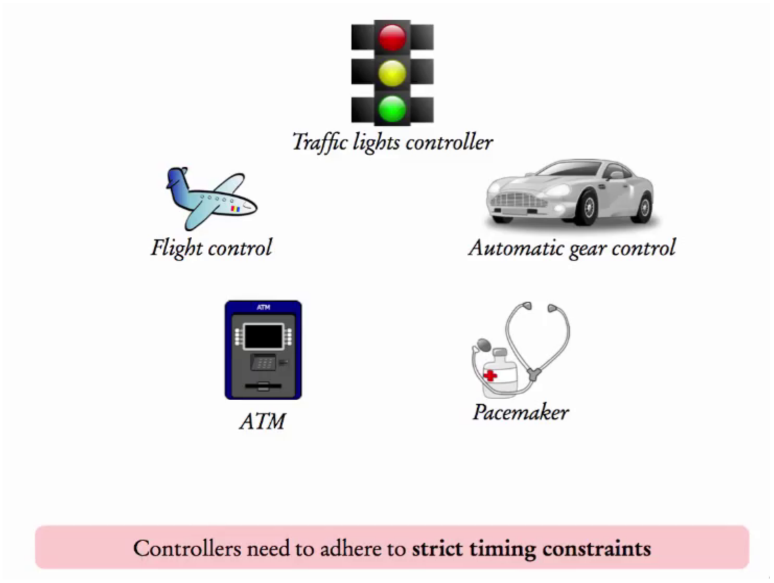
Model Checking
Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology- Madras

Lecture – 54
Timed Transition Systems

Welcome to unit 12 of this course in this unit we will be looking at how to model Controllers with timing constraints.

(Refer Slide Time: 00:14)

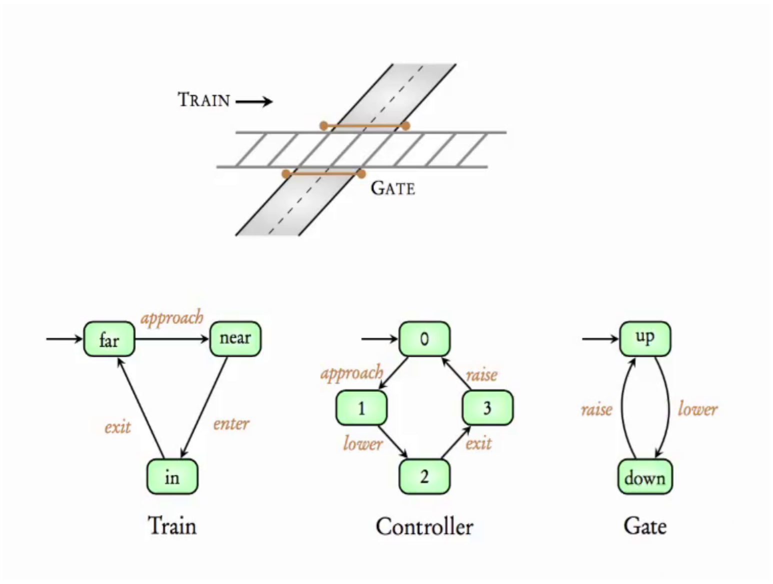

Traffic lights controller
Flight control
Automatic gear control
ATM
Pacemaker
Controllers need to adhere to **strict timing constraints**

We started off this course with the goal to come up with reliable Controllers right, so we said that there is code controlling the functioning of many of the devices that we see around us. In particular Controllers need to stick to strict timing constraints for example a Controller say for example this Automatic gear control when a request for gear changes made the response should be within certain time limit.

A Controller for a pacemaker should listen to heart beats and respond if there is no signal within a certain time limit so clearly many of the safety critical systems that we see around is have strict timing constraints and the question of Model Checking systems with timing constraints is extremely important we need good solutions to model check systems with timing constraints.

This is what we will see in this unit we will see one way of adding time to transition systems how do we model controllers where time comes into picture. Let us start with the first example.

(Refer Slide Time: 01:50)

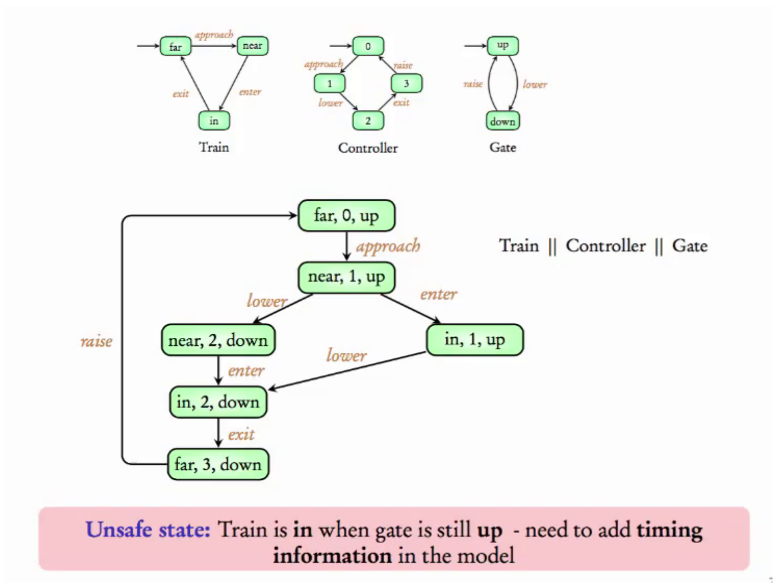


We had seen this Train Gate example in a previous unit let me recall so this is a railway crossing and there is a Gate which is controlled by a software we need to make sure that the Gate is down whenever the Train is in, let us see a model of this system. So let us start with the Train, so the Train is initially far that's denoted by this far state when its approaching close to the crossing it senses a signal approach and comes to the state near this means that it is near the crossing.

