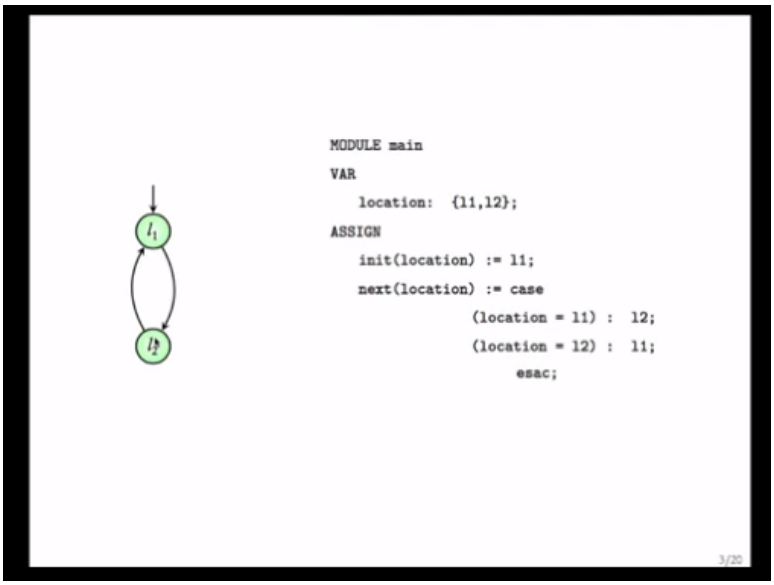
Model Checking
Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology – Madras

Lecture - 08
Simple Models in NuSMV

Welcome to Module 2 of this unit. In this module we will be seeing the format for specifying transition systems in the tool NuSMV.

(Refer Slide Time: 00:21)



Let us start with a simple transition system which has 2 states l1 and l2. l1 is the initial state there is a transition from 11 to l2 and there is a transition from l2 back to l1. How do we describe this transition system in the tool NuSMV?. NuSMV has a specific form for writing a transition system. Each NuSMV code starts with the statement MODULE main this is followed by a description of the states.

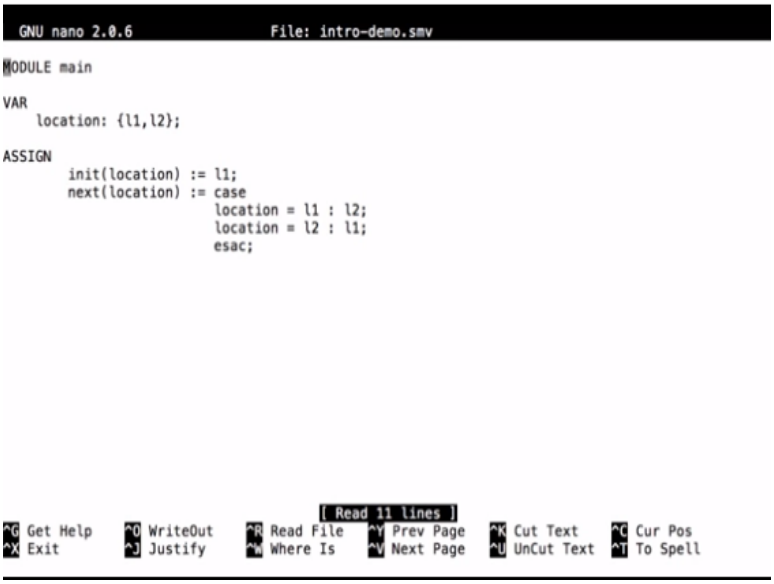
The description of the states is given the VAR block, here there are 2 states l1 and l2. A variable called location is defined and it can take the values l1 and l2. This variable location is of the enum type. So far this description says that the model has states given by location equal to 11 and location equal to l2.

It now remains to describe the transitions this is done by the assign block. Firstly, we have to specify that l1 is the initial state so the initial value of the variable location is l1.

What are the transition? If location is l1 then the next value of location is l2, if location is l2 then the next value of location is l1 this is what is written here. Next of location there is a case and esac statement we have to write it within these 2.

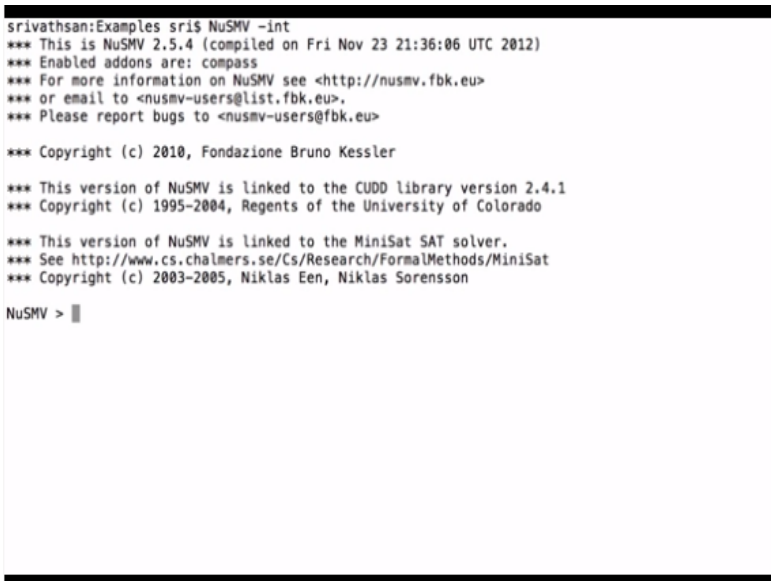
If location is l1 then the next value of location will be l2, if location is l2 then the next value of location is l1 this is the format for specifying transitions. This entire text describes the simple transition system given by this picture.

(Refer Slide Time: 02:52)



Let us now see a demo of NuSMV. I have written the NuSMV code describing the simple transition system in a separate file with the extension dot smv. Dot smv is the extension for NuSMV files let us now try to execute this code using NuSMV. If NuSMV has been rightly installed you should be able to execute the following command. This will bring us into the interactive mode of NuSMV.

(Refer Slide Time: 03:30)



The first task would be to read the model intro demo dot smv. The next command is flatten hierarchy do not worry about what it means just enter this command. The next is encode variables yet again do not worry what this means just enter it and the final task is to build the model.

Okay, now we can try to simulate and see whether we have entered the right model. To see what is the initial state described by the model and intro demo give this command pick state minus I it says there is only one initial state and the state is given by location equal to l1.

Let us now see, what are its transitions? For doing this we will give this command. We will try to simulate the transition system this says that we will simulate the transition system for 10 steps.

