TOMER ULLMAN: What we're going to do in the last section is not so much have a lecture as a debate. And it's going to be a debate between myself and Laura Schultz. And hopefully at the end, all of you can join in for some sort of free for all. Although, of course, you're welcome to ask questions at any moment.

Now, Laura and I have had this debate a few times now. It's a debate about where do new ideas come from. It's about theories. It's about imagination.

The last time that we were supposed to have the debate was in SRCD, which is a child development conference. But Laura couldn't make it. So I ended up pulling a Monty Python man wrestles with himself routine where I was arguing both my point and Laura's point. So it's nice to have a sparring partner again after that experience.

I am still a little intimidated. Just to give you a sense of the caliber of my opponent-- the reason Laura didn't show up to our SRCD debate was that she was giving a talk of her own in something called TED. I don't know if you've heard of SRCD, but presumably you've heard of TED. OK.

So what's this debate about? What am I talking about? Here's a 65 second prologue sort of setting it up. And then we'll get into it. The background to this debate is that, you know, we have this sort of really nice picture coming out of development and models.

And sort of Laura and Josh have set up the perfect intro to that, which is from development we have this idea of children are sort of like scientists or hackers or what have you. They come up with these theories to explain the world. And they do it beautifully. And that's what they do in development.

And then we have this idea coming out of computational land which is, well, what we mean by theories-- how do we actually capture that computationally? How do we formalize that? Well, maybe it's something like hierarchical learning over space of programs, right?

Because programs are this the thing that's kind of like theories. And they kind of explain the world. And you can learn them via Bayesian inference and programs over programs and learning over learning.

And that's sort of a really nice picture. But a lot of people-- Josh I think was using the phrase-- or Nour was using the phrase-- "give us grief." A lot of people give grief to these computation models correctly.

They say something like, wait a minute. These computational models that you've described, like Josh was describing earlier, these hierarchical programs over programs, what's the theory space on that, right? Like, what's the space of all possible programs?

Remind me. Isn't that infinite? And isn't infinite really, really big? What are you trying to suggest, that children are just searching through this giant space of programs? How could they possibly be doing that?

And what we'd like to argue and I'll spell out is that while we don't search through the space of all programs, we don't do that when we write out our programs, right? We somehow figure it out. We're writing out these infinite spaces. And yet we search them in our computers.

That's what we're try to suggest. The same way that we search them in our computers might be something like the way the children actually have to search for their theories. And it's hard, grueling work for computers and children.

And the algorithms that we're, in particular, interested in-- there's all sorts of learning algorithms, of course. But one of the most influential learning algorithms that people have used for learning these programs and hierarchical programs over programs and things like that is just sort of this version of stochastic search. And I've told you a little bit about that-- those of you who went to the church tutorial-- about Markov Chain Monte Carlo, and all that stuff.

But the point is something-- well, the way to search large, complicated theory spaces is through something like stochastic search. And, well, if it's something like stochastic search algorithms for our theories, maybe children are doing something like stochastic search, right? No is what Laura's going to say. Or she is going to say that can't possibly be the whole story, and here's what's missing.

So does everyone sort of understand what the debate is going to be about more or less? I'm going to spell out, of course, all of this in the next few minutes. OK. So as I said, we're going to switch back and forth, Laura and I.

The outline of the debate is that I'm going to give some background for like what good our theory is or presenting a good theory-- just really sort of covered that. So I'll be zooming through it just to give you an example of what I'm talking about. Because I want to tell you what stochastic search in theory space looks like.

After I do that, Laura is going to step in. And she's going to rudely interrupt me and say, no, no, no. That can't possibly be true. Here's all the things that are wrong with that.

I will then give a rebuttal. And then Laura will end and summarize. Beyond all these things, something I didn't write was all of you. Jump in and say what you think.

OK. So what good is a theory? Like I said, Josh sort of went through this. I don't think you guys need that much convincing at this point. But by theory, I mean some sort of structured knowledge that goes beyond the data, compresses the data in some way, and is able to predict new things.

My running example is going to be much simpler than character recognition or anything like that. It's going to be about magnets and, in particular, very simplified theory of magnetism. So suppose we bring a child into the lab. And we tell her, look at these blocks.

They all look the same. Play with them. See if you can figure out what's going on here. OK? And unbeknownst to the child, but beknownst to you, is that these things are magnets.

They're not just blocks. Some of them are metal. Some of them are magnets. Some of them are plastic. OK?

So she's going to start playing with them. And she's going to start noticing something. Like, sometimes none of these things do anything, right? They don't stick.

But sometimes, huh, they do stick. And she starts collecting observations in something like this. Like, how could she explain the data?

Well, she could explain the data in this. You can just write it down. She could like label these somehow. She could have all these, A to I.

And she could say, well, A and B attract one another, B and A attract one another, B and C attract one another, and so on. OK? That's her theory. That's her explanation of the data.

This is horrible, of course, right? Like, this is not an explanation in any sense of the word. It's just a table of data.

But in one sense, why is it not a theory? It's because it's just writing down what you've already seen. It doesn't compress it in any way. And it can't predict anything new.

If I now give you a new block X and I tell you, listen, X attracts B, what else can you tell me about it? And she'll say, well, X attracts B. Yeah. OK. That's what my table tells me.

You can't really predict anything new from a giant table like that. You want some sort of compression. You want some sort of theory.

But what else could she come up with, right? There's all sorts of things you could come up with. But she could have come up with a very simplified theory that goes like this.

Suppose that we imagine that there are certain things in the world. We're just going to call them, like, shmagnet and shmetal. Because I don't want to confuse it with actual magnets and actual metals. But she comes up with a name.

She says there are several things in the world. And how do these things interact with one another? Well, let's just hypothesize some rules.

And we can talk about how does she come up with these rules. How does she know that there's two things in the world? Let's say for some reason by dumb luck she hits upon this, which is actually a really good theory for explaining that, which is to say there are two things in the world. And the way they interact is through these rules.

If something is a magnet and another thing is a magnet, if X is a magnet and Y is the magnet, they're going to interact. They're going to stick. If X is a magnet and Y in the metal they're going to stick. And interactions are symmetric.

OK. So metals don't stick to metals, but metals stick to magnets. Magnets sticks to metals. And the interactions are symmetric.

And if you have these rules, you need to pay some overhead, obviously, for remembering the rules. But you can compress this, you know, n by n thing into just the vectors of remembering

what are the magnets, what are the magnets. So you've achieved some compression.

And if I give you something new and I tell you, look, this is X, X attracts A, and you're like, wait a minute. A is a magnet. So this is either a magnet or a metal.

You can probably predict a lot of different things about this thing. You can go and design your little experiment to figure out if this new thing is itself a magnet or a metal. Great. So this is wonderful.

I won't go through something. Like, there's this added thing of finding out which one are the actual shmagnets and shmetals. And then you can predict the observed data. OK.

So now, what we've done is we've set up this sort of space of possible theories, right? Imagine like all the possible logical theories that you could have written out, of all the possible logical predicates, there's an infinite amount of them. But now we've turned this into a rational inference problem, right?

The problem for you as the learner is out of the space of all possible theories, just find the one that best explains the observed data, right? Where, by best we mean something like Bayes' law, right? Try to find the best theory to predict the data, whereby we mean the theory that's sort of the shortest and most compressed in itself a priori.

You've all seen Bayes' rule. You've all heard Josh talk about this and other people. But it also explains the data itself the best.

That's the problem. Go figure it out. We formalize it for you. We've solved it.

And, of course, you know, this elides a lot of things. It doesn't really solve anything. But we'll get to that in a second. So you might wonder like how do we build a good space for possible theories.

And one way to do then is to say, well, how do you generate all possible theories? How do you define a space of all possible theories? In this case, we're just going to go with grammar. But you could imagine something like a generator for all possible programs, right?

So a grammar, to put it very, very, very shortly, is something that you can run forward and will generate a sentence. Or it's a way of sort of looking at the sentence and figuring out what its underlying rules are. Let's put it this way, OK?

So in a grammar, you would start with a sort of a sentence node. And then you know that a sentence can go into several other nodes in some probability. For example, a sentence node can go into something like a verb phrase and a noun phrase. The noun phrase goes into a noun and another thing.

