Artificial Intelligence:

The Event Calculus: Reasoning about Change

Prof. Deepak Khemani

Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Module – 04

Lecture - 06

okay so we are looking at knowledge representation and I want to introduce a new dimension to representation now which is what do you do if you want to reason about change. So far we have not talked about the changing world essentially. We have okay this is how we describe the world, this is how we describe relations between elements of the world and so on. But we have not really given some thought to when the world is changing how do we represent that and how do we talk about that. So for example I might say that John is happy or Peter is happy lets say. And that’s a statement I can make, Happy Peter in adhoc way of representing

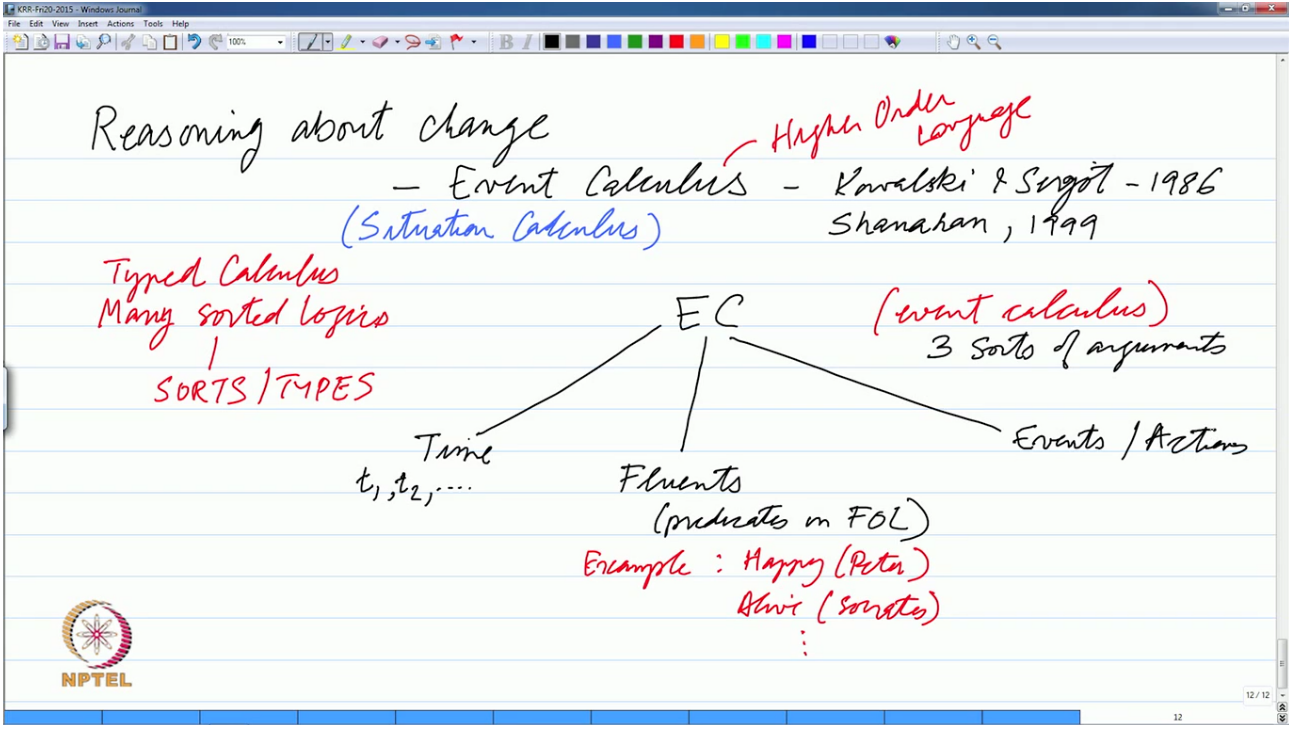
And then I say Mary hit Peter and then I want to say that after that Peter is no longer happy. So how do I represent change here. How do I capture the fact that certain things are true at certain point of time and not true at different point of time. So far we have not talked about time essentially. And we want to look at a mechanism for doing that. And the mechanism that we are looking at is called event calculus. We will focus on the representation part today. The reasoning especially in an open world is a little bit more complex because you dont know what else is happening. We will come to reasoning when we have looked at some ways of handling uncertainties.

So this event calculus was proposed by Kovalski and Sergol in 1986. Robert Kovalski was a , he has done a lot of work in logic and reasoning and he is one of the founder of the idea of logic programming, which we will look at may be in a few lectures. But there is an interesting tutorial by somebody called Shanahan. And we will put up this material along with the tutorial so you can read about it. There are other variations that people have talked about, for example McCarthy has talked about something called Situation Calculus. But for all purpose its that we look at one flavour and imagine what the other is like essentially. Now in some sense event calculus is a higher order logic.

