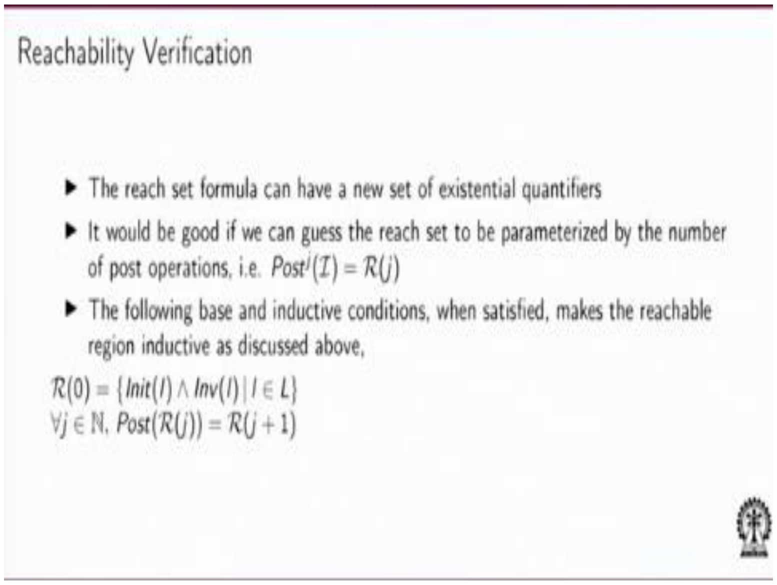
Foundation of Cyber Physical Systems

Prof. Soumyajit Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 35
Reachability Analysis (Continued)

(Refer Slide Time: 00:31)



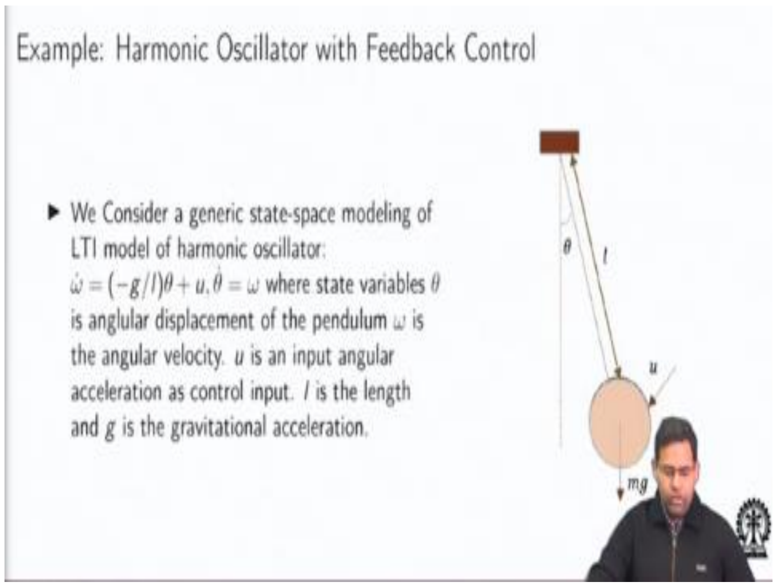
Reachability Verification

- The reach set formula can have a new set of existential quantifiers
- It would be good if we can guess the reach set to be parameterized by the number of post operations, i.e. $Post^j(\mathcal{I}) = \mathcal{R}(j)$
- The following base and inductive conditions, when satisfied, makes the reachable region inductive as discussed above,

$$\mathcal{R}(0) = \{Init(l) \wedge Inv(l) \mid l \in L\}$$
$$\forall j \in \mathbb{N}, Post(\mathcal{R}(j)) = \mathcal{R}(j+1)$$

Hello and welcome back to this lecture series on foundations of cyber physical systems. So, in the previous lecture we have been talking about methods for reachability verification. We came up with a very basic idea and we were trying to say that well how reach sets can be made. And also, we are not going into the actual optimized algorithms that are used but we are trying to give you an idea that how reach set constructions can be done well there are various optimized algorithms for this.

(Refer Slide Time: 00:53)



Example: Harmonic Oscillator with Feedback Control

- We Consider a generic state-space modeling of LTI model of harmonic oscillator: $\dot{\omega} = (-g/l)\theta + u, \dot{\theta} = \omega$ where state variables θ is anglular displacement of the pendulum ω is the angular velocity. u is an input angular acceleration as control input. l is the length and g is the gravitational acceleration.

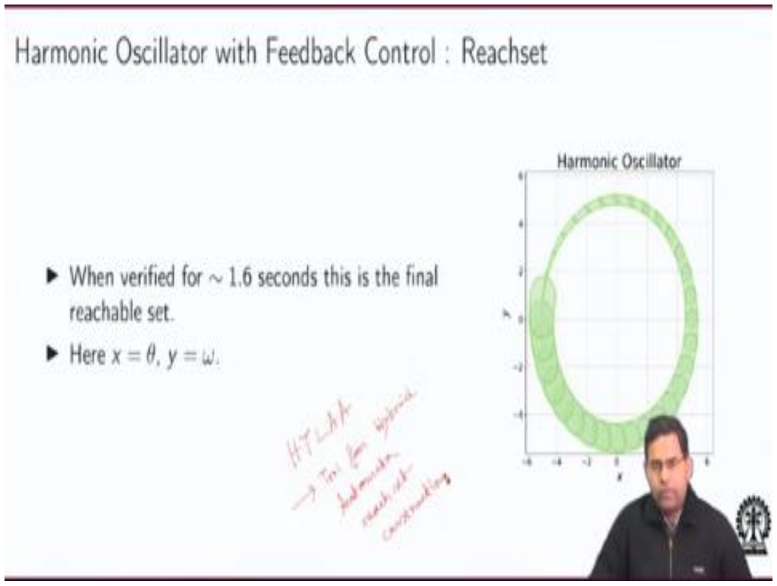
So, what we will do is for the sake of creating some interest in this topic of how to study reachability analysis and stuff we will take up few examples and we will show you the output of certain tools which are very popular in this domain which actually do this kind of reach set computation internally and try to figure out well if the reach set is going outside some safe zone and stuff like that.

So, let us take a very small example we have a generic, so we have a harmonic oscillator here as you can see that you are going to give a control force u from outside and you are if you give the force here it will overshoot to this side if you then you change the direction of the force you will overshoot to this side and stuff like that. And given the length of this rod as l and the mass of this ball as m this angular velocity ω will be following this kind of equation.

$$\dot{\omega} = \left(-\frac{g}{l}\right)\theta + u, \;\; \dot{\theta} = \omega$$

Where this angle is nothing but I mean the rate the $\dot{\theta}$ will of course be ω and we have theta that is the angular displacement of the pendulum and ω as the angular velocity and these are the two state variables in this system and u is our control input like we said.

(Refer Slide Time: 02:14)



Harmonic Oscillator with Feedback Control : Reachset

- When verified for ~ 1.6 seconds this is the final reachable set.
- Here $x = \theta$, $y = \omega$.