And you sort generate through this grammar until you end up with a sentence. And it can be any sort of sentence. It can be the clownfish ascended the stairs darkly, right?

And you can run this grammar forward and generate whole new sentences that you've never thought of it before. And you could then also use that grammar to figure out sentences. So if you see the sentence, the clownfish ascended the stairs darkly, you can use something like a grammar to figure out, well, you know, how is this sentence constructed.

This is a grammar for logical predicates. It just means that if we run it forward, it starts out with some sort of abstract thing like, you know, it's going to be a law or some sort of set of logical predicates. You run it forward, and it ends up not with a sentence, like the clownfish blah, blah, blah. It ends up with a particular law that's relating a few particular predicates like, you know-- sorry-- like X and Y interact symmetrically, right?

Like, there are things in the world. There are X and Y. If X interacts with Y, Y interacts with X. That's a law. We run through the grammar, we generate that.

We could run through the grammar again, and it will say something completely different. Like, of all the things that I've related, there are such things as magnets. There are metals. And they should interact in this way or that way.

Once we define a grammar like that, you can predict all sorts of different theories, right? I gave simplified magnetism, but you could also capture kinship with a set of four laws and several predicates. You could capture taxonomy. You could capture very, very simplified psychology of preference.

OK. So as I said, this is another view of theory search. I haven't yet gotten to the algorithmic level of how do you actually search for these things. But I did want to set up the grammar, because it's going to be a little bit important later on.

The grammar is just a way of saying, you know, you start with something. You generate it forward until you end up with a particular set of logical rules. So, again, to phrase the problem

in a particular way of logical inference is to say we have the space of theories. It's an infinite space, right?

And before I've seen any data, there is some probability of the right theory is to explain my data, where the data can be something like-- play with these blogs, figure them out. Before I've even seen the blocks, before I've seen anything that I need to explain yet, I have some sort of space of all possible theories. It can be all the logical theories, all the possible programs in the world.

And I have some sort of probability distribution over which one of these are more likely than others, which comes from the prior. And the prior can be something on simplicity, like shorter programs are more likely, or the less free parameters the better. Something like that, right?

So before I've seen any data, I can already score some programs as being unlikely, because they're too long and too ungainly. Does that make sense to everyone? OK. And that's just a representation of that in 2D space.

It's saying, like, over here is theory space. And the size of the hills over 2D space is how likely each point, each point in theory space, is a theory, where theory can be, as I said, a set of logical laws or program or anything. And the height of the hill over that is the amount of probability you should assign to that theory, how much you should believe in it.

And as data comes in, what happens is you become more or less certain of certain programs. Like, suddenly you shift probability distributions around. You say, oh, these things that I didn't think were that likely, well, the data's really pushing me to accepting these theories. OK?

And that's what learning is, right? That's the view of learning. You just start with particular probabilities. You get some data. And then you shift that around, and you get other probabilities.

And that's sort of a very beautiful picture. And the only problem with it is that it is clearly false. And the reason it's false is because this is an infinite space.

And there's no way that you could have scored the entire infinite space. There's no way that what children are doing, what adults are doing, what computers are doing is to instantly shift probability mass over this entire space, right? New data comes in. They're holding in this infinite space and shifting exactly the probabilities around, right?

This view of learning is ridiculous, because, well, it's patently absurd. And also, people have taken issue with it even though it was never meant to be the story of learning that is happening in the real-- well, how should I put this, in the real world? Do you guys know the difference between the computational level and the algorithmic level when I say something like that?

Show of hands-- who knows about Marr's three levels? OK. Who doesn't know about Marr's three levels of explanation? OK. The point is to say when you try to explain the phenomenon, you're going to give it several levels of explanation.

You're going to give it sort of the functional level, what we might call the computational level. And then you're going to actually say how does this actually get implemented in an actual machine. And there are many ways of implementing things.

Like you might say, well, look, the general problem for vision is this, or this thing needs to be in addition function. These are the sort of things I wanted to do. But there are many different ways to implement that function.

Some of them are better than others. That's the algorithmic level. How does this actually get implemented in an algorithm? Then you can implement the algorithm in many different mechanistic ways.

You can go and implement it in neurons or in silicon chips or things like that. There are many different ways of implementing a particular algorithm. This view of, you know, you have some theories, you have some prior over theories, you shift the probabilities around as data comes in-- that's an explanation on the computational level.

That's not an explanation of how we actually shift them around, how we actually search for these theories. And really the computational level sometimes gets grief, because people say, well, what are you saying? Are you saying that Einstein somehow had the theory of relativity, we all had the theory of relativity, in our head?

And his process of discovery was just to say, well, I believe in Newton's theory. And I had some low prior on relativity. But then the data came in, and I shifted my probability to the theory of relativity.

That sounds not the way people actually learn. That doesn't sound like the way we discover things. That doesn't sound like the way we come up with new theories. That doesn't sound like the way that children learn.

And as I said, that's not exactly what we think. And that's not what happens in computers either. That's not the algorithmic level.

So what happens in the algorithm level is you actually have to search for your theories, OK? And this is what happens in stochastic search in particular. What how does in stochastic search-- those of you who are in tutorial remember this-- you have some space of theories.

OK, I'm still giving you the space of possible theories. Each dot in theory space is, let's say, a program. Or in this case, it's something like a theory in the sense I defined it before. It's a set of logical laws relating these predicates together.

OK. So let me use-- I'm not sure you can follow my hand like this, right-- gaze detection or whatever. So theory A is, let's say, a theory that has three possible laws. It says if X has a P, P can be anything. And Y is a P. It's the same sort of thing. Then they're going to interact.

The second law says if X is P and Y is a Q, they will interact. It doesn't matter. You don't have to figure this theory out, right? I'm just trying to give you an example of what a theory could be theoretically.

And they interact symmetrically. That's one dot. Let's call that dot number one. That's theory A. Here is dot number two, theory B. OK?

You as the learner, as the stochastic search learner in the algorithmic level, don't have access to the full theory space. All you have is A. That's where you are right now. That's all you have of the world. OK. That's how you can try to explain the world.

And what you can do is you can try proposing certain changes to your theory. You can try taking out a law or putting in a law or taking out a predicate, somehow sort of messing with your theory, hacking with your theory somehow, coming up with a new theory that gives you theory B. It's a different point in theory space.

And what you do then is you compare your two theories. You say, how well does this predict the data? So and so, you give it some score. You say, how well does this theory predict the data? So and so, you give it some score.

You say, how likely is the theory a priori? How short is it? How simple is it? OK. How short is this theory a priori? OK.

So you get some score for this theory that's based on the data in the prior. You get some score for this theory based on the data in the prior. And you basically decide which one of these two theories to accept.

You could either stay with your old theory before you proposed any changes. Or you could decide that, wait a minute, that theory that I just proposed, the new one, is actually a bit better. Or even if it's not better, it's not doing that much worse, so I'll jump to it.

OK. That's the stochastic part in stochastic search or in like these Metropolis-Hastings algorithm. This way of sort of jumping around in the space is the stochastic search I'm talking about. You're here in theory space. You have one theory. You propose a change to that theory.

And we'll get in a second to how you propose a change to that theory. You propose a change. On end up here. You say, should I move here? Let's check. Is this theory doing any better?

If it's doing better, move there. If it's doing worse, well, maybe still move there depending on how much worse it's doing. And this process is sort of jumping around in theory space. Probabilistically proposing theories and accepting them is not that far from what MCMC is doing or optimization through MCMC.

Oh, sorry. I'm clicking on this. OK. And you can notice that this sort of search is somewhat different from-- this is the picture that Josh was showing before-- gradient descent or convex optimization, the sort of thing that you might do neural networks. I'm not trying to suggest that this is exactly what happens in neural networks.

I mean, the more complicated neural network stuff, the energy landscape, can be quite complicated. And they still need to do stochastic gradient descent. But it doesn't look as fully connected and as horrible as these sort of theories on top look like.

Because the space there is much more-- I don't want to say well-defined. They're both well-defined. But it's sort of much easier to search through, right? It's sort of easy to get in these neural networks to know where you're going to sort of differentiate and quickly get to your target.