(Refer Slide Time: 05:02)

```
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

NuSMV > read_model -i intro-demo.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > pick_state -i

**************  AVAILABLE STATES  *************

================= State =================
0) -------------------------
  location = l1


There's only one available state. Press Return to Proceed.

Chosen state is: 0
NuSMV > simulate -i -k 10
********  Simulation Starting From State 1.1   ********

**************  AVAILABLE STATES  *************

================= State =================
0) -------------------------
  location = l2


There's only one available state. Press Return to Proceed.
```

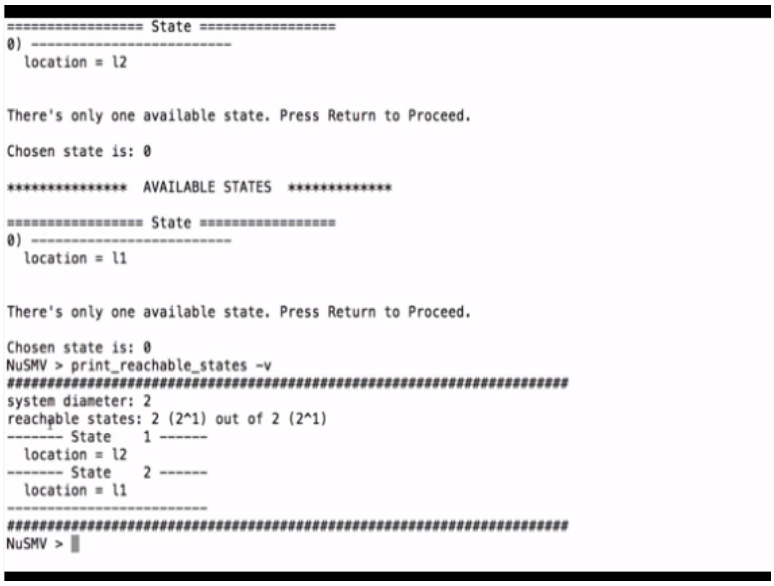
We started with the initial state, which had location equal to l1 and from that there is only one available state which is location equal to l2. Now, we pick location equal to l2 and from that there is only one available state which is location equal to l1, from that there is only one available state location equal to l2 and so on.

(Refer Slide Time: 05:27)

```
**************  AVAILABLE STATES  *************

================= State =================
0) -------------------------
  location = l2


There's only one available state. Press Return to Proceed.

Chosen state is: 0

**************  AVAILABLE STATES  *************

================= State =================
0) -------------------------
  location = l1


There's only one available state. Press Return to Proceed.

Chosen state is: 0

**************  AVAILABLE STATES  *************

================= State =================
0) -------------------------
  location = l2


There's only one available state. Press Return to Proceed.
```

So, this says this the path from l1 to l2 to l1 to l2 and so on. From this we can infer that the model that we have given is the one that was shown in the picture.

(Refer Slide Time: 05:53)



You could also give this command, print reachable states minus v. This will say the number of states and the description of states; this says that there are 2 reachable states, one which has location equal to l2 and one which has location equal to l1 do not worry about the other things.

As of now try to understand that this command print reachable states minus v will give us the number of reachable states which is 2 and the description of the each of the states. Let us now look at a slightly more complicated example. In addition to the location and the simple transitions we have a variable x which is being checked for in this transition. This transition can be taken only when x is less than 10.

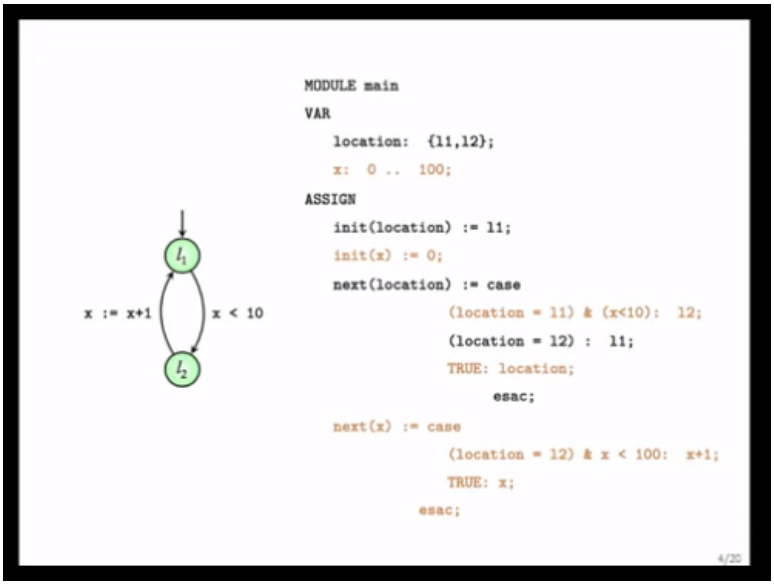
While this transition is taken the value of variable is set to x plus 1. If you remember this was called a program graph in the previous unit. Let us now see how to describe this program graph using NuSMV. This is the code from before it just describes that there are 2 locations and there is a transition from l1 to l2 and transition from l2 to l1.

Now we need to incorporate the variable x. We need to first define x in the VAR block NuSMV supports only bounded integers. So far x we need to give a bound I have given it from zero to 100. In general NuSMV can take values from minus 2 power 31 plus 1 till 2 power 31 minus 1.

Next, we say that the initial value of x is 0, what about the transitions? The first change is here instead of just saying when location is l1 go to l2 you need to say that when location is l1 and x is less than 10 go to l2. Notice that these 2 cases do not exhaust all possibilities in particular when location is l1 and x is bigger than or equal to 10 what do we do? To say that we need to add this statement which says that if these 2 conditions failed then do not change your location.

Finally we need to describe the next values for x, when location is l2 and x is strictly less than 100. So, 100 is the bound for the variable x that's why we have to add this. If location is l2 and x is strictly less than 100 then increase x by y, otherwise do not do anything to the variable x.

(Refer Slide Time: 09:12)



(Refer Slide Time: 09:15)

```
 GNU nano 2.0.6                File: pg-demo.smv

MODULE main

VAR

        location: {l1, l2};
        x: 0 .. 100;

ASSIGN

        init(location) := l1;
        init(x) := 0;
        next(location) := case
                          (location = l1) & (x < 10) : l2;
                          (location = l2) : l1;
                          TRUE: location;
                          esac;
         next(x) := case
                       (location = l1) : x;
                       (location = l2) & (x < 100) : x + 1;
                       TRUE : x;
                  esac;


                        [ Read 21 lines ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

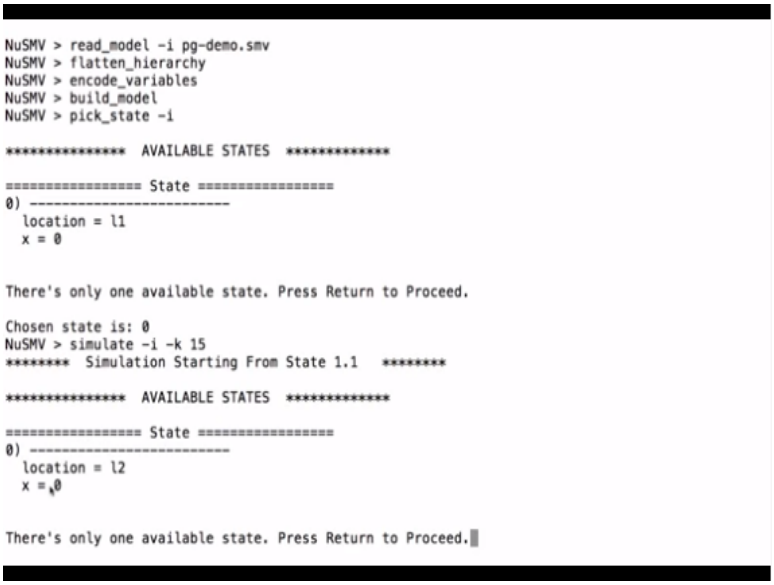
Let us now see a demo of this example the code has been saved under the file name pg demo dot smv. Let us now execute this code using NuSMV. The first task is to go into the interactive mode of NuSMV, read model, flatten hierarchy, encode variables and then build model. Now the information about the transition system written by the code and pg demo dot smv is fully here.

(Refer Slide Time: 09:55)

