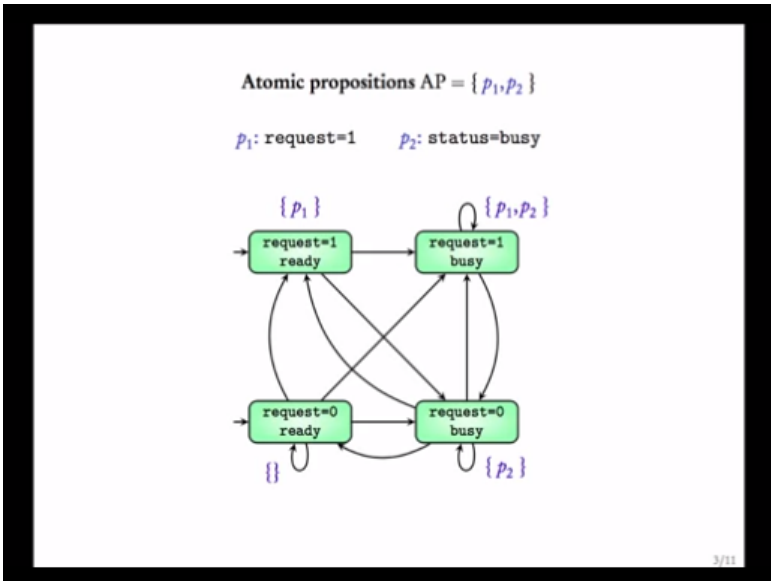
Model Checking
Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology – Madras

Lecture - 14
Invariants

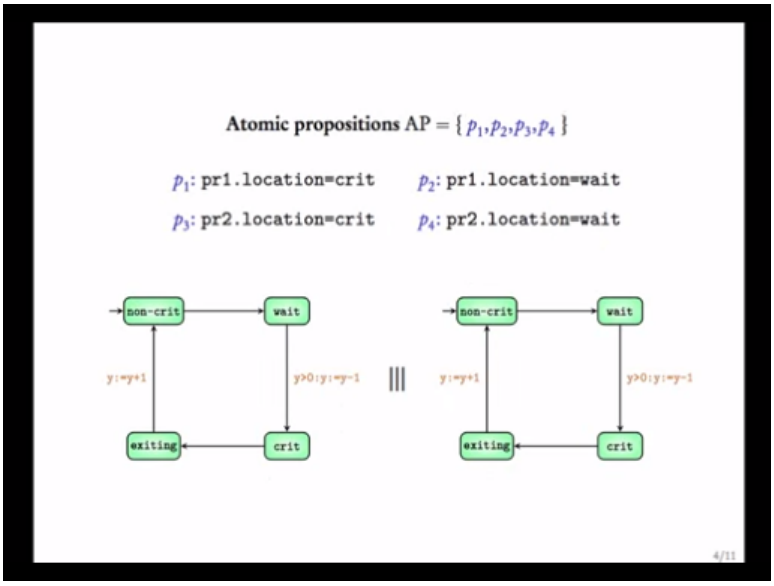
In the last module we gave a mathematical meaning to the word properties. We said that a property is nothing but a set of words. We are now going to look at different kinds of properties. The first kind of properties that we are going to look at are called invariants. In this module we will see what invariants are we will see how they can be checked in NuSMV and finally we will look at an algorithm that checks invariant properties on models.

(Refer Slide Time: 00:44)



We will use 2 running examples in this module, the first one is our favorite transition system here with the request and the status variables. We will use the same atomic propositions that we used last time p1 and p2, p1 says that request is 1 and p2 says that the status is busy. Here are the states and the corresponding evaluations of the atomic propositions.

