

Foundation of Cyber Physical Systems
Prof. Soumyajit Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 30
Hybrid Automata Based Modelling of CPS (Continued)



Hello and welcome back to the course on Foundations of Cyber Physical Systems. So, in this lecture we will be introducing you to this formal semantics and definitions of hybrid automaton. (Refer Slide Time: 00:42)

HA formal definition

A hybrid automaton H is a collection $H = \langle Q, X, f, Init, Inv, E, G, R \rangle$ where

- ▶ $Q = \{q_1, q_2, \dots\}$ is a finite set of discrete locations
- ▶ $X \subseteq \mathbb{R}^n$ represents the state/value space of the continuous state variables
- ▶ $f : Q \times X \rightarrow \mathbb{R}^n$ assigns to each discrete state (location) $q \in Q$ a vector field $f(q, \cdot)$
- ▶ $Init \subseteq Q \times X$ is the set of initial states;
- ▶ $Inv : Q \rightarrow 2^X$ assigns to each discrete state $q \in Q$ a set $Inv(q) \subseteq X$ called the invariant set;
- ▶ $E \subseteq Q \times Q$ is the set of discrete transitions;
- ▶ $G : E \rightarrow 2^X$ assigns to each discrete transition $(q, q') \in E$ a guard set $G(q, q') \subseteq X$
- ▶ $R : E \rightarrow 2^X$ is a reset map.

Uppanil \rightarrow T.A.



So, we will first start with the formal definition of what a hybrid automaton is, and then we will I mean we will just introduce the tuple elements here just like we did for finite automaton. That is the first affair. So, a hybrid automaton is like this, it is a tuple with quite a lot of elements here.

$$H = \langle Q, X, f, Init, Inv, E, G, R \rangle$$

So, as you can see that, first we have Q , which is a finite set of discrete locations. So, here we have a terminology change.

Instead of calling states, we are calling what we kind of denoted as states using the circular box as I mean in the circles in the finite automaton, we are calling them locations. Why? The reason

is simple, now we are trying to talk about systems where there is some continuous dynamics as well as discrete switches. So, the state of a system will comprise two pieces of information, one is in which discrete position I mean, let us call that as location, and what is my current value set of the different continuous variables in my system. Now these two pieces of information together gives me the state of the system. And since we have variables who can take continuous values, now that immediately tells me that we are talking about infinite state system. That means that the state space is infinite because we have these variables for which the value set is the real set.

So, let us say we have a set of continuous variables and they are value set given by X is basically the value space or state space of the continuous state variables. So, that this being a subset of \mathbb{R} to the power n means that we are talking about a set of n continuous variables, and their corresponding value set. The next thing is this flow function f . So, that means given a state, that means, sorry given a discrete location for the set of continuous variables, we have a vector field given which is denoted like this. We will soon see with the tuple example. So, that essentially is a mapping which tells me what is the gradient, or what is the rate of change of the corresponding continuous variables. For each of the continuous variables we will have a differential equation which will give me its rate of change. Now we are showing it as a mapping to \mathbb{R} to be power n because well for that set of continuous variables we can take their values at any point. And for those points, if I compute the gradient that will also be an n tuple in the real set which will give me a direction for the each of these continuous variables on the trajectory of the system. I mean typically this is represented by phase portrait diagrams which we will see later on. But for the time being, we can just think like this, that f is a flow function which is given as a function of every location Q , and in that location f will tell me what is the derivative function of each of the continuous variables. We will soon see with some examples. What we have next is a set of initial states. It is nothing but the set of discrete location cross product with the value set of the reals and any a possible subset. So, this essentially tells me that there are certain discrete locations where the hybrid automata can start.

And it can also start with certain possible values being assigned to the set of the continuous variables. Now wherever we are talking about this subset relations here, as well as here, as you can understand that how will this be represented. Now all this same they say is that well there

is R (04:45) to the power n . Its n dimensional space and I am talking about one partition, one subset of that space.

So, it maybe I mean typically it will be represented by some equations which will capture that subspace. Or for example, let us say I am talking about R square as this place and there are two variables X and Y . So, $X < Y$ is a relation which captures a subspace of R square. So, in the same spirit we will have a set of inequalities or a set of some complex relations which will typically tell me which subspace I am talking about here, as well as here like that.