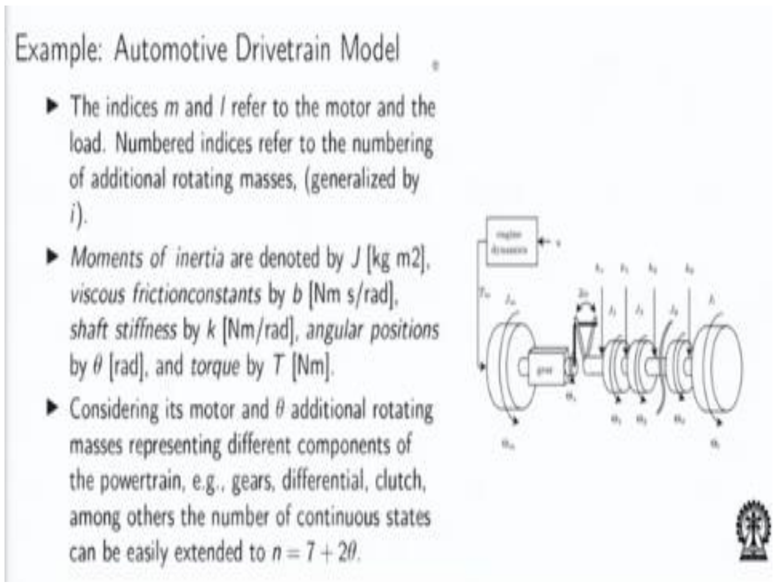
HYLAA
→ Tool for hybrid automata reach set constructing

So, for this system if we try to create a reach set, we use a tool called, so this tool you can just check it out over internet. So, this is one of the latest tools that have come up. I mean this is a very simple one to use that is why I am talking about this. All you need to do is you have to write a simple python script where you express the dynamics of the system and some desired property the tool will just run and it will create this kind of reach set for you.

And there are several other tools in this domain which does reach set construction formal verification and all this stuff this is just one very simple to use tool you can just download if you have some python interpreter in your computer you can just download and install and run this thing and get a check. So, you can check that well what is the final reach set here. So, this is a plot we are doing so in this direction we are plotting the angle.

And in this direction, we are plotting the angular velocity and this shows that with time, starting with time how the reach set has changed and as you can see so I mean initially if you have a very small subset from that you can start. And these are the possible combinations of θ and ω through which your harmonic oscillator can move about and it will not ever go outside these reach sets. So, that is an example we thought would be interesting.

(Refer Slide Time: 04:27)



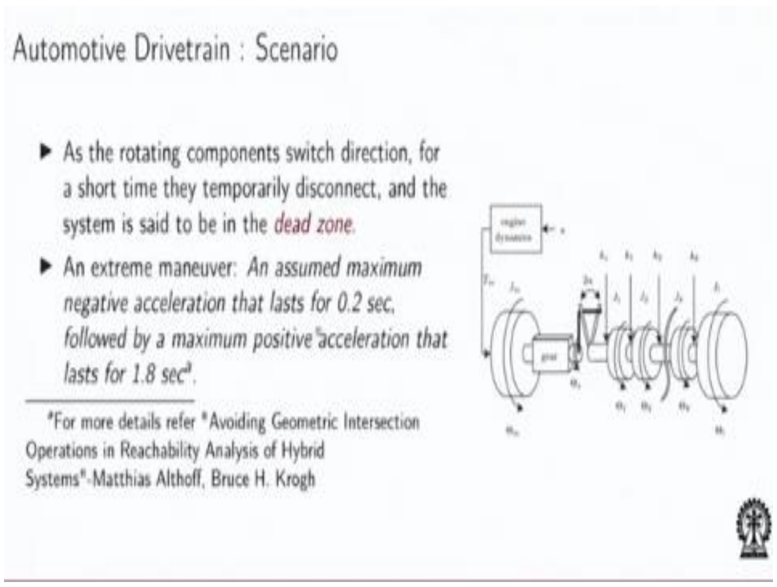
Example: Automotive Drivetrain Model

- The indices m and l refer to the motor and the load. Numbered indices refer to the numbering of additional rotating masses, (generalized by i).
- *Moments of inertia* are denoted by J [kg m2], *viscous frictionconstants* by b [Nm s/rad], *shaft stiffness* by k [Nm/rad], *angular positions* by θ [rad], and *torque* by T [Nm].
- Considering its motor and θ additional rotating masses representing different components of the powertrain, e.g., gears, differential, clutch, among others the number of continuous states can be easily extended to $n = 7 + 2\theta$.

But is it all? No. I mean you can actually use this tool for more important kind of applications also. For example, this is an automotive drivetrain model. So, if you see you have this kind of a gear mechanism which is kind of connecting through these two levers and it is going to engage each of these gears here so, these are these numbered indices are referring to the numbering of the additional rotating masses.

And you have a moment of inertia here denoted by J for this for this shaft here and there is you have viscous friction constants given by b and because and those are used for creating the mechanical equations for the system which we are not doing.

(Refer Slide Time: 05:10)



Automotive Drivetrain : Scenario

- As the rotating components switch direction, for a short time they temporarily disconnect, and the system is said to be in the *dead zone*.
- An extreme maneuver: *An assumed maximum negative acceleration that lasts for 0.2 sec, followed by a maximum positive acceleration that lasts for 1.8 sec*[a].

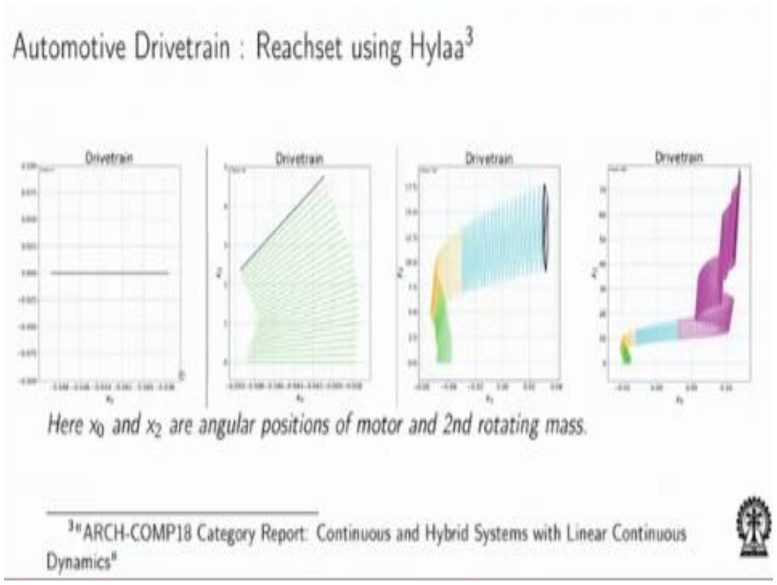
[a]For more details refer "Avoiding Geometric Intersection Operations in Reachability Analysis of Hybrid Systems"-Matthias Althoff, Bruce H. Krogh

And there is a reference paper which we will provide you this one. So, I mean if you look at this paper it actually contains the detailed analogue equations and the auto and the controller model for this kind of a system. Let us not get into that all we are trying to tell you is that if you see that when this thing rotates this shaft will either connect this way or if you suddenly change direction, it will connect the opposite way.

So, that is how the clutch mechanism and the differential mechanism will work here and we are trying to show all these different components of the powertrain gears the differential here and the clutch among others. And so, you have a very this is a significantly large system and there are a lot of continuous states here. So, what can happen is when these rotating components are going to switch direction for a short time they will temporarily disconnect.

As you can understand that this shaft, once it was driving in this direction suddenly it changes to the other direction it will move from here and up to here to engage this opposite direction, I mean this stick here in the opposite direction. So, in between what we have here is called the dead zone and suppose there is an extreme manoeuvre that can assume maximum negative acceleration that will last for 0.2 second. And it is followed by a maximum positive acceleration that is going to last for 1.8 second. Let us say you want to check if that is something safe or not.

(Refer Slide Time: 06:40)



Automotive Drivetrain : Reachset using Hylaa[3]

Here x_0 and x_2 are angular positions of motor and 2nd rotating mass.

[3] "ARCH-COMP18 Category Report: Continuous and Hybrid Systems with Linear Continuous Dynamics"

So, you can actually do a reach that analysis using this tool where you are trying to create the reach set for the angular position of let us say the motor. This motor the angular position and one of the rotating mass whichever is engaged here through this gearing mechanism whichever is engaged. For it what is the angular for I mean what is the angular position. So, let us say you are plotting the angular position of these two things that is the original motor and one of the shafts.