You're not so much doing this sort of hard laborious stochastic search. You sort of have this notion of, well, it's immediately going to be the best direction to go in is this. OK? I'm just going

to do gradient decent. I'm going to roll downhill. And then that's some sort of good point in neural network land.

OK. So how do we actually propose alternative theories? Well, you say, look, I have some sort of, let's say-- let's do this. OK. You have some sort of particular rule.

And then what you do-- remember when I said you have some sort of grammar over theories, kind of like a grammar in language for those of you who know? The grammar describes a particular tree that you walk through to end up with your theory. What you do to propose a change to it-- and this is true for both these logical theories, but also for programs.

You go to any [INAUDIBLE]. You go to any sort of node in that tree that generated your program or generated your theory, and you change it. You sort of re-sample from it or, you know, cut it and regenerate.

And then what that ends up doing is it ends up, for example, adding a new rule or, for example, changing the predicate, or adding a new rule again, or deleting a rule, or deleting a predicate, or changing a predicate, deleting a predicate by sort of changing predicates, deleting rules, adding rules, adding predicates. I wasn't covering the full space. You jump around in theory land.

OK. And this sort of dynamic of moving around in theory land stochastically-- one step forward, two steps back, you know, you don't know the target ahead of time, when you actually propose something, you proposed something new-- different learners might take different paths to get to the same target ultimately starting from an equal state of ignorance. You sort of propose different theories. You end up in different ways.

But usually, if you get the right data, eventually you end up in the same spot. These discrete moments of sort of, you know, faffing about and sort of saying, well, this rule, that rule, that doesn't explain it. This explains it, doesn't explain it.

Suddenly, you propose something that sort of clicks into place. You say, ah, it's not that there's two things in the world. There's three things in the world, right? It's not just metals and plastics. There's actually metals and magnets and plastics.

Ah. And suddenly, you rearrange the data and things like that, right? All these things, all these dynamics, have something of the flavor of children's learning. And I didn't give you the full spiel. And you can go and read some of our papers.

There's also been other people who have been very interested in sort of looking at the dynamics of stochastic search, how well it predicts what children are doing. Among them, I'll just mention people like Liz Bonawitz and Stephanie Denison, and also Tom Griffiths, and finally Alison Gopnik. So I also have TED speaker on my side. And not just any TED speaker-- it's Laura's former advisor.

So at that point, it's a good point to hand it off to Laura and see what she has to say about this. But a midpoint summary is to say theories are useful, right? I think Josh has already convinced you of that.

We want theories. The problem is that they define these rich structured complicated landscapes much more rich, much more complicated than anything that you might find in the neural network. Well, yes. And it's hard to search for these rich, complicated, fully connected landscapes, like the space of all programs or the space of all possible logic theories.

And the way to sort through it is stochastic search, which can be horribly slow and wrong and things like that. But the claim is something like, what are you going to do, right? I mean, we want these rich theories.

Rich theories define rich landscapes. And you just have to get away right now with stochastic search. Our algorithm solution for these spaces are stochastic search in that rich landscape. And, well, why shouldn't that apply to what children are doing? So here with a why not is Laura.

LAURA SCHULZ: I think I have one. I'm hooked up, right? OK. So when I first engaged in this debate with Tomer, I was stuck on this kind of a thought that what's going to happen here is-- in which, following an eloquent exposition of a former model, attendant experiments, and quantitative data, Laura proceeds to wave her hands around.

Someone was asking about intuitions. And I just had this strong compelling intuition this was completely wrong. But mainly, I was puzzled as to why I was stuck on this archaic locution. What was I doing with this "in which" situation?

And it occurred to me the reason I was stuck on this particular locution is that I was thinking of a particular story that nicely illustrates exactly what I think the problem is with stochastic search. It's from a classic in developmental literature, the stories, of course, A. A. Milne and

Winnie the Pooh. And this is the particular problem of "in which" Christopher Robin and Pooh and all go on an expedition to the North Pole.

And so they're organizing a search. And like Tomer's algorithms, they don't actually know where the North Pole is. And it turns out, as Christopher Robin confides in a moment to rabbit, they also don't know what the North Pole is, also like Tomer's algorithms.

But they do something about the terrain. And they know something about search. You gather all the friends and relations. And you engage in an iterative process over and over and over again until you succeed, and you find yourself at the North Pole.

And Eeyore is a little bit skeptical about this. He says, well, you can call this you know, expo- whatever or gathering nuts in May. It's all the same to him.

And at the of the day, I actually think Christopher Robin and colleagues here have a certain set of advantages over Tomer and colleagues. And that is that it is a Hundred Acre Wood. And so if the North Pole's really there, and they engage in that process, and they have all of rabbits friends and relations, they probably will find it.

But it also turns out that, unlike Tomer and colleagues, they do know something about what they're searching for. Now, they're wrong about this. But nonetheless, I think they access an important constraint on the search process and actually helps them out quite a lot.

So that's what I'm going to try to talk to you about today. So here are the issues with stochastic search. I think there are two big ones. Grant everything-- grant grammar, grant prior knowledge, grant templates, grant a bias towards simplicity.

The problem with an infinite search space is infinite is a very, very, very big space to search. It's a very big space to search. And children seem to do remarkably well with it.

So I'm going to, I guess, give you a toy fictional example. I'm going to give you a non-fiction example from my child. We were riding on an airplane.

She was about 3 and 1/2 at the time. And she knows a lot about airplanes. She has a lot of folk physics, prior knowledge about airplanes.

And she also knows a lot about phones. She's had a lot of experience of phones. But nothing in her prior knowledge predicted the announcement that you have to turn off your cell phone

when you fly a plane, right? So this was surprising.

And she immediately said, well, I know the answer. I know why you have to do that. And I know that she doesn't know anything about radio transmission or government bureaucracy. So I said, how do you know? Why do you think?

And she said, well, because when the plane takes off, it's too noisy to hear. You know, that example is not especially clever or especially adorable. But what is really, really interesting about that answer is that although it is wrong-- it's even wrong as to the causal direction-- it is a good wrong answer compared to all of the other things that are consistent with her prior knowledge and the grammar of her intuitive theories that she didn't say.

She didn't say, because airplanes are made of metal and so are phones. Because airplanes fly over the Earth, and the Earth has phones. Because airplanes are big and phones are small. Because airplanes-- her grandfather lives in Ohio and has led her to believe that everything is made in Ohio. Because airplanes and phones are both made in Ohio.

Infinite is a very big space. There are a lot of things that you could say consistent with prior knowledge where you're making random changes in your intuitive theories that are not even wrong. They're not even wrong.

And so the real question is, how did she converge at a good wrong answer, at an answer that, although it is wrong, makes sense? It isn't just, I think, a toy problem. But here is the problem.

There are innumerable logical constitutive causal and relational hypotheses consistent with the grammar of intuitive theories. How do we so rapidly, literally between the announcement and the next thing out of our mouths, converge on ones that, if they were true, might explain the data? They might not be true. But if they were, they could work.

They could solve problems. And that I think is the really hard mystery. And again, it's not just a toy problem.

Modeling even relatively simple well-understood problems takes time. I would often come across Josh's students wandering in the hallways. And I say, what are you doing? They're like, oh, I'm waiting for my model to run.

I'm like, waiting for your model? Computers are really fast. They're fast information processing. What is it doing?

Well, it turns out it's generating a lot of answers that aren't even wrong. That's what it's doing, right? It's spending a lot of time sitting around sifting through things that aren't even right. Iterations are spent searching in hopeless places.

And this is true of some of the best and the brightest. This was like a fantastic NIPS paper, a major advance. It's a probabilistic graphics program. And it's solving the really deep theory problem of CAPTCHAs.

So it's trying to figure out in this case-- but it doesn't just do-- in fairness-- well, let me return to that. Let me show you what it does do. There's a rectangle over there.

It wants to be able to figure out what is in that space. And it wants to model it. It wants to find a rectangle in the lower left-hand corner of a scene.

So what does it do? It generates pixels all over the map until it finds the rectangle. And I saw this in [INAUDIBLE]. And I said, why is it looking all-- can it at least confine its search to the lower left-hand corner?

