JOSH TENENBAUM: So where we left off was, you know, again, I was telling you a story, both conceptual and motivational and a little bit technical about how we got to the things we're trying to do now as part of the center.

And it involves, again both the problems we want to solve. We want to understand what is this common sense knowledge about the physical world and the psychological world that you can see in some form even in young infants? And what are the learning mechanisms that build it and grow it? And then what's the kind of technical ideas that are going to be also hopefully useful for building intelligent robots or other AI systems that can explain on the scientific side how this stuff works?

All right, so that was all this business. And what I was suggesting, or I'll start to suggest here now, is that it goes back to that quote I gave at the beginning from Craik, right? The guy who in 1943 wrote this book called *The Nature of Explanation.* And he was saying that's the essence of intelligence is this ability to build models that allow you to explain the world, to then reason, simulate, plan, and so on.

And I think we need tools for understanding how the brain is a modeling engine or an explaining engine. Or to get a little bit recursive about it, since what we're doing in science is also an explanatory activity, we need modeling engines in which we can build models of the brain as a modeling engine. And that's where the probabilistic programs are going to come in.

So part of why I spent a while in the morning talking about these graphical models and the ways that we tried and, I think, made progress on, but ultimately we're dissatisfied with, talking about how we're modeling various aspects of cognition with these kinds of graphical models. I put up-- I didn't say too much about the technical details. That's fine. You can read a lot about it or not.

But these ways of using graphs, mostly directed graphs, to capture something about the structure of the world. And then you put probabilities on it in some way, like a diffusion process

or a noisy transmission process for a food web. That's a style of reasoning that sometimes goes by the name of Bayesian networks or causal graphical models.

It's been hugely influential in computer science, many different fields, not just AI, and many fields outside of computer science. Not just cognitive science, neuroscience-- many areas of science and engineering. Here are just a few examples of some Bayesian networks you get if you search on Google image for Bayesian networks. And if you look carefully, you'll see they come from biology, economics, many chemical engineering, whatever.

They're due to many people. Maybe more than anyone, the person who's most associated with this idea and with the name Bayesian networks is Judea Pearl. He received the Turing Award, which is like the highest reward in computer science.

This is a language that we were using in all the projects you saw up until now in some form that we and many others used. Because they provide a powerful set of tools for general purpose tools. It goes back to this dream of building general purpose systems for understanding the world. So these provide general purpose languages for representing causal structure-- I'll say a little bit more about that-- and general purpose algorithms for doing the probabilistic inference on this.

So we talked about ways of combining sophisticated statistical inference with knowledge representation that's causal and compositional. These models-- I'll just tell you a little bit about the one in the upper left up there, that thing that says diseases and symptoms. It is causal. It is compositional. It does support probabilistic inference.

And it was the heart of why we were doing what we were doing and showing you how different kinds of causal graphical models basically could capture different modes of people's reasoning. And the idea that maybe learning about different domains was learning those different kinds of graphs structures.

So let me say a little bit about how it works and then why it's not enough, because it really isn't enough. I mean, it's the right start. It's definitely in the right direction. But we need to go beyond it. That's where the problematic programs come in.

So look at that network up there on the upper left. It's one of the most famous Bayesian networks. It's a textbook example. One of the first actually implemented AI systems was based on this for a system for medical diagnosis. Sort of a simple approximation to what a general

practitioner might be doing if a patient comes in and reports some pattern of symptoms, and they want to figure out what's wrong. So diagnosis of a disease to explain the symptoms.

The graph is a bipartite graph. So two sets of nodes with the arrows, again, going down in the causal direction. The bottom layer, the symptoms, are the things that you can nominally observe. A patient comes in reporting some symptoms. Not all are observed, but others maybe are things that you could test, like medical test results.

And then the top level is this level of latent structure, the causes, the things that cause the symptoms. The arrows represent basically which diseases cause which symptoms. In this model there's roughly 500, 600 diseases-- you know, the commonish ones, not all that common-- and 4,000 symptoms. So it's a big model.

And in some sense, you can think of it as a big probability model. It's a way of specifying a joint distribution on this 4,600-dimensional space. But it's a very particular one that's causally structured. It represents only the minimal causal dependencies and really only the minimal probabilistic dependencies.