And when we say higher order we mean higher than first order logic. So remember our notion of first order logic is that predicates take terms as arguments and there are basically defined relations between elements in a domain. Now in this event calculus we will take those predicates themselves as arguments. So we may want to say that Happy John is true or Happy John is false and things like that. So in that sense its a higher order calculus. Now when we talked about logics, people often talked about typed systems or types calculus or some people use the term Many Sorted logics. So what we are essentially talking about is things like Sorts, so Sorts is a noun here. Its just another word for Types essentially. And people have explored languages in which you separate things of different types. So for example you might say Type Human is one thing, Type Number, we have already introduced the notion of a number, so that’s a different type of an object altogether. So people like to sometimes breakup languages into many sorted languages or many types of languages, many sorts of objects essentially.

In event calculus which we will use the term EC, it uses three types or sorts of things, one is of sort Time, we will use notations t1, t2,..to represent Time. Or we can use uppercase it doesn’t matter. Then the second sort is of type Fluents , so by fluents we basically mean predicates in FOL. So and things like this. We use fluents for predicate which are amenable to change, which can change. So it may be Peter is happy now but he may not be happy at a different time. So such predicates which can change we will call them fluents essentially. So in general we will call all predicates as fluents. And then the third type is Events or you might say Actions.

(Refer Slide Time: 7:22)


KRR-Fri20-2015 - Windows Journal
File Edit View Insert Actions Tools Help
NPTEL
12 / 12
12

Reasoning about change

— Event Calculus — Kowalski & Sergot - 1986

Higher Order Language

(Situation Calculus)

Shanahan, 1999

Typed Calculus

Many sorted logics

SORTS / TYPES

EC

(event calculus)

3 sorts of arguments

Time

$t_1, t_2, \ldots$

Fluents

(predicates in FOL)

Example : Happy (Peter)

Alive (socrates)

⋮

Events / Actions

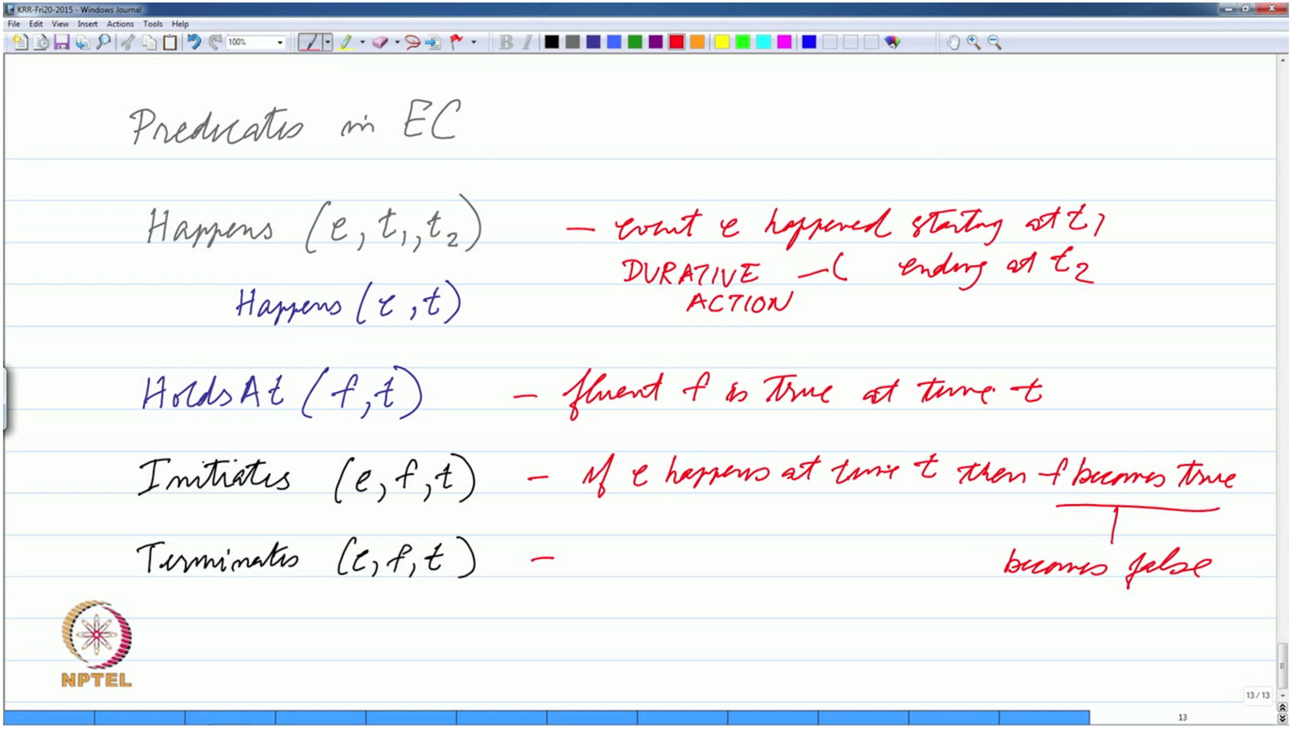
so for example Hit Move Eat anything which is an action or an event is the third type essentially. Now you can see that both these fluent types and event types are defined on the domain. The predicates are defined on the domain and actions are also defined on the domain. If we have an action lets say to continue using the Hit Mary Peter is defined on the domain in which Peter and Mary are objects and there is some relation we have defined as Hit, only thing is that now we are separating relations for example Brother Peter Mary if I had said, its a relation on the domain essentially. And its relation in the true sense of word whereas Hit Mary Peter, we have modelled so far as a relation in the domain. Remember that the semantics of binary predicates is that they are subsets of D cross D. So its just some mathematical relation but eventually we would like to think about that as separate from a descriptive relation like Brother to something which describes action or event essentially. So the event calculus separates events or actions from relations of the other kind which we will call as Fluents and this we call as Events.