(Refer Slide Time: 01:18)



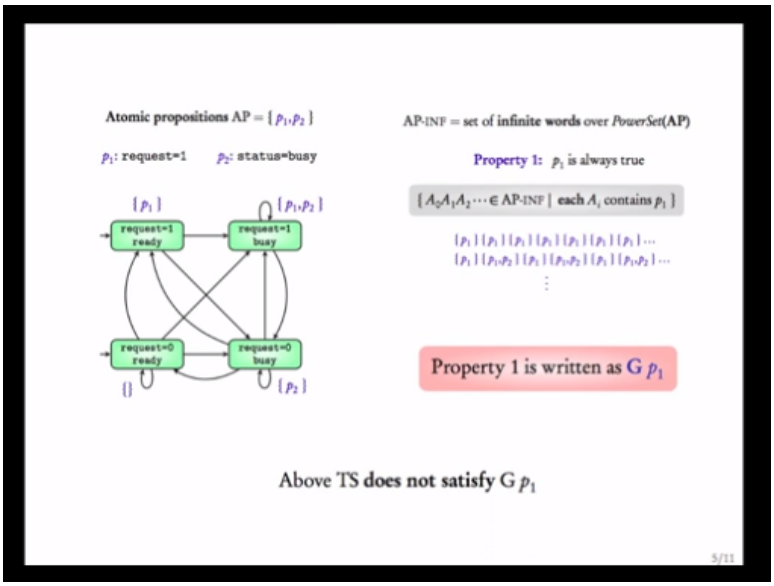
The second example that we will use in this module is the mutual exclusion. Here are 2 processes along with a shared variable y which is there to ensure mutual exclusion. We had seen this example in the previous unit, the transition system that we are interested in is the interleaving of these 2 program graphs.

Each program can be in 4 possible states, either it is non-critical after a while it moves to a waiting state where it wants to move to the critical section. It can move to the critical section only if certain conditions are satisfied and in the process it makes some assignments. It stays in its critical section for a while and then when it is about to exit it moves to an exciting state.

From the exciting state it goes back to the non-critical location and in the process makes an assignment. The transition system that we are interested in is the interleaving of these 2 program graphs. Let us define some atomic propositions, p1 is the proposition which says that the first process is in its critical section, p2 is the proposition which says that the first process is currently in its wait location and finally p4 is the atomic proposition which says that process 2 is in its wait location.

We have chosen these atomic proposition so the properties that we would be giving on this transition system would depend on these atomic propositions. So, these 2 are the running examples for this module. Let us now see some example properties.

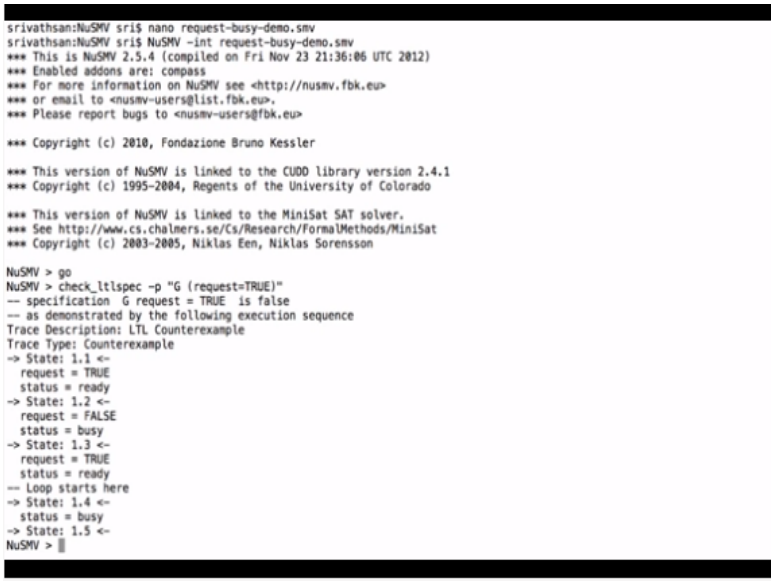
(Refer Slide Time: 03:27)



Here is the request busy transition system, let us look at the property that we have been looking at before as well the property which says that p1 is always true. The property is of a special form it says that something should always be true. In particular the word that belong to this property should satisfy the fact that each set contains p1.

The way we write this is G of p1, as we had seen before this transition system does not satisfy G p1. Let us now see a demo of NuSMV where we give it this transition system and we check this property. Let us see what the output of NuSMV is.