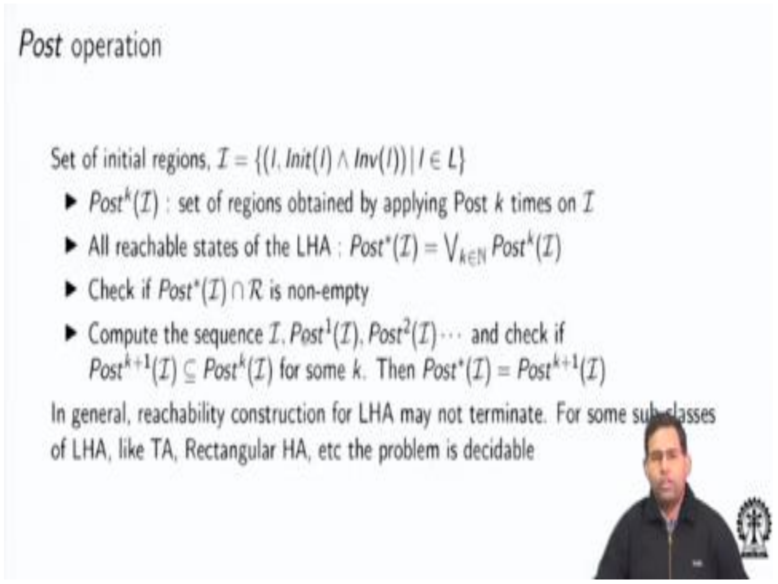
Let us say the second shaft here and they are denoted by x_0 and x_2 here let us say. So, initially it was all zero. So the motor was at zero and for the second rotating and so the second rotating mass is not rotating but the motor can have a small variance here as you can see and with time, we see that these two quantities can move like this. They are values they are possible set of values can move like this following this kind of a frontier.

And if we simulate further than with time once it has started moving like this then it can switch direction so, this is where the drivetrain which kind of switch the direction and they both are now going to go in the same direction and if we just want to simulate further this flow pipe you can keep on building and you can create this nice graph using this tool for which you can have this category report from this conference paper.

So, what I am trying to say is even in very large significant industry specifically the automotive industry this kind of formal methods can actually be applicable. If you try to check for this kind of an engine shaft connecting through this clutch and differential to the different gears. If we try to see that well under this reverse movement scenario what are the angle values and what are the maximum differences between these two angle values and what is the range of those values is it safe and all those things.

We can actually create these reach sets and we can just simply verify. So, that is one important application I just trying to wanted to touch on may not but note that the reach set construction algorithms which were used by these tools are kind of state of the art and we are not touching them. What we did initially was in our course we just try to give you an idea that how reach sets can be formalized.

(Refer Slide Time: 09:22)



Post operation

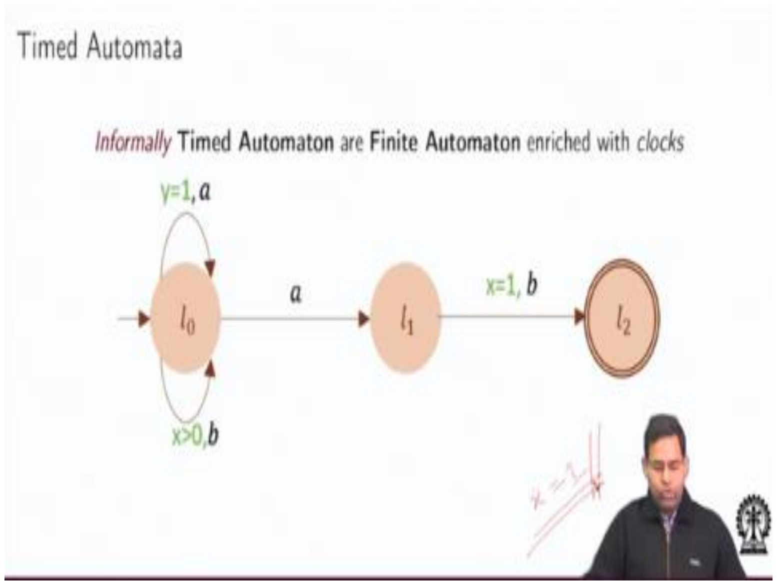
Set of initial regions, $\mathcal{I} = \{(l, Init(l) \wedge Inv(l)) \mid l \in L\}$

- $Post^k(\mathcal{I})$: set of regions obtained by applying Post k times on $\mathcal{I}$
- All reachable states of the LHA : $Post^*(\mathcal{I}) = \bigvee_{k \in \mathbb{N}} Post^k(\mathcal{I})$
- Check if $Post^*(\mathcal{I}) \cap \mathcal{R}$ is non-empty
- Compute the sequence $\mathcal{I}, Post^1(\mathcal{I}), Post^2(\mathcal{I}) \cdots$ and check if $Post^{k+1}(\mathcal{I}) \subseteq Post^k(\mathcal{I})$ for some k. Then $Post^*(\mathcal{I}) = Post^{k+1}(\mathcal{I})$

In general, reachability construction for LHA may not terminate. For some sub-classes of LHA, like TA, Rectangular HA, etc the problem is decidable

And how reach sets can be constructed through this kind of techniques like the post computation and etcetera. But of course, for using them practically there are a lot of optimized algorithms which will be employed for this purpose. So, fine from this we will just move on to another model of computation which is timed automata which is kind of simpler than hybrid automata but it also has got its usage.

(Refer Slide Time: 09:46)



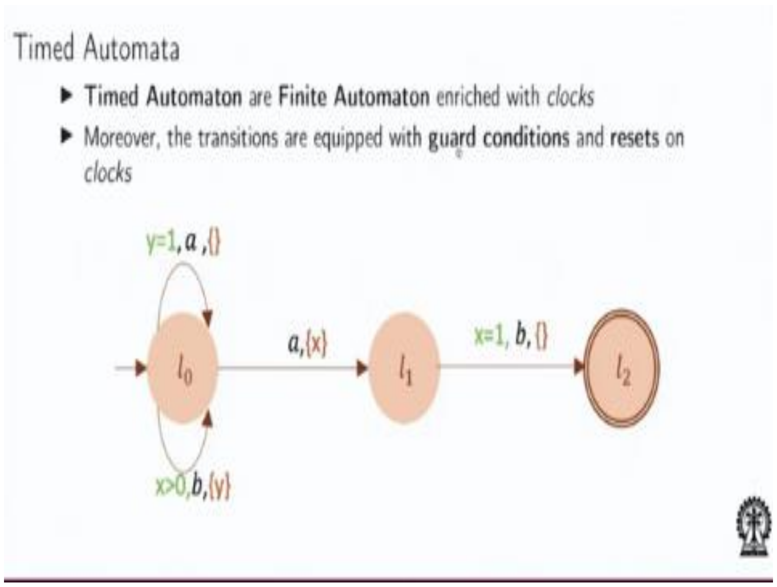
Timed Automata

Informally Timed Automaton are Finite Automaton enriched with *clocks*

So, formally what we what timed automaton. It is just finite automata and enriched with clocks. So, is very simple you consider you already know hybrid automaton and in hybrid automata we had any kind of continuous variables. Now let us just say that well whenever you are having a hybrid automaton with a continuous variable the continuous variable is only having one restriction that it cannot have any vector field of its choice.

And if the x is a continuous variable the differential of x is always 1. If I put on this restriction what we have is a timed automaton. Basically, what we are doing is all the variables can only be clocks and they can be reset like a stopwatch and they can start counting something like that. So, if we do like that then we have this kind of a simple timed automaton.