```
srivathsan:Examples sri$ nano pg-demo.smv
srivathsan:Examples sri$ NuSMV -int
*** This is NuSMV 2.5.4 (compiled on Fri Nov 23 21:36:06 UTC 2012)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

NuSMV > read_model -i pg-demo.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV >
```

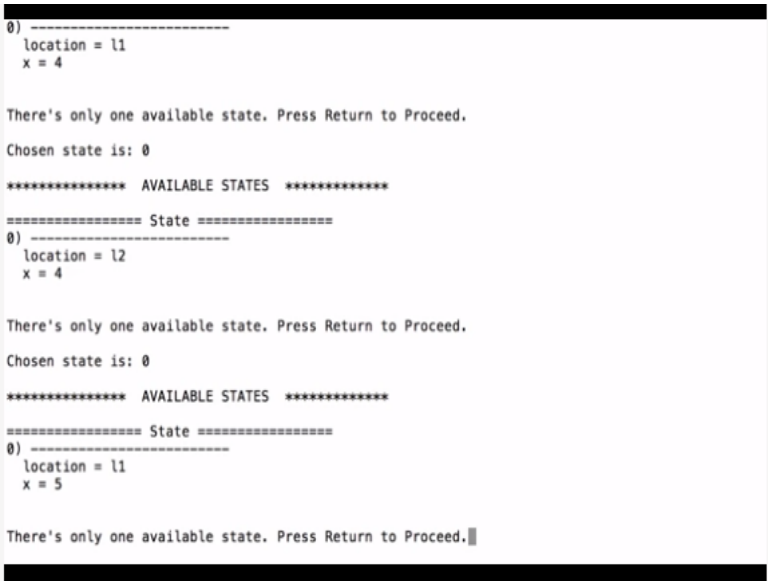
To get the initial state we need to say pick state minus I. The initial state is given by location equal to l1 and x equal to zero this is fine. Let us now simulate the transition system for say 15 steps. From location l1 and x equal to 0 we went to l2 and x equal to 0.

(Refer Slide Time: 10:30)



Then the location goes back to l1 in the process the value of x becomes 1. The next is there is no change to x but the location becomes l2 after this x becomes 2 and location is 1. So this keeps going on we have given 15 steps.

(Refer Slide Time: 10:56)



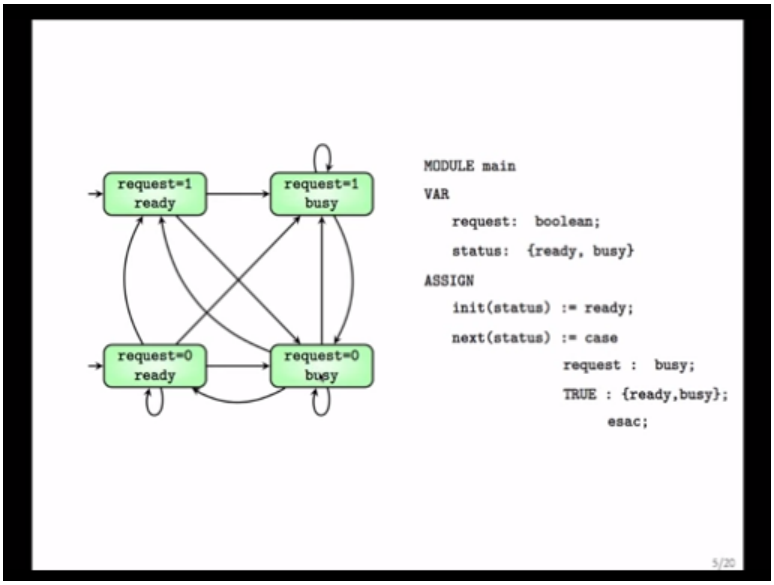
Yeah, let us try to print what the reachable states are.

(Refer Slide Time: 11:19)

```
There's only one available state. Press Return to Proceed.