(Refer Slide Time: 04:36)



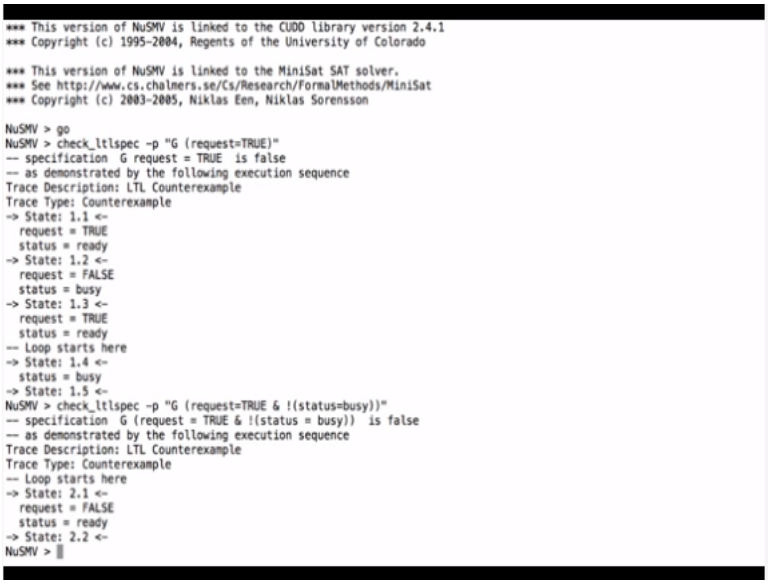
Here is the code, I will run this file now. No problems so far, let us now check the ltl specification, G of request equal to 1 the way we write it in NuSMV is request equal to true. It says that this specification is false and it says why it is false. It is giving us an example of an execution which does not satisfy this property. Let us look at the counterexample, it starts with a state request equal to true and status is ready.

And goes to the state where request equal to false status is busy, goes to request equal to true and then there is some loop, however already request has been made false in this state. So, this clearly says that there is an execution where the property is not satisfied. The thing to note here is that given a model and a property of this kind NuSMV checks if this property is true on that model and if it is not true it returns an execution which does not satisfy this property.

Let us now look at another example of a property. This property says that p1 and not p2 is always true, what are the words that satisfies this property. The words that satisfy should contain p1 in every set and there should not be p2 anywhere in any of these sets. So, there is only one word which satisfies this property and that word contains the atomic proposition p1 in every set.

This is the description A0, A1, A2 such that each Ai satisfies p1 and not p2. How do we write this property? This property is written as G p1 and not p2 inside the G there is a Boolean expressions formed out of the atomic propositions. We claimed that this transition system does not satisfy this property. Let us check what NuSMV says, and if it says NO, what is the counterexample?

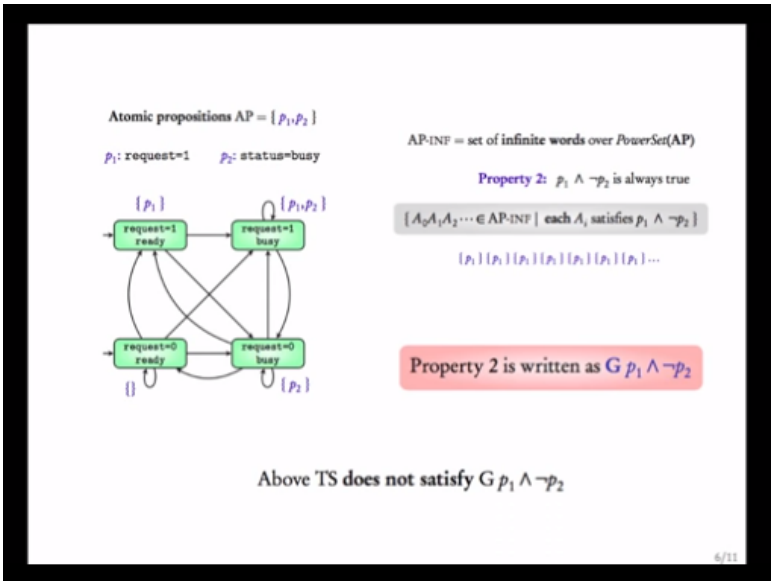
(Refer Slide Time: 07:55)



Let us now check the ltl specification G of requests equal to true and p2 was the atomic proposition which said that status is busy. We want not of status equal to busy, it gives a very short counterexample and the counterexample is a loop in the state where request is false and status is ready. This state clearly does not satisfy the expression request equal to true and not of status equal to busy, because both of them need to be true request is false here so already p1 is false.