We will see in future with some examples. And similarly, we have invariance, similar just like I said that for the previous cases here invariants are also captured as a subset of this possible value space of continuous variables. So, they are also being given by some relation which tells me that relation is true as long as the system is in some discrete state q . When I go to another discrete state q_1 another relation in q_1 will become true.

And that is also given by another set of relations which actually provide me a subspace of X . Now we have a set of discrete transitions which dictate that based on which input event I can go from q to another's position q . Now so apart from these discrete transitions, we have two important new things here over and above finite automaton. One is the guard set, that means with every transition we can have a relation which tells me that well the that this transition is enabled or not.

So, it is typically given by this kind of $G_{q, q'}$, subset of X . So, that would mean that this is also another relation which will encompass a substate a subspace of X . And similarly, what we will have is another new thing here which is a reset map, that for every such transition, the set of continuous variables may be re-initialized to some values and that is what we call as a reset map. So, we will soon see what that means.

Now one thing you can see here that this definition of hybrid automata which we have, there is we do not have any notion of an input event or an output event stuff like that. Now in various

books you may see different variations in the definition of the automaton. There would be the books where they talk about the alphabet set that means which it has some input events. There may be books where they also talk about that hybrid automata producing some hybrid output events, or hybrid input output automata.

All these different classes or subclasses classifications of hybrid automata exist. We are just going forward with one possible definition that is there. So, this is one possible definition in a restricted space. There can be many variations based on the different classifications, different classes of hybrid automata people talk about. And there has been a lot of research in this area of hybrid automata modelling of hybrid automata.

I mean modelling control systems using hybrid automata, modelling in general reactive systems using hybrid automata. There has been a lot of theoretical CS theory research, language theoretic research in the last 20 years. And there are tons of papers, on different classifications of hybrid automata, different ways to represent hybrid automata, attaching hybrid automata with like I said input or output events.

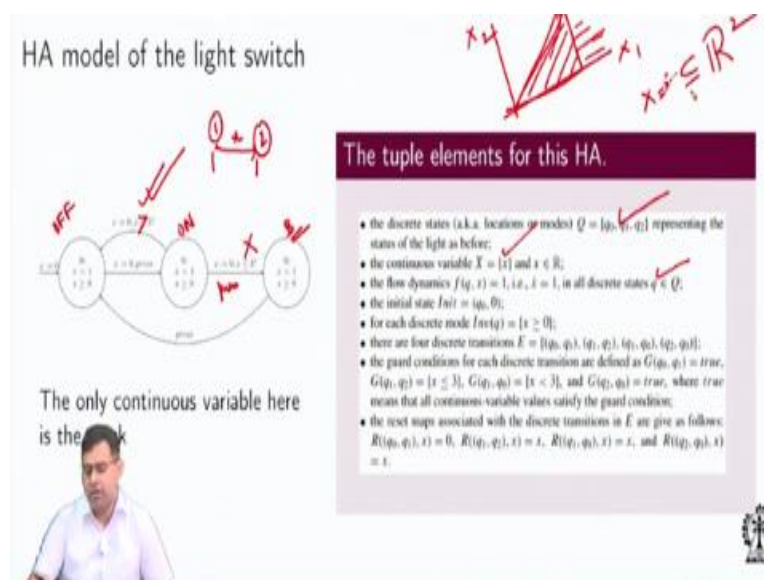
There is a subclass of hybrid automata called timed automata. There are timed IO automata, all these things. There is a nice tool called UPPAL, which you can actually download and have fun with. It captures a specific class subclass of hybrid automata called timed automata. You can do lot of modelling, there are lot of modelling examples. And you can actually see hybrid automata, I mean timed automata plots, different kind of property verification methods etcetera.

We will cover some of them not all of them here in this course as we go forward. So, just to summarize what we just said is, this is a tuple definition of hybrid automata. We have the set of locations, the set of continuous state space, and then you have this vector field. That means at every location, the continuous variables in the system will have a vector field assigned to it. And that vector and if I assign values, that means if I assign a value to that vector field by assuming valuations from the set X , then I also get, the direction of the directional gradients for each of the containers variables. We have location invariance basically relations which tell

me that it must be true if the system has to reside in that location. That means if I am in the location q in Q , I mean my valuation of the continuous variables must be inside this set-in Q . That means it should satisfy a corresponding equation or relation. This was my definition of discrete transition.