(Refer Slide Time: 10:38)

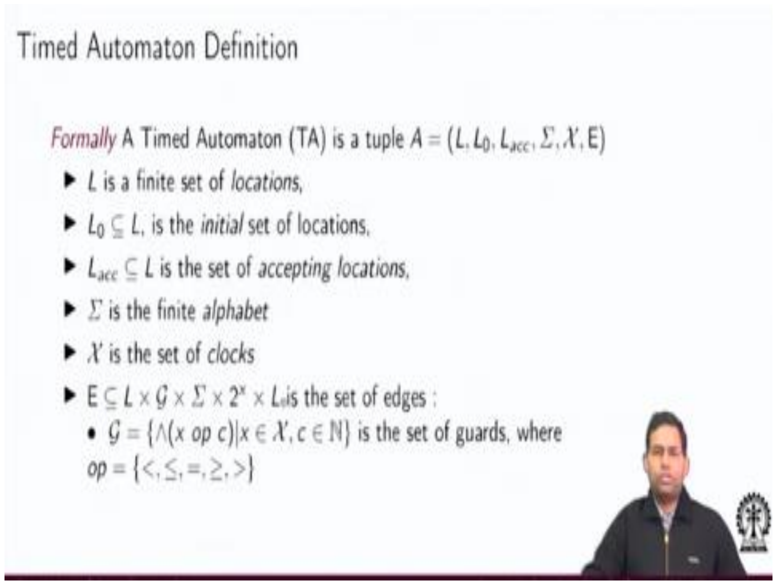


Timed Automata

- **Timed Automaton** are **Finite Automaton** enriched with *clocks*
- Moreover, the transitions are equipped with **guard conditions** and **resets** on *clocks*

So, let us say you have this timed automaton as an example and you already know that transitions are equipped with guard conditions and resets happen on clocks etcetera.

(Refer Slide Time: 10:50)



Timed Automaton Definition

Formally A Timed Automaton (TA) is a tuple $A = (L, L_0, L_{acc}, \Sigma, \mathcal{X}, E)$

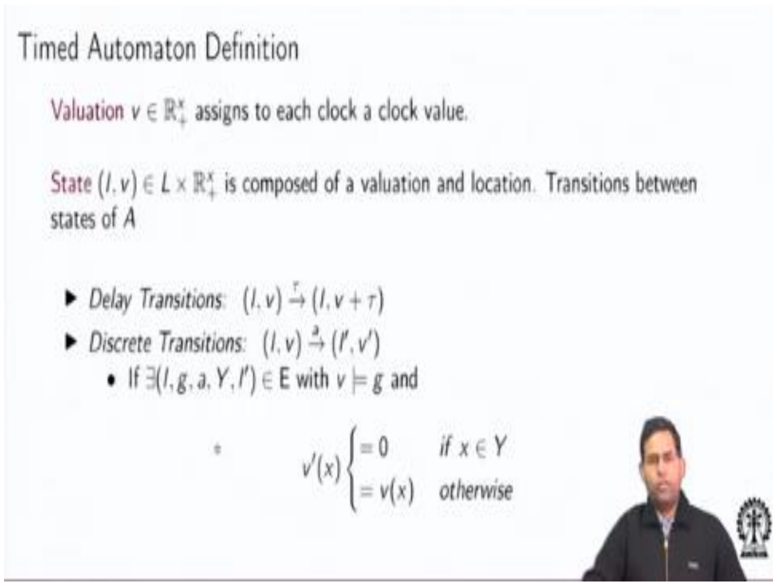
- L is a finite set of *locations*,
- $L_0 \subseteq L$, is the *initial* set of locations,
- $L_{acc} \subseteq L$ is the set of *accepting locations*,
- Σ is the finite *alphabet*
- $\mathcal{X}$ is the set of *clocks*
- $E \subseteq L \times \mathcal{G} \times \Sigma \times 2^{\mathcal{X}} \times L$ is the set of edges :
  - $\mathcal{G} = \{\wedge(x \; op \; c) | x \in \mathcal{X}, c \in \mathbb{N}\}$ is the set of guards, where $op = \{<, \leq, =, \geq, >\}$

So, just for the sake of definition a timed automaton will have a tuple which is similar to hybrid automaton. You will have the usual set of locations the set of initial locations, the set of accepting locations, you will have a finite alphabet, a set of clocks and you will have this. So, this is the set of clocks is whatever originally was your set of continuous variables and you will have your set of edges now when you have an edge basically it is a transition from one location to another location.

And when taking this transition your clock variables I mean the continuous variables they should satisfy some guard condition. And there should be an input event part of the alphabet based on which the transition will happen. And why this transmission happens? So, the set of clocks a subset of them may or may not be reset. So, that is why you have two to three power X the power set of all possible subsets of X.

And what are guard conditions they are just linear relations like we discussed earlier constants. Linear relations among clock variables with using clock variables and some constant values so, the clock variables and some constant values they will be operated by some of this one of these linear arithmetic operators and there can be multiple construct constants which can be in conjunction.

(Refer Slide Time: 12:13)



Timed Automaton Definition

Valuation $v \in \mathbb{R}_+^X$ assigns to each clock a clock value.

State $(l, v) \in L \times \mathbb{R}_+^X$ is composed of a valuation and location. Transitions between states of A

- *Delay Transitions*: $(l, v) \xrightarrow{\tau} (l, v + \tau)$
- *Discrete Transitions*: $(l, v) \xrightarrow{a} (l', v')$
    - If $\exists (l, g, a, Y, l') \in E$ with $v \models g$ and

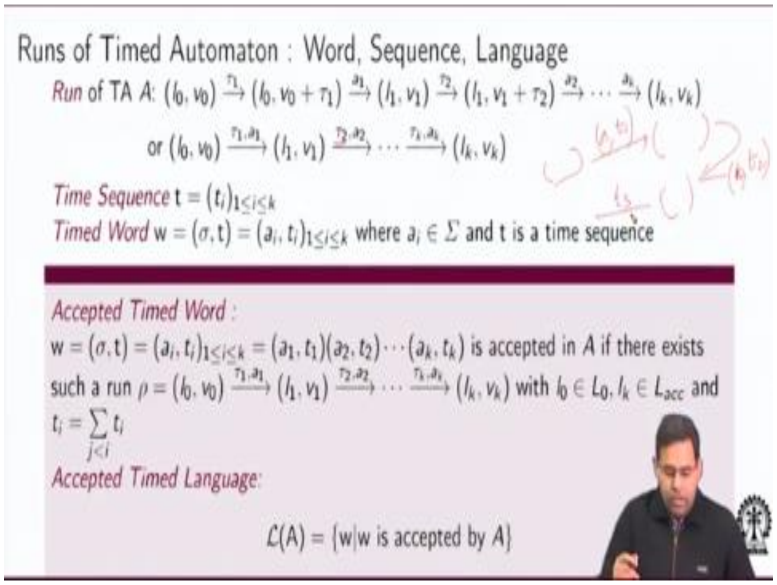
$$v'(x) \begin{cases} = 0 & \textit{if } x \in Y \\ = v(x) & \textit{otherwise} \end{cases}$$

So, that is how we define guard condition and next we have valuation of I mean let us say you have a valuation I mean it is also something we have already discussed that earlier valuations assigned values to any variable and now it will just assign a value to a clock variable. And with such valuations you can have delay transitions like with time just relapsing your clock variables will just increase by some time τ.

And if you have a discrete transition from some state like this. So, let us say your location is l you are jumping on to l' there is a guard that you satisfy that means the valuation right now that you have in l just immediately before the transition will satisfy this guard condition. And when the guard condition has, I mean satisfied then you are also supposed to do the required resetting. So, after doing the resetting the valuation will get modified.

So, you as per this formula v'(x) will be zero if they are reset and they will be v(x) that means they will be same because the transition is instantaneous. And in that way your modified state would be l' v'.