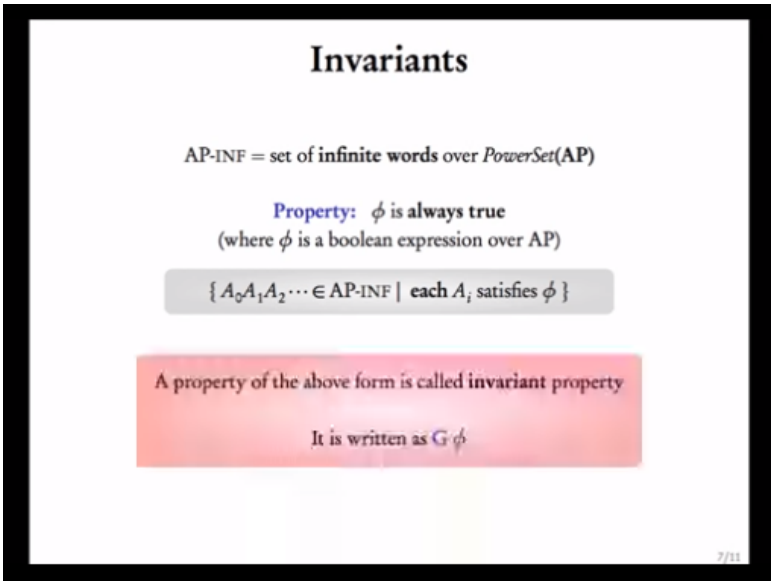
So, this state does not satisfy the expression inside G and the execution formed by looping around the state does not satisfy the property.

(Refer Slide Time: 09:12)



This is the execution given by NuSMV as you see it does not contain p1.

(Refer Slide Time: 09:22)

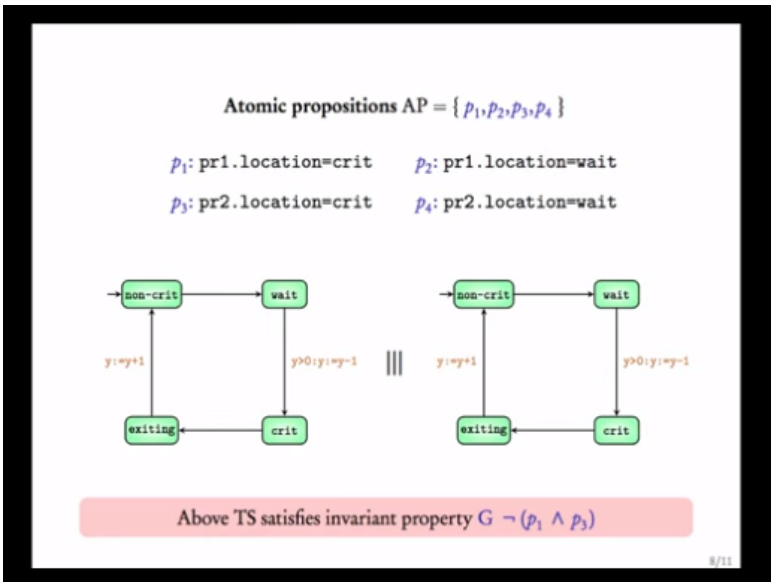


If you saw the previous 2 properties they were of a special form they asked if something is always true and the something was a Boolean expression over the atomic propositions. For example here we asked if p1 and not p2 is always true, in this property we asked if p1 is always true. Properties of this kind where we asked if a formula phi is always true where phi is a Boolean expression over the atomic propositions.

This kind of a property is called invariant and the way it is written is G of the Boolean expression. Let me repeat, properties of the form which ask for some Boolean expression

is always true are called invariant properties and the way we write them is using the G operator G phi.