So lets talk about the kind of Predicates or vocabularies in event calculus. What are the predicate names. So we will have a well defined set of predicates which we will use. So the first one is called Happens. Fluents we will denote by f1, f2 and so on. And events or actions we will denote by symbols like e1, e2 and so on. So whenever we say e1, e2 we are talking about the type event, whenever we say f1, f2 we are talking about type fluents and whenever we say t1, t2 we are talking about type time. So the first predicate that we are interested in is say that. So this says event e happened starting at t1 and ending at t2. You can see that this is a Durative action which means it has a duration associated with it. Otherwise we could have talked about something like Happens e at time t. If actions are instantaneous then we can say that, if we are modelling actions which are instantaneous then we could have simply said event e has happened at time t. If you want to model the duration of actions then we would event Happens e1 starting at t1 and ending at t2. But this is one predicate that we will use which is happens.

Then we have HoldsAt some fluent at time t. So this stands for the fact that fluent f is true at time t. So observe that already fluent has become an argument to this predicate HoldsAt which is why we can think of this as a higher order language essentially. So these are the two basic predicate, the first predicate says that some event has happened at certain time or may be over a certain time duration. The second one allows us to say that some thing is true at a given time. Then ofcourse we want to find connections between things, so the predicate that we are interested in is called Initiates. And the meaning of this predicate is that if e happens at time t, and we could talk about durative actions here but lets just keep it simple. Then f becomes true.

Strictly speaking f becomes true not at time t but after time t so depending on whether you are talking about discrete event calculus or continuous event calculus you have this notion of causality that event is causing that fluent to become true. So strictly speaking it becomes true after that. If you are talking about discrete event calculus, then essentially what you mean is that if event e happens at time t then fluent f becomes true at time t plus 1. but we will not go into that detail at this moment essentially. Then we have Terminates c, f, t which is similar to Initiates but instead it becomes false.

(Refer Slide Time: 13:58)


KRR-Fri20-2015 - Windows Journal
File Edit View Insert Actions Tools Help
100%
NPTEL
13 / 13
13

Predicates in EC

Happens (e, t_1, t_2) — event e happened starting at t_1 ending at t_2

DURATIVE ACTION —(

Happens (e, t)

HoldsAt (f,t) — fluent f is True at time t

Initiates (e,f,t) — if e happens at time t then f becomes true

Terminates (e,f,t) — becomes false

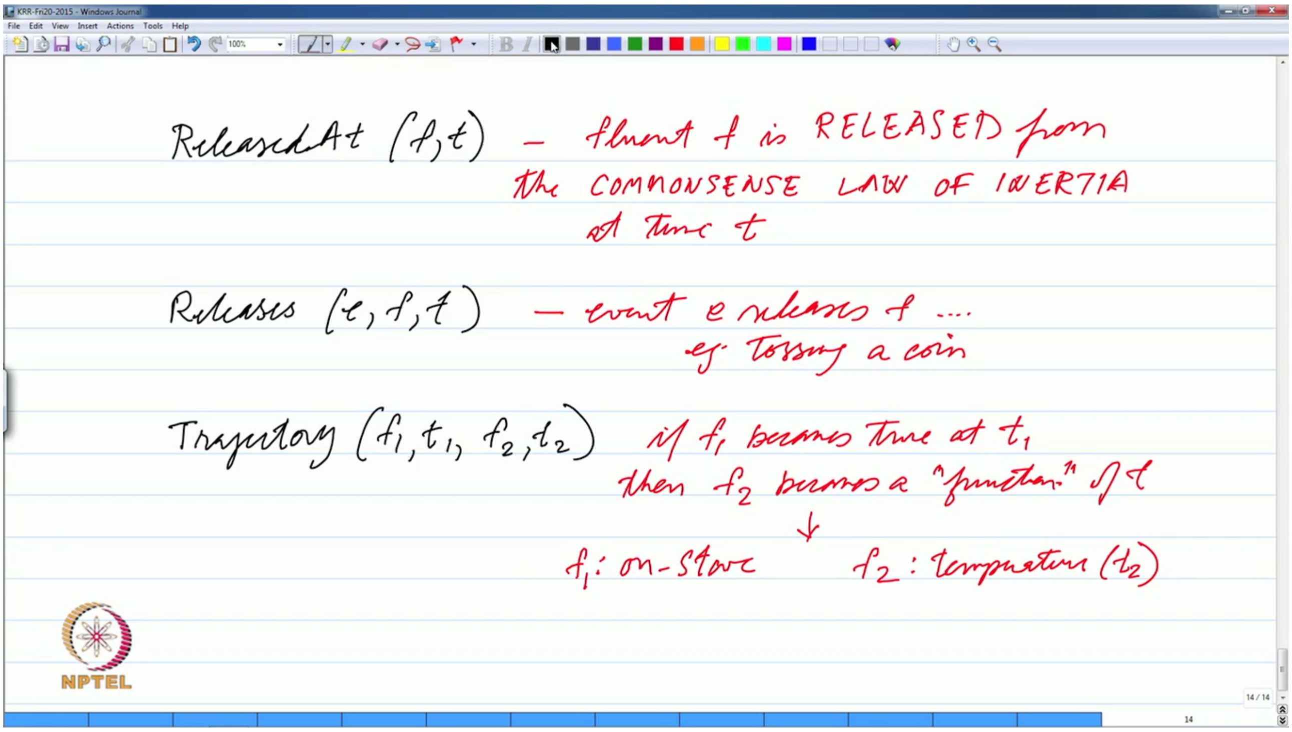
So Initiates and Terminates are two predicates that we will choose to say that this event is causing this to become true. So our knowledge base for example might say that if somebody hits somebody else then that somebody else stops being happy essentially. So continuing with our study of predicates we have a predicate called ReleasedAt . So we are saying so we are talking about handling uncertainty, trying to see how can we model uncertainty here essentially. In the sense that sometimes we don’t know what will be the value of a fluent essentially. So for example if we have a coin and we toss it up essentially then you don’t know whether its going to be heads or whether its going to be tails or something like that and things like that essentially. So ReleasedAt is one attempt to being able to model uncertainty and this predicate says that fluent f is RELEASED from what we call as the COMMONSENSE LAW OF INERTIA.