But, of course, the algorithm doesn't know from lower left-hand corners. It's a powerful algorithm. If you wanted to solve a CAPTCHA, you could do-- see look at it. It's all over the place.

Now, this is a virtue. And, yeah, it converges. And that's just great, right? But why doesn't it search in lower left-hand corner?

Well, the answer is it doesn't know from lower left-hand corners. And it's a feature, not a bug, that it doesn't know from this. Because that means it doesn't just solve CAPTCHAs, which you can do with edge detection.

It can find, you know, objects in the road. And it can do all kinds of other things that I'm sure Josh and Vikash would be happy to talk to you about. It's a pretty general thing.

But it's not constrained. And that's a feature. But the interesting thing about humans, including human children, is they are both flexible and constrained. They can both solve a whole bunch of problems, and they can converge on them very quickly, right? Whereas, here, it trades off.

And, again, that is a simpler problem. That is a square and a bunch of pixels. The kind of learning we're talking about when we're trying to talk about theory generation or real world

learning is a really, really, really, really big space.

So one problem is just how are you going to get at least to answers that, if they were true, might work? But if one problem is that the theory space is really big, the other problem is that human learners are not that dumb. We have a lot of knowledge.

And we have a lot of knowledge that these algorithms are not making use of. And the question is, why not? And is there any way that we could develop models that did make use of it?

In particular, we know a lot about our problems. Our problems are actually our friends. We know about our problems and our goals. And we know about our problems well before we can solve those problems.

An abstract representation of what the solution might look like, what it ought to do, what the criteria it's trying to satisfy are, could help constrain and guide the search. It matters about it though, not just that she had prior knowledge about airplanes and about phones, but that she had prior knowledge that the problem she was solving was an unpredicted incompatibility between airplanes and phones. You have to turn one off when the other is going on.

That's information that's not in her general background knowledge. It's about the particular problem that she has. And the question is how could you use that to make good proposals and make better proposals?

So the proposal I have is that when you know something about what you're looking for, then that can help you find it. And this is the kind of knowledge that Christopher Robin and colleagues had that Tomer and colleagues did not. They at least eventually decided that what they were looking for was, of course, a pole. They know what poles are.

Therefore, when they find a pole, they can be quite confident that here they are. And this is a good candidate solution to their problem. That solution is wrong, but at least it's wrong.

OK. So the argument here, which I'm going to try to get slightly more precise, is that the form of the problem as an input to the algorithm should increase the probability that proposes useful ideas. And you can consider this even in the simplest form in the kind of information that is contained in our question words.

So I think it's an interesting feature about human cognition that we have a very, very small handful of question words, which we use to query the entire universe. And you know what?

Those question words do a lot of work for us.

When I tell you I'm asking a question about who, you might propose that we ought to be looking for some kind of answer that's something like a social network. And a social network might be more likely as an answer than a 2D map. Whereas, if I ask you a question about where, well, you really do want to consider 2D maps.

If I'm asking when, you're talking about a time line answer. If I ask you a why question, maybe it's a causal network. And if I'm asking you a how question, maybe it's a circuit diagram, right?

You don't know anything about the content at this point. I could be asking you anything. But I ask you who was Christopher Columbus, and you answer 1492. That's the kind of thing that our algorithms are doing. That's not even wrong, right?

It's consistent with your prior knowledge. And it's the kind of thing Watson does as good [INAUDIBLE] solutions. But it's not what children do. It's not what children do.

I think that the issue is that this is actually a friendly amendment, right? Because in what we have shown time and again-- by we I mean not me, but computational modeling folks-- is that we can use lots of information out there for hypothesis evaluation, right? Once we have a theory and we have the data, we can select and use this information and say, well, does it answer the problem or not, right? Does it improve? Does it make better predictions?

So we use this kind of information that we have. Even formally, we can say that we can use it to select among hypotheses. We can use information about the structural form of problem to represent them.

The question is, can we use the same kind of information to constrain the search space? And it's easy for me to say. I don't have to do that, right? But it's the kind of proposal that I think is missing.

Because we have rich constraints that go far, far, far beyond our question words. The kinds of problems that we have and the criteria for solving them derive from all kinds of sources. We try to solve different kinds of problems-- navigation in some cases, explanation in other cases.

And some of those are epistemic ends. I want to persuade you of something. I want to instruct you in something. I want to deceive you in some ways.

But we also have all kinds of non-epistemic goals. I want to impress you. I want to soothe you. I want to entertain you. Each of these goals is actually a constraint on what is going to count as the solution.

Our goals are innumerable. But there are only a small handful of ways you can solve any given goal. So when you're dealing with an infinite search space, having a goal, having a problem, actually could act as a constraint on how you search for the solution.

And it is an interesting feature of human cognition that our goals can be noble or venial. They can be impressive or trivial. And it may not matter with respect to the solution we have.

We have analytic logic, because the medieval monks wanted to find incontrovertible proof for the existence of God. We don't hold onto their goal, but we hold onto their solution. We are here in the East Coast of Massachusetts, because of the search for the West Indies, right?

So our goals act as constraints on the solution and on the search process. And the importance of our goals may be that they do exactly that, that they help leverage some new search in a way that at least helps us make progress. So the argument here is instead of stochastic search, that we have-- I don't call it goal oriented. I call it goal constrained now-- goal-constrained hypothesis generation.

And the idea here is that at least we know something about where we want to go. Now, this is not a total argument against stochastic search. It's just a way of getting stochastic search into a much smaller search space, right?

Once you know what things count, then you can do everything Tomer says you do. I actually agree with him. But you don't want to do it over all possibilities, because you know a lot more than that, right?

You know what kinds of things are going to count. You should do it over that space, not over-- should look in the left-hand corner. Then you can iterate all the pixels you want. But if the thing's on the left-hand corner, that's where you ought to be looking.

I'm going to give you a corollary to this, which is if you don't have any idea what the search space is, you are going to resort to an extremely inefficient, extremely frustrating search and, actually, the kinds of conditions under which I think human beings quit. So I will give you an example from my personal experience, as Jessica Sommerville knows. Because we were trying to get my child's booster seat-- because children now need booster seats until they're

14-- re-attached from my plane flight.

Couldn't do it, right? One thing goes down, two things go up. It's a spatial problem. We spent 10 minutes, two PhDs, on it-- threw the thing out and just had her ride in the bottom of the booster seat without the top.

If I have 1,000 piece puzzle and I have to find a puzzle piece, people who are good at puzzles know before they find that piece something about what it's going to look like. Like, OK, well, the edge has to be angled like this. It has to have a concavity here and a convexity there. And that's what I'm looking for.

Me-- soon as I look away from the piece, I have no idea what I'm looking for. And so I do what Tomer would do. I do a stochastic search over all the puzzles, And with 1,000 pieces and many permutations, that is not a good way to solve a puzzle. As a result, I never do puzzles, right?

As entertainment, I just don't understand. But if you do know, it's very satisfying, right? You know what you're looking for. And now you can constrain your search space much more effectively to those kinds of things.

Indeed, when I say we are smarter than that, human beings know. We have metacognitive principles around these kinds of things, right? This knowledge is in human minds, what it might mean for us to think that a problem is tractable. What does that word mean, right?

Sometimes it means we have the financial resources or the technology to carry it off. But often, it means we have a well-posed problem. We don't know the answer, but we know what the answer needs to do.

We know what the answer needs to look like. We have criteria for what would count. At least we have a precise enough representation of the problem to effectively and efficiently guide the search.

And I think that's the kind of thing we would ideally like our models and algorithms to have. At that point, we may have to bounce around a lot. But we're bouncing in a pretty well-defined space.

So to the degree that this is true, it actually explains a lot of otherwise peculiar features of human cognition. For instance, we had this weird sense that we're on the right track, right?

Well, what does it mean to be on the right track?

It surely doesn't mean you're better at explaining the data, right? You may be nowhere close to having an answer that makes better predictions or gets it right. But you're like, oh, that's a really good idea, you know? You get excited. Or you're like, no, that's a non-answer.

What does it mean? It might mean that at least it fits the abstract form of solution. If it were true, it might work.

We can tell our students in an undergraduate class that that was a great idea even when we know it's been disproven, right? So it's actually false. It doesn't explain. It's just not true.

