PROFESSOR: Here we have a near miss learning tree. It's a little bit different but a little bit similar. We've got different types. We're trying to learn about different light sources. So we're concerned about different lamps, flashlights, all sorts of things. So these guys can have had types of support, a base or wires. The base can be flat, clamped, or have legs. The legs can be flat bottomed or wheeled.

The light source can be incandescent, fluorescent, or sodium vapor. And the energy source that powers it can be electric, battery, oil, or gas. So there's also some other things. But these are the trees that we might be climbing up with the climb-tree heuristic.

Our starting model is we have an incandescent bulb. The height is equal to 24 inches exactly. They have a flat base. There's electricity to power it. And it has a shade. So our starting model is basically like a table lamp that you set on the table. It's got the shade over the top of it. You know, the standard room table lamp is our starting model.

So recall in arch learning, which is what we are using here, that there are several heuristics that we might want to use. Let's see, well, the heuristics are-- we've got require-link, forbid-link. We've got climb-tree, extend-set, close-interval. And we also have drop-link-- forgot about that one. That's sort of with the links. I can put it here.

OK, so if you want to do well on these questions if they appear on the final, first of all, you better know what those six heuristics do. So basically require-link is a heuristic. It goes into your model as a requirement, where before your model just didn't care. So let's say there was something about color for your light, like for your lamp's base or for the lamp. And right, now it doesn't care about color at all.

But let's say that you wound up having an example that was, for instance, blue or white or whatever these colors are. And it wound up getting that blue was OK for-- in its model, it wound up getting, OK, blue and white are OK.

Or actually, no, an even more basic example-- let's say that it just kept getting blue lamps as examples, and all of them were blue. And it just didn't care about the fact that they were blue. But they did turn out to be blue. And then eventually it found a red one that was exactly the same as its model, but the red one was a near miss. Then it might require blue, which was already an acceptable, permissible thing in its model. But it could be required.

Now, another thing it can do is forbid. Some people ask me, why would you bother saying, forbid red, say, if you can already just require blue? Together they cover the same ground. The answer is because this system, this arch learning system-- remember, it's Patrick's doctoral thesis. It's an old system.

Because it was built on old systems and because also it's generally a good idea, it was very parsimonious. Let's say you had a set of colors-- red, blue, yellow, green, pink, purple, orange-- and you found that only orange was giving you problems, you could try to require that it be in this. Or you could even use extend-set, make a giant set of red, blue, yellow, green, purple, pink, but not orange. But if you just forbid orange, you've just saved a whole bunch of space, particularly if there's a large possible series of set. You want to have the ability to forbid. Because it needs to have the smallest model that it can that covers all of its samples.

So we also have climb-tree. Climb-tree takes one of the elements in our model and moves up the tree one level. So let's say we said, only incandescent lights are good. But we found that fluorescent lights are also good. Whoop, we move up to light source, climb-tree.

Extend-set-- we don't have any sets here. But let's go back to red, blue, orange, yellow. Extend-set is when we say, only red is good as an example. But we see another positive example that's yellow. So we say, OK, I'll extend the set. Red and yellow are both good.

All right, close-interval-- close-interval is really obvious when you need to use it. Because not only is it only used for intervals like height equals 24 inches or some kind of numbers, but it is the only thing you can use for when height equals some kind of numbers. Close-interval will let you fuss around with the intervals and say, oh, well I guess one with a 20 inch height is OK, too. So we'll make the whole interval 20 to 24 inches good. Because it doesn't make any sense for it to be like, 20 is good, 24 is good, but nothing in between is good. No, those are awful. So that close-interval covers the entire interval.

Last but not least is drop-link. The thing about drop-link is that drop-link is, again, only really used due to the fact that the system wants to be as parsimonious as possible with its model. Drop-link is-- let's say you have a color, and you're like, OK, only red is acceptable. And then you see a blue one. It's OK.

And let's say red, blue, and yellow are your only colors. Only red and blue are acceptable. Hey, wait a minute. No, we can just say that only yellow is not acceptable, or something like that, all right? But then you see yellow as a positive example. Actually, you can't even switch to the yellow being not acceptable, because you've only seen positive of red and blue. So you say, only red and blue are acceptable. Question?

