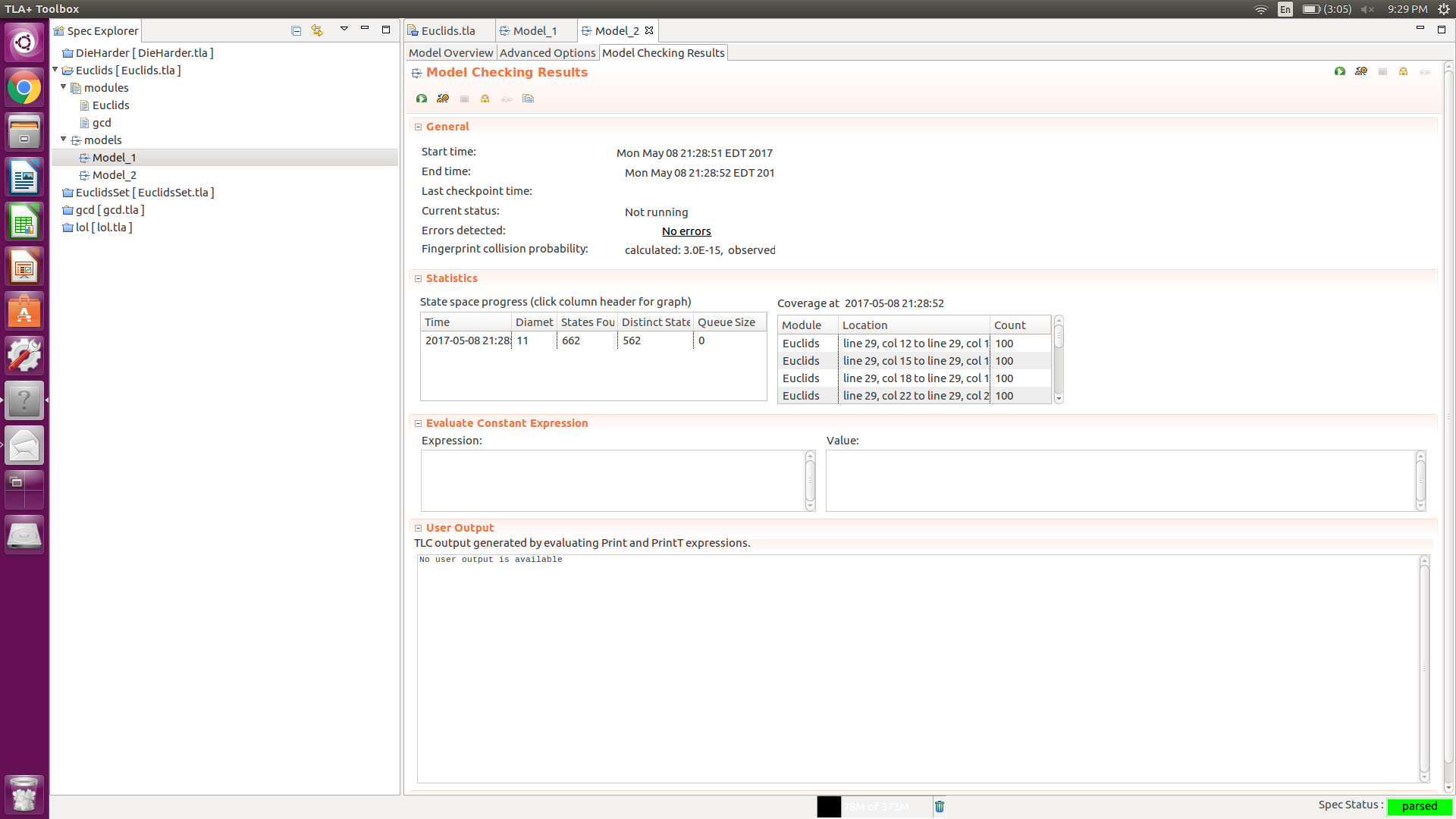


Lists the divisors of 493.