Chosen state is: 0
NuSMV > print_reachable_states -v
################################################################
system diameter: 21
reachable states: 21 (2^4.39232) out of 202 (2^7.65821)
------- State    1 ------
  location = l1
  x = 10
------- State    2 ------
  location = l1
  x = 8
------- State    3 ------
  location = l1
  x = 2
------- State    4 ------
  location = l1
  x = 0
------- State    5 ------
  location = l1
  x = 6
------- State    6 ------
  location = l1
  x = 4
------- State    7 ------
  location = l1
  x = 9
------- State    8 ------
  location = l1
```

There are 21 states can you figure out why there are 21 states? In location l1 the value of x could be from zero till 10 because once it reaches 10 it cannot take the next value and the value of x cannot be incremented. At location l2 the value of x can range from x equal to 0 till x equal to nine. So this is 10 states in l2 and 11 states in l1 totally we get 21 states.

(Refer Slide Time: 12:00)



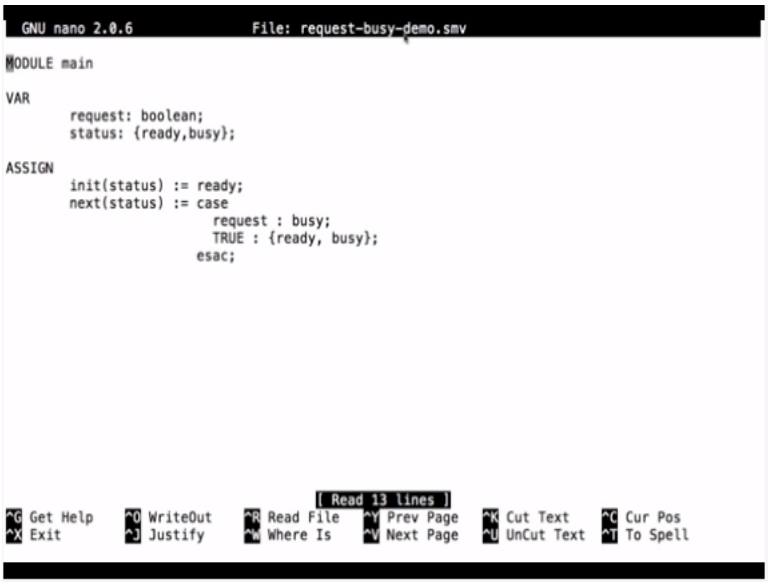
Let us now go to the next example, this time I will start with a NuSMV code and try to build the transition system represented by this NuSMV code. There are 2 variables request which is of the boolean type and state is which is of the enumeration type status can take either ready or busy.

Given these 2 variables these are the states request can be either true or false represented as request equal to 1 and request equal to 0 and state is can be either ready or busy so this gives us 4 states. Here are the transitions, first the initial state is given by init of state is ready. We have not specified what the init of request is, so there are 2 initial states one with ready and request equal to 1 and the other with ready and request equal to 0.

Here is the definition of the transition relation the next of status is defined as follows. If request is true status should become busy, otherwise status is either ready or busy. This is what it says if request is true from ready the status should go to busy. However there is no condition on the request so request could either be 1 or 0. Look at this state in this state request is true so the next value for state is should be busy. So you will not have the transition going to a state where ready is true.

From request equal to 1 and busy, you either go back to the state which is busy or you go to another busy state however here the request is 0. The other condition states that if request is not true you can go anywhere that means you have all possibilities from request equal to zero you can go to all 4 states here, and here. If request is 0 you can go here, here, here as well this one.

(Refer Slide Time: 14:49)



Let us now see a demo of this example the code has been saved under request hyphen, busy, hyphen demo dot smv. Let us now run this. As usual we start with NuSMV minus int to get into the interactive mode of NuSMV, read model, flatten hierarchy, encode variables, build model.

(Refer Slide Time: 15:26)