AUDIENCE: You mentioned that if we had seen incandescent and fluorescent, it's OK that we could climb the tree to light source, because it's more parsimonious. If we then see one that's sodium vapor, say a street lamp, and we reject it, do we have to go back down the tree, or do we just add a forbid-link?

PROFESSOR: Then you would add a forbid-link. The question is, we climb incandescent, fluorescent, to light source. When we see sodium vapor, what do we do? The answer is, our system, unlike with lattice learning, in arch learning, our system is memoryless of its previous examples. It's incapable of going back down the tree. And so if you see incandescent and fluorescent, climb up to light source, you're only recourse, if you see that sodium vapor doesn't work, is to forbid-link sodium vapor. That's a good question. It shows good understanding of how we work.

So drop-link, as you see, red and blue, they both are OK. You don't know about yellow, so you can't just switch that to a forbid-link to yellow. But then you see yellow is also OK. What do you do? You can drop the link altogether, save yourself space.

If all the colors that you have in the entire world are fine, you can just not have part of your model be the color, and just say, any color is fine. Color must not be really part of what defines a lamp. And I'd say it's not. I mean, it's what defines a tacky lamp, I guess. But it's not what defines a lamp.

So that might be why color isn't here to begin with. So to make yourself do better on this, more knowledge equals less search. Let's have more knowledge. There's only some of these are used to generalize, to make our model more accepting of new examples. And those are ones we would use after we've seen a positive hit, sort of generalize, learn more things in the model.

Some of these are used to specialize and make our model refined, make our model more specific. You'd only use those after you saw a near miss. So let's actually separate those. That way you guys will never make the mistake of using one of these in the wrong situation. And more knowledge, less search. And less search means a faster quiz time. So require-link, what do you guys think? Is that a specializer or a generalizer?

AUDIENCE: Specializer.

PROFESSOR: Specializer, that's right. We'd only use it when we saw a near miss. Forbid-link, specializer or generalizer?

AUDIENCE: Specializer.

PROFESSOR: Specializer. We'd only forbid something if we saw a miss. Climb-tree, specializer or generalizer?

AUDIENCE: Generalizer

PROFESSOR: Generalizer, that's right. We'd only climb up to a more generic thing in the tree if we saw a positive example. Extend-set?

AUDIENCE: Generalizer.

PROFESSOR: Generalizer. We'd only extend the things in our set if we saw a positive example. Close-interval?

AUDIENCE: Generalizer.

PROFESSOR: That's a generalizer. We'd only make the interval bigger if we saw a positive example. Someone asks, what do you do if you have like 10 to 30, and then you find a negative example in 20? It's generally pretty annoying for this system to have to deal with that. There's a variety of things you could do based on your implementation.

I've never seen us ask it in a quiz. But one thing you could do is forbid 20, exactly just 20, and just have a little hole at exactly 20. So then drop-link-- specializer or generalizer?

AUDIENCE: Generalizer.

PROFESSOR: Generalizer. This one's an easy one to mess up if you don't understand the system. Because dropping, you think, oh, getting rid of something. That's specializing. But actually no, you're generalizing the entire area saying, we can forget about it. Because they're all good.

All right, so given that, we are set to do this problem. Hey, we have a pretty reasonable amount of time considering that we don't have to do our sums. It's great. So our first example is sort of one of those stand reading lights that you sort of have it on a stand, and you can adjust the little light bulb up or down. It is an incandescent bulb. It has a height of 11 inches. And it's got a flat base.

It's electric. It's got a shade. And it's a hit. It's a hit, a positive example, a plus. It's good. So right away, we know that we can only use the S's or the G's? The G's. We can only use the G's. Because it's a hit. So we're not going to be requiring. We're not to be forbidding today. We're going to be happy and using a generalizer.

So I'll help do the first one. You guys will help me do the next one. Or someone can say, we didn't cover this, and I'll do the first one, and we'll stop. So we've got an incandescent light. That's the same as our model. We've got a height of 11 inches. Oh, our model only covers 24. Flat base-- that's the same as our model. Electricity-- same as our model. Shade-- same as our model.