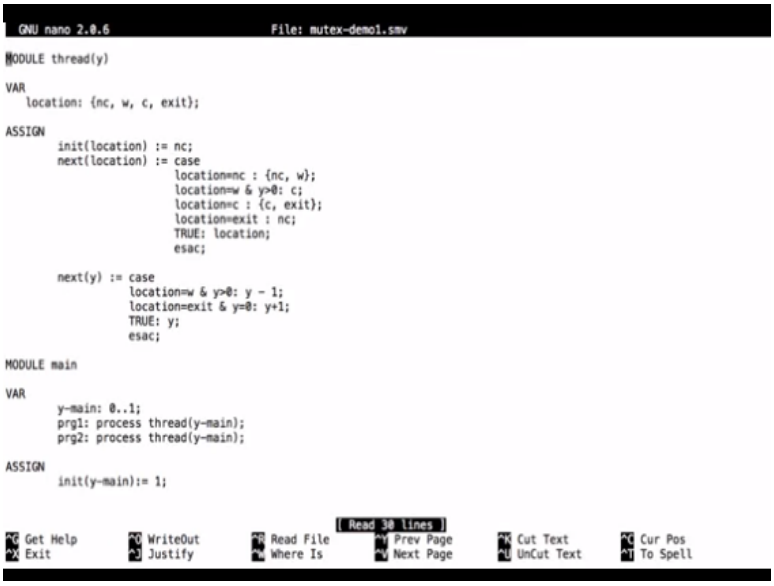
(Refer Slide Time: 10:43)



Let us come to the other example, we want to ask if the transition system of the mutual exclusion problem satisfies the invariant property G of not of p1 and p3. Is it the case that in every execution not of p1 and p3 is true, that means p1 says that process one is in critical location, p3 says that process 2 is in the critical location, p1 and p3 together say that process 1 and process 2 are in its critical location together and the not says that it is not the case that both processes are in their critical location.

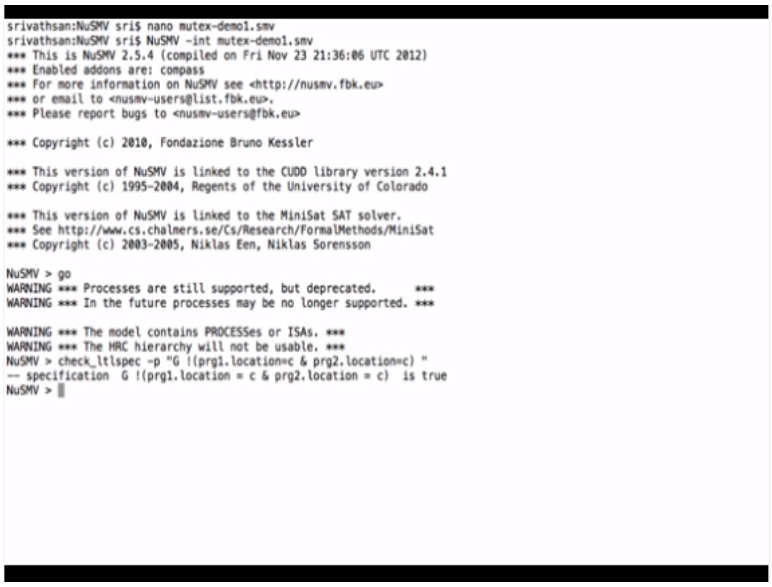
So, this property asks is it the case that in every execution in every state both process 1, process 2 cannot simultaneously be in their critical location. Let us check this property using NuSMV.

(Refer Slide Time: 10:51)



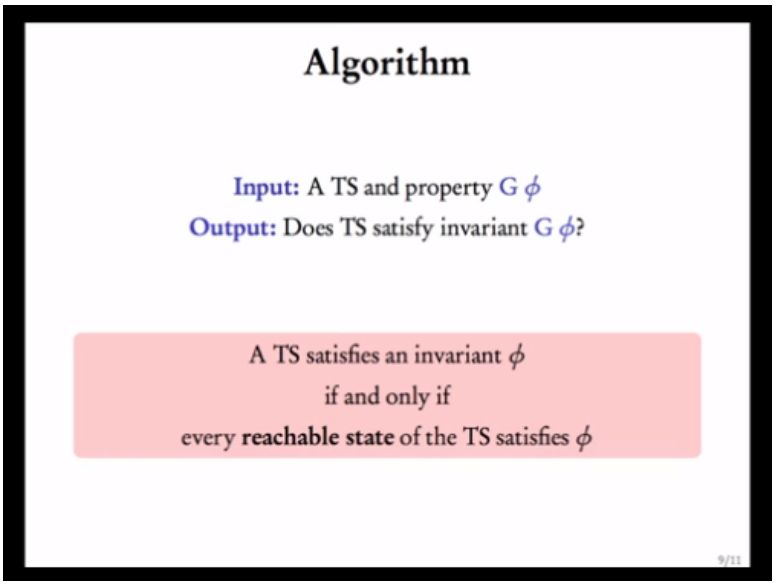
Here is the code that describes the program graphs, there are 2 variables rather 2 processors prg1 and prg2. We need to check that both are them are not in their critical location.