Let us first see the states, there are 4 states given by request equal to true status busy, request equal to true status ready, request false status busy, request false status ready, this is what we expected.

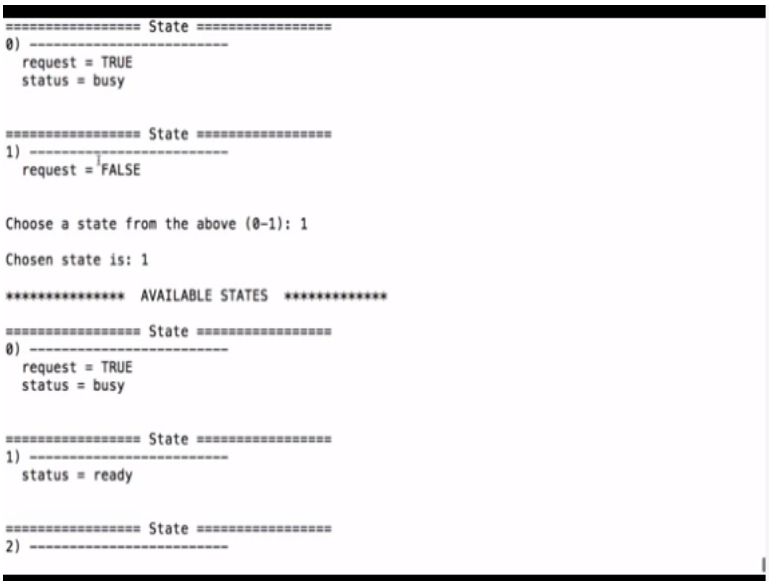
(Refer Slide Time: 15:59)

Let us now simulate this model, we first pick an initial state. There are 2 states request equal to true status is ready and in the other one request is false nothing is given about the status so this is how NuSMV prints its states. If the value of a variable is not given here then you should look at the previous state 1 represents request equal to false and status equal to ready. Yet again this is what we expected we did not give an initial value for request we just said that init of status is ready.

(Refer Slide Time: 16:51)

Let us pick state 0 and simulate for 3 steps from here. From request true state is ready there are 2 transitions, one transition leads to request equal to true and status equal to busy, the other transition leads to request equal to false and value of status here is given by this status equal to busy.

(Refer Slide Time: 17:21)

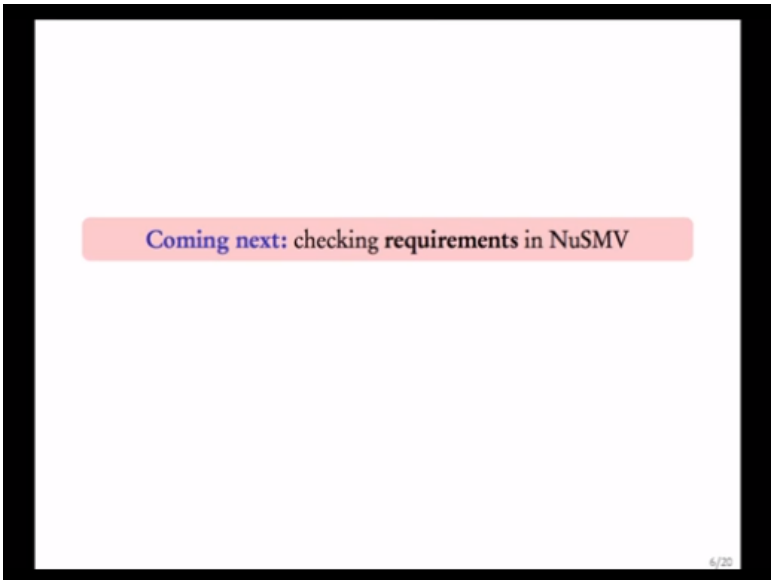


(Refer Slide Time: 17:24)

Suppose I chose 1, I chose a state where request is false and from this state there are 4 possible transitions. Let me now choose a state with request equal to true and then there are only 2 states as expected.

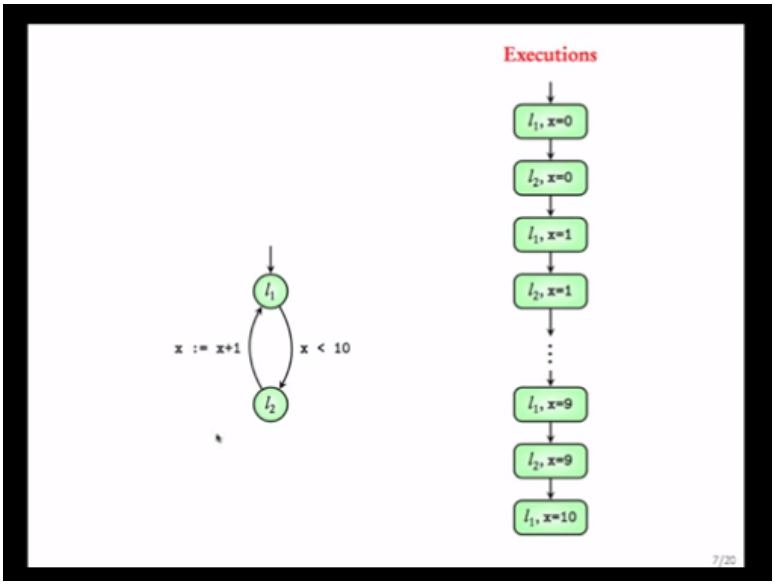
(Refer Slide Time: 17:36)

(Refer Slide Time: 17:39)



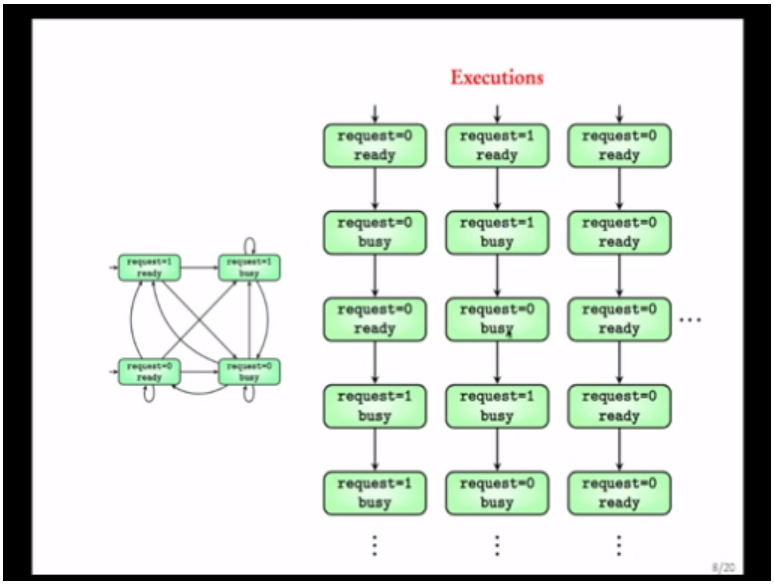
So far in this module we have seen how to write transition systems in NuSMV. In the next part we will see how we can check requirements on these modules using the tool NuSMV. We will see 2 kinds of requirements but before that let we recap what executions of a transition system are?

(Refer Slide Time: 18:06)



Look at this program graph this picture represents the transition system of this program graph starting from the initial value x equal to 0. Since, there is a single path there is only 1 execution in this transition system. An execution is just a path of this transition system.

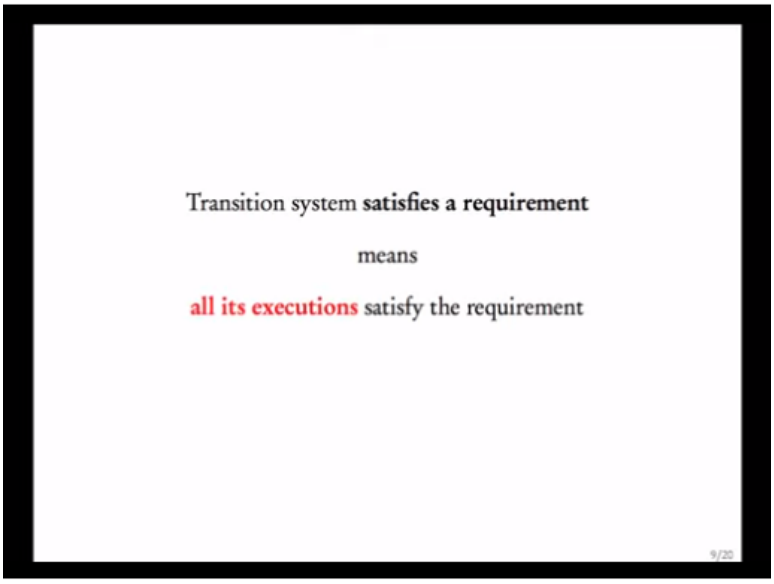
(Refer Slide Time: 18:35)



Consider this transition system there are many executions I have listed some of them. The first execution starts from the initial state request equal to zero ready, it goes to request equal to 0 busy, goes back to request equal to zero ready then takes the transition the request equal to 1 busy stays here and so on.

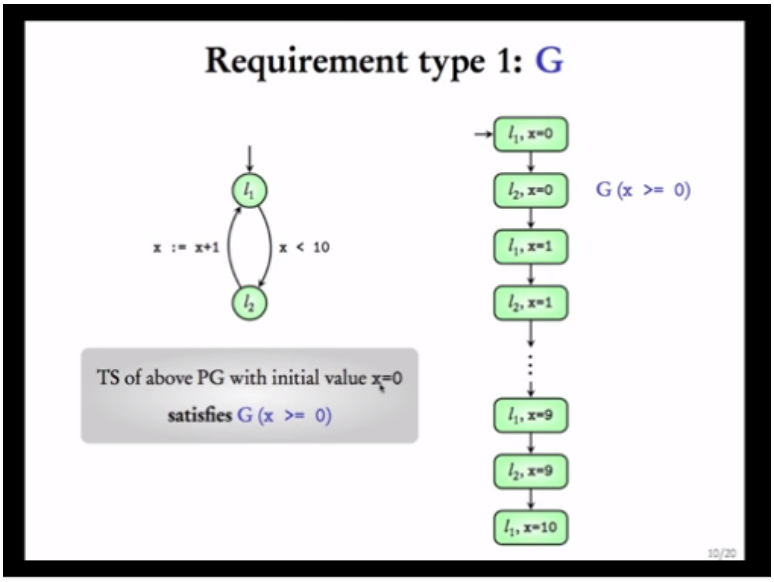
The next execution starts from the initial state request equal to 1 ready, goes to request equal to 1 busy, goes to request equal to 0 busy, comes back to request equal to 1 busy goes again to request equal to 0 busy and so on. This execution starts from request equal to 0 ready and stays there forever.

(Refer Slide Time: 19:39)



When we say that a transition system satisfies a requirement we mean that all its executions satisfy the requirement. We will see 2 kinds of requirements and this concept will become clearer. The first kind of requirement that we are going to look at is called G. G stands for global.