(Refer Slide Time: 13:20)



Runs of Timed Automaton : Word, Sequence, Language

Run of TA A: $(l_0, v_0) \xrightarrow{\tau_1} (l_0, v_0 + \tau_1) \xrightarrow{a_1} (l_1, v_1) \xrightarrow{\tau_2} (l_1, v_1 + \tau_2) \xrightarrow{a_2} \cdots \xrightarrow{a_k} (l_k, v_k)$

or $(l_0, v_0) \xrightarrow{\tau_1, a_1} (l_1, v_1) \xrightarrow{\tau_2, a_2} \cdots \xrightarrow{\tau_k, a_k} (l_k, v_k)$

Time Sequence $t = (t_i)_{1 \leq i \leq k}$

Timed Word $w = (\sigma, t) = (a_i, t_i)_{1 \leq i \leq k}$ where $a_i \in \Sigma$ and t is a time sequence

Accepted Timed Word :

$w = (\sigma, t) = (a_i, t_i)_{1 \leq i \leq k} = (a_1, t_1)(a_2, t_2) \cdots (a_k, t_k)$ is accepted in A if there exists such a run $\rho = (l_0, v_0) \xrightarrow{\tau_1, a_1} (l_1, v_1) \xrightarrow{\tau_2, a_2} \cdots \xrightarrow{\tau_k, a_k} (l_k, v_k)$ with $l_0 \in L_0, l_k \in L_{acc}$ and $t_i = \sum_{j<i} t_j$

Accepted Timed Language:

$$\mathcal{L}(A) = \{w | w \text{ is accepted by } A\}$$

So, just similar to hybrid you will also have a definition of a run of a timed automaton. A run is nothing but well you are in a state l_0, v_0 from there you can have a time elapsing of time t_1 τ_1. So, that is a delay transition. So all that will happen is if you are in still in l_0. Your valuation will increase by τ_1 and then due to some input event belonging to this set sigma or there is a finite alphabet of the automaton.

You can have a transition to some l_1 now whenever such transitions happens this rules have to be satisfied. And then again you can have a delay transition so that v_1 gets updated by τ_2. Then again you can have an action transition or an instantaneous discrete transition. So, overall, what you have is when you when that automaton here you can I mean you can just show like this I mean in different books you can see that well there are different ways to show.

For example, here what we are showing is with respect to the initial location we are spending here τ_1 amount of time and then we are taking a transition based on the arrival of the input event a_1. And then again, we are spending some time here the amount of time we spend is τ_2 and then based on the input action into we are doing a transition. You can show like this. Or the other standard that maybe you will see is that well there is a location and valuation.

Then some input a happens at some global time t_1 and then you go some other location valuation. From there then again there is some input a and which is happening at some global time t_2 and based on that you it happening. Here we are just showing the delays the time spend in each case but this is also perfectly fine. You just show the time at which this input event happened then, again you go to this some updated look updated state again you show the time at which some other input event let us say b happened.

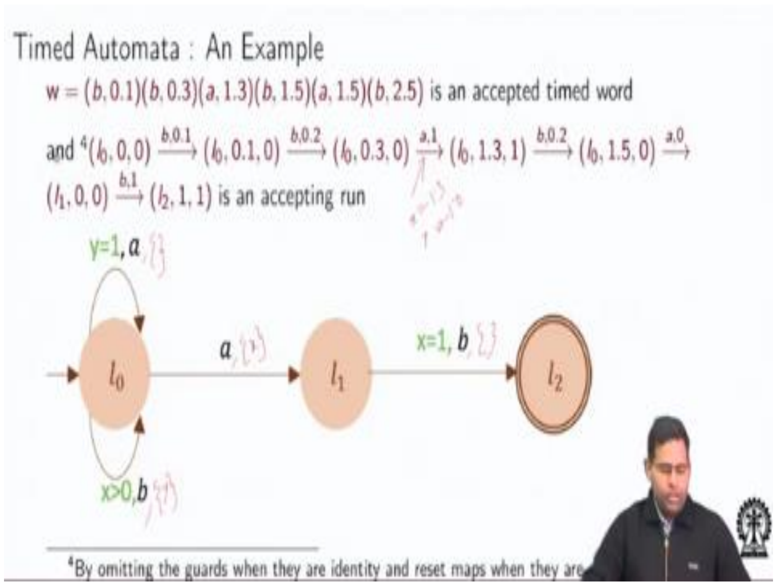
So, that is also quite fine I mean there I mean many books follow this thing also. So, overall, this different time sequences I mean you can look at this runs as a collection of this time spent in each of these states or you can also look at it as those global time instance when these transitions were taken. Let us say this happened that τ_1, this happened that this happened at t_1, this happened at t_2.

So, essentially t_2 is nothing but $\tau_1 + \tau_2$. If we are considering this representation and t_2 here in this representation is nothing but $\tau_1 + \tau_2$. Next if we have a t_3 it is nothing but $\tau_1 + \tau_2 + \tau_3$. So, either you write those absolute time sequence that is one idea of representing the time sequence or you just write the sequence of intervals spent in each of these discrete locations. So, that is also a way to represent the time sequence both are fine.

Now when you have such time sequences and with them these input events together like written like a string then what you have is what we call as a timed word. So, just like a finite automaton will have an input string a time to automaton we will have a timed what basically it will be a string where each element of the alphabet will also be paired up with a timestamp that this is when it happens.

Now the timestamp can be based on this semantics of the global timestamp or the timestamp can be based on the intervals, the amount of time stamp in the previous location something like that. So, then we talk about accepted timed word. So, let us say you have a timed word and based on that timed word you simulate your timed automaton and your timed automaton will have a run something like this. And at the end of the run if you are reaching an accept state then we will say that well this is an accepted run or this is this timed word is a member of the language of the automaton.

(Refer Slide Time: 17:32)



Timed Automata : An Example

$w = (b, 0.1)(b, 0.3)(a, 1.3)(b, 1.5)(a, 1.5)(b, 2.5)$ is an accepted timed word

and [4] $(l_0, 0, 0) \xrightarrow{b,0.1} (l_0, 0.1, 0) \xrightarrow{b,0.2} (l_0, 0.3, 0) \xrightarrow{a,1} (l_0, 1.3, 1) \xrightarrow{b,0.2} (l_0, 1.5, 0) \xrightarrow{a,0}$
$(l_1, 0, 0) \xrightarrow{b,1} (l_2, 1, 1)$ is an accepting run

[4]By omitting the guards when they are identity and reset maps when they are

So, this is a simple example like these are word which is an accepted timed word and with respect to this word if you just simulating this automaton and if it is going to eventually reach. So, we have some things missing in this picture sorry let me just correct it up. It will be the reset sets.

Here the results set are blank. So, if you can see here, you have one example. So, if you are at l 0 you can either take this self-loop based on an input event a.

And you have two clock variables x and y so you can take this self-loop provided the time of y is 1 or otherwise if b comes as an input event then whatever is the time of y and what whatever I mean as long as x is positive you can just take this self-loop. Now from l_0 based on a, you can also move on to l_1 and while you do that you will reset x and when x is again elapsed to one and if b comes then you can move on to l_2.

So, let us say l 0, 0, 0 so when this is your initial state you are in the initial location and all the clocks are just at 0. And then there is a b_1 input event coming after an elapsing of time that is 0.1 after an elapsing of time that is 0.1 an input event has come b so; what you do is you take this transition and while you take this transition. So, x will climb up to 0.1 and y will be reset as per this so y remains 0.

And then after this transition let us say you spend another 0.2 amount of time here and then a b event comes again. So, with that b event what will happen is well x will again climb up from 0.1 to 0.3, addition of 0.2. But again, y is reset so you are still here. And now suppose a event comes when after an elapsing of time = 1. So, as you can see that with a coming and with time = 1 being spent in this location.

So, your x at this moment. So at this moment when this is being happening your x is what your x is 1.3 and y is 1.0. So, what happens is which guard do you satisfy? Well the event is a so you have to take this transition or this transition. But with y = 1 you are satisfying this guard also and here there is no guard to satisfy here you do not have any guard to satisfy. So, let us say I take this transition this is a non-deterministic choice I could have taken any word, I take this transition.

So, I satisfy this guard nothing is being reset. So, I am here again with 1.3 and 1 as clock valuations of x and y. Now let us say after spending another 0.2 second let us say I have an event b that comes. So, again I am in l_0, I have taken this self-loop now again you will have a y being reset and x has climbed up to 1.5. Now let us say an a again comes. Now you see I do not satisfy this guard but here there is no guard.