We still think it's a great idea in virtue of what? Not it's fit to the data, right? Not how well it's predicting things.

It's actually false and still good. What could that possibly mean, right? It must mean there's some other constraint on hypothesis generation that we are sensitive to.

So I want to suggest that there are actually two constraints for our hypotheses. One is how well they fit prior knowledge and data. That's the one we know something about. That is, for instance, truth.

But Stephen Colbert in his infinite wisdom proposed something else. He said, well, we're also sensitive to this thing called truthiness, right? You know, like how good the story sounds, how good the argument. And of course, in politics, you know, he makes massive and effective fun of this.

But I think this is a feature of human cognition, not a bug. I think it is extremely important that we can generate ideas that are truthy, right? Because they're plausible.

They're interesting. They're informative. They tell good story. They may be false, but it could be worse than false. It could be not even false.

So I want to make an important point here. Generating new ideas is not just about Einstein versus Newton. And it's not about going from an undifferentiated concept of heat and temperature to a modern scientific one. It's just about radical conceptual change.

This is the stuff of ordinary everyday thought. It is our ability to reliably make up new relevant answers to basically any ad hoc question. The answers may be trivial.

They may be false. But they are genuinely new. And that we didn't have them until we thought of them, they're genuinely made up.

We didn't learn them from evidence or testimony. And they answer the question. They're not non-sequiturs.

And I think this is important. And this is only possible if we can use the form of our problems to guide search. So let me give you a few examples.

What's a good name for a theater company? None of you know. You haven't thought about that problem before. But now you're thinking about it. Well, what's a good name for a theater company?

How do you get stripes on peppermints? This is not a problem you walked in thinking about. None of you are working on this as your independent project. But you can think about it.

And you already know enough, knowing nothing about theater and nothing about peppermints maybe, to know what the constraints, what counts, right? That's the kind of information you already have. It's not prior knowledge about theater companies or peppermints. It's part knowledge about what's going to work for a solution.

So for instance, you know that McDonald's is a nonstarter. And [INAUDIBLE] or whatever is also not a good name for a theater company. You know that getting strips on peppermints you don't want to do it with a spatter [INAUDIBLE].

You don't want to just spray things at it, right? Because that wouldn't even count as a solution. That's not the kind of thing you're looking for. You're looking for something more like fresh ink.

It's new. It's novel. It's familiar. It makes some reference. You're looking for something more like a pendulum approach, which at least could generate periodicity.

I'm sure they don't use a pendulum to spray paint peppermints. So are you. But it's not a bad answer, right?

So is there any evidence that kids can do this, that information contained in only that strict form of the problem can help learners converge on solutions? We wanted to find out. So I'm going to show a little baby attempt to start getting at this problem.

We gave kids a machine. And we gave kids some things that could work the machine. And the machine had two visual effects.

You could make a ball appear to flow up and down on that screen there. Or you could make the ball appear at the bottom and then flash up at the top, right? Because we put a computer behind, right?

So woo, the ball is moving continuously, or the ball is moving discretely. And the affordance, as you might note, are also continuous or discrete, right? There's a rolly ball, or there's this peg that you can move back and forth continuously. Or there's a peg you can pull in and out, or a drawer you can pull in and out.

So there's continuous and discrete affordances and also continuous and discrete effects. So we also had an auditory tone that varied continuously and an auditory tone that went from high to low. Everyone got it?

And we just showed the children all the parts that connected to the machine. And then we showed the kids the effects, but we had hid the affordances. And we said, well, which part made the ball go? And we asked them either about the visual or auditory affordance?

They'd seen no covariation data, prior knowledge- you know, agnostic about all of this. And the question is, well, would they say, well, I'm trying to solve a problem about a continuous effect. I should use a continuous affordance.

I'm trying to solve a problem about a discrete effect. I should use a discrete affordance. Would they use something about the form of the problem to answer the solution if they knew nothing else about it?

So there's no fact of the matter here, right? Because, obviously, we're not using either of these really. But the prediction was that they would indeed make this kind of mapping. And they did so.

Now, what you might worry about was that, well, there is no fact of the matter. So they're just doing some kind of cross-modal mapping, right? If you don't have a way to answer the problem, they're just saying something like, well, you know, this has this property, so does this. Let me go ahead and make a mapping from one to the other.

It's a weird kind of cross-modal mapping. Because, usually, you integrate things that are in a

single stimulus actually contained together, like the sound of a ball dropping and the sight of a ball dropping, not two different things. But maybe it's something kind of like that.

So if they're actually using the form of a problem, then if you change the problems, they should generate different solutions. So what we did in the second experiment is we said-- oh. Yeah. So what we did here is we showed the children the continuous visual stimuli, and then we asked them to generate the continuous auditory stimuli.

OK. So all of these are now continuous. They could still just go ahead and make the continuous map. But if you represent the problem as changing from visual to audition, that feels like a discrete problem, right? You're completely changing modalities. So now at this case, you might expect the kids to resist making just the continuous mapping if they're using something about the kind of problem they have to constrain how they search for the solution.

Now, again, there's no real right answer here. But what you see is the kids shifted their responses in response to this. So this is just a tiny bit of suggestion that when there's no differentiating prior knowledge and there's no differentiating evidence, children take into account what kind of problem they're trying to solve, and what the information is, and the problem itself that can help constrain their search for a solution.

I'm going to give you another example for some more recent work by Pedro Tsividis in our lab. He varied the dynamics of a scene. He had here some bugs.

And in one scene, those bugs in green, they varied periodically. So they just went from having very few spots to having a whole lot of spots to having a very few spots to having a whole lot of spots, right? That's what these green bugs do.

And the other bugs just got faster over time. So those longer vectors are supposed to be indicating the bugs are getting faster continuously. And then he said, well, you know, here are some bugs in these rooms. And I have two kinds of lights in the room.

One set of lights looks like those on top. The other set of lights looks like those on the bottom. Can you tell me which lights are responsible for the behavior of which bugs? Again, a sort of very similar kind of problem.

No fact to the matter that if children can represent something about the abstract form of the problem and use that to constrain their search for the solution, then the periodic lights should reflect the periodic change in the bug's behavior. The continuous light should reflect the

continuous change in the bug's behavior. We are, of course, not using the words periodic or continuous with us or anything like that.

They have to pull that out and say, this is the kind of problem it is. This is a feature of a possible solution. Let's go ahead and make that mapping. And indeed, the kids are doing that well above chance.

And obviously, in this case, we're giving the kids some possible solutions. They're not generating it whole cloth. But minimally, it's a different way of thinking about problems and search. They're not using most of our sort of traditional ways of figuring it out. They're just using something about the kind of problem they have and what's available in the problem to help sort out the solution.

So is this analogical reasoning, right? It feels kind of an analogy. But it's a funny kind of analogy.

Because what it isn't is a mapping between a known problem and a known solution to a new problem and a new solution. It's rather a mapping from the kind of problem you have to the kind of solution you have. It's using, again, the problem or the query itself as your friend. It has information in it about how it wants to be solved.

How are you going to use that to solve it? And, again, the virtue being that even if your answer is wrong, if it were true, it would work. So the argument also, of course, is that this applies to any possible goal we might have including those cases where it's just not at all obvious how an analogy would apply.

So what is a good name for a theater company, right? You're using something about what would count as a solution to constraint something about what you think would be a good answer. But there's not an easy way to tell that story as analogical reasoning.

So their argument is children seem to have data independent criteria for the evaluation of hypotheses. And these criteria extend beyond simplicity, grammaticality, or compatibility with prior knowledge. They consider the extent to which a hypothesis fulfills the abstract goals of a solution of a problem, not just the degree to which it fits the data.

And I will suggest that this is maybe deeply part of an important mystery of human cognition, which is-- our most powerful learners spend a lot of their time doing something that has defied

most of our best attempts to explain it, which is they've spent a lot of time, many hours a day, just pretending and making up stories. Stories have some interesting properties. They do not have to be true, right?

They don't have to fit the world or fit the data. But they do have to set up a problem and a solution that, if true, would solve the problem. Most of play has that.