(Refer Slide Time: 12:50)



Check ltlspec minus p G of not prg1 dot location equal to critical. We use the work crit and prg2 dot location equal to crit, rather we use c sorry and NuSMV says that the specification is true. So as we have seen when we give an invariant property to NuSMV, if it is true it states that the specification is true on this model and if it is false it gives us a counterexample which does not satisfy the property. In particular here we are focusing on invariant properties.

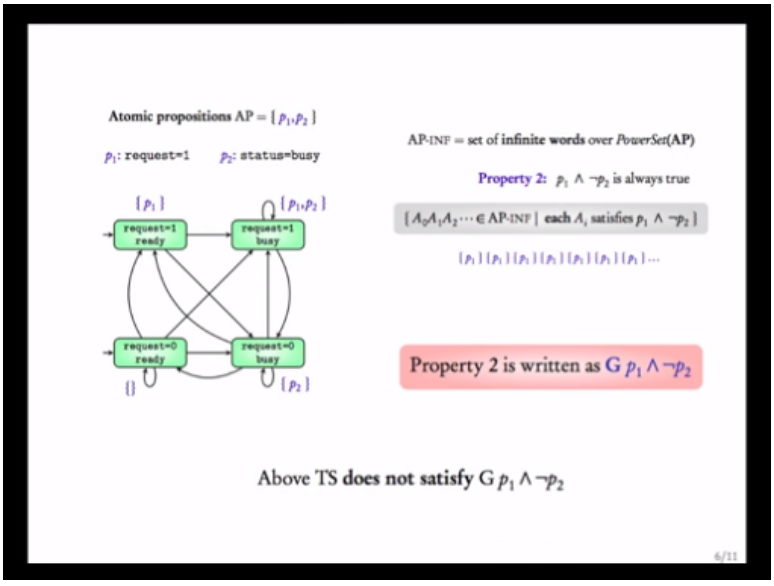
(Refer Slide Time: 14:14)



We will now ask this question what is the algorithm that is used to give this answer. Given a transition system and a property G phi where phi is a Boolean expression over atomic propositions given these 2 as inputs, my output should be does the transition system satisfy the invariant G phi.

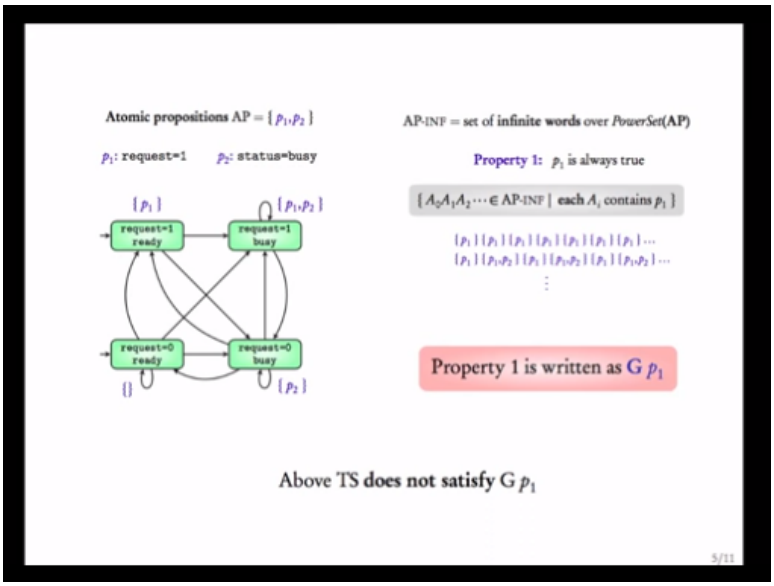
What should be the algorithm for this? The first observation is that a transition system satisfies an invariant if and only if every reachable state of the transition system satisfies phi.