So, I can always take this transition so I choose to take this one. So now I am in l 1 and both might see x y was reset here and then immediately I mean when I took this transition y was reset and then you see this is zero that means I did not spend any time in this transition in this state I immediately had another, a came. So, b came and immediately after that a came that is what this is been modelled by this zero.

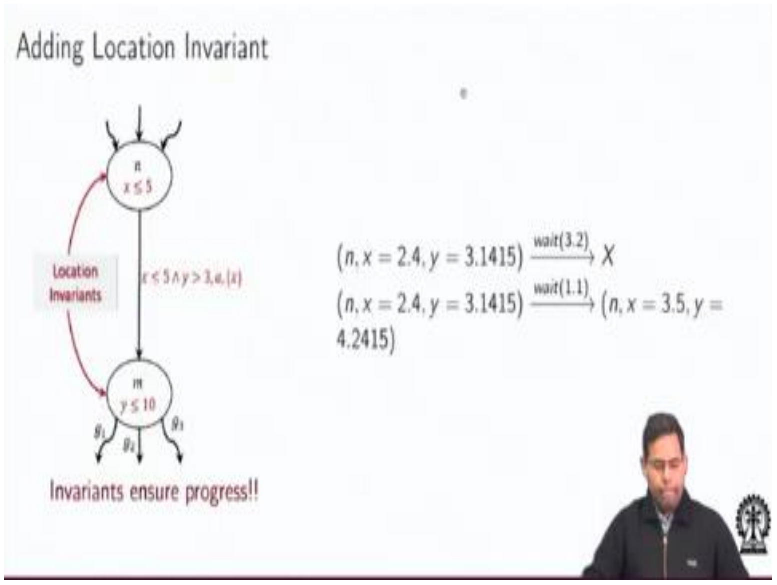
So, with b I climbed up here I got y being reset and then with a I immediately moved off to l_1 and I got x being reset. So, both x and y are zero. So as you can see that these are instantaneous transitions and here there you do not have any delay. But there is a sequence that is why you came from here to here and then you came from here to here so, you have two consecutive instantaneous transitions, so l_0 to l_0 and then again from l_0 to l_1.

And but in that sequence although they are instantaneous, they are in that sequence. So, you can understand these things these issues of having instantaneous but sequences and etcetera they will again lead to Zeno conditions like just we had in hybrid automata but we have already discussed that there. We are not discussing hybrid or I mean Zeno time divergence etcetera in the context of timed automata you are interested.

If you are interested you can just consult the relevant material that we will site here. so, from here if another event comes after another time elapsing of one unit of time, then you see that I go to l_2

but when I go to l_2 then well, I satisfy this guard because x = 1 so and I will end up here so, this is like an accepting run. So, that is how our timed automaton typically will simulate.

(Refer Slide Time: 22:59)



Adding Location Invariant

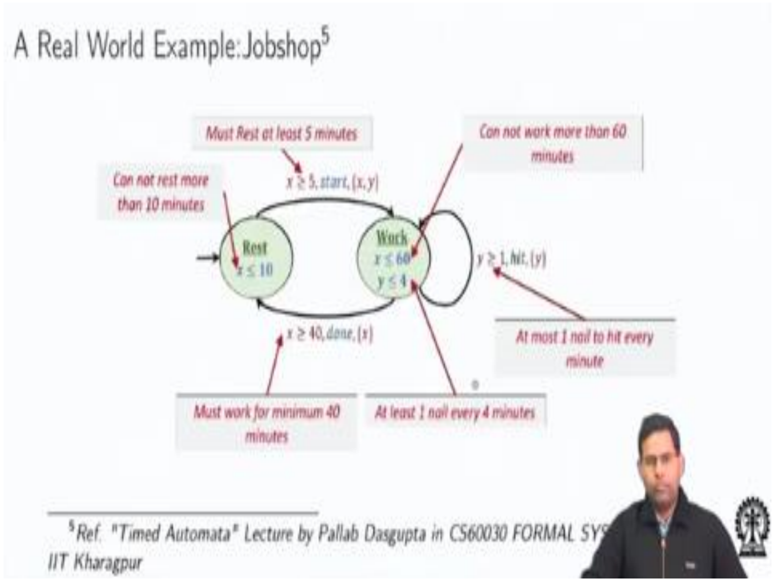
$(n, x = 2.4, y = 3.1415) \xrightarrow{wait(3.2)} X$

$(n, x = 2.4, y = 3.1415) \xrightarrow{wait(1.1)} (n, x = 3.5, y = 4.2415)$

Invariants ensure progress!!

Now just like hybrid you can have location invariance like for example if you are here and you are waiting you are spending 3.2 amount of time the invariant will get dissatisfied. So, then you are kind of going to be pushed out of the state. If you are here up to after you elapsing of time for 1.1 unit and then find your variables the clock variables will get updated.

(Refer Slide Time: 23:25)



A Real World Example:Jobshop[5]

Must Rest at least 5 minutes
Can not work more than 60 minutes
Can not rest more than 10 minutes
$x \geq 5$, start, $\{x, y\}$
Rest
$x \leq 10$
Work
$x \leq 60$
$y \leq 4$
$y \geq 1$, hit, $\{y\}$
At most 1 nail to hit every minute
$x \geq 40$, done, $\{x\}$
Must work for minimum 40 minutes
At least 1 nail every 4 minutes

[5]Ref. "Timed Automata" Lecture by Pallab Dasgupta in CS60030 FORMAL SYS
IIT Kharagpur

So, here we have a small example of a jobshop scheduling. We will show you some few examples of timed automata based modelling. So, for example here we have a jobshop scheduling problem and how that is being modelled here so, we are trying to show a factory where there are a set of people who are hammering nails and we are trying to have automata-based specification of when exactly they work and how much they work stuff like that.

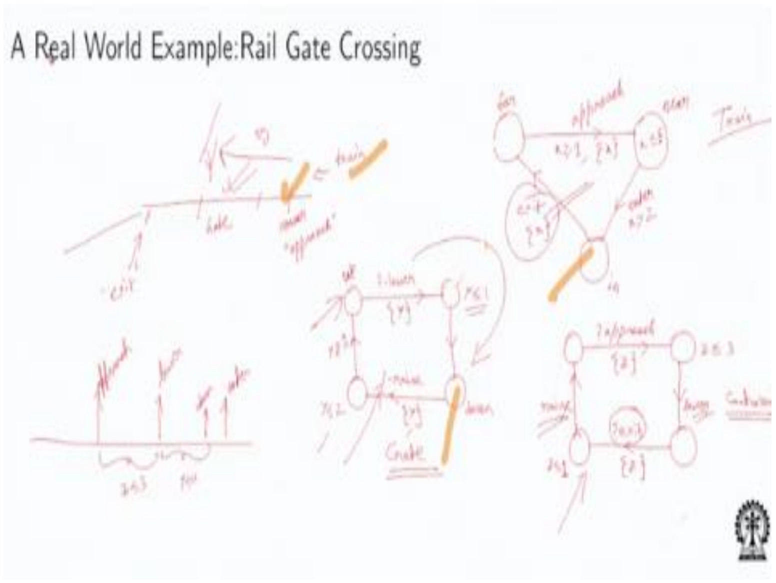
So, they can be in any of these two states they can either be resting or they can be working so this is an automatic which kind of nicely captures different constraints on that. So, let us say I do not allow them to rest for more than 10 minutes so that is how I modelling. But they must rest at least for five minutes so that would be a guard here then with the start even they will start working both the clocks x and y will be reset.

During work they have given our specification of how much they should work. They must hammer one nail every 4 minute. So, that is specified by y less than 4 and whenever they hammer one nail there is a hit event. But at the same time, they should not be hammering a nail too fast because that can kind of compromise your work quality. So, you should not hit more than one nail in every minute.