And then it sends the signal and enter and gets into the crossing that's when it is in the in state when it sufficiently outside the crossing it will say that it has exit and it will go to the far state again okay far near in again far. What about the Controller so these are some discrete states of the Controller, when the Controller listens to this approach signal of the Train it will go to some state called 1.

Now in this state, it will start lowering the Gate it will send the signal lower to the Gate and then when the Controller receives the exit signal from the Train it will raise the Gate back and this is the model of the Gate if it initially its up if it receives the signal lower from the Controller it will go down when it receives the signal raise it will go up. So this is the usual transition system that we have been seeing so far.

(Refer Slide Time: 03:59)

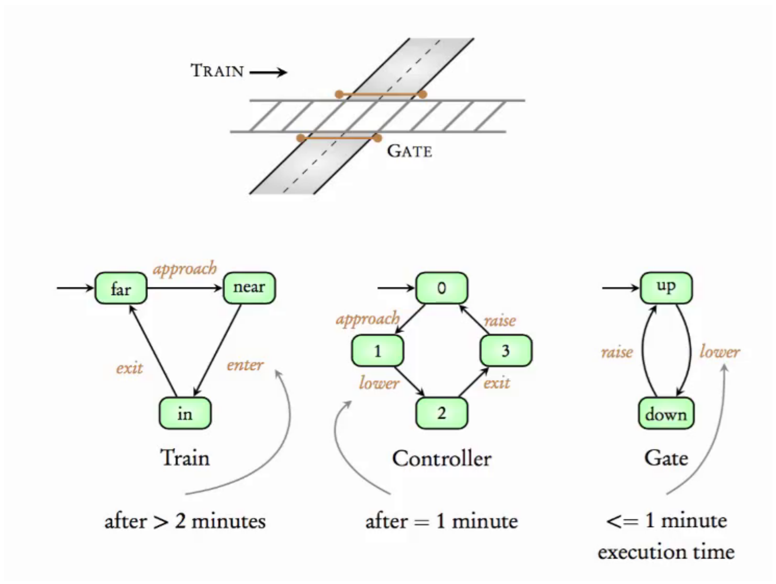


Unsafe state: Train is **in** when gate is still **up** - need to add **timing information** in the model

The joint behaviour is given by this handshake operator which is some synchronous product so from far, 0, up when there is an approach far goes to near and 0 goes to 1 okay, that's the state change so based on this idea this is the transition system of the joint behaviour, if you see there is a way in which the Train can be in, but still the Gate is up, this is not a safe state.

In this very abstract model where we have not taken care of timing it is possible that the Train is in but the Gate is still up and this is an unsafe state. So what we will do we will add some timing information into the model.

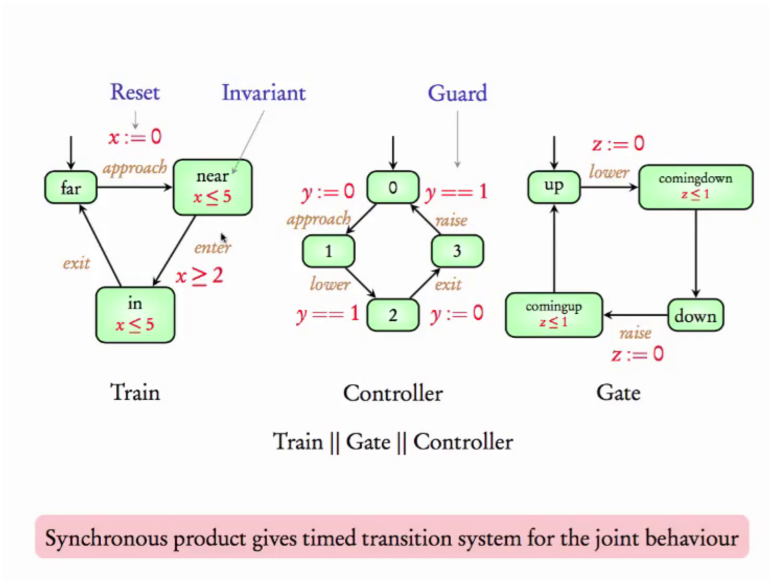
(Refer Slide Time: 05:06)



Let us see how to do it firstly, based on the speed of the Train and other physical properties we can make certain assumptions about the time taken for the Train between approach and the enter signal okay, assume that once the Train sense the approach signal to the Gate it takes at least two minutes to actually come into the crossing now what about the Controller once the Controller receives the approach signal in exactly one minute it will send the lower signal to the Gate.

And the Gate after receiving the lower signal takes at most one minute to bring it down okay these are the assumptions that we are making or the timing based on other parameters. Now the question that we are interested in is suppose you are given this timing information how do you model it in this transition system that is the goal of this unit. We will be seeing what are called Timed Transition Systems.

(Refer Slide Time: 06:35)



So this is the usual one we wanted to say that between approach and enter there is at least two time units delay to do that what we will do, we will add a clock to this model now what is this clock it is a real value variable and it is assumed to increase at the rate of global time in the sense that initially the clock is 0 forget about these two assume that this transition system has a clock initially it is 0 and has and when time lapses the value of the clock also increases at the same rate as time.

Now what happens when the Train gives the approach signal the value of this clock is set to 0 and you make sure that this transition is taken only when the value of the clock is bigger than or equal to 2 let me show this diagram we wanted to say that between approach and enter at least two time units have to elapse and for doing that we used a clock in this model and we set the clock to 0.

When the approach was given and we say that this transition can be taken only when the value of x is bigger than or equal to 2, so once x becomes 0 in this state the value of x increases as in when time increases and once two time units have elapsed the value of x becomes bigger than or equal to 2 and this transition is allowed to be taken only after this okay.