(Refer Slide Time: 14:21)



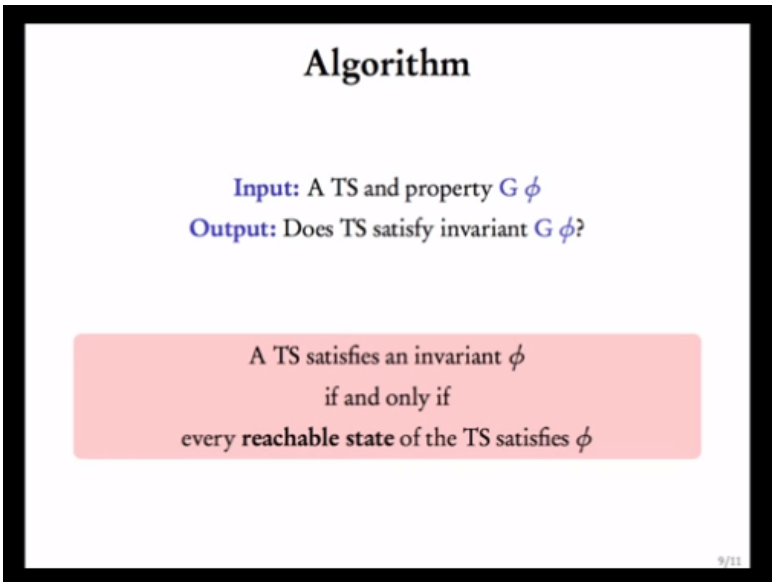
For example here this transition system satisfies this property only if every reachable state satisfies the property. Here there was a reachable state which did not satisfy it and hence this was a counterexample to this property.

(Refer Slide Time: 14:40)



Similarly, here this state did not satisfy p1, hence and it was reachable from the initial state, hence this transition system did not satisfy the invariant.

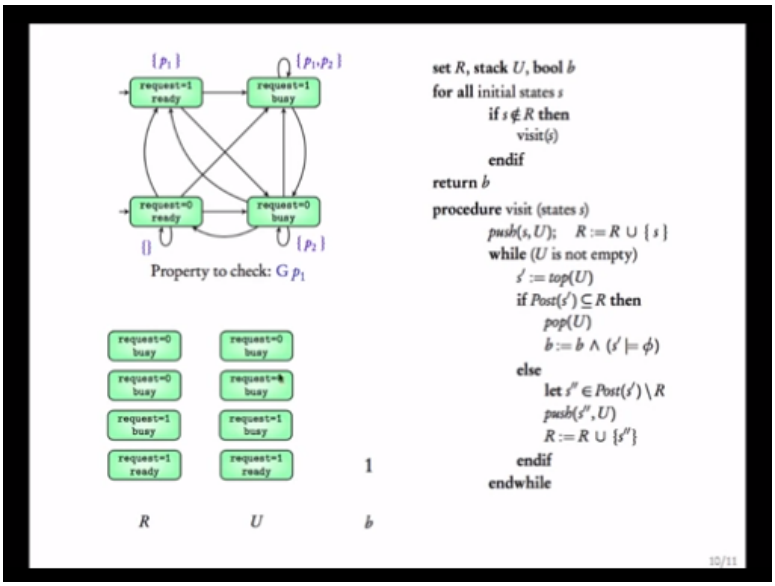
(Refer Slide Time: 15:00)



So, this statement says that in order to check if the invariant is true on a transition system explore all the reachable states of the transition system and in each reachable state check if phi is true. This is going to be the algorithm for this problem. For doing this we could

do a depth first search on the transition system. Let us now see a concrete algorithm for this problem.

(Refer Slide Time: 15:37)



Here is a pseudo code, these are the inputs we will try to understand the pseudo code by evaluating it on these inputs. This is the transition system and the property to check this Gp1. Let us start executing, the algorithm uses 3 data structures, a set are a stack u and a boolean variable b which is initially 1. Here there is a loop over all initial states s, if s is not in r then visit s.