So I say the only difference is the height. I say we need to close the interval, close-interval. And I say that our model is the same as before, except for that height is an element of 11 to 24. Simple enough-- I picked the easy one for myself.

Next one. So this one is going to be-- the picture here didn't turn out. But it's basically a lamp that doesn't have a shade. It just sort of has the light shining on you. So it is a positive example. It's incandescent, height equals 11.5, flat base, and electricity.

All right, so first of all, which kind would we use, specializing or generalizing?

AUDIENCE: Generalizing.

PROFESSOR: Generalizing, and particularly, anyone want to take a look at our model, which is the starting model, except for the height can go 11 to 24? Which heuristic do we need to use here?

AUDIENCE: Drop-link.

PROFESSOR: All right, people are all saying drop-link. Yes, that's right. Someone said extend-set. Well, I suppose we could extend the set to shade or not shade. But that's everything. So drop-link is the answer. We can drop the shade. Shade or not, this is still pretty much a lamp. So that's correct. We drop-link. So the model-- all right, there's been enough changes to it that I think I'll write it out again.

The model is incandescent, height equals an element from 11 to 24, and flat base, and electricity. Good, so question?

AUDIENCE: So during this all, we're editing our model based on the order that the model is in, right?

PROFESSOR: So the question is, we're editing the model based on the order of the examples?

AUDIENCE: What happens if there's the same thing except with no shade or something like that afterwards?

PROFESSOR: So the question is, what happens if you have the same thing except for that there's one that's the same with no shades and is a near miss or something so that you need to-- or I'm sorry, you're saying, what if there's one that has a shade now that's a near miss? And so it would probably try to forbid-link shades. That's sort of the question?

You're right. That's an inconsistency in the data. The system is very fragile. It's very fragile to ordering in particular. And you'd be surprised how awesome this does considering that it was made in, like, the '60s. It's pretty impressive stuff. But it's old. We're not saying that this is a way that you should do all of your learning nowadays. Because it has some serious issues.

It's old, but it's pretty damn good for what it did. A newer style of this kind of learning made by me and another friend who's now working at Google is lattice learning, which has its own share of issues, one of which is that since it's trying to act like a little kid, at first it tries to claim that everything is OK until you show it some good negative examples. But very particularly one way gets around this problem is it's not memoryless.

It in fact stores all of the examples it's ever seen and compares and contrasts them to what it sees in the new example. And this allows it to say something like-- let's say you're trying to teach it what can fly. In lattice learning, you say, all right, well, a blue jay can fly. And then if you ask it, can a cow fly, sure, a cow jumped over the moon. Everything can fly. So that's its problem.

But if you say, actually, that was a nursery rhyme, a cow cannot in fact fly, it will then say, only birds can fly. Because it remembers that the blue jay can fly. And then you can eventually say, well, actually bats can fly, a fruit bat can fly. Well, OK, so chiropterans can fly and birds can fly. But that's it. And you can give it some other examples.

But there are other styles of learning. You're absolutely right. In fact, if there's a multiple choice asking you about the strengths and weaknesses of arch learning, one of the weaknesses-- it's very vulnerable to ordering of what you teach it. But think about it. If our first example was one that's exactly the same as this but with a shade, and it was a miss, the system would just yell at you. Like, you're kind of being an asshole to it to give it two things that are inconsistent.

You might say in the real world that happens. Well, arch learning, not great with real world messy data. It just goes ballistic. It's very OCD. It wants everything to match up. If it gets two things that are inconsistent, it'll just yell at you. It's like, you're wrong. This can't be true. Because you said it's OK with the shade. So that's a very good question. Another question?

AUDIENCE: So if you have a sample that had two different things for the model, and it's a near miss--

PROFESSOR: It's not a near miss. The question is, if you have something with two different things in the model, and it's a near miss-- and I'm sorry to cut you off. Based on timing, I'm just going to say, whatever you said afterwards doesn't apply due to the fact at that point it's not a near miss. It's very important. It's only a near miss if there's only one change.

Otherwise, it's just a miss. And if it's a miss with more than one change, you were probably going to say, how do you know what to change? The answer is, you can't change anything. Because any of them could have been what was wrong.