So we have this notion of commonsense law of inertia which basically says that if something was true it will continue to remain true. If something was false it will continue to remain false except when we model certain kinds of uncertainties. When we say that this fluent has been released from common sense law of inertia which means that if has been released from common sense law of inertia at time t then after that you cannot state whether it is true or it is false. It is a little bit like Scrodinger's kitten essentially, if you know about that, you don’t know whether it is going to be true or false. You cannot determine whats the state. Even you may have started with certain state but once you are released from the commonsense law of inertia you don’t know what the value is, whether it is true or false. And this is used to model some kind of uncertainty. We will come back to this a little bit later.

Then we have RELEASES. How does it get Released? An event can release essentially. So we can say that event e released f. So for example tossing a coin, we may try to model the process of tossing a coin by saying that it is released, we don’t know whether its heads or tails. But we will also see that this will not really serve a purpose because eventually the outcome of tossing a coin should be heads or tails essentially and releasing it doesn’t really serve a purpose but that’s a first attempt at trying to do that essentially. But there are other examples, for example if you are addicted to playing Russian roulette which is not a good thing to get addicted to ofcourse which is a game in which you have a revolver with lets say one or two bullets and you spin it randomly and you fire it at yourself. Its something that you don’t want to imagine but anyway that’s a game. So if you are spinning the barrel then you are releasing it from the law of inertia. You dont know at the end of it whether its going to be loaded or not loaded essentially.

Then there is another thing called TRAJECTORY. So i just very briefly said that if f1 becomes true at time t1 then f2 becomes a function of t2 which you can describe because its after all a predicate, f2 is also a predicate or fluent essentially. So as an example if f1 is lets say on the stove, if you kept a pan of water on the stove or something like that then f2 could be temperature as a function of t2. The moment you put a pan of water on the stove its temperature will depend on time, so that’s described by the predicate called trajectory.

(Refer Slide Time: 20:09)


KRR-Fn20-2015 - Windows Journal
File Edit View Insert Actions Tools Help
100%
NPTEL
14 / 14
14

ReleasedAt (f, t) – fluent f is RELEASED from the COMMONSENSE LAW OF INERTIA at time t

Releases (e, f, t) – event e releases f ....
eg. Tossing a coin

Trajectory (f_1, t_1, f_2, t_2) if f_1 becomes true at t_1 then f_2 becomes a "function" of t
↓
f_1: on-Stove f_2: temperature (t_2)