And similarly, we had this guard condition that means with for every transition to be enabled some guard conditions should be satisfied. That means the continuous variables must be assuming values inside this subset of X , which will again be given by an equational or in equation or relation which needs to be essentially true for the guard transition to be enabled. And the when a transition is taken, the some of the variables from the continuous variables can be re-initialized. So, that is what we call as a reset map. So, that is my overall summary.

(Refer Slide Time: 10:55)



Now let us look into this example of that light switch model. And let us see how hybrid automata can be used to actually create a proper model of that light switch example. So, what we are doing is well first let us label these different states. So, this was our OFF state, this is ON state and this is the BRIGHTER state. So, what we are doing with the initial version is we are attaching this one continuous variable clock.

And with each transition we are attaching reset maps, we are attaching this guard conditions. So, this is a guard condition. So, what this guard condition is testing is whether the value of x is less equals 3. And what this so, this guard condition should be that whether it is greater than

3. So, if this guard condition is satisfied that, the value the time between the two press events is greater than 3 then I go, I mean, less than three, then the light becomes brighter.

Also, one thing is missing here we will have that press event. So, like I was saying that if you see our previous picture example there was no concept of input event. But when I am drawing this automaton, I am introducing an input event. That means in certain tuple definitions it maybe it may require that you have that set of alpha, the alphabet set or the set of input events attached inside that the tuple definition as we have it here.

So, let us see how this works. So, we have some guard conditions, and what is important is we have some invariance, location invariance like we discussed earlier although they do not play much of a role in this example. But what is important is, since this is the clock variable, I mean you are taking the derivative of time with respect to time that is your derivative of X . If you remember our definition in the previous slide, we said that we will attach a vector field with every continuous variable.

That means that will tell me what is the gradient of that continuous variable. So, for this continuous variable clock that we have attached to the original automata here, we have to take its derivative. That means what is its rate of change with time, this being itself time the rate of change is 1. So, $\dot{x} = 1$. So, my flow function here is f of q_0 , $\dot{x} = 1$. And that is so, in all the other cases in place of q_1 in place of q_2 , because this is the time variable.

So, everywhere it is 1. So, that is my flow function of the vector field., is for only one continuous variable. And it is simple, $\dot{x} = 1$ everywhere. Now what is happening additionally is that, if you remember we said that there will be guards and there will also be resets. That means, when I take the transition I can re-initialize this continuous variable x , and I am always reinitializing it to the value 0.

Why? The reason is I want to, what is my objective here? My objective is that, I want to find out that timing difference between two consecutive press events. So, when the first press event happens, I am resetting this to 0. This is my first press event I have a reset. So, this is my time

axis, time is increasing. Suddenly I just reset it. So, I just stop time here, its reinitialized, again it will start increasing.

But here I have reset it at that time, when the first press event happened. So, when the next press event happens, it we will just check what is the value of x this time variable x . I mean starting from the time when the first press event happened. This is the second press event. This is the first press event, this is the second press event. Now if that difference is less equals 3, then we have this guard satisfied and I will again just make this change, here it is x greater than 3.

So, if the difference between the two consecutive press events is less than equal to 3, then from OFF we will directly come to I mean we came to ON, and since this difference is less than equal to 3, we go to this BRIGHT state. And if the difference is greater than 3, so, that means the user's intention is to turn it OFF. So, this condition is not satisfied, but this guard condition is satisfied. So, we will just reset the clock again and we will go back here.

Why reset it again? Because again from here the it will start its own counting. I mean this is not necessary here but anyway. So, that is how, the point we are trying to make a here is, using guard conditions and using this reset maps, we are able to operate on the continuous variable clock whose gradient is given by $\dot{x} = 1$ of course because times change with respect to time. But our objective is satisfied.

We are able to model the light switch behaviour, with respect to what I mean in terms of capturing its behaviour with respect to time which we are unable to do using the original finite automaton model. Now if we want overall description of these hybrid automata's tuples. So, let us just see what it should be. So, you have three discrete states. So, this Q set is this. And you have only one continuous variable, x .

You have this flow dynamics which is same for all the discrete states like I said. So, it is q is given to q_0, q_1, q_2 all this. Everywhere this is only one. So, that is your flow dynamics. Your initial state is you are always starting from q_0 here, and when you are starting you are starting

with the continuous variables value being set to 0. So, let us just try to match this with the previous things.

Here we always said that if you look at this Init is a subset of $Q \times X$. That means, I am saying a subset of the initial states and the subset of the valuations of the continuous variable. Note that x is not the state of continuous variables, but rather it is the set of valuations of the continuous variable. That is very important to remember here. Now we needed the invariant condition if you see.