It is not at all obvious why you would think it was important that you want to balance a twisty on the top of the candle stick in order to shoot pee through it. But it's a problem. And guess what?

If you can set up the problem and find the solution, you've just accessed a really important ability about setting up problems and setting up solutions that, if they worked, would solve that kind of problem. And, of course, imagination, most of our narratives, most of our fictions, have at least those sets of properties. So I'm going to go ahead and stop there.

[APPLAUSE]

TOMER ULLMAN: OK. So if you remember the structure we said at the beginning, there will now be a short rebuttal. And then Laura will give an even shorter summary, and then the free for all. So most of you are probably thinking all sorts of ways that Laura is wrong.

But wait, let me get through it first, and then see if I didn't cover something. But, actually, my response to this, to all of what Laura's said, is not you're wrong, but you're right. So you're right, Laura.

You're right. Other people in the audience who think that stochastic search by itself-- if you have some sort of infinite theory space that was supposed to account for all possible problems and for any new problem what you did was to just completely at random try to search through that space anew, then that wouldn't work-- that's fine. And I agree that there's something inherently wrong about an algorithm that can take some problem, like why these two blocks are sticking together, and say, well, maybe it's because the moon is bigger than a piece of cheese, right?

Like, as Laura said, it just seems like it's not even wrong. Or maybe it's because people have more than two children on average. No. And there's also something wrong. Laura didn't quite get to this or maybe not emphasize it. But she emphasizes it sometimes, which is there's something wrong about-- well, she did a little, but-- an algorithm that makes sort of dumb proposals.

Dumb proposals of all sorts of things-- things like you try to explain something in theory space, and you say, well, maybe it's because of X. And you check it, and it's not X. And you say, oh well, oh well. Maybe it's because of X, right? There's something wrong about stochastic search.

Although, I have to say, Laura, you have an eight-year-old. And, you know, when we gave this first, I had a two-year-old. And I actually think maybe it's X, maybe it's X, maybe it's X is not such a crazy way of describing what two-year-olds are doing.

LAURA SCHULZ: Well, [INAUDIBLE]. So, you know, like, what if there's noise?

TOMER ULLMAN: Yeah. Yeah, yeah. OK. But I do want to give an actual response. So, you know, I think I have some responses for Laura.

And these are responses that, importantly, you know, they'll try to address what Laura is saying. They'll try to take it into account. They'll try to give new answers to it that will importantly leave her unhappy.

So what I'm going to try to do is take a page out of Hannibal Barca at the Battle of Cannae and try to envelop. So I'm going to highlight of work by other people. From one direction is work by Steve Piantadosi, which is about making these, you know, algorithmic search part of the problem, whether it's too slow, students are wandering around the halls doing nothing waiting for it to converge. What if we just did it really, really fast?

Another way to address this is to say, what if we made actually good proposals? OK. The problem of making proposals that are just ridiculous-- what if we made proposals that actually take the data into account a little bit or the previous data, the stuff that we're trying to explain? Another way to address what Laura is saying is to say, well, maybe we can make better primitives.

And better primitives mean that your search space is actually more confined to the right sort of things. And finally, I'm going to highlight some new thoughts by myself about ad hoc spaces and how we might construct them to get at this problem of this thing, like how would you come up with a new name for theater company or a name for a new theater company? So I'll go through these somewhat fast.

You're welcome to come and talk about any of them. And I'll point you to the people who are

actually doing the work, which I've said before. So let's see. This is just sort of the rebuttal.

One of the rebuttals is to say, well, here's this work by Steve Piantadosi, which is, you know, introspection is really actually a poor guide. So when [INAUDIBLE] was giving that beautiful example of cell phones, why you have to shut them off on airplanes, you say, oh, and she came up with this example that it's not true. But if it were true, it would be nice.

Well, maybe she went through a billion other things before she came up with that? Laura's like, she did it like that. But we don't have a good sense for what is actually fast, what is actually slow.

And there might be a case where we don't actually introspect about a lot of things. The things that bubble up into consciousness that you might actually accept or reject rely on actually a ton of other proposals that they don't even bubble up into introspection that you're making very quickly and just rejecting even before that. And Steve came up with this really interesting way.

You guys have probably heard about deep learning and the sort of the GPU revolution for deep learning and things like that. So the point is if instead of we use CPUs we would use graphical processing units, then we can make stochastic search algorithms in parallel in some cases. And once you can make them in parallel, then you can put them on a GPU.

And once you put them on a GPU, a GPU is sort of a way of taking something that's supposed to be a CPU. And instead of having a CPU that can do something sort of complicated on a sort of task, you can do a lot of really simple tasks in parallel very fast. So if you could make that proposal, that thing I said before-- take a theory, make a change to it-- if you could make that into the sort of thing that you could put on a GPU, then you can make a ton of those proposals very quickly in parallel.

And that's sort of what he figured out how to do for a bunch of spaces. It's much faster than the CPU. And the main advantage is that it's also much cheaper. And you can cram a whole bunch of together.

And you can get to something like-- I forget the exact numbers. You can make like a million of these theories proposals a second. And that's just with today's technology, right? We don't know what's coming around the corner a few years from now.

You know, Steve plus GPUs is awesome. And you could think of it like these various problems

like you're trying to fit these data points on the bottom. Can people see that?

This is sort of a classic problem. You have some data points. You're trying to fit a polynomial to it. And you're trying to say, well, how will we do that?

The truth is there's a lot of very clever ways of doing that. But let's assume that you're even doing random search in polynomial space-- not the sort of thing you want to do. Those of you who have been to the tutorial, I mentioned that if you have an actual better way than stochastic search, you should probably do that.

But suppose you didn't know and you wanted to do stochastic search. You could still do a million moves a second and quickly converge on something like that line that you see. OK. And that line is actually taken from, I think, Galileo Galilei's data for how things slide on a hill.

I'm not Galileo Galilei sat around and said, maybe it's x to the square, maybe it's x to the 2.1, maybe it's x to the 2.3, maybe it's x to the 2.1, maybe it's x to the 2, and then finally converged like after a million moves to that. That's not exactly scientific discovery. But for a lot of everyday thinking, you might actually be proposing things very fast and rejecting them. OK. That's Steve.

Some things from Owen Lewis about making maybe smarter proposals-- and this gets at that point of, like, maybe it's an x, maybe not. So I suppose that I'm trying to teach you a concept. OK. I'm trying to teach you a particular concept.

I'm going to give you some positive examples of the concept. OK? This is the sort of thing that psychologists really like to study. OK? So this is a room-- no, Roomba's an actual thing. Blick gets overused.

Can someone give me a nonsense term? This is a Jabberwock. OK? This is a Jabberwock. I'm going to give you another example of a Jabberwock.

Who thinks they know what Jabberwocks are? You have some sense of what a Jabberwock is? OK. Huh, that's also a Jabberwock. Wait a minute.

OK. The sense that I had for maybe what a Jabberwock is is maybe not that great. That's a Jabberwock. That's a Jabberwock. That's a Jabberwock. That's a Jabberwock.

OK. And you might think at this point, well, fine. Jabberwocks are either squares of any color

or red circles. Or maybe they're squares or circles. I don't know exactly.

You're building up some sort of theory for that concept, which can be described in something like a grammar for your current hypothesis. You might say it's either a red circle, or it's the square of any color. OK. And that's sort of your grammar for these concepts.

And now you could sort of change that grammar, right? You could sort of excise these nodes and that tree to come up with new things. Why would you want to come up with new things?

Because, look, that's a Jabberwock. OK. I just gave you a new example. It's something you didn't know before. It's sort of confounding with your theory. You have to come up with a new theory on the fly. OK?

Theory-- again, used in a very, very minimal sense here. But if you accept that this is something like a theory, you have to come up with a new theory for explaining why that's a Jabberwock and all the other things that you've seen are Jabberwocks. What is a bad idea?

A bad idea is to just cut and generate randomly, right? You might come up with something like it's a triangle or something like that. But you might come up with, well, maybe it's a square. No, we already did that.

Well, maybe it's, you know, just triangles. Maybe it's just a square. Like, you could spend all this time not taking into account your previous theory and the fact that your new example had something to do with triangles, something to do with red triangles. You want to be able to make proposals that take into account this new data. Does everyone understand what the problem we're trying to get at is?