Let us continue let us look at another feature we want to say that in at most 5 units the Train will enter the crossing so once x is set to 0 it means at least two time units to enter but it cannot be in the near state for more than 5 time units that is modelled by attaching a constraint to the state, so this was a constraint on the transition this transition can be taken only when x is bigger than or equal to 2.

This constraint tells that the system can stay in this state only till the value of x is less than or equal to 5..Let us continue what about this here once again we said that between approach and lower there is exactly 1 minute delay to do that we add a clock to this system this is a clock different from here however all clocks are assumed to go at the same rate at the same rate of time okay.

So when initially y would be 0 and as and when time lapses the value of y also increases and when this transition happens that is when the approach signal comes then the value of this clock y is said to 0, and then from 0 the value of y keeps increasing along with time and this transition can be taken only when the value of y is exactly equal to 1 okay.

Similarly, between exit and raise there is exactly one time now you can reuse the same clock we do not have to – firstly, let me tell you that you can add multiple clocks for example I could have said - say y 1 is set to 0 and then y 1 equal to equal to one will - will be the constraint for this transition but that does not need because we can reuse the same clock so in this transition y is set to 0 and this transition y is asked if its value is 1.

Now after this we are not worried about the time at which the Train approached so we can reuse the same clock we can say that when this transition is taken the value of y is reset and then this transition can be taken only when the value of y equals, equals 1. Let us continue this is about the Gate we need to make a small change you say it says that the time taken for the Gate to come down the execution time is at most one minute.

So how do we model it we say that the Gate is up when it receives the signal to lower it goes to this coming down state and here are the timing constraints when it receives the signal to lower it will reset a clock said to 0 so this is the clock of this system and it can stay in the coming down state for up to one-time unit, okay.

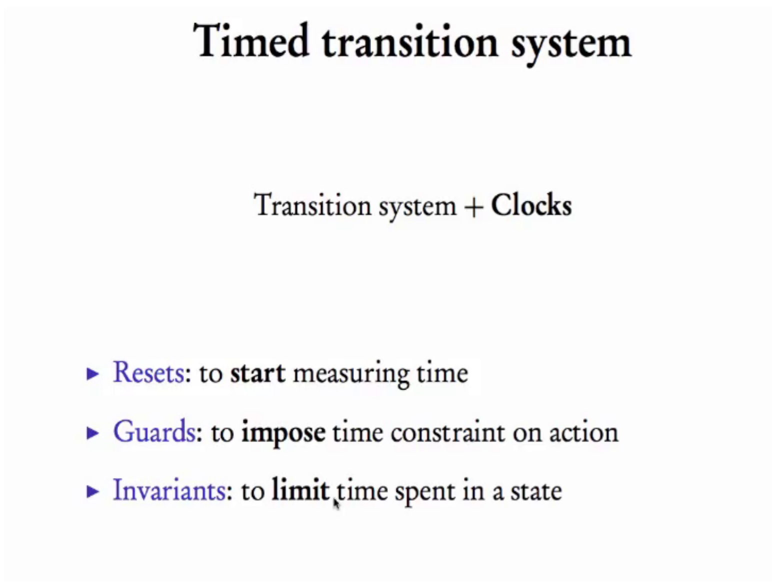
And again once the time is 1 it has to take this transition and it will go to down now from the down state in the down state it can stay as long as it wants when it listens to the raise signal it will reset a clock said and it will go to the coming up state, now it is coming up and how long can it stay here it can stay for at most 1 time unit and when the time becomes 1 then it has to go here okay, so this is a way in which we can add timing constraints into the model.

What are the basic mechanisms that we have seen we have what are called clocks, clocks are real valued variables they are part of the model and then as and when time increases the values of the clocks increase at the same rate and how can we use the clocks? we can reset the clocks during transitions. That is, this reset operation, we can use constraints on clocks in states.

This will say that the system can stay in this state as long as the value of x is less than or equal to 5. And on transitions we can have Guards this says that this transition can be taken only when the value of y equal to 1 okay, and now you can use the usual composition operator for these timed program graphs in some sense these are program graphs with timing constraints.

And this synchronous product will give the timed transition system for the joint behaviour. By synchronous what we mean is initially the system is in far, 0, up when there is an approach for goes to near and 0 goes to 1 and then both the clocks are reset, because of the invariants present in the state we need to be a bit more careful when we define the synchronous product.

But let us not worry about it for now as long as you understand that there is a way in which we can add timing constraints by making use of clocks we can proceed to the next part.
(Refer Slide Time: 15:06)



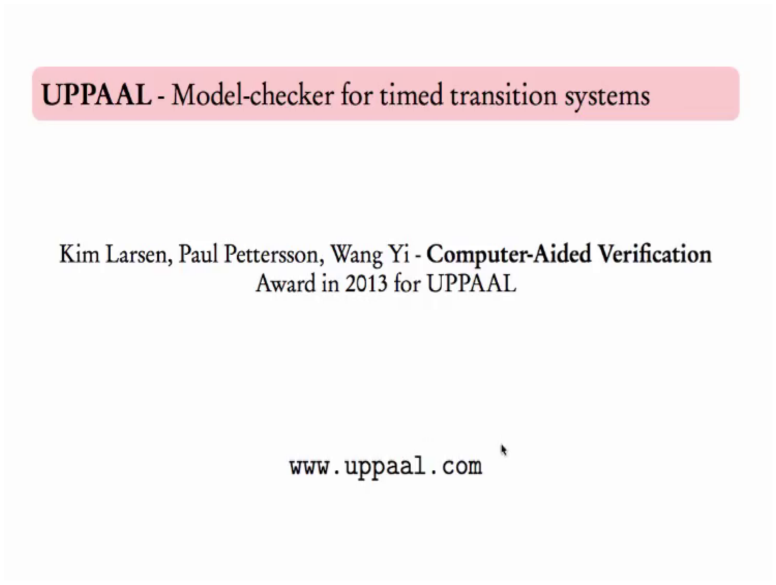
Timed transition system

Transition system + **Clocks**

- Resets: to **start** measuring time
- Guards: to **impose** time constraint on action
- Invariants: to **limit** time spent in a state

So here is a summary Timed transition systems are transition systems with clocks with a finite number of clocks and clocks can be use in the following ways they can be reset and transitions to start measuring the time and there can be guards on transitions to impose timing constraints on actions and guards can be used I mean constraints can be used on states to limit the time spent in a state.