So, whenever you are hitting the nail it y this value that has been reset after the previous nail was hit, this must have time elapsed at least by one. So, you do not hammer more than one inside a minute that will be classified here and at the same time you must have more one inside every 4 minute that is kind of captured in this location environment. Overall, the person cannot work continuously for more than 60 units of time in minutes let us say then you can have a invariant like this here y less than equal to 60.

So, cannot work more than 60 like that and then when this done event comes, you have to transit to rest at the same time while I cannot work more than 60, I must work at least for minimum 40 units of time so if I want to specify that. I can have a guard like this here so, you see that lot of timing specifications that is the point here lot of timing specifications of real time systems can be captured by using a timed automaton. As long as only time is the continuous variable timed automaton is a fine modelling formalism.

(Refer Slide Time: 25:57)



A Real World Example:Rail Gate Crossing

So, let us pick up some other examples here. For example let us say you have a rail gate crossing. So, let us say in this crossing whenever there is a gate and whenever the train is coming there is a sensor which is sensing whether this approaching or not. And similarly, you can have a sensor to sense whether you have the train have exited or not. So, now for the train gate and the controller if I am trying to model their behaviours using this kind of timed automaton-based mechanisms.

So, let us say I am creating a train automaton. So, let us say sorry it took some time you have these three automata modelling the train the gate and the controller. So, let us say initially you are sitting here, so the train is far away. So, the train is far away you are in this state and the gate is open. So, let us say you are here and the controller right now is I mean is in this state here. Now so that is like your state when the train is somewhere here and let us say now this sensor goes ON with approach that means the train is nearby.

So, once the sensor is saying your nearby all these states are now going to change and let us see how. So, when this approach event is true, so this due to this approach event happening so that is like an input event coming from the sensor the strain automaton let us say it is switching to near and along with that with this approach is also an input event. So, this is this question marks model input event and if some somebody is outputting an event its true an exclamation mark followed by that event.

That is a standard which is followed in automata theory in general. So, this controller also then change its state here. Now you can see that when I am the train has changed to this state it has set up a counter. So, x is being reset and you can only be in this near state up to this x less than equal to 5 and the controller can also be only in this state up to z this its own local clock less than equal to 3.

So, it is something like you have a timeline an approach event has happened and you are monitoring this said less than equal to 3 and also the train must you know that within x less than equal to 5 the train will definitely change from its near state and really enter the gate. It will really enter here and that is known to the designer of the system here. So, next what will happen is the train will come in and in between the controller needs to give suitable commands.

So, that the gate is being lowered. So before the train moves from this thing actually the train moves from this location to this enter state the controller inside a smaller amount of time so as you can see that the time given for the trend to move from here to here is 5. Inside after 5 it must be here so that is the timing model. So, if you see what we are saying is even before that the controller within 3 time units it must go to a state where it will signal the gate to lower down it will signal the gate to lower down.

And then it will be in that state and when this lower signal comes from the controller inside 1 time unit as you can see because of this clock constant model. Inside 1 time unit the gate must move from this up state to the down state. So, you see that is a sequence of events to be modelled. Inside

5 time units from being near the train should be here inside 5 time units. So, even before that the gate which was raised must go down.

So, the way it is modelled is inside these five two things will happen inside 3 time units the controller must come to a state where it is emanating this signal and this signal is an input event for the gate subsystem here so, the gate system must react to this signal and it should have its internal clock reset so that and that clock condition here will monitor that the gate must go down inside 1 time unit.

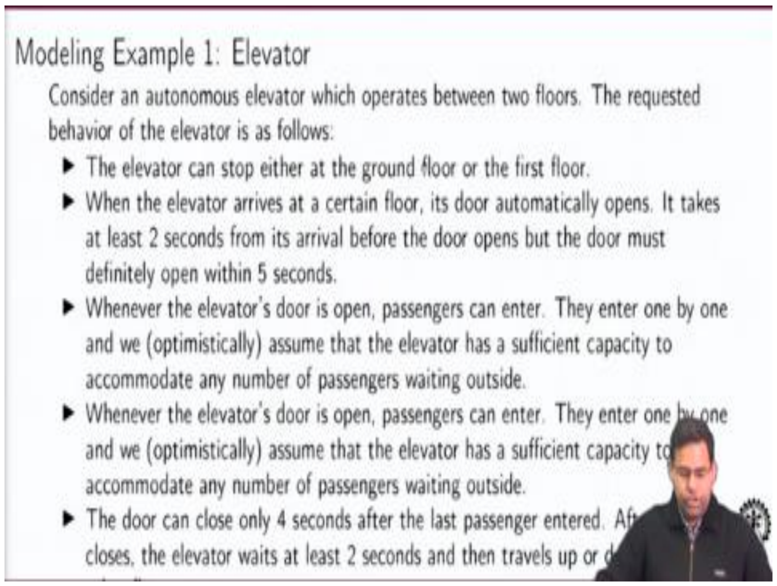
So, what this will in effect ensure is that before the train is in the gate is actually down. So, this is now using this automata model you can create the specification of a time of the timings of a real name system and just like we did reachability analysis that existing similar techniques there exists well known logic based or algorithms through which you can actually use several automated decision procedures to check well if you have a real time system for which you have a timed automaton model whether that system works correctly whether the system is safe whether that system satisfies some timing specification and all these things. So, that is one primary reason why people use this kind of models. Model usage also another important region model there are tools which support these models and they can do code generation for the source models using several kind of modern embedded system design tools more for model based design primarily.

So, let us see that how these things will move now what will happen is well, inside this time the controller will move from here to likewise saying it will move to this and once it has shifted to this the gate will move from up to this and then inside this timeline as I was saying it must go to the down state. So, here inside this 3 time you will have this lower signal coming and then inside some y equal to less than 1 from starting from here inside this interval you will have the down event really occurring. So, and then at some time the train will actually enter and before that this is the gate is actually down here. So, eventually what will happen is the train will come here and by that time the controller will ensure I mean this gets downward movement as has created this down event to happen before the entire event. And then if you see that well the train has entered and eventually the train is supposed to leave.

So, when this train is leaving what will happen is it will give this exit signal and once this exit signal is given it is like an input for the controller. So, with the exit signal the controller will move to this state and as you can see that here again it is reset a clock and inside 1 unit of time of getting the exit signal the controller will output this raise signal and this raise signal is basically an input for the gate's physical subsystem.

And with that after getting this raise signal inside write these two units of time you have a resetting and you have a guard the gate will go to this up state. So, once it has exited it gives a raise signal and based on the exit event and inside after sometime within these two units of time of getting the raise signal the gate will go to an up state.

(Refer Slide Time: 38:45)



Modeling Example 1: Elevator

Consider an autonomous elevator which operates between two floors. The requested behavior of the elevator is as follows:

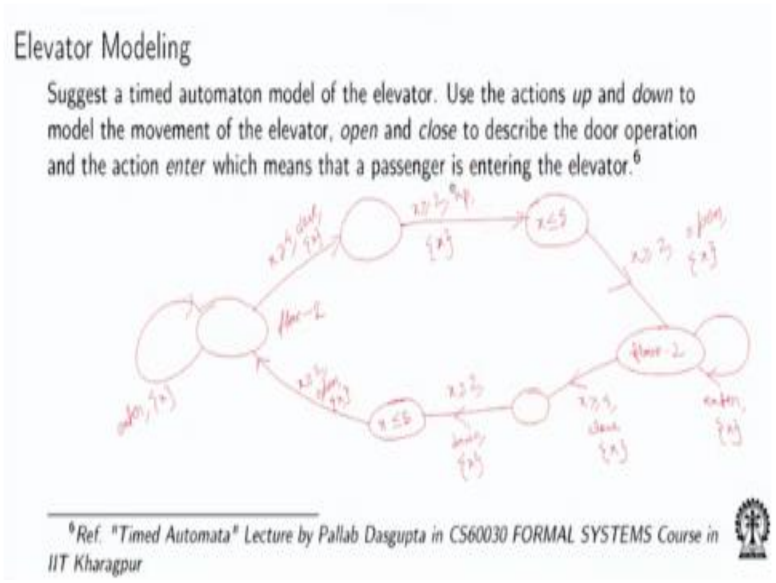
- The elevator can stop either at the ground floor or the first floor.
- When the elevator arrives at a certain floor, its door automatically opens. It takes at least 2 seconds from its arrival before the door opens but the door must definitely open within 5 seconds.
- Whenever the elevator's door is open, passengers can enter. They enter one by one and we (optimistically) assume that the elevator has a sufficient capacity to accommodate any number of passengers waiting outside.
- Whenever the elevator's door is open, passengers can enter. They enter one by one and we (optimistically) assume that the elevator has a sufficient capacity to accommodate any number of passengers waiting outside.
- The door can close only 4 seconds after the last passenger entered. Af closes, the elevator waits at least 2 seconds and then travels up or d