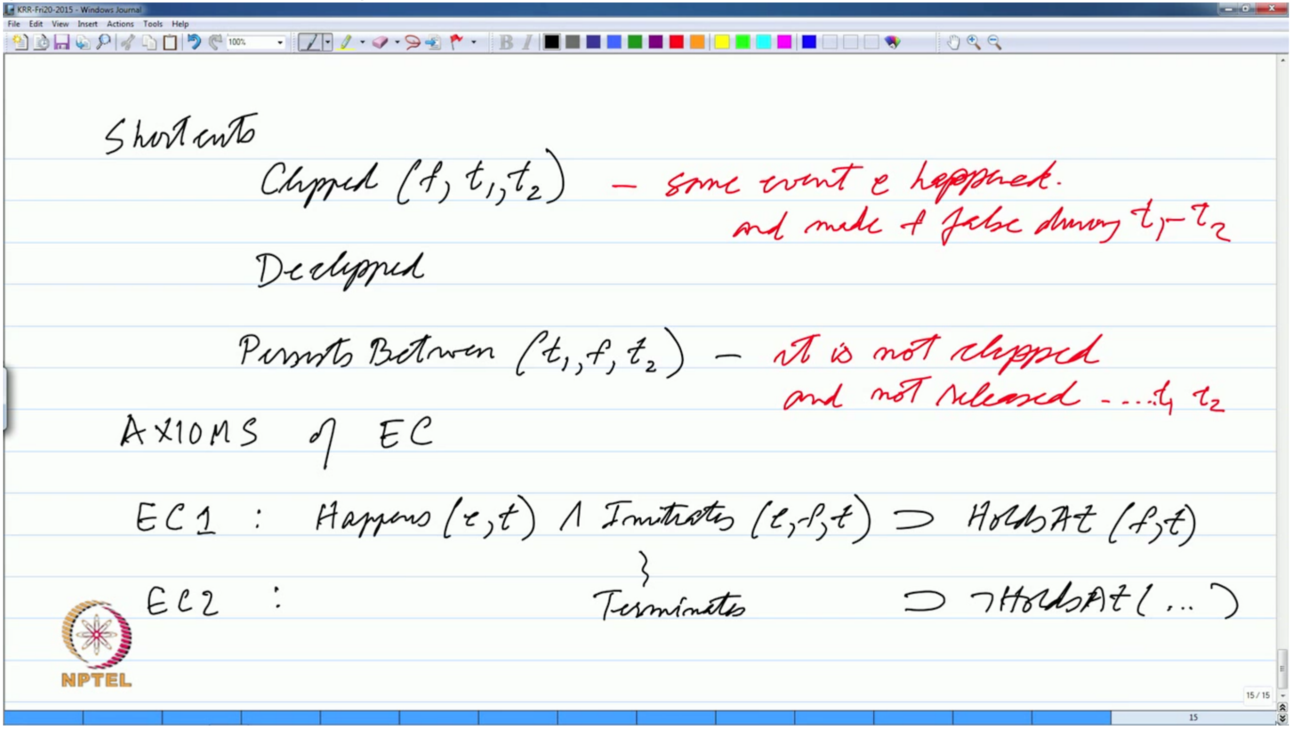
then we have some shortcuts so for example you may say Clipped fluent between t1 and t2. Its a shortcut for saying some event e happened and made f false during t1 to t2. So i am not writing the logical expression. You can actually write a logical expression for this that there exists a event e and there exists a time t such that event e happened at time t and terminates this fluent f. So i am not writing that i am just writing the English version which says that some event happened which made the fluent false at time t.

And there is another one called Declipped which is the opposite of Clipped in the sense that it made it true essentially. And there is one more called Persists Between, so we can say that between time t1 and fluent persists and time time t2. Which means it was not clipped and not released. Between t1 and t2 essentially. So when i say that a fluent f persisted from time t1 to t2 we are essentially saying that nothing has made it false which means no event has made it which terminated that fluent from becoming true and also it was not released in this time period essentially. So nothing happened which released it from the commonsense law of inertia. Then we are asserting that persists between t1 amd t2.

Then we have the axioms of event calculus. One thing that i could have mentioned that when we were talking about HoldsAt we can add another predicate called Initially which is equivalent to saying HoldsAt f t0 where t0 is some starting time. So initially, just a short form again. You could have just used HoldsAt time t0 but typically we will say Initially, that in the beginning that was the case essentially.

There are certain sets of events, axioms of EC that we use to describe the effects of, which are basically used to reason about change. So these axioms are called lets say EC1. It says that if an event e happens at time t and if we know from the domain that e initiates f then we can infer that HoldsAt f at time t. There is one way of figuring out what is true essentially. So if some event has happened which makes this particular fluent true then we can infer that it is true. Is similar but we have Terminates here and then we have not HoldsAt

(Refer Slide Time: 25:02)



KRR-Fri20-2015 - Windows Journal
File Edit View Insert Actions Tools Help
15 / 15
15

Shortcuts

Clipped (f, t_1, t_2) — some event e happened. and made f false during $t_1 - t_2$