(Refer Slide Time: 15:39)



UPPAAL - Model-checker for timed transition systems

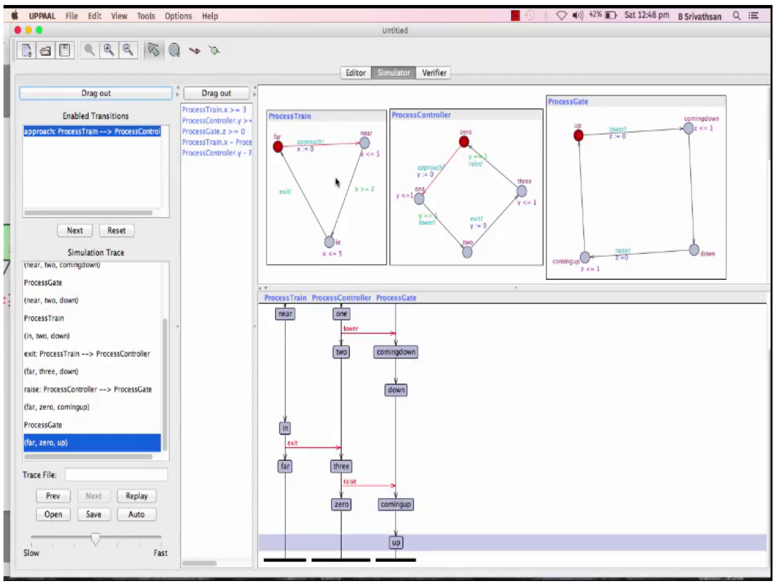
Kim Larsen, Paul Pettersson, Wang Yi - **Computer-Aided Verification** Award in 2013 for UPPAAL

www.uppaal.com

Like NuSMV which corresponds to transition system there is a model checker for Timed transition systems it is called UPPAAL you can go to this website UPPAAL.com it is a commercial software however for academic use they have a free version so you can try to download this version and check it out we will now see a demo of the example that we saw in UPPAAL.

But before that let me mention that the founders of UPPAAL Kim Larsen, Paul Pettersson, and Wang Yi were awarded the Computer - Aided Verification award for the tool UPPAAL so Computer - Aided Verification is - is a popular conference dealing with formal verification.

(Refer Slide Time: 16:38)



Enabled Transitions:

```
approach: ProcessTrain --> ProcessControl
```

Simulation Trace:

```
(near, two, comingdown)
ProcessGate
(near, two, down)
ProcessTrain
(in, two, down)
exit: ProcessTrain --> ProcessController
(far, three, down)
raise: ProcessController --> ProcessGate
(far, zero, comingup)
ProcessGate
(far, zero, up)
```

Let us now see a demo of open so this is how the UPPAAL model checker looks like when you open it when you download the software from the web page that I have mentioned it will have installation instructions and when you open it this is the screen that you will see, so this is a Graphical User Interface you can make transition systems by making use of this GUI.

Let us first start making this Train transition system so this is the transition system that we want to make it has three states and these are the actions and research invariants and guards so each transition system is called the template let us call this template as Train now this is the initial state when they have a double circle It means it is the initial state there are three states here for adding states.

They have a button here you just click on it and do this three states have been added. Now what we will do we will edit the states by giving them means so whenever you want to do that you have to click this button now let us edit location we will give the name for its initial there are no invariants for now don't worry about what urgent or committed is let us press okay so that there you go here you have got a name for this state.

Now let us do the same thing here edit location call this location near and the invariant is x less than or equal to 5. Now there is an option to check syntax go to tools and check syntax it shows something in red it says unknown when you - when you - when you bring it close it says unknown identifier yes indeed we need to declare a clock called x either you can declare it along with the Train or you can place a global declaration in this case.

We could have declared it with the Train because x is not used anywhere else, okay. However, you could as well declare in a global fashion does not make a difference for this example, let us continue - let us add transitions for adding transitions this is the button you need to press let us first make the diagram okay we have not yet given names to this so this will be in and the invariant will be x less than or equal to 5.

Here you can also move the names wherever you want for clarity. Now let us add the following information to this edge you do edit edge forget what the select is there is no guard but there is a reset and this reset is called an update what you do is you update the clock x to 0 now there is something called as sync this says is there a synchronizing action if you see the Train sense a signal approach and the Controller has to listen to the signal.

So the Train is sending this signal. So we will in UPPAAL whenever it is a signal which is being sent you have to add this exclamation and declare what is called a channel - channel approach so it is this channel means that it is a means of communication between different transition systems this app - and the Train and Controller communicate via approach which is one of the channels that Train sends the approach signal and the Controller listens to it.

when we do the transition system of the Controller it will become clearer let us finish of the Train what about this edge it has a guard x bigger than or equal to 2, now what about this enter note that this is not a synchronizing action enter is present only in Train it is not present anywhere else so we do not have to give any synchronization and there is no update as well let us leave this things blank there is no update.

And there is no synchronizing channel in this transition okay, so this is the guard. And what about this translation here there are no guards are updates but there is an exit signal and this is a synchronizing signal because the Train sense the signal and the Controller has to listen to it so we need to give exit and this is the sending so we need to declare a channel called exit as well, okay.