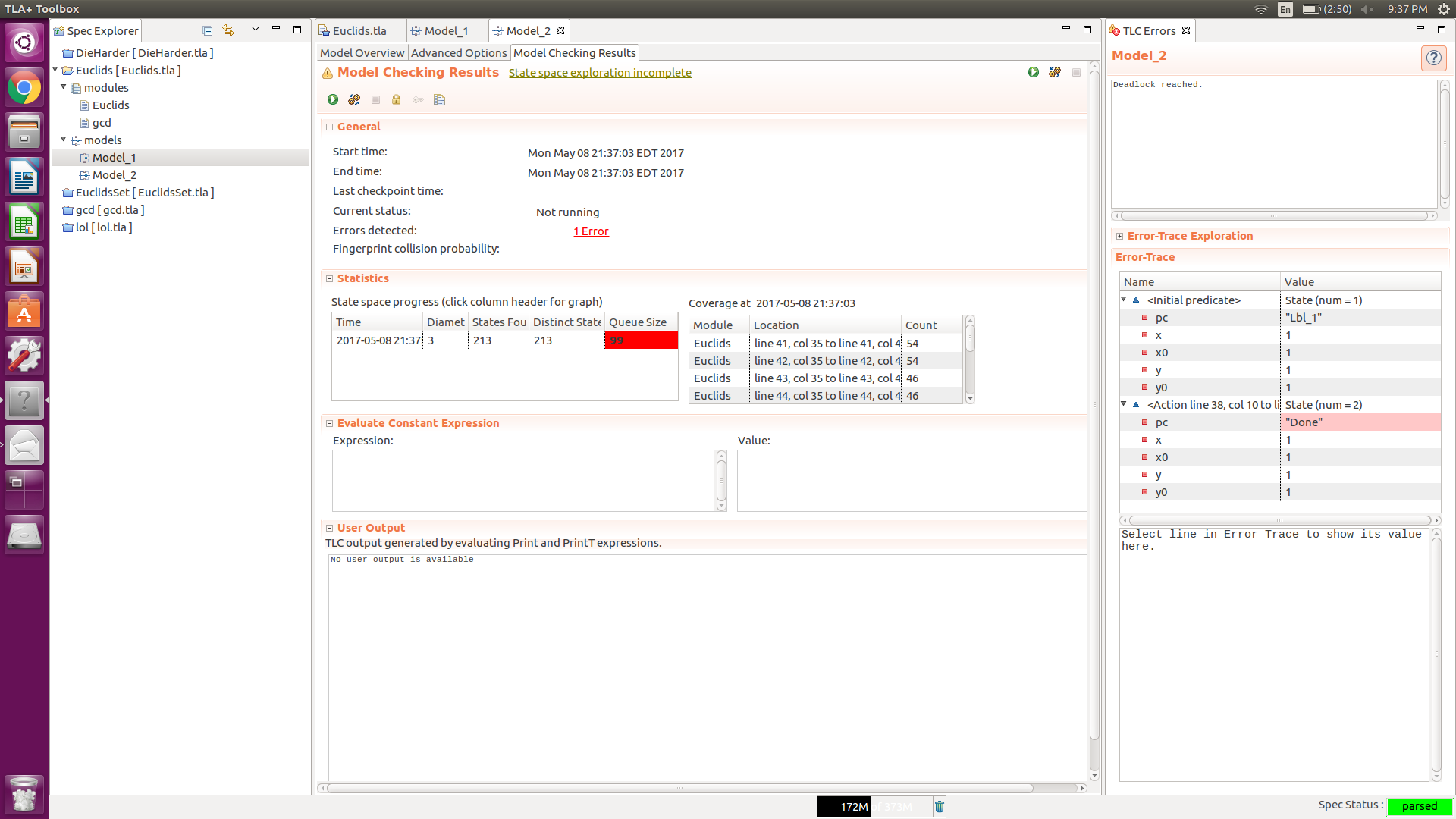
  
Gets the max of the elements in the set from the previous one