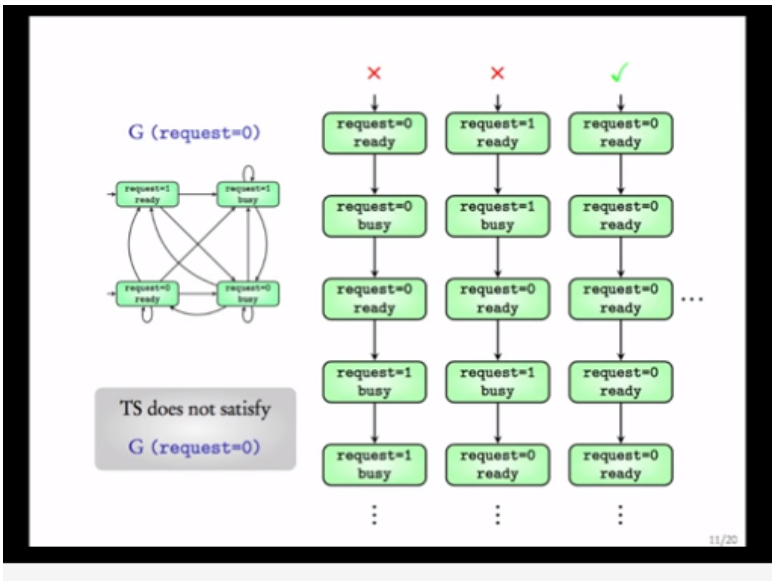
(Refer Slide Time: 20:09)



We will explain this using the example of the program graph. When we write G followed by an expression we are checking if this expression is true globally. This is the single execution of this transition system starting from x equal to 0. In this state x is bigger than or equal to zero, in this state x is bigger than or equal to zero here again it is true and so on.

Therefore, this requirement checking G x bigger than or equal to zero is true on this execution. Since this is the only execution it is also true on the transition system. We will say that this program graph satisfies G x greater than or equal to zero when the initial value of x is zero.

(Refer Slide Time: 21:11)



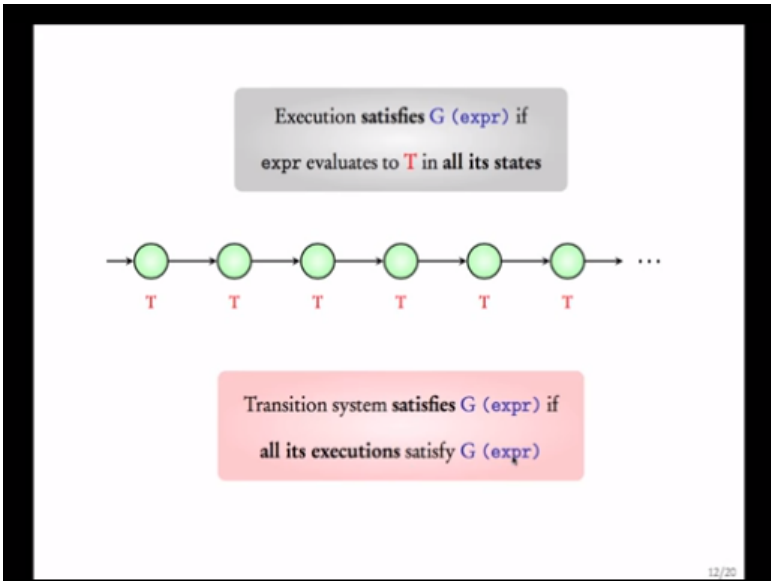
Let us see another example, suppose I write the requirement that G request equal to 0. Look at this execution is request equal to zero always during this execution no because here request becomes 1. So this execution does not satisfy the requirement G of request equal to 0. Similarly, this execution also does not satisfy the requirement G request equal to 0. The final one here indeed satisfies G request equal to 0 because in every state of this execution request is 0.

However, the transition system on the whole does not satisfy this requirement because there are executions which do not satisfy this requirement. Let me summarize what we

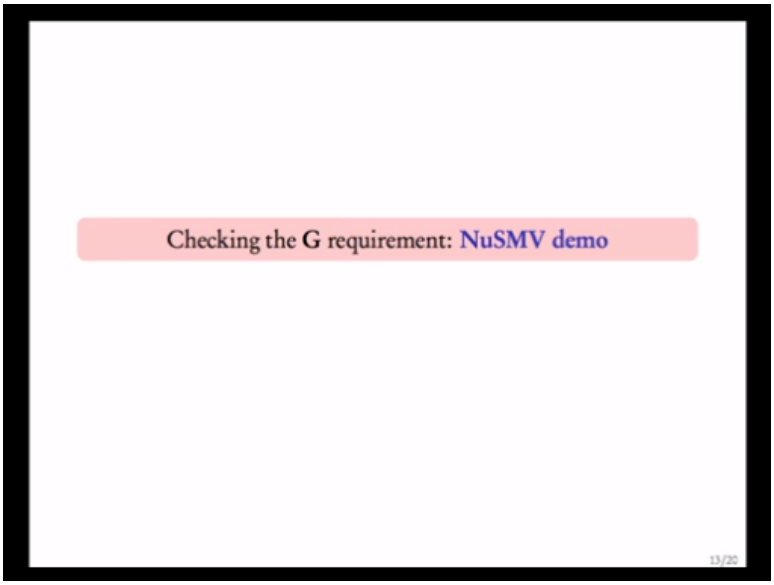
saw in the previous 2 examples. Suppose, this is an execution of a transition system, the execution satisfies G of a boolean expression if that expression evaluates to true in all its states.

Given an execution and an expression you should evaluate and see if it is true in all the states. If it is true then this execution is set to satisfy g of expression and the transition system satisfies G of expression if all its executions satisfy g of expression.

(Refer Slide Time: 23:09)

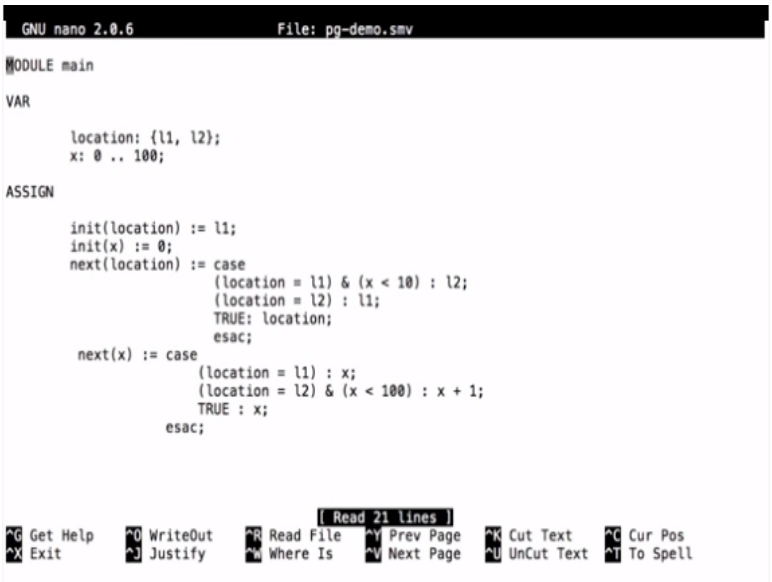


(Refer Slide Time: 23:11)



We will now see how we can check the G requirement using NuSMV. NuSMV will automatically check whether the requirement G is true on a model.

(Refer Slide Time: 23:29)



Recall that this is the code for the program graph given in the slides. We want to check if G of x bigger than or equal to 0 is true on this model. First let's get to the interactive model as usual and read the model. To check the requirement, this is the command say check ltl spec minus p and within quotes write your requirement do not worry about what ltl spec means?

Just assume that this is the command for checking if G x bigger than or equal to zero is true on the model given by pg demo dot smv.

(Refer Slide Time: 24:44)

Let us now enter, it says that the specification is true which is what we expected. Let us now look at the other example which was the one with the request and busy.

(Refer Slide Time: 25:03)

(Refer Slide Time: 25:05)