Now we want to add another transition system so you say go to edit and say insert template and there you can change the name of the template to Controller and Controller has four states so this is the initial one click on this to add states let us give name 0 1 2 3 there are no invariants you can just delete this so UPPAAL does not allow you to call it just 0 so just write it like this it is the initial there is no invariant this is 1 2 and 3

Now let us add transitions note that Controller as a clock y, so we can make a local declaration here saying that clock y for the Train also let us change the global declaration to a local declaration saying clock x okay, now let us add information on the transitions on this transition there is an approach so here let us finish of the update y is set to 0 and what about the sync it is a it listens to the approach signal.

So here, we have to say approach followed by? This is how UPPAAL talks about listening signals, okay. And what about this here the guard is y equal to 1 and the synchronizing so here it sends the signal lower okay and we need to add a channel lower okay this is a global declaration because it is between transition systems. now what about this one here it listens to exit and does not update of y to 0.

And finally here there is a guard y equal to 1 and it sends the signal raise lets add the declaration raise, okay. Let us check syntax so nothing red here nothing, red here now there is something called system declarations. You need to define the processes that are part of your system there is one which is Train call it process one or maybe let us call process Train then process Controller is the template Controller.

And the system is made of process Train and process Controller for now we also need to add the Gate so what we will do we will insert template and call it Gate how many states there are 1 2 3 4 so up coming down, down coming up, this is up coming down and invariant was z less than or equal to 1 this means we need to add a clock z to the Gate. And now this is up there is no invariant sorry this is down.

And there is no invariant and now the invariant is okay the state is coming up and invariant is z less than or equal to 1 yet again, let us now add the edges so this edge updates z to 0 and listens to the signal lower from the Controller. You can even redesign your state diagram now this is lower and z equal to 0 edit edge update is z equal to 0 and the synchronization is it listens to the lower from the Gate and here there is no action.

And here it is update to z equal to 0 and it listens to raise now in the system declarations we also need to add this process so the entire system is made of Train Controller and the Gate. Let us now check syntax nothing wrong here, nothing wrong here as well, and nothing wrong here as well. Now what we can do is we have defined three transition systems this was our editor we can now simulate it in the simulator, yes.

So you see this is how it looks like initially these - these are the states and you can also see it in a different way here so process Train is in far, process Controller is in 0, process Gate is in up. And this will also give you the values of the - the constraints satisfied by the clocks there are three clocks here x y z and this will tell you the constraints satisfied by clocks initially all of them are bigger than or equal to 0 and all of them are equal to each other.

Because nothing has changed and they are just evolving with time then you can do next so the Train has sent the approach signal so the Train has sent approach signal to the Controller so the Train has gone to near and the Controller has gone to 1 and the Gate is still up, you can keep doing this. Now the Train is in and you see that something is going wrong. So the Train is in, the Controller is still in this state and the Gate is still up.

This means that we are allowing for the Controller to stay in the state as long as we want so we should have added an invariant to this Controller and I mean in this state let us add it and see what happens you can stay here only as long as y is less than or equal to 1 and then you need to take the transition. Similarly, here we need to add the invariant y less than or equal to 1 okay.

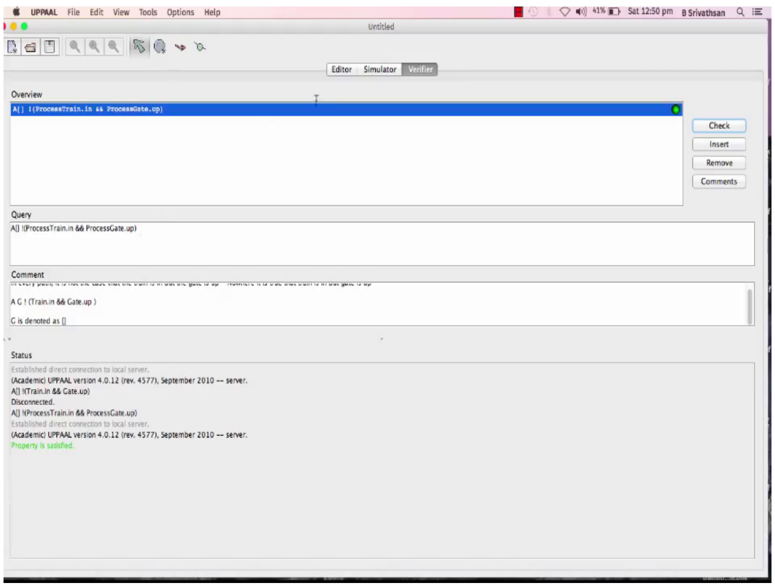
Let us now do the simulation so initially far, 0, up then the Train sense the signal approach to the Controller, Controller goes to 1 and now the Controller sense the lower signal because the time is now 1 and the Train is still near hear the Controller sense the lower and the Gate is coming down now the Gate is down and now the Train comes in okay.

Let us continue the Train as exit so when this sense the exit signal the Controller goes from 2 to 3 and now what happens the Controller asks the Train to raise sorry the Controller asks the

Gate to raise itself so it senses raise and goes back to 0 right now the Gate is coming up and now it is up so we are back to far 0 up. So you can do many more simulations here there were just three processes you can have multiple Trains and then you can see what happens.

So for defining multiple Trains there are parameters you can take a look at UPPAAL reference manual to see how you can do it well this was the simulator we can also do verification.