That's why for a hit, oh, it can have as many differences as you want. You'll just generalize them all. For a miss, it can only have one thing different. In fact, there might be one here that's not a near miss, and you just say, oh, we don't do anything. We'll keep going. We might see it if we have time. So question?

AUDIENCE: Because ordering is an important factor for this bigger thing, and you encounter an item which has two discrepancies in values, but if the ordering were different such that that would be accounted for, do you remember if you turn back to that [INAUDIBLE]?

PROFESSOR: So the question is, let's say you have a non-near miss. Can you hold onto it and use it later when it would be a near miss? The answer is, nope. If we're going here, for a system that doesn't remember things, you can put in some kludges. That definitely makes it smarter. It uses all its data.

But Patrick in the '60s making this up strove for elegance. And the elegant solution is, let's just be memoryless completely, the idea being, well, little babies can't tell you every experience they had with playing with blocks with an arch. So let's remember nothing. With lattice learning, our idea is that people sort of somewhere in the subconscious maybe do store all the examples, or at least a lot more than you give them credit for. So why not store them?

But with arch learning, it's, little babies can't tell you, oh yeah, there's that one time I played with a lamp, and it didn't have a shade, and so it wasn't a lamp, or something like that. They're not going to be able to store it. They're not going to be able to save it for later.

Sort of in the style of Turing, who believed a human would be a teacher to a computer, arch learning really focuses on the fact that the human is a kind and good teacher who offers examples that are exactly appropriate at the time. And that puts a lot of pressure on you as the trainer of an arch learning system.

Good questions, all. Let's continue working this guy out. So the next example is basically another one of these stand lamps. It's got a light shining down, which is fluorescent. It looks like this. It's fluorescent. Height equals 13 inches, flat base, electric, shade.

And yeah, as I put here, it's a hit. So as you guys have told me countless times, so I believe you that you remember, we're going to generalize for the hit. So what should we do here? So some people are saying extend-set.

AUDIENCE: Climb-tree.

PROFESSOR: The people who are saying climb-tree are correct. Why do we climb-tree instead of extend-set? Because of the fact that it's a tree, not a set. And it is more parsimonious to climb the tree than it is to treat it as a set and extend the set.

Sets don't have a hierarchy. It's just like red comma blue comma yellow, or shade comma not shade. Whenever you have a tree, it's a fact that it's more parsimonious to just climb it than it is to treat it as a set and extend it, even though you could do that, I suppose. And arch learning, it's parsimonious. It's elegant. It's simple. And the elegant thing to do-- just climb up.

So yes, we're going to climb up the tree. So our model is the same as last time except that this time it has a light source instead of saying, incandescent. It just climbed up to light source, sure.

All right, so the next one is pretty cool. It's another one of those fluorescent shine down lamps. But in space, it has a tripod with three legs. And those legs all have wheels.

So it is also a hit. So it is fluorescent, height equals 14 inches, wheeled legs, and electric. So obviously we're generalizing. What heuristic will we use this time?

AUDIENCE: Climb-tree.

PROFESSOR: Climb-tree again, that's right. We're going to climb from flat base to base support itself. Because wheeled legs and flat base, base support is an ancestor of both. So our model is the same as before. But at this point, it's changed enough that I might as well write it out. So we'll say that it is light source, height between 11 and 24 inches, base support, some sort of support, and electricity.

Awesome, now, come on parity. Ah, no, back up, you. There we go. So that down there, don't get confused. That down there is our most recent model. We now have-- oh, oops, hehe. OK, I actually missed one of the examples. However, we were actually completely correct, which gives away quite a lot.

We had a miss, incandescent bulb-- back in the old days when incandescent was the only kind of light we had. Height equals 60 inches, flat, back in the old days when flat was the only one we had, electric, back in the old days when we had electric. And we also have shade after we've dropped the link for shade. So we know that shade or not shade are both OK. So the question is, what do we do? First of all, do we specialize or generalize with this miss?

AUDIENCE: Specialize.