Declipped

PersistsBetween (t_1, f, t_2) — it is not clipped and not released ...t_1 t_2

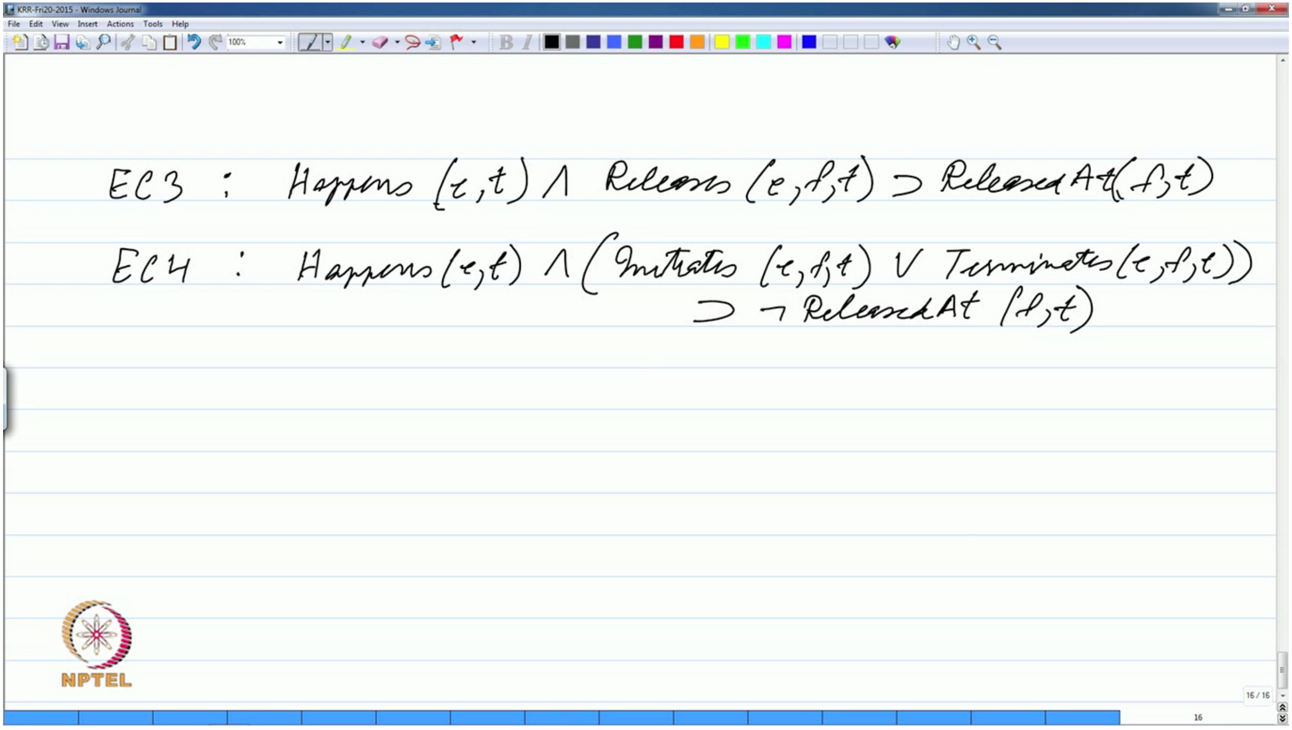
AXIOMS of EC

EC1 : $\text{Happens}(e,t) \wedge \text{Initiates}(e,f,t) \supset \text{HoldsAt}(f,t)$

EC2 : $\text{Terminates} \supset \neg\text{HoldsAt}(\ldots)$

Then we have EC3. There is an event which happened, see now we are making a distinction between the facts that events have to happen only then their effects are seen essentially. So we are saying that if an event e has happened at time t and if we know that this event will release the fluent f at time t, then we can say that it has been released at time t. Now we are saying that if an event e happens and it either initiates the fluent f or it terminates the fluent f then it is not longer ReleasedAt. That the fluent is no longer random.

(Refer Slide Time: 27:01)



KRR-Fri20-2015 - Windows Journal
File Edit View Insert Actions Tools Help
16 / 16
16

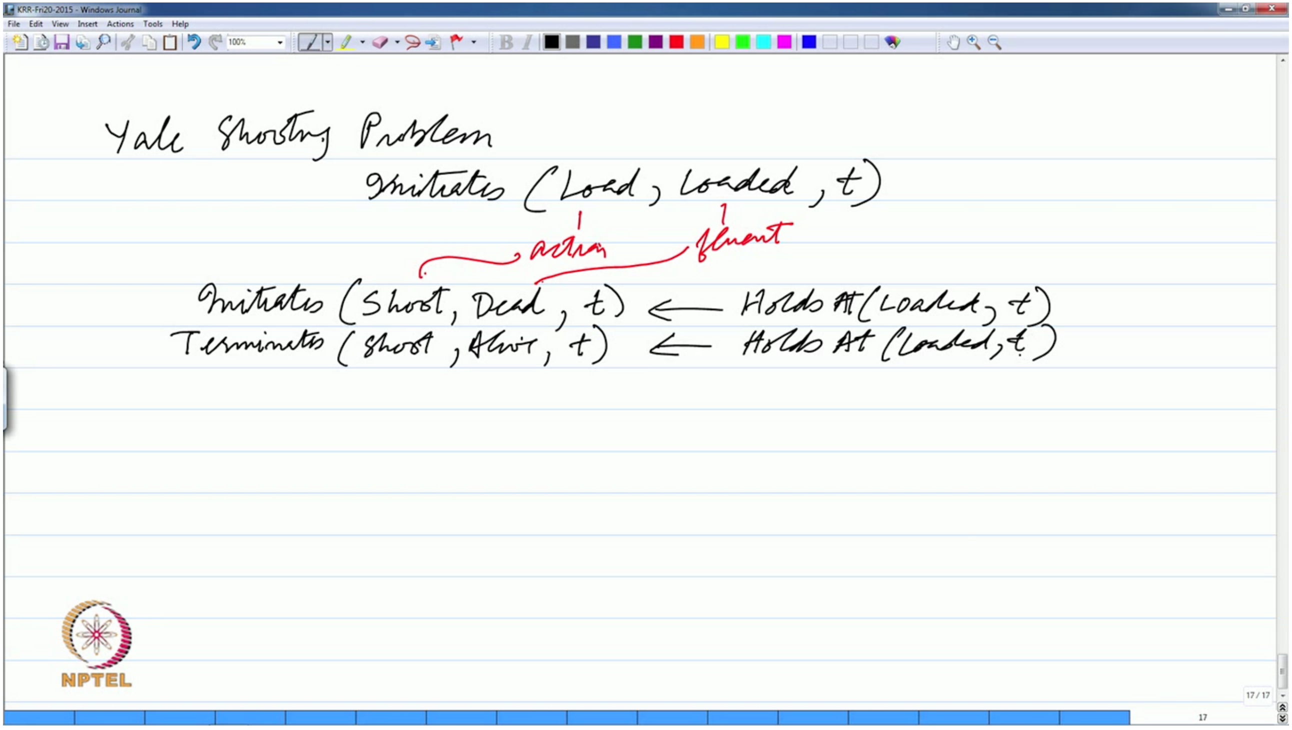
EC3 : $\text{Happens}(e,t) \wedge \text{Releases}(e,f,t) \supset \text{ReleasedAt}(f,t)$