(Refer Slide Time: 34:18)



UPPAAL File Edit View Tools Options Help
Sat 12:50 pm B Srivathsan
Untitled
Editor Simulator Verifier
Check
Insert
Remove
Comments

Overview

```
A[] !(ProcessTrain.in && ProcessGate.up)
```

Query

```
A[] !(ProcessTrain.in && ProcessGate.up)
```

Comment

```
A G ! (Train.in && Gate.up )

G is denoted as []
```

Status

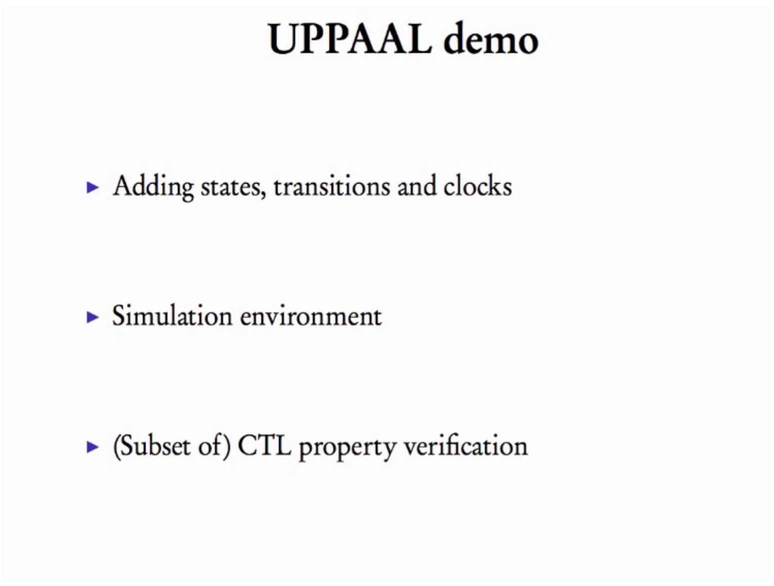
```
Established direct connection to local server.
(Academic) UPPAAL version 4.0.12 (rev. 4577), September 2010 -- server.
A[] !(Train.in && Gate.up)
Disconnected.
A[] !(ProcessTrain.in && ProcessGate.up)
Established direct connection to local server.
(Academic) UPPAAL version 4.0.12 (rev. 4577), September 2010 -- server.
Property is satisfied.
```

Verification can be done using some kind of CTL properties what do we want let see. in every - in every path let me write the property that I want to check in every path it is not the case that the Train is in, but the Gate is up essentially nowhere it should be true I mean nowhere it is true that Train is in but Gate is up. So in CTL this will be written as A G not the case that Train.in.

And so here you have to use two ANDS in path and Gate.up in UPPAAL G is denoted as [ ] okay, so A [ ] it is not the case the Train.in and Gate.up, so this is the query that we want to check let us check it. So here is the error it is says the Train is not a structure well the mistake that I made was this would be process Train process Gate because this where the processes let us now check and it says that the property is satisfied.

So it is never the case that you will reach a state where the Trains in and the Gate is up.

(Refer Slide Time: 36:21)



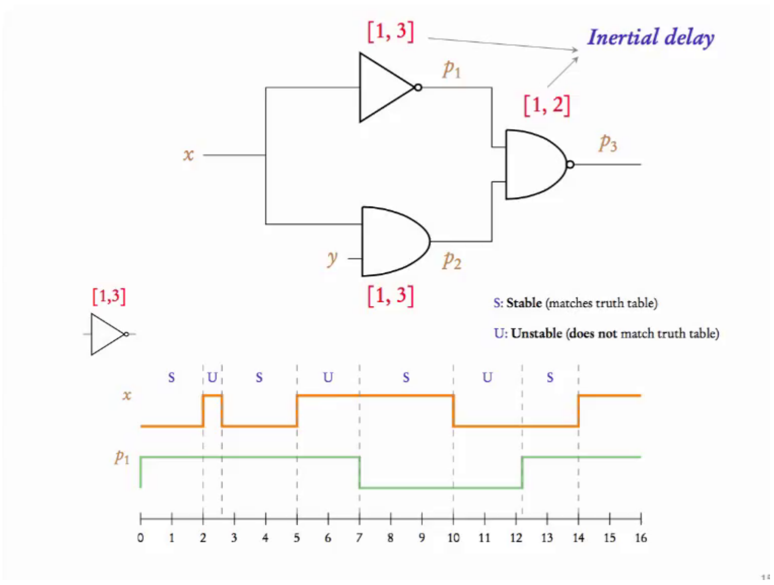
UPPAAL demo

- Adding states, transitions and clocks
- Simulation environment
- (Subset of) CTL property verification

So we saw a demo of UPPAAL we saw how to add states transitions and clocks we model the Train Gate Controller example, we had a look at it simulation environment this graphical user interface is very helpful and then we saw how we can check certain properties for more information about what kinds of properties you can check and what else can you do with the states and transitions you can have a look at the manual which comes along with UPPAAL.

So this - in this unit - this unit was meant to give you an introduction to UPPAAL. So let us now see one more example of a Timed transition system.

(Refer Slide Time: 37:18)



Here is a circuit this is a NOT Gate this is a NAND Gate and this is an NAND Gate, you are familiar with this, suppose I also give you some intervals along with this circuit let us see what it means usually when the signal is 0 here immediately the signal becomes one when the signal is one here immediately the signal becomes 0.

Similarly, if x and y are one immediately p 2 becomes 1 and so on, so the change to the output is immediate you assume that there is 0 delay in the change however there could be some non-negative time between the change to the input and the change to the output so this is called Inertial delay, this says that when the signal changes the NOT Gate needs at least one time unit to change its output moreover within three time units it will change.

For example, it says that supposed this is 0 and this is 1 suppose x becomes 1 then p 1 does not become 0 immediately the system is like this 1 1 for at least one time unit and between 1 and 3 time units gap the - the output changes that is what this inertial delay means let - let me explain it with a picture so suppose this is the NOT Gate assume that this is the timeline assume that this is how the input x changes 01101 and so on so here it is 0 and p 1 is 1 so up to this its fine.