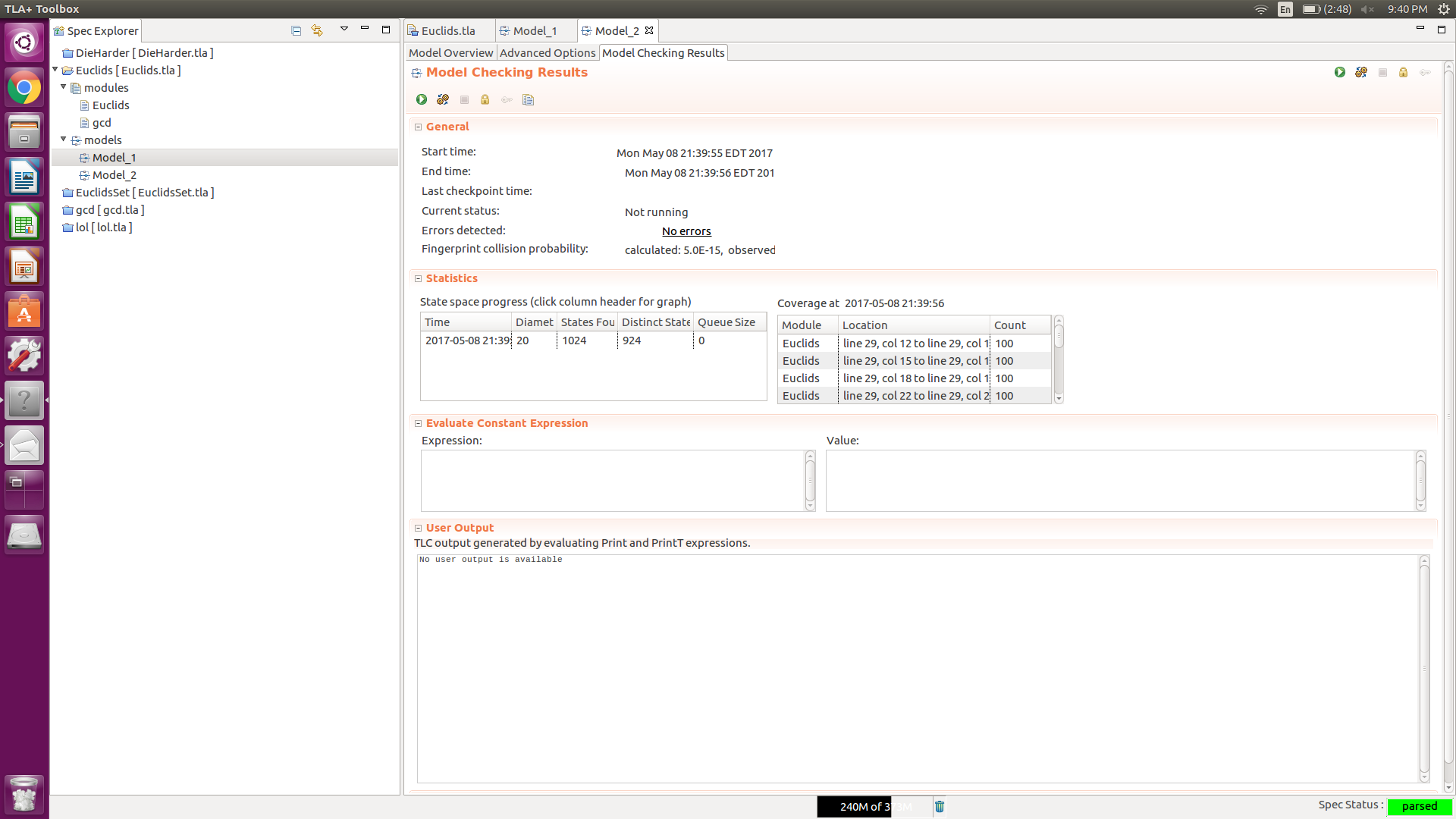
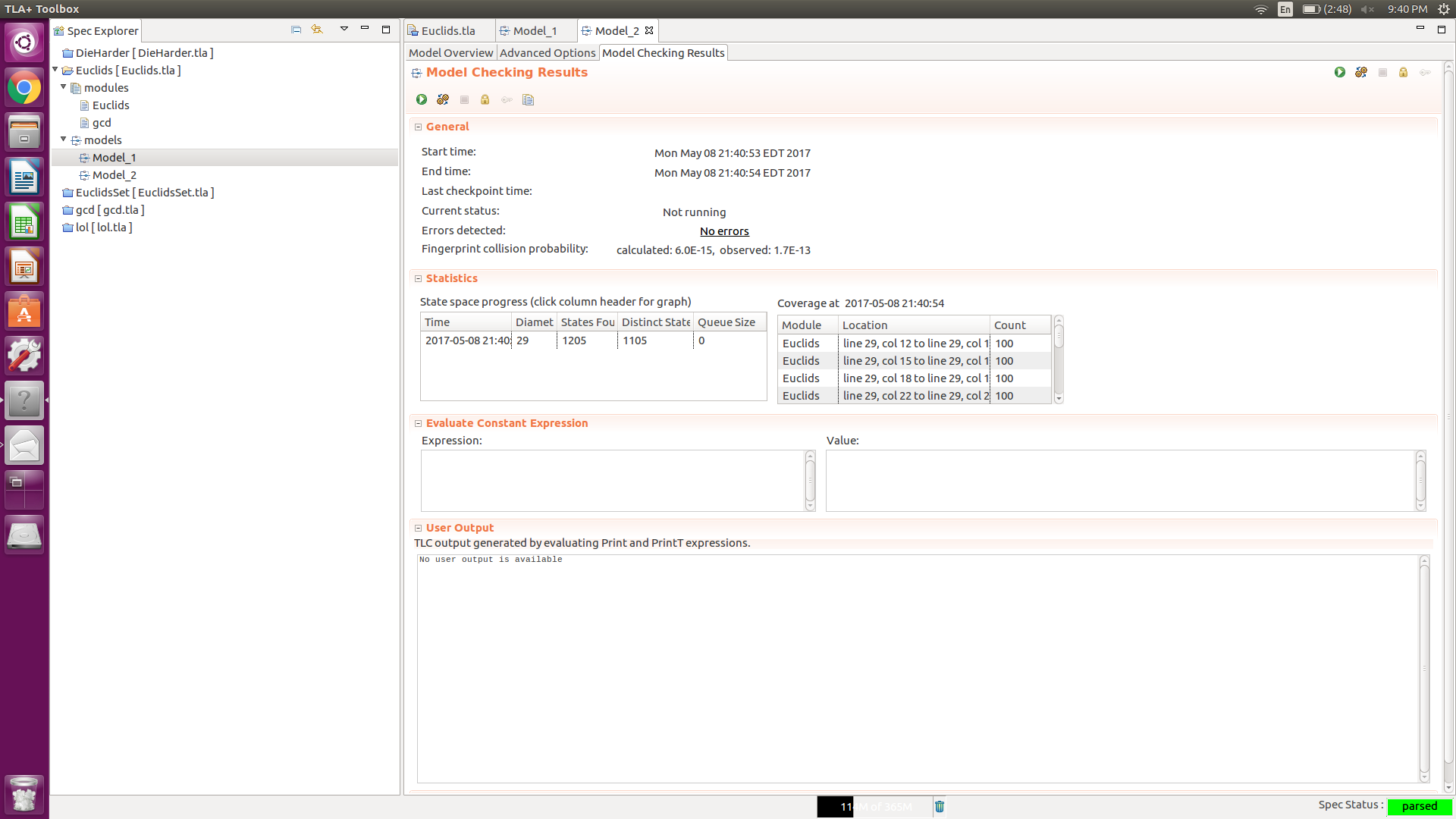
This is checking safety of Euclid’s algorithm for 30 and 18. There are no errors for our model with INT = -1000..1000