EC4 : $\text{Happens}(e,t) \wedge (\text{Initiates}(e,f,t) \vee \text{Terminates}(e,f,t)) \supset \neg\text{ReleasedAt}(f,t)$

Now that event e has happened which either makes it true or makes it false then it must be true or false, it no longer random. So this is the basic language of event calculus. Now reasoning in event calculus is not such a straight forward thing. We have to worry about the fact that we don’t know what all happened and we do not what are the effects of what all has been happened. If our description of the world is open or it is not complete then we have to take recourse to slightly more sophisticated forms of reasoning that we will come to later. But i just want to illustrate something which is a well known example which is called the YALE SHOOTING problem. So i will urge you to go to the web and search for this and you will find nice description of this essentially.

So the yale shooting problem is as follows. It kind of illustrates this language. So if we have a statement which says that its a propositional language where we don’t have variables. So you can imagine what we are talking about. One of the actions is called Load and one of the fluents is Loaded and we say that if you load then loaded becomes true. So we are talking about a gun as you can imagine. Its a shooting problem. Then we say that Initiates so that another action, so thats another fluent. So we say that if you Shoot then Dead becomes true. So obviously we are talking about someone or something. But its not unconditional it is only if the gun is loaded. Which means if the gun is loaded and you shoot then dead will become true at time t. And you can say likewise, thats another predicate, another fluent.

(Refer Slide Time: 30:18)


KRR-Fri20-2015 - Windows Journal
File Edit View Insert Actions Tools Help
100%
NPTEL
17 / 17
17

Yale Shooting Problem

Initiates (Load, Loaded, t)