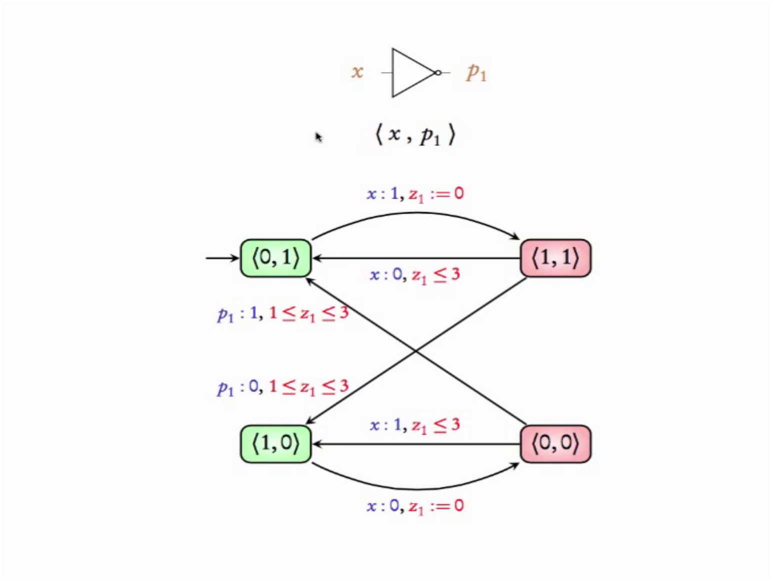
However, the value of x changes to 1 so in a normal circuit immediately this should have become 0 however let us look at the time for which x is 1 so the signal is 1 for less than one time unit and this is the bound on the NOR Gate so p 1 does not change at all and now x becomes 0 again so p 1 is still 1.

Now you see x becomes 1 and it staying at 1 for more than one-time unit so within three time units p 1 should change so p 1 becomes 0 as x becomes 1, p 1 becomes 0 and now it stays here yet again x changes and signal persists for at least one time unit and within three time units p 1 becomes 1 and so on, okay. I hope you understand what inertial delay is, so some notation we say that this the Gate is stable if the outputs and inputs match the truth table.

For example, 10 01 it said to be unstable if the input output does not match the truth table so here in this portion the output is stable I mean the system is stable here the system is unstable the Gate is unstable here the Gate is stable here the Gate is unstable because both input and output are 1, here it is table because input I mean input is 1 and output is 0 in this part it is unstable since both input and output are 0 here it is stable and so on.

So this inertial delay just say is that when the input to a Gate changes the Gate will be unstable for at least one time unit and within three time units it has to become stable that is what it means okay. Let us know model the circuit using Timed transition systems what we will do is we will give a Timed transition system for each of these Gates and the synchronous product will give us the Timed transition system for the entire circuit.

(Refer Slide Time: 41:56)



So let us start with the NOT Gate it has a timing delay 1, 3 now how do we give the state of this NOT Gate the state is given by the value of the input the current value of the input and the current value of the output so x, p 1 and there are four possibilities the red ones are the unstable states, the green ones are the stable states.

Assume that initially the Gate starts with x equal to 0 and a stable state suppose x becomes 1 so this is the signal that x is becoming 1 the value of x should change to 1and it should stay in this unstable state for at least one time unit and how do we take care of this reset a clock is z 1 to 0 during this transition and we allow this transition to be taken only when the value of z 1 is bigger than or equal to 1 and at most 3.

And what is this transition this transition is setting p 1 to 0 so this is the signal so you can look at it as an alphabet you can look at it as a letter actually okay just for clarity I have returned it has x: 1 but you can think of it as an action name of an action like approach enter exit and so on, okay.

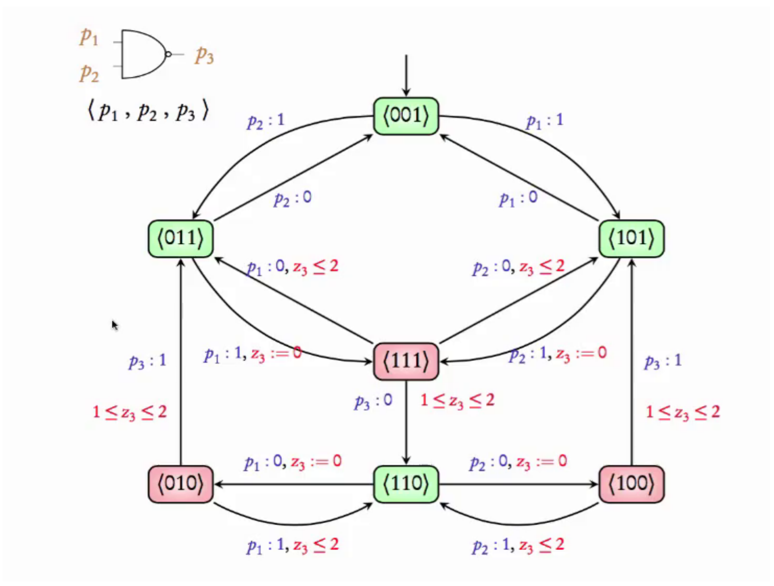
So when x became 1 the value of z 1 was set to 0 and p 1 can become 0 only when z 1 is bigger than or equal to 3 and less - and it has to be less than or equal to 3 okay, however when the system is unstable here however when the system is here it might so happened that the value of x changes back to 0 in that case the system can go back to it stable state look at this it says that the system can stay unstable for at most three time units.

Note that the circuit is not forced to become stable within this time unit what we say is that it will becomes I mean for it to become stable it needs at least one time unit but it may choose not to become stable within this time gap and go back here however if the time become equals to 3 then it has to go to either one of these two states so there is an in variant z 1 less than or equal to 3 in this state.