Let us start with this initial state. This initial state is not in r, so we will visit this initial state. Now, what is procedure visit? The procedure visit takes as input a state, it first pushes the state in u and in r. Let us do that, the first state is pushed into the stack u and it is added to the set r. There is a loop now over the condition u not being empty, yes u is not empty now what you do, you pick the top of u so s prime is going to be this. If post of s prime is contained in r.

Let me explained this, post of s prime gives us all successors of s prime. Now, s prime is the state what are its successors, its successors are the state and the state, these are the 2 transitions so it has 2 successors post of s prime will be this union this. This condition

checks if post of s prime is already in r NO. So we have to go to the else, in the else part you pick some state in the post which is not in r already.

Here we have 2 choices, let us pick the state and then push it to the stack added to r, the else part will be done. Now, we go back to the beginning of the loop and check the loop condition, YES u is not empty. Now s prime is going to be top of u which is this, we check if the successors of this state are already in r, what are the successors? This itself is a successor and this is another successor however this is not in r.

So this if condition does not hold we go to the else part we choose a successor which is not in r, which is this and push it on to the stack and add it to r. People familiar with depth first search would have already recognized that this is just a depth first search algorithm. Let us continue, while u is not empty you pick the top of the stack which is this, check if all its successors are already in r this is the successor, this the successor, NO.

So, go to the else part choose a successor which is not in r it will be this and add it to u and r. Now we go back to the beginning of the while loop, we look at the top of u which is this. Now, we check if post of the state is already in r. Yes all its successors are already in r. now we enter the if part. Once we entered the if part we will pop the top of the stack and now we will check if the state satisfies the invariant phi. What is phi? Phi here is p1 and p1 said request equal to 1.

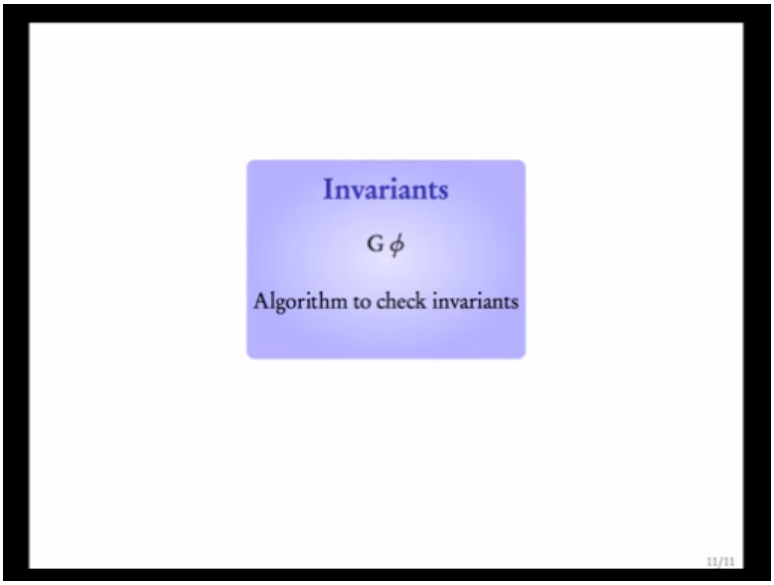
This state request equal to 0 busy does not satisfy p1, hence we need to pop the state out and change the value of b to 0, that is what happens. The if part is over and we go back to the beginning of the loop, what is the top, the top is the state all its successors are already in r. Now, you have to pop that state and change the value of the Boolean, but the Boolean variable is already 0 and AND of 0 with anything is going to remain 0.

So, the only task left now is to pop this out, go back to the beginning of the loop check this, you will pop the next state, you will pop the next state, and so on. The while end and you will go back to this location of the program. Since there is for loop you will check

the other initial state and the other initial state is this but this is already in r. So, this part will not be satisfied and now the for loop ends and the procedure returns the Boolean b and the value of b is 0.

It says that this transition system does not satisfy the property. This algorithm can be modified slightly to also give us the counter example. This was the place where the Boolean change to 0 and the stack at that point will give us the counter example to the property. Yes, this is what we said, since this is 0 property is not satisfied.

(Refer Slide Time: 23:01)



This brings us to the end of this module. We looked at particular kinds of properties called invariants, invariants specify that some Boolean expression over the atomic propositions is always true. To check this we need to check if phi which is this condition is true on every reachable state of the transition system. We saw a depth first search algorithm to check invariants.