```
  GNU nano 2.0.6              File: request-busy-demo.smv

MODULE main

VAR
        request: boolean;
        status: {ready,busy};

ASSIGN
        init(status) := ready;
        next(status) := case
                          request : busy;
                          TRUE : {ready, busy};
                        esac;

                        [ Read 13 lines ]
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

This example we want to check if G of request equal to 0 is true.

(Refer Slide Time: 25:33)

```
NuSMV > read_model -i pg-demo.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > check_ltlspec -p "G (x >=0)"
-- specification  G x >= 0  is true
NuSMV > quit
srivathsan:Examples sri$ nano request-busy-demo.smv
srivathsan:Examples sri$ NuSMV -int
*** This is NuSMV 2.5.4 (compiled on Fri Nov 23 21:36:06 UTC 2012)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

NuSMV > read_model -i request-busy-demo.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV >
```

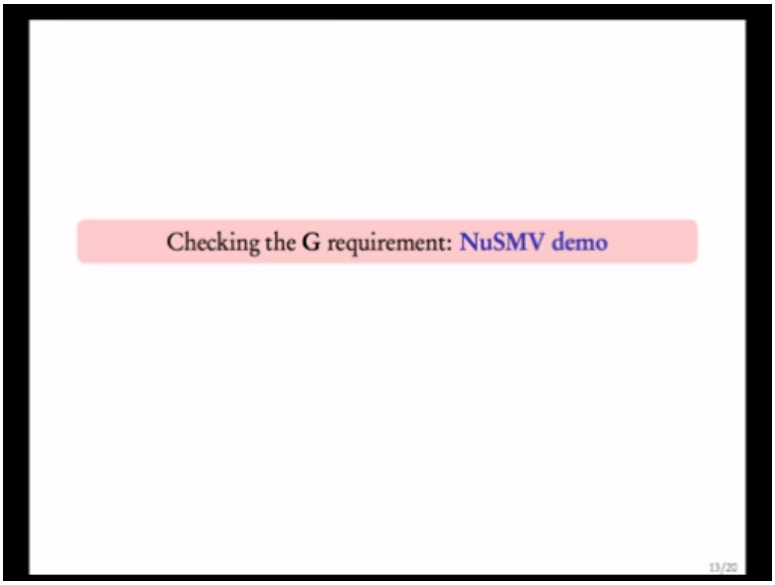
Let's first read the model to check if G of request equal to zero is true we have to give the following command check ltl spec minus p G of request equal to should not a zero. Since request is a boolean we should say request equal to false.

(Refer Slide Time: 25:56)

```
NuSMV > read_model -i pg-demo.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > check_ltlspec -p "G (x >=0)"
-- specification  G x >= 0  is true
NuSMV > quit
srivathsan:Examples sri$ nano request-busy-demo.smv
srivathsan:Examples sri$ NuSMV -int
*** This is NuSMV 2.5.4 (compiled on Fri Nov 23 21:36:06 UTC 2012)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

NuSMV > read_model -i request-busy-demo.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > check_ltlspec -p "G (request = FALSE)"
```

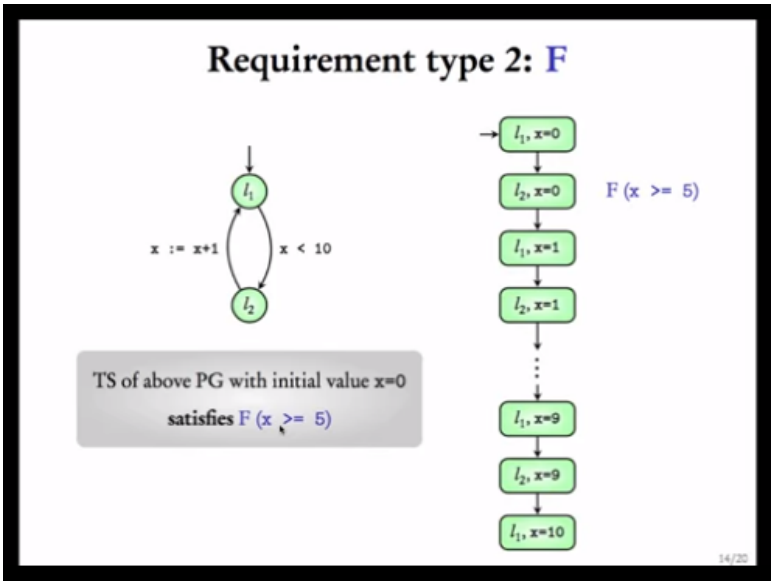
Rightly, so NuSMV says that the specification is false on the model. Moreover, NuSMV will give you an execution on which this requirement is false. In particular it gives us the execution that starts from request equal to true status equal to ready and goes to request equal to false status equal to busy and loops here. Clearly this execution starts from request equal to true which already violates the property given by the requirement.

(Refer Slide Time: 26:41)



We saw how to check the G requirement on NuSMV models. Let us now look at another type of requirement which is called F. F stands for future, lets understand F through examples.

(Refer Slide Time: 28:12)



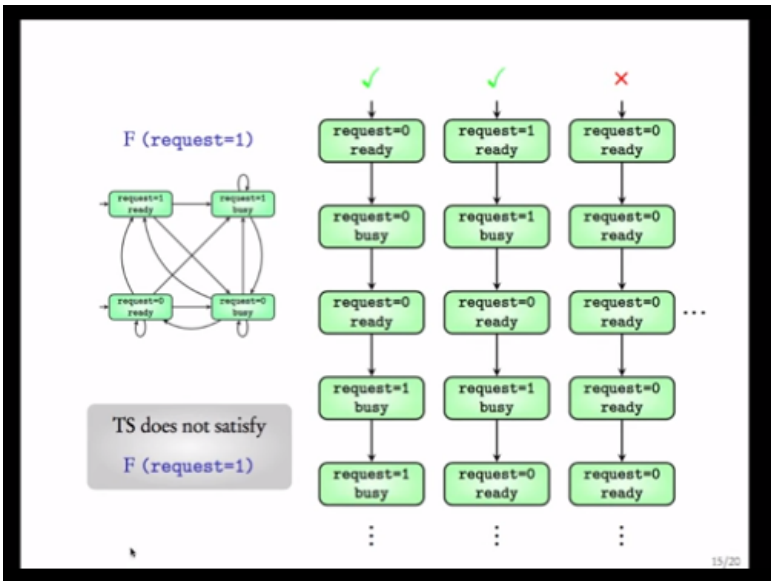
Consider the same program graph again and suppose I write the requirement F of x bigger than or equal to 5 this means that sometime during this execution the value of x becomes bigger than or equal to 5, x bigger than equal to 5 is not true here, not true here, not true here, not true here but true here in fact it would become true sometime before.

This execution satisfies F x bigger than or equal to 5 because sometime during this execution x becomes bigger than or equal to 5 we do not care what happens before or after. In some state x bigger than or equal to 5 should be true. Since this is the only execution the transition system of the above program graph with the initial value x equal to 0 satisfies the requirement F x bigger than or equal to 5.

Let see the other example, suppose for this transition system I asked the requirement F of request equal to 1 what about this? If you look at this execution sometime during this execution request equal to 1 is true.