So what Owen has done is to sort of take these stochastic search algorithms and say, if you get a new piece of data that contradicts, that sort of interferes with what you had before, how would you make proposals that must take into account this new data? I'm going to recut and generate.

But I'm going to identify the places in my theory that would take into account this new piece of data. And I'm going to make smarter proposals. They might still be wrong.

And there's better and worse ways of doing this. It's still going to be a randomized search and theory space. But it's at least going to take into account this new data.

OK. And that's just-- I'm afraid I don't have time. But, look, it works. And it's much better than just bouncing around completely around random as you might guess.

Another response, this work by Eyal Dechter. Let me skip over this for a little bit. This is work Eyal Dechter, which is to say, what if you wanted to use better and better primitives?

OK. So before we have this notion of you're just bouncing around in theory space, you're making all sorts of notions, let me put it this way. Suppose that you're searching through the entire space of programs. OK?

And the only thing that you had to work with is something like that lambda expression before, right? You didn't have the notion of plus, minus, multiplication, sine waves, things like that. And you're trying to figure out something about an equation.

And you work through it. And there's a way of doing it. There's a way of, like, generating functions that rely on other functions that rely on other functions in the complicated way that will give you the plus function. OK?

And you do that. And then you generate a lot. And you somehow manage to find out the sinus function. And you finally figure out that this function you're trying to describe is sinus x plus sinus y.

Let's say something like that. OK? The exact example doesn't matter. But you worked really hard, and you figured that out.

Now, you get a new example. And underneath the hood, it's actually just sinus x minus sinus y. Or let's say it's sinus y plus sinus z or something like that. And now you start all over again. You're like, fine. OK, lambda something, something, something.

Like, if you could only use the fact that you've already discovered the sign function, you've already discovered plus and minus and things like that, and now when you come to try and explain a new problem, you actually have a lot of previous knowledge. OK? So when you're trying to describe why airplanes take off and how they do that, you're actually going to rely on previous knowledge. You're not going to search through your entire theory space starting from nowhere.

You're going to rely on primitives before that have been useful. So you might notice that actually plus and minus and sines and things like that and cosine and exponentiation are really

useful. Let's save those as primitives.

So that next time that we make random proposals in theory space, you can think of it of making like a whole bunch of moves at once that were useful. OK. Like, a whole bunch of stuff at once that was useful that shows up all the time-- you want to make that again. OK? So I try that, for example.

And the examples that we gave in theory space-- like you might find out that actually a lot of theories use transitivity, or a lot of theories use reflection, right? In the particular magnet case, if X attracts Y, then Y will attract X, in general, the law of if X blahs Y, then Y will blah X, that turns out to be useful in a lot of domains if only there was a way of reusing them, right? There is.

And what Eyal was doing was to basically use an explanation compression algorithm, the EC algorithm. And what it does is it tries to encapsulate useful concepts. And he used it on a whole bunch of stuff.

One of the nice domains he used it on was these circuit diagrams. Have any of you actually had to solve circuit diagrams? This is the sort of stuff that people at MIT do. You're given a particular input-output function.

And again, like I said, under the hood it might be something like you're just told something like here's X and Y, OK? X and Y can each be 1 or 0. OK? And now I'm going to give you combinations of values of X and Y and what that spits out.

OK. So X and Y are both 1-- light turns on. X and Y are both 0-- light turns on. X is 1, Y is 0-- light turns off. OK. And you're trying to find out some sort of circuit that will explain this behavior, some sort of combination logic gates, like ANDs and ORs and NANDs and things like that. OK?

Do people sort of understand the problem for these circuits? And you can get a long list of things, X, Y, Z, T, some sort of complicated behavior. You might not even get the full behavior. And you're trying to find sort of the minimal set of logical predicates, or in this case circuits, that will explain that behavior.

And now suppose that you only have the gate NAND to work with. Do people know what NAND is? OK. It's a sort of logic gate.

It's a very simple logic gate that you can build up all the other logic gates from. But it will take you a while to build up AND from NAND or OR from NAND. But you can do it.

And so what he did was, and his colleagues, they started out sort of giving this algorithm a lot of different problems. Like, here's a bunch of circuit diagrams that you need to solve or circuit problems that you need to find the diagram for. What the algorithm was doing was that each time it solved a problem or set of problems, it would go back and look, huh, which parts of this can I encapsulate, right?

Which parts of this can I sort of use again? I can carve off a chunk of something that was useful. And now, when I make a new proposal, I'm not going to say put a NAND here or a NAND there.

Stochastically, I'm going to sort of put in a whole chunk that I've already used before. OK? I'm going to sort of call that a new primitive. Cut out this part of the space. Under the hood, it's actually an AND or something like that.

And discover-- so discover is not an AND. And discover is this really useful thing that they called E2, which doesn't appear in logic books. But it turns out to be really useful for certain diagrams, which is take an input, split it into two, do something on this part, do something on that part, and recombine it.

It turns out to be a hugely useful concept for circuit diagrams. And this thing discovers it. And once it discovers it, it sort of reuses it. And what that does is it turns an infinite and unmanageable space into infinite, but a bit more manageable. OK?

So your space is infinite. You're not going to search the full length of it. Imagine that this is a space of all possible programs. As you go down, the programs get way too long. You're never going to reach them.

But some of them are really good. Some of them are really good explanations. And the only way to get to them is if you had some sort of way of chunking the problem, of saying, yes, it looks like a long program. But actually half of it I've already used before to solve a different thing. And half of it is less long than two times.

So you might discover, you know, you might have an effective search area. You find out all the problems you can solve there. Yeah, this is an interesting thing, choice color.

So imagine that of this blue thing over here is describing, within the space of all possible programs, the sort of programs that you want to find. So there's the effective search area. They only cover part of this blue thing.

You can think of it like the probability is really high over there. You really want to find all of them. But by searching that small space and, within that small space, finding the right primitives and encapsulating them, you can now actually search more efficiently the rest of the space.

And the rest of the space sort of compresses and compresses until it's all within your effective search area. Do people sort of understand that? It's sort of there were long programs before that you never would have gotten to. But by searching these small spaces that you could search through before, discovering the useful parts there, these new things that seem really long before are actually short. Because they can be described by just a few chunks. OK?

This is a really interesting work. And I encourage you all to read it, those of you who find it interesting. The last thing I'm going to do-- and then that'll leaves 10 minutes to discussion, which is great-- is this problem that I guess maybe it's really the heart of what Laura is getting at.

I think she was not satisfied by any of these things. And she was sort of pointing out, well, fine, you can do stochastic search all you want. But the really hard problem is constructing the space itself on the fly.

You're not going to use one infinite space for all possible problems. You're going to use the right spaces for the right problem. And how do you do that?

In this case, we're going to do, give me a good name for a romantic drama. All right. And your search space is going to be imagined that-- can people see sort of the border? Like, there's this whole space of uselessness.

And what you really want to do is focus in on that tiny part of useful things. If only there was a way of just on the fly, you know, zooming in on that thing and then bouncing around in that. And the point is to say, well, when we construct the space, we can just use previous examples.

I don't think it's the case that we just knew something necessarily completely new in these sort of everyday thinking. Well, maybe. We can argue about that in the discussion.

What you actually start out with is actually taking a few examples that you find relevant in some way and using those examples to then construct your space on the fly, right? You might think about things like, what other romantic dramas do I remember in the past? What do they share in common?

What movie names do I know of in the past-- quickly finding the sort of relevant thing for all these things, and then having the space for those, and then searching around stochastically. Because you're not going to do better than stochastic search. There will come a point where you're just bouncing around at random.

So I used this actually, forgive me, a paper title for SRCD and came up with some amusing things. You guys can play with that online if you want. But let's do, give me a good name for a new romantic drama.

So as I said, what you would do is you would just think about all the romantic dramas that you know, like *The Climbers, Christine* of *The Big Tops, Cupid's*-- these are all actual romantic dramas pulled off of Wikipedia-- then use those to construct your space. Don't care about all the things that could happen in the world. OK?

And what do we mean construct your space? Well, there's a bunch of ways to look the space. What ideally we would want and what I didn't do, but what we're thinking of, is to construct a very, very simple grammar which instead of all possible sentences is a grammar for movie titles.