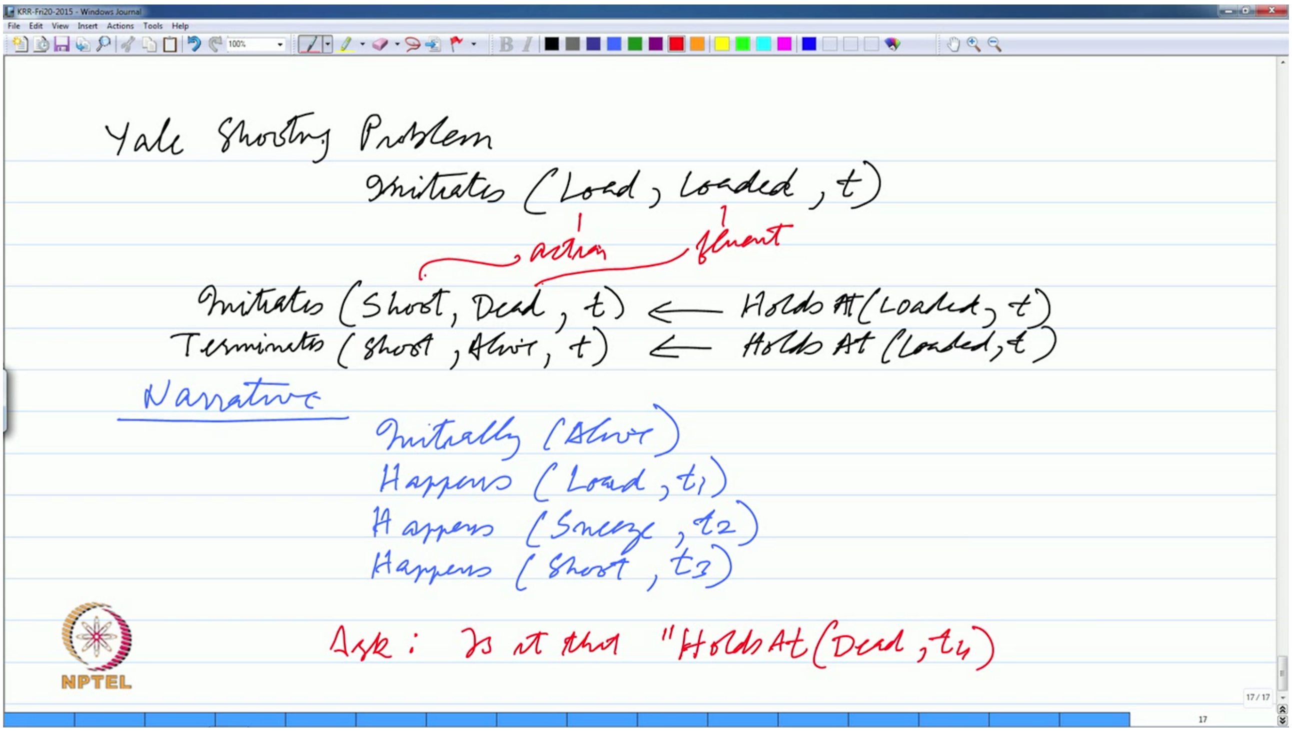
action fluent

Initiates (Shoot, Dead, t) ← HoldsAt(Loaded, t)

Terminates (Shoot, Alive, t) ← HoldsAt (Loaded, t)

so with these three statements we have described the domain. So our domain is very simple. If you do the action Load then the gun will become loaded. If the gun is loaded and he shoots somebody then that person will be dead. So its a very simple domain here. And then here is a Narrative. It says that Initially alive. Remember this is a short form for HoldsAt time t0. Then Happens Load at time t1, Happens, and suddenly we have introduced an action called Sneeze. We have not said anything about what is the effect of Sneeze and things like that. But we have simply said that there is an action called sneeze. And it has happened at time t2 essentially. And then we ask.

(Refer Slide Time: 32:14)


KRR-Fn20-2015 - Windows Journal
File Edit View Insert Actions Tools Help
100%
NPTEL
17 / 17
17

Yale Shooting Problem

Initiates (Load, Loaded, t)

action fluent

Initiates (Shoot, Dead, t) ← HoldsAt(Loaded, t)

Terminates (Shoot, Alive, t) ← HoldsAt (Loaded, t)

Narrative

Initially (Alive)

Happens (Load, t_1)

Happens (Sneeze, t_2)

Happens (Shoot, t_3)

Ask: Is it that "HoldsAt(Dead, t_4)

there is something I have not stated which you will have to state is that t1 is before t2 is before t3 is before t4. These are also separate statements that we have to make, each one individually. So that yale shooting problem basically says this Initially Alive was true, then gun was loaded then Sneeze happened and then the Shoot action happened and the question we are asking is Is it true that Dead will be true after time t4? Which is equivalent to saying that Is it true that Alive will be false in time t4?

Now the problem with trying to come to a conclusion here is that we do not know what are the effects of Sneeze? We don't know we have not stated this. If we have not stated this what is the influence you can make essentially? So we will come to this later. So the default inference that we would like to make is the sneezing doesnt affect the status of the gun essentially. But who knows that sneezing could have somehow unloaded the gun essentially. So in some sane world that you live in if you sneeze then the gun goes off automatically and therefore it is unloaded. We have not even stated that shooting will make the gun unloaded. That could be another domain statement that we could make. So there are many things we have not stated essentially. What if we have introduced another action. If Happens SPIN in time t2. Ofcourse we have not talked about Spin but we did talk about Russian Roulette. So what if I introduce a third action which says that you are spinning the barrel at time t2 and then you are shooting. Then how do you figure out. What can you conclude at the end of the day whether the poor victim is dead or alive essentially.

So i have introduced the language here, the event calculus language which allows us to talk about events and the effect events have on fluents and so on and so forth. You can say that if you for example Eating a dosa results in you becoming happy so you can make such statements. And then you can give a story and ask is john happy or not? But things become complicated when the description of the world is incomplete essentially so we do not know what is the effect of sneeze and we do not know if spin has actually happened or not. Just because we have treated that these are the four events which have happened doesn't mean other things have not happened. So we have to implicitly make an assumption that the only things that we know are the ones that are stated. And if we can say that okay we dont know effect of sneeze so there is no effect of sneeze or loaded. We dont know spin happened so spin did not happen.

And under those assumptions we can say yes if we make those assumptions then it is true that the person is dead. But it is subject to only those assumptions and reasoning in an open world forces you to make those assumptions.

We will come to this not in the next class but much later when we look at what is called as default reasoning. How can you make inferences which are default inferences. Where such assumptions are made essentially which says that everything we know is the only thing we need to know essentially. We will look at mechanisms of that towards the end of course. In the next class i want to continue with representation and I want to look at one very commendable effort in representation in which they try to choose a very small set of predicates and talk about everyday action in the world. Thats called Conceptual Dependency Theory and we will look at it in the next class.