So, invariant condition meant again that for every q , it is possible that you have a set of possible valuations. $\text{Inv } q$ is what? $\text{Inv } q$ is a subset of X . That is why the mapping is to the power set of X . That means from this side you pick up a small q , and from this side you pick up a subset of X . A subset of x means, thus entire valuation space of continuous variables, and you are picking up one specific part of it, which may be given by some equational relation on the continuous variables that you have in the system.

Here you have only one continuous variable X . And you have this equation relation $x \geq 0$, which represents all possible values of a real time clock. Because it will be, it cannot be negative. It will always be positive or it will be reset to 0. So, the only practical relation for a real time clock that you can express is $x \geq 0$. So, there is why you have this same invariant everywhere in all the locations, so, you have $x \geq 0$.

So, I am just trying to communicate here that why this is mapping nicely with this definition. If you remember, let us say if I just take an example; suppose I have two continuous variables in my system. Suppose I have two continuous variables X_1 and X_2 . So, let us say I am interested in these value space. So, this is I am interested in only this value space, I mean this is continuing in this side.

So, if you can see that this is given by what? This is given by $X_1 > X_2$, this side is given by $X_2 > X_1$. So, that is why I keep on saying that it is an equational relation, or in equational relation among the variables, through which you can actually capture in a

symbolic manner a subspace of the state of all possible values these variables may get. In this case, it is a subset of \mathbb{R}^2 , X is a subset of $X = X_1 \times X_2$ I mean this is subset of \mathbb{R}^2 here.

So, and you can represent it algebraically using this kind of equational relations that is what we are doing here. So, this is the single variable. This is a relation which is kind of representing it. So, this works $\text{Inv } q \subseteq X$, why the notation is like this. I hope we are able to explain it here. Similarly, I mean this reasoning will hold for all the other definitions. If you have guards, guards are also you can see there.

How are we expressing guards? As equational relations, right. That is why they are all mapping to from edge to 2 to the power X . Because every guard, we will have a guard set given by this $G q, q \text{ prime} \subseteq X$. That means we are expecting this $G q, q \text{ prime}$. We are expecting in the automaton that between these two, I mean states q_0 and q_1 that is q or $q \text{ prime}$. The guard is going to be something which is a subset of X .

The set of all possible evaluations of the continuous variables. And it is represented by X greater than three here at X less than equal to 3 here. So, they are represented by the equation relations here. Similarly, here the reset maps. You can see now when you are resetting your kind of reinitializing them. So, here the re-initialization just means resetting the clock, making the clock, forcing the clock to start from 0.

So, if you look at the guard condition definitions before that the discrete transition. So, that is very simple you have the transition from q_0 to q_1 , q_1 to q_2 and q_2 to q_0 and q_1 to q_0 . So, you have these four guard conditions so, four transitions. And your guard conditions are well not all of the transitions I mean have guard conditions here. So, whenever they are not explicitly mentioned, that means the guard is always true. So, there is no condition here.

So, q_0 to q_1 is always true, q_1 to q_2 you have a condition X less than equals 3. Similarly, here q_1 to q_0 you would have this x greater than 3. And if you see the next one here again from q

2 to q_0 you do not have any guard condition. So, it is a true. True means, that all continuous variable values satisfy the guard condition like we said. And then the reset maps, well in this case here only you have a reset happening.

And then here also I mean when you are going from q_1 to q_2 , for all practical purposes you only need the reset right here. So, this is actually where you need the reset and at the other places. We I think we have an error here. This should also be they just remain the same. Now in some books you may find that the reset is only specified where the result actually happens. And in some books, you may find that it is specified everywhere.

So, if there is no resetting state level writers x being assigned x . So, it is fine if you just mention this and you do not mention any of the others. Because they simply mean that the variable has not been disturbed. And in this case, as you can see that here we only need the reset to happen here. Because I am only interested in measuring the time difference between the two successive press events. So, I only need the reset to happen after the first press event.

Nothing nowhere else is the reset required. And well when I go back here, when the automaton goes back here, well the clock may increase. There is no problem because again I will reset it just again when the first press event happens here again. So, that is all. Here it is fine to have only the reset here, or if you want you can also do a reset here it does not change the language of the automaton it just changes its description.