And this grammar usually tends to generate things like the, right? The something something. And [AUDIO OUT] long. And then it just stops. OK? And it turns out that something like the adjective noun is a really good way of generating names for pubs-- The White Queen, The Blond Tiger, The Bleeding Bottle, I don't know, something.

Right? That's really useful if only it could do that. If it could construct these tiny grammars, it'll still give you an infinite number of things.

But, you know, [AUDIO OUT] movie names. Or things like verbing proper name turns out to be a really good thing for like, you know, *Amy Stopping, Interrupting Timmy.* It's so bad. And you could find that from looking at these things.

And just to show you how much I think that this is, you know, actually not that bad of a problem, I did not this grammar thing. I did something even simpler, which is to take all the

other names that I could find on Wikipedia for different movie genres throughout the ages, and then I looked at things like romantic dramas. And what I did was construct a very simple n-gram which just takes those words and just sort of does random walk on those words. OK?

And you could imagine complicating this immediately by taking something like embedding those words in the high-dimensional space and actually picking words that aren't close to that. So you could get new words that were never in there before. I'm not even doing that.

I'm doing something ridiculously simple that I don't think people are doing. But let me show you how reasonable it is. OK? And what I'm going to compare it to is some stuff that we ask people on Mechanical Turk to give us names for a new romantic drama.

Ah, the only thing I forgot was the right labeling for these things. So Laura, what do you think? Is this from Turk, or is this my algorithm?

LAURA SCHULZ: I am 50/50 [AUDIO OUT].

TOMER ULLMAN: So how about we'll have by, not show of hands, but people just shout it out. Like, if it's, I don't know, Turk or-- I'm looking for a short word which is like a Turk or Tomer. Let's do it that way, so Tomer just standing in for Tomer' simple silly algorithm.

So who thinks that this was created by someone an actual human on Mechanical Turk? And who thinks it was created by Tomer mechanically running through an algorithm? OK. So in 3, 2, 1 you're either going to shout Turk or Tomer. 3, 2, 1.

[INTERPOSING VOICES]

TOMER ULLMAN: OK. That was actually someone at Mechanical Turk. Let's do this again. *Girls In Ships* for a romantic drama, 3, 2, 1--

AUDIENCE: Tomer.

TOMER ULLMAN: This was an algorithm. *Value Of Love,* 3, 2, 1--

AUDIENCE: Turk.

TOMER ULLMAN: That was Turk, good. *Endless Love,* 3, 2, 1--

AUDIENCE: Turk.

TOMER ULLMAN: Good. OK. How about *Legend of Paris?* 3, 2, 1--

[INTERPOSING VOICES]

TOMER ULLMAN: Nobody knows. This is actually me. OK. Who's enjoying this and wants to do a few more? *Land of Roses,* 3, 2, 1--

AUDIENCE: Tomer.

TOMER ULLMAN: Tomer. And finally, *Those We Meet Again,* 3, 2, 1--

AUDIENCE: Turk.

TOMER ULLMAN: No, it wasn't me. Oh, sorry, one more. *Love Lightly,* 3, 2, 1--

AUDIENCE: Turk.

TOMER ULLMAN: Yeah, Turk. It seems like Turkers were actually doing better than the algorithm, which is romantic is love. And I'm just going to put something with love in the title.

So who wants to do this action movies, and then we'll start stop? OK. Let's do this for action. How about *The Chase?* 3, 2, 1--

AUDIENCE: Turk.

TOMER ULLMAN: Yes, how about *Who, The Annihilation?*

[LAUGHTER]

TOMER ULLMAN: OK, that's me. *The Oversight?* Turk. *The Edge.*

AUDIENCE: Turk.

TOMER ULLMAN: Turk. *Jack Death?* Tomer. *Among Heroes.*

AUDIENCE: Turk.

TOMER ULLMAN: No, it was me. *Swordmen in China Three?*

AUDIENCE: Tomer

TOMER ULLMAN: Tomer. And *The Hit?*

AUDIENCE: Turk.

TOMER ULLMAN: People on Turk. You can probably [AUDIO OUT] than four in each one. And, again, people are like, *The Oversight, The Hit, The Chase, The Edge.* That's the only thing they did. They actually came up with some clever stuff as well.

But, you know, it's interesting. And, of course, I'm cheating. Because the algorithm did a bunch of really dumb stuff that I didn't put in here, like *Hunchback of Monte Cristo, Get it Did, Bell of a Lesser God, Eagles Shooting Heroes, Tomb Raider, The Raging God Of Violence,* and *Legend of Legend,* my favorite.

But my point is to say, you know, in the same sense that Joshua's saying, you know, imagine that you could use something like a ConvNet to quickly cede your proposals-- imagine if you could think of like a random dumb algorithm that could then [AUDIO OUT] and say Legend of something. And then you start to say, no. That's not really a great idea.

What you're trying to get at here is not 100% accuracy with these silly things, but something like 1 in 5. 1 in 5 is better than 1 in 0, or 1 in a million or something like that, which is what Laura was pointing out. OK. So as I said, we still have a long, long way to go to model children to meet Laura's critique.

It's hard to say what's hard. I think that's what I was trying to hint at with Steve Piantadosi's point. We don't really know what's easy. We don't really know what's hard.

But people in development and in computational land should continue to care about stochastic algorithms. And people in computational land should continue to care about children to everyone's benefit. And that's it-- so, Laura.

LAURA SCHULZ: This will be very short.

[APPLAUSE]

So I didn't know. Or rather, I did know. But I didn't know it was going to be part of this debate about Max Siegel's thing. So I'll say something briefly about that.

But you've had three really good approaches to each of these. So I'll speak briefly about them. I think what Owen is doing is totally great, but still driven in some sense by the data, not by the question, right?

And I think the point that I'm going to make just continually here is that the way we think is driven by the goals that we have, right? And each of these solutions in some ways is failing to use what is most salient to it as humans, which is we have problems. We have questions, right?

And what I would like to do is see us move to a case where it's not just the data that's causing us to generate new ideas. And we're not just trying to deal with that. It is actually the information in the problem itself.

Similarly, I think what Eyal's doing is totally beautiful. And the representational compression is really, really interesting. But a lot of learning problems can't be solved.

Most of the ones I was gesturing at are not really problems of changing the representational format. It matters hugely that we have an Arabic numeral system instead of a Roman numeral system. That changes the kinds of problems that we can solve. And so that represents a huge advancement.

And for many kinds of problems, it will make search much more efficient. But a lot of problems just don't have that property. So it's, in some sense, an answer to a different kind of problem.

Steve's proposal-- what can you say, right? It could be true. There are a billion, billion neurons. You get more synaptic connections than there are stars in the known universe. Of course, it could be true.

That's what an expedition means-- a long line of everybody, says Pooh. But it's not as good a story if it's true. So it could be true. You could do a billion things really really fast and just think about the ones that you arrive at. But I think the jury's out on that one.

Max and Tomer and this-- and while ago I think Sam Gershman also, right? So Sam Gershman came up to us and we spent a while talking about how you would invent what we were affectionately calling a bullshit generator, our ability to [AUDIO OUT]. Somebody asked me about anything, you know-- tell me about Ionic and Doric columns, you remember something from sixth grade. And you start talking, right?

So the question is, what can you do? And I think this is a really nice attempt. And I think the idea of seeding it from past examples to help construct a search space is a really beautiful idea.

Again, the question is, how do you make that. My feeling is still we can do something that works for those kinds of problems where we have past examples. We can do it for any kind of problem.

And so what I really want to push for is use the problem, right? Use the information and the problem. Because for those problems, like romantic movies, we have some existing examples.

We can the search space. We can do that. And for my theater company example, it's perfect.

But for the peppermint example, not so much, right? You're not going to see the search space from examples of candies, or what you know about the construction of candies. If you did, it wouldn't generate the pendulum answer, which we think is a good wrong answer.

So it's not just that. It is I have a problem of a particular kind. It is going to be satisfied by some kind of an answer and not others. How can I use that to help my [INAUDIBLE]? So that's I think the end of what I have to say. And we'll return to questions.