PROFESSOR: So obviously we're only to be able to require or forbid. So standardly, traditionally, the answer would be nothing. However, we accept it as sane the possibility that you might put a require onto the height. Some students did that, like say, oh gosh, that's a hard set, 11 to 24. But that's not what you would normally do. So just to show that we tried to understand that there might be multiple ideas in a weird situation, we did accept that that one time.

But generally, there's not anything you have to do. This system at the time, and still now, said 11 to 24. This is spookily exactly the correct place where this example would go. But you don't have to do anything to the model. You don't have to use any heuristics. OK, so last step, -- the question is-- oh, question.

AUDIENCE: So here, if we did it in this particular order, having it be the next [INAUDIBLE] in our example list, then it only differs by one from our existing model, right?

PROFESSOR: No, it differs by a lot. It actually wouldn't be a near miss if we did it out.

AUDIENCE: It would be a near miss if we did it out.

PROFESSOR: No, incandescent is different than light source. Height is different than height. Flat is different than base support. So it would differ by a huge number, and it would not be a near miss.

AUDIENCE: OK, then here's a question. Even though on the tree incandescent is a child branching off of light source, if we instead encountered-- because our current model says light source comma whatnot, if instead we got a negative thing that said, fluorescent, again, would that be considered not a near miss, because fluorescent differs from light source?

PROFESSOR: So the question is, would it not be considered a near miss if you had, say, fluorescent rather than light source, because fluorescent differs from light source? So the answer there is--

AUDIENCE: Since it's memoryless, it doesn't know that fluorescent--

PROFESSOR: Well sure, it is memoryless. It doesn't know that fluorescent used to be a positive example before. Unfortunately, let's say we did have an example that was fluorescent, height is an element of 11 to 24, base support, electric. Actually, let's say we have one that was sodium vapor, height is 13.

Sodium vapor, height is 13, base support-- for some reason it was called base support and electric. That's possible, right? We haven't seen a positive of sodium vapor. Let's say that sodium vapor was not allowed. If we don't say that that's a near miss and then forbid-link sodium vapor, how are we ever going to get rid of sodium vapor?

Unfortunately, because of that, we lose some amount of expressiveness. Actually, what you're saying is very logical and makes sense, that you would want to have the ability to say, OK, this is a subset. This should be OK. But unfortunately, because of the fact that we climbed the tree even when we haven't seen a positive example for all things in the bottom on that tree, we actually lose that ability. It's a trade-off. It is most certainly a weakness of arch learning that you point out here. Lattice learning fixes that, but it has its own problems.

So you're right, that you'd want to be able to do that. But you can't. Because it's memoryless. So last but not least, which model would we present in order to teach the system, right now from its final system, that a lamp requires a base support? So it would need to be require-link for the base support.

So really fast-- incandescent bulb with a height of 6, a flat base, and electricity. Would that do it? Incandescent bulb with a height of 8 and a battery, incandescent bulb with a height of 12, wire support, and electricity-- oh yeah, by the way, this will have to be a miss to teach a require.

Fluorescent bulb, height of 21, clamp base, and electricity, or incandescent bulb, height of 12, flat bottomed legs, and electricity-- so did anyone pick that out over the fastness of me saying it? Because [INAUDIBLE] is going to want to come in. Question or answer?

AUDIENCE: [INAUDIBLE].

PROFESSOR: It's going to have a miss.

AUDIENCE: If it's on the wire support?

PROFESSOR: Yes. So I know it was fast. You can look through quiz 2007 final. Because [INAUDIBLE] is going to come in in a minute. But it would be a miss on the one that was correct on everything except for it was a wire support. Then the question you have is, wait a minute, it says incandescent bulb. That is now a subset of light source. You have his question again.

The answer to that is they must have decided here that that was OK. But on the other hand, how would it learn sodium vapor if that wasn't decided to be OK? That's going to have to be an implementation detail. So one thing you can do is try to treat subsets as OK in the logical way. Because questions may do that.

And then I guess you've lost the ability to forbid, say, sodium vapor. So you have to lose one or the other. I think you're definitely OK to ask TAs or someone who's proctoring if a situation like that comes up, which implementation detail is chosen. Because you could choose either when you're running arch learning.