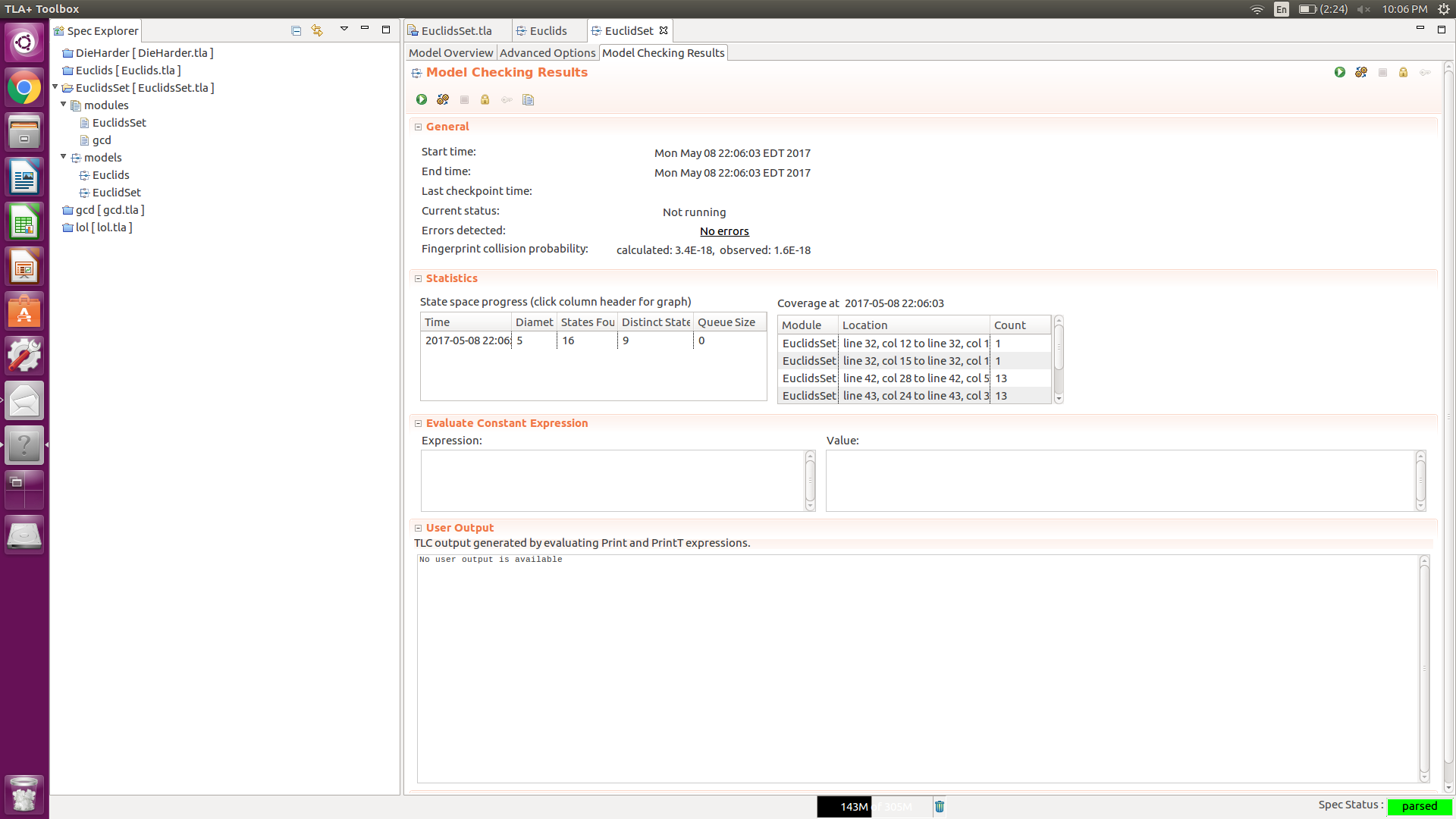
This is the test for checking for liveliness. The Termination property is now checked, guaranteeing that the algorithm halts.



It turns out that we need to disjunct saying that we are done or the variables are unchanged to prevent deadlock



The top picture has three labels and the bottom has two. More labels -> more states -> slower calculation



Euclid’s set is all good

Checking Safety

This property says that an algorithm is correct when it returns an answer. The way that we do it in this assignment in TLA+ is by defining a GCD function that we consider correct and comparing the value that it returns to the value returned by the algorithm.

Checking Liveness

This property is about guaranteeing that the function will return in a finite amount of time. The way that we do this in TLA+ is with a Termination property that will guarantee that our PC variable will reach the value “Done”

Proving Partial Correctness

To prove partial correctness is to prove that the safety property is satisfied. This means that if the algorithm returns, it will have the right answer. The way that we prove this was by adding a property that states that if pc = “done” then the values returned by Euclid’s algorithm match what we got with our GCD function that we considered correct.

Proving Total Correctness

Proving total correctness is the same as proving partial correctness, but in addition, you must also prove that the algorithm will indeed terminate. To ensure that this happens, we do the same kind of thing that we did in Alloy with a progress predicate. You say that the sum of the values in the gcd vector must get smaller with each iteration or you will be at the completed step.