That sparsity is really important for how you use it, whether you're talking about inference or learning. So inference means observing patterns of symptoms or just observing the values of some of those variables and making guesses about the others. Like observing some symptoms and making guesses about the diseases that are most likely to have explained those.

Or you might make a prediction about other symptoms you could observe. So you could go up and then back down. You could say, well, from these symptoms, I think the patient might have one of these two rare diseases. I don't know which one. But if it was this disease, then it would predict that symptom or that test maybe. But this disease wouldn't, so then that suggests a way to plan an action you could take to figure things out. So then I could go test for that symptom, and that would tell me which of these diseases the patient has.

They're also useful in planning other kinds of treatments, interventions. Like if you want to cure someone-- again, we all know this intuitively-- you should try to cure the disease, not the symptom. If you have some way to act to change the state of one of those disease variables to kind of turn it off, reasonably that should relieve the symptoms. If that disease gets turned off, these symptoms should turn off.

Whereas just treating the symptom like taking Advil for a headache is fine if that's all the problem is. But if it's being caused by something, you know, god forbid, like a brain tumor, it's not going to help. It's not going to cure the problem in the long term.

OK, so all those patterns of causal inference, reasoning, prediction, action planning, exploration is a beautiful language for capturing all of those. You can automate all those inferences. Why isn't it enough, then, for capturing commonsense reasoning or this approach to cognition? Which I'm calling the kind of model-building explaining part, as opposed to the pattern recognition part.

I mean, again, I don't want to get too far behind in talking about this. But that example is so rich. Like if you can build a neural network, you can just turn the arrows around to learn a mapping from symptoms to diseases, and that would be a pattern classifier.

So these two different paradigms for intelligence-- as some of the questions we're getting at, and as I will show versions of that with some more interesting examples in a little bit-- often it's very subtle, and the relations between them are quite valuable.

So one way to work with such a model, for example, or one nice way-- I mean, I mentioned a lot of people want to know-- and I'll keep talking about this for the rest of the hour-- productive ways to combine these powerful generative models with more pattern recognition approaches.

For some classes of this model-- there are always general purpose algorithms that can support these inferences, that can tell you what diseases you're likely to have given what symptoms. But in some cases, they could be very fast. In other cases, they could be very slow. Whereas if you could imagine trying to learn a neural network that looks just like that, only the arrows go up, so they implement a mapping from data to diseases, that could help to do much faster inference in the cases where that's possible.

So that's just one example of where a model, which might be not a crazy way to think about, for example, more generally the way top-down and bottom-up connections work in the brain. I'll take that a little bit more literally in a vision example in a second.

So there's a lot you can get from studying these causal graphical models, including some version of what it is for the mind to explain the world and how that explanation and pattern recognition approach can work together. But it's not enough to really get at the heart of common sense The mental generative models we build are more richly structured. They're

more like programs.

What do I mean by that? Well, here I'm giving a bunch of examples of scientific theories or models. Not commonsense ones, but I think the same idea applies. Ways of, again, explaining the world, not just describing the pattern.

So we went at the beginning through Newton's laws versus Kepler's laws. That's just one example. And you might not have thought of those laws as a program, but they're certainly not a graph. On the first slide when I showed Newton's laws, there was a bunch of symbols, statements in English, some math.

But what it comes down to is basically a set of pieces of code that you could run to generate the orbits. It doesn't describe the sheep or the velocities, but it's a machine that you plug in some things. You plug in some masses, some objects, some initial conditions. And you press run, and it generates the orbits, just like what you're seeing there. Although those probably weren't generated. That's a GIF. OK. That's more like Kepler or Ptolemy.

But anyway, it's a powerful machine. It's a machine, which if you put down the right masses in the right position, they don't just all go around in ellipses. Some of them are like moons, and they will go around the things that will go around the others. And some of them will be like apples on the Earth, and they won't go around anything. They'll just fall down. So that's the powerful machine.

And in the really simplest cases, that machine-- those equations can be solved analytically. You can use calculus or other methods of analysis like Newton did. He didn't have a computer. And you can show that for a two-body system, one planet and one sun, you can solve those equations to show that you get Kepler's law. Amazing.

And under the approximation that only the sun is-- for every other planet, it's only the sun that's exerting a significant influence, you can describe all of Kepler's laws this way.

But once you have more than two bodies interacting in some complex way, like three masses similar in size near each other, you can't solve the equations analytically anymore. You basically just have to run a simulation.