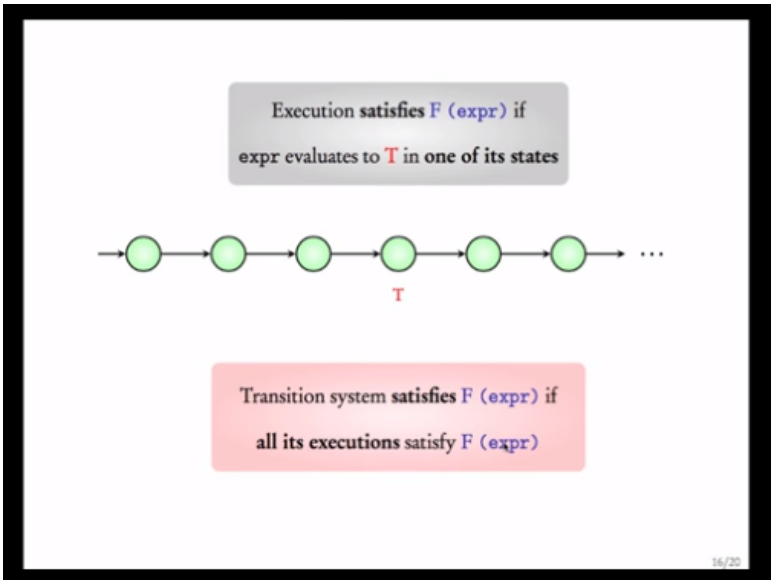
So, this execution satisfies the requirement F of request equal to 1 what about this? Already in the first state request is 1. So, this execution also satisfies the requirement F of request equal to 1. What about this execution? It always stays in 0, so this execution does not satisfy the requirement F request equal to 1.

(Refer Slide Time: 29:19)



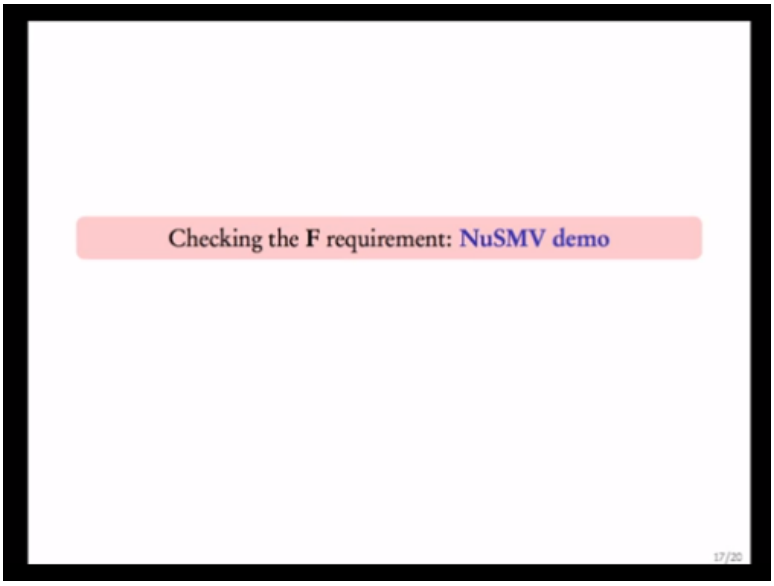
Hence, the transition system does not satisfy the requirement.

(Refer Slide Time: 30:01)



An execution satisfies F of an expression if the expression evaluates to true in one of its states. We do not care what happens elsewhere, if there is one state where expression is true the execution is set to satisfy F of this boolean expression. And a transition system satisfies the requirement F of an expression if all its executions satisfy F of expression.

(Refer Slide Time: 30:05)



Let us now see a demo of checking the F requirement on our NuSMV models. Let us start with program graph example, we do the initial steps as before the command for checking the requirement F is similar, Check ltl spec minus p and now give the requirement F, F of x bigger than or equal to 5 as expected NuSMV says that the specification is true.

(Refer Slide Time: 30:57)

Let us now look at the case of the request and busy example.

(Refer Slide Time: 31:06)

```
 GNU nano 2.0.6              File: request-busy-demo.smv

MODULE main

VAR
        request: boolean;
        status: {ready,busy};

ASSIGN
        init(status) := ready;
        next(status) := case
                          request : busy;
                          TRUE : {ready, busy};
                        esac;

                        [ Read 13 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

Now, let us check the specification check ltl spec minus p F request equal to true and it says that the specification is false and the counter example is the loop where request is false always this is what we saw in our slides.

(Refer Slide Time: 32:04)

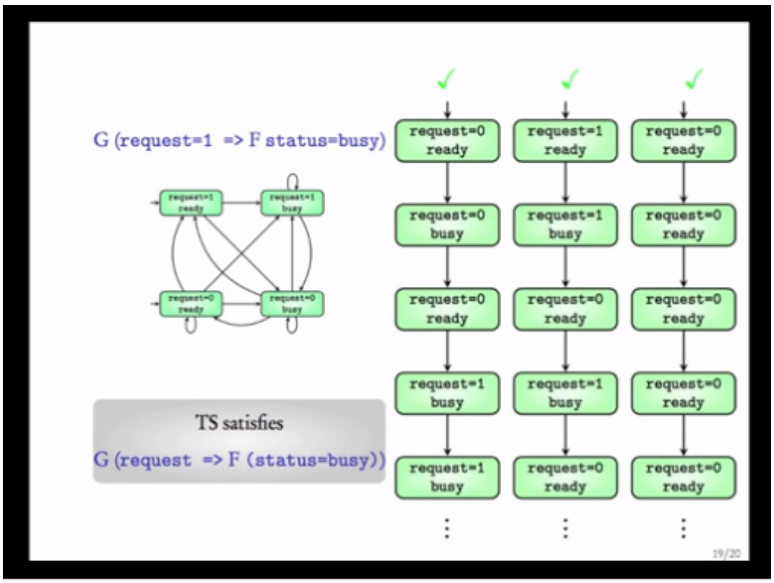
We have seen a demo of the G and the F requirement more interesting requirements can be written by combining G and F. We will look at one example consider the same transition system the requirement that we asked now reads as follows G of request equal to 1 implies F of status equal to busy. In English this says whenever request is 1 status becomes busy sometime in the future.

Consider the first execution in this state request is 0. So, the expression will automatically be through we do not have to do anything in this state. In this state yet again request is 0 we go to the state with the request equal to 1. If request equal to 1 then the requirement says that in the future of that state status should become busy which is true for this execution.

So, this execution would satisfy this requirement, what about this execution? request is 1 here. Does the status become busy in the future? Yes it does. Request becomes 1 here. Does the status become busy in the future? Yes. One can count this state itself as its future. Request is one here, Does the status become busy in the future? Yes. Because the state already has the status to be busy.

Let me say that this execution satisfies the requirement. What about this execution? Since request is 0 always we have nothing to check, so this execution also satisfies the requirement. We have seen that these 3 executions satisfy this requirement. I claim that the entire transition system satisfies this requirement that means all the executions satisfy the requirement that whenever request becomes true, the status becomes busy in the future.

(Refer Slide Time: 34:39)



It is not easy to check this requirement just by looking at this example this is where the use NuSMV becomes important. We will run this example on NuSMV, I have already read the model request busy demo and have executed the preliminary commands so the model is already built.

Let us now check the specification G request equal to true implies. The implies NuSMV is written as a hyphen and a greater than implies F status equal to busy NuSMV says that this requirement is true on the model.

(Refer Slide Time: 35:38)

The goal of this module was to introduce you to writing models and specifications in the tool NuSMV. In the rest of the unit we will see how to write more complicated models. More complicated requirements would be taken care of later during the course.