So, that is how it works here. So, let us check another simple example suppose you are given another real time system specification for let us say an elevator system. So, you have an autonomous elevator which operates between two floors and it is supposed to have the following behaviour. The elevator can stop either at the ground floor or at the first floor. Now when the elevator arrives at a certain floor its door will automatically open.

And in takes at least 2 seconds from its arrival before the door opens but the door must definitely open within five seconds. Now whenever the elevator's door is open passengers can enter and they enter one by one and we assume that elevator has a sufficient capacity to accommodate any number of passengers who are waiting for it I mean of course there is not an ideal scenario here.

And whenever the door is open passengers can enter and they can enter one by one just like that. Sorry we have a repetition here so you can just ignore this thing. And then the important timing behaviour you have here again that the door can close only 4 seconds after the last passenger is entered. So, once the last passenger is entered the door can close again and after the door closes the elevator will wait at least 2 seconds and then it will just travel up or down to the other floor.

(Refer Slide Time: 40:20)



Elevator Modeling

Suggest a timed automaton model of the elevator. Use the actions *up* and *down* to model the movement of the elevator, *open* and *close* to describe the door operation and the action *enter* which means that a passenger is entering the elevator.[6]

[6]*Ref. "Timed Automata" Lecture by Pallab Dasgupta in CS60030 FORMAL SYSTEMS Course in IIT Kharagpur*

So, let us say you just want to create a timed automaton and you are using these actions up and down to model the movement of the elevator and open and close to describe the door operations and the action enter which will mean that the passenger is entering the elevator. So, as you can see its very easy actually. What you can do is you can have a state so, let us say here passengers are entering, so there is an entered event.

And let us say this is representing floor 1 and then eventually there will be enough action you will move. So, if you check our constraints like earlier. So the door can close only four seconds after the last passenger has entered. So, after the passenger has entered the door will close so you can put a constraint like x greater than equal to 4 and this is the guard which must be satisfied with the close event.

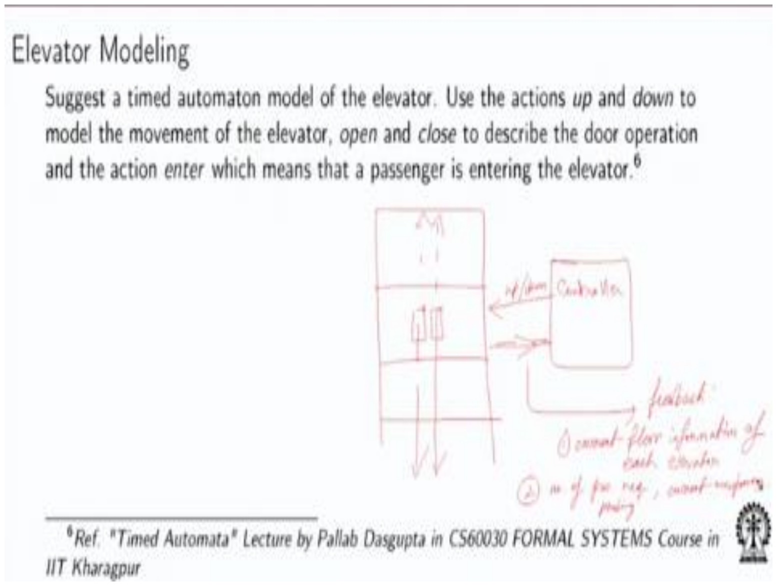
And then while this happens you can take a reset of this clock. So, if you look at this the elevator arrives to the floor and the door opens and exactly to, I mean and that would happen inside 2 seconds of its arrival. So, that also we have to model and it should happen between these 2 and 5 seconds should at least take 2 seconds to open but it should open within 5 seconds. So, these are the two kind of constraints we have.

So, I think with all these constraints we can just model them like this. Once again so we have this as a floor 1 state and this is the floor 2 state and these are the in between states we have drawn. So, passengers entering is considered here as an event. It can stop either of these floors. So, when an elevator comes this is where we say that well it has arrived. Now once it has arrived it is going to take this the I mean the door can close only 4 seconds after the last passenger has entered.

So, that is modelled here and after the door closes the elevator waits at least 2 second so it will just wait at least two second. So, only after x is reset here and there in this transition you have x greater than equal to 2 and after that the up action should happen and it will travel upwards. Now it will take at least 2 seconds from the arrival before the door opens but the door must definitely open within 5 seconds.

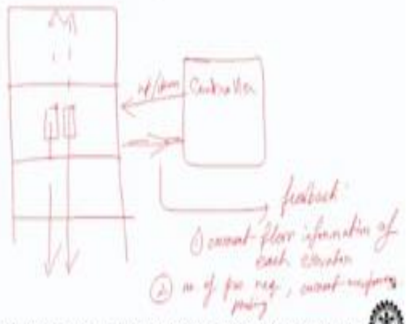
So, that is modelled here and after that the door will open in the second floor. Now in the second floor like you see that the door can close 4 seconds again here after the last passenger entered. So, only after being in this floor and all the inter events with each enter event I am just looping here and whenever one passenger is entering this clock is being reset. After the last passenger is entered there is no one coming in so there is no resetting, so clock will go up and once it expires 4 then the door will close and I come here and then from floor 2 I will be starting to go down and once I have gone down, I will again reset and then we will reset and I will move to opening the door inside an interval from 2 to a max of 5. So, that is why 5 is the invariant and 2 is the guard condition. And again, the entering logic will be just like for floor 2 in floor 1. And again all the passengers get in then x greater than equal to floor then I just start going up. And that is how it works so these are very simple model. But just to insight your mind, there are several interesting problems around elevators that can actually be done.

(Refer Slide Time: 46:48)



Elevator Modeling

Suggest a timed automaton model of the elevator. Use the actions *up* and *down* to model the movement of the elevator, *open* and *close* to describe the door operation and the action *enter* which means that a passenger is entering the elevator.[6]



up/down
Controller
feedback
① current floor information of each elevator
② no. of floor req., current ... pending

[6] Ref. "Timed Automata" Lecture by Pallab Dasgupta in CS60030 FORMAL SYSTEMS Course in IIT Kharagpur

For example, you can have multiple let us consider a situation you have a say large multi storeyed building. Large multi storeyed building of many floors and you have two elevators running. So, they can both go up and they can both go down. Let us say you are trying to build a control algorithm which is going to control the up or down signals which is going to give this up and down signals for the elevators.

And as feedback it gets to know what is the current floor information of the, of the elevator, feedback is current floor information of each elevator. And then number of passenger requests for each elevator and current occupancy, passenger requests that are pending, for each elevator current occupancy etcetera. So, I can design a central controller which will work in real time based on these feedbacks and it will decide which elevator goes up and which elevator goes down from which floor.

So, designing such controllers is a very important real time programming problem and just to pick your minds you can think about it you can design a small simulation C program or let us say an Arduino based controller of such a toy system stuff like that. So, fine with this we will end this lecture, thank you for your attention.