Let me explain it with this the symmetric thing here and we will understand now suppose x is 1 and y is 0 this is a stable state when x becomes 0 the circuit goes to this state, this is unstable x is 0 and p 1 is 0 now there are two choices either the circuit become stable because the value of p 1 changes and for this to happen value of z 1 should be between 1 and 3 or the circuit become stable because the value of the input changes before it becomes table, okay.

And that is modelled by saying that there is a transition here which makes x to 1 so this is the alphabet this is the action marking x to 1 and this should happen within three time units so within three time units either p 1 becomes one or the input changes for this to happen for the system to become stable you need at least one time unit so you can assume that there are invariants z 1 less than or equal to 3 on both these states.

So this is the time that automaton model of this simple Gate you can see that there is some notion of non-determinism coming because of time, okay the system can choose to go to the state non deterministically within one to three time units also there is non determinism due to action at sometime between 1 and 3 it can either choose to do this or choose to do this okay so this is the Timed transition system model of this circuit rather this Gate.
(Refer Slide Time: 46:39)

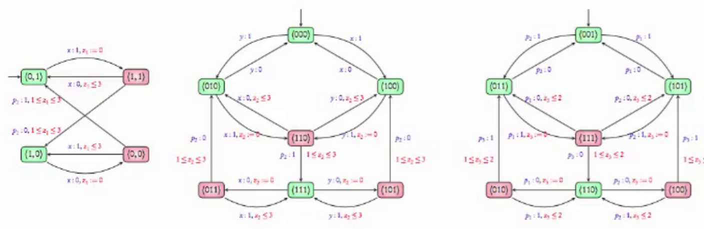


Now you can also do the same thing for this Gate for this Gate you have two inputs and one output so a state is given by value of x, y and p 2 so there are eight states however one of the states was not reachable so we have not written that so see 000 matches the truth table so this is stable however 110 does not match the truth table which is unstable. You can go through the transitions and see that they match the inertial delay of 1, 3 for this Gate.

So it says that for example when the Gate is at 010 and if the value of x becomes 1 then the Gate is unstable now it can become stable in multiple ways either the value of the inputs can change or the output changes for the output to change there is a constraint, okay. And you can assume that there are invariants on every unstable state you can be unstable for at most three time units so there is a bounded inertial delay.

Similarly, for NAND Gate you can draw a similar Timed transition systems.

(Refer Slide Time: 48:04)



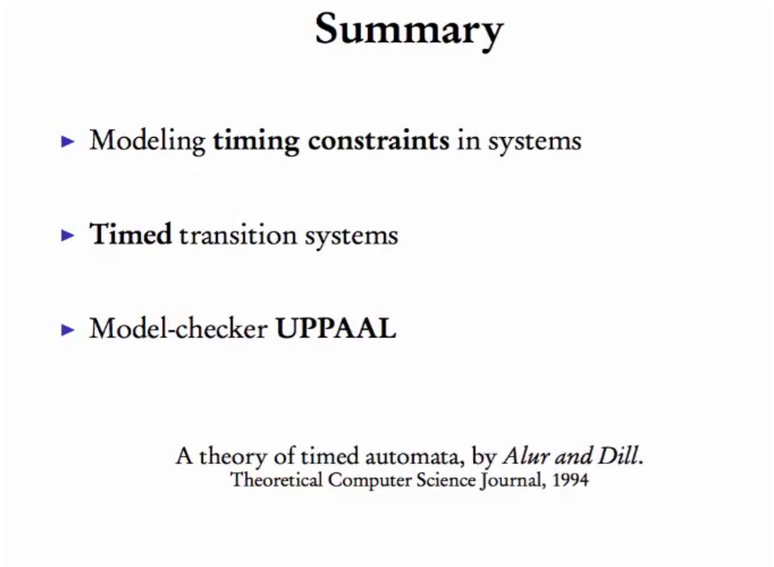
Synchronous product of above will give timed transition system for circuit

Now the synchronous product of these three Timed transition systems whether give us the time transition system for the entire circuit and the synchronization is happening through this actions for example this one says that x is becoming 1 and here there will be synchronizations on if x becomes 1 what to do on so on.

So if you want to model this in UPPAAL you might have to have - you might need to have another automaton which keeps sending the signals x becoming 1, x becoming 0 and this will all be receiving the signals that x has become 1 and y has become, so the synchronization is over these actions, and UPPAAL also allows you to do the synchronization over multiple transition system.

For example, if you have another transition system which is sending the signal x: 1? These three can sorry x: 1! These three can use x: 1? So they all can listen to the synchronization what you need to know is that this is just the handshake operator which we had seen in the first unit on a state when you see this action all the transition systems that can play this action will take the next step that is what it means okay that is what synchronous product means.

(Refer Slide Time: 49:43)



Summary

- Modeling **timing constraints** in systems
- **Timed** transition systems
- Model-checker **UPPAAL**

A theory of timed automata, by *Alur and Dill*.
Theoretical Computer Science Journal, 1994

Let me summarize in this unit we have seen how to model timing constraints in systems and for doing that we have given this definition of Timed transition systems we have given you an idea of what time transition systems are and how you can model it and verify certain properties using UPPAAL, so you can - so this unit gave you a glimpse.

If you want to know more about the Timed transition system this is this seminal paper which introduced the notion of timed automata so when we wanted to model check usual transition systems, we made use of results from Buchi automata and Finite automata for Timed transition systems. There is something called timed automata.

And all model checking results are based on the results on timed automata for more information about Timed transition system and UPPAAL you can find a lot of online references you could also take a look at the book principles of model checking which we had prescribed there is an entire lesson on Timed transition systems.