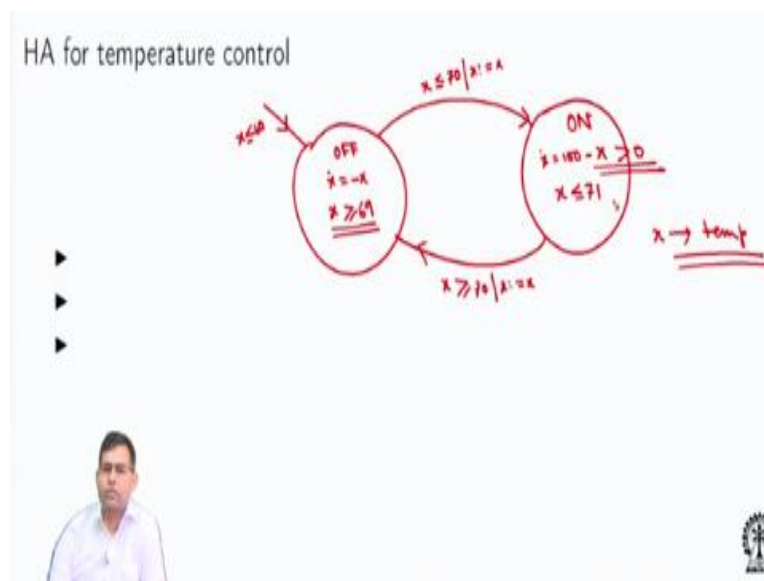
It only means that after the end of a full run, I mean you can I can reset here, as well as this set here and I again start counting the time here. But just remember that in this case my key functionality is that, I just, why did I really include the time variable? I want to know the time difference between the two press events. So, as long as I am resetting the clock after the first press event, my objective is nicely satisfied, I do not need any more of them.

But if I am interested in bounding the clock, I mean you do not know what is your application you may want to bound the clock value. So, then you may want to reset it here, or here, at this

place or at this place. That means when you are going back to the initial state, you may want to reset it, that will depend on your application. If you if there is you have some interest in bounding the clock value.

Because otherwise I mean if you have this second press happening after a long time, the clock value is increasing. So, anyway, so we can just say in another way that this can be 0 or it cannot be, I mean that does not matter. But this is the key event in this specific case.

(Refer Slide Time: 26:33)



So, let us look at another example of temperature control system. We will look at some more examples of hybrid automata now. We will start with a simple temperature control system. So, let us first understand what we really want. We want the automaton to represent a controller which is trying to regulate the temperature in a room. So, it is assumed that, I mean what it does is it basically turns ON or turns OFF a furnace.

And it is trying to maintain the temperature of a room around something like a 70 degree. If the temperature shoots below that, it will go to some I mean it will turn the furnace ON and if the temperature shoots over that I mean it will just turn the furnace OFF. That it is a simple kind of example. So, let us just look at the automaton then. So, we have this system. So, initially your furnace is off the control I mean, so, the temperature is kind of decreasing.

Now you are allowed to be in the state. So, this is your state invariant as you can see. You are allowed to be in the state as long as your temperature is greater than or equal to 69. That means, I mean there are two ways you are forcing the transitions here. And if you if your invariant is not satisfied that means, the automaton will be pushed out of this state by the invariant. It will be asked to try and satisfy one of the guards.

Because if your invariant is not satisfied and any guard condition is also not enabled that means your automata simulation or the execution of the automata is stuck. So, that is I mean that means it is not a correct model or there are behaviours in the model which are not really, I mean acceptable because the automata cannot go forward. So, you are allowed to be in the offset, in this OFF state.

As long as this is true that your temperature is greater than or equal to 69. If it falls below 69, then you can see that you have a matching guard condition that the sooner even if the temperature goes below or equal to 70 you are turning this thing to on, you are turning the furnace to on. Now whenever you are turning it on you can see that the temperature is below something below or equal to 70 which means this is a positive quantity.

So, you will have a positive gradient in the temperature now. So, x represents temperature of the room let us say. So, that means now the temperature will increase, but at the same time I do not want the temperature to increase too much. So, I put an invariant condition here $x \leq 71$, that means I can be in this state as long as the temperature does not shoot beyond 71. And in fact, at 70 itself I have enabled the guard here.

That means, the system can move to OFF the moment. So, eventually what comes out of this design is my target is to keep the temperature around 70. It goes below, I turn it ON, it goes above, I turn it OFF, and that is how this hybrid automata kind of operates. So, with this we will end this lecture. Thank you.