For the most part, the world is complicated, and our models have to be run. Here's a model of a riverbed formation. Or these are snapshots of a model of galaxy collision, you know, and climate modeling or aerodynamics.

So basically what most modern science is is you write down descriptions of the causal processes, something going on in the world, and you study that through some combination of analysis and simulation to see what would happen. If you want to estimate parameters, you try out some guesses of the parameters. And you run this thing, and you see if its behavior looks like the data you observe.

If you are trying to decide between two different models, you simulate each of them, and you see which one looks more like the data you observe. If you think there's something wrong with your model-- it doesn't quite look like the data you observe. You think, how could I change my model, which basically if I run it, it'll look more like the data I observe in some important way?

Those activities of science-- those are, in some form I'm arguing, the activities of common sense explanation. So when I'm talking about the child as scientist, that's what I'm basically talking about. It's some version of that.

And so that includes both using-- describing the causal processes with a program that you run. Or if you want to talk about learning, the scientific analog is building one of these theories. You don't build a theory, whether it's Newton's laws or Mendel's laws or any of these things, by just finding patterns and data. You do something like this program thing, but kind of recursively.

Think of you having some kind of paradigm, some program that generates programs, and you use it to try to somehow search the space of programs to come up with a program that fits your data well. OK, so that's, again, kind of the big picture. And now, let's talk about how we can actually do something with this idea-- use these programs. And you might be wondering, OK, maybe I understand--

I'm realizing I didn't say the main thing I want you to understand. The main thing I want you to get from this is how programs go beyond graphs. So none of these processes here can nicely describe with a graph the way we have in the language of graphical models. So the interesting causality-- I mean, in some sense, there's kind of a graph. You can talk about the state of the world at time T, and I'll show you graphs like this in a second. The state of the world at time T plus 1 and an arrow forward in time.

But all the interesting stuff that science really gains power from are the much more fine-grained structure captured in equations or functions that describe exactly how all this stuff works. And it needs languages like math or C++ or LISP. It needs a symbolic language of

processes to really do justice to.

The second thing I want to get, which will take a minute to get, but let's put it out there. As yes, OK, maybe you get the idea that programs can be used to describe causal processes in interesting ways. But where is the probability part come in?

So the same thing is actually true in graphical models. How many people have read Judea Pearl's 2000 book called *Causality?* How many people have read his '88 book? Or nobody's read anything.

But, OK, so what Pearl is most famous for-- I mean, when we say Pearl's famous for inventing Bayesian networks, that's based on work he did in the '80s, in which, yes, they were all probability models.

But then he came to what he calls, and I would call, too, a deeper view in which it was really about basically deterministic causal relations. Basically it was a graphical language for equations-- certain classes of equations like structural equations. If you know about linear structural equations, it was sort of like nonlinear structural equations. And then probabilities are these things you put on just on top of it to capture the things you don't know that you're uncertain about.

And I think he was he was getting at the fact that to scientists, and also to people-- there's some very nice work by Laura Schultz and Jessica Sommerville, both of whom will be here next week actually, on how children's concepts of causality are basically deterministic at the core.

And where the probabilities come in is on the things that we don't observe or the things we don't know, the uncertainty. It's not that the world is noisy. It's that we believe, at least-- except for quantum mechanics-- but our intuitive notions are that the world is basically deterministic, but with a lot of stuff we don't know. This was, for example, Laplace's view in philosophy of science.

And really until quantum mechanics, it was broadly the Enlightenment science view that the world is full of all these complicated deterministic machines, and where uncertainty comes from the things that we can't observe or that we can't measure finely enough, or they're just in some form unknown or unknowable to us. Does that make sense?

So you'll see more of this in a second. But where the probabilities are going to come from is basically if there are inputs to the program that we don't know or parameters we don't know that in order to simulate them we're going to have to put distributions on those and make some guesses and then see what happens for different guesses. Does that make sense? OK. Good. So again, that's most of the technical stuff I need to say.

And you'll learn about how this works in much more concrete details if you go to the tutorial afterwards that Tomer is going to run. What you'll see there is this. So here are just a few examples. Many of you hopefully already looked at the web pages from this probmods.org thing.

And what you see here is basically each of these boxes is a probabilistic program model. Most of it is a bunch of defined statements. So if you look here, you'll see these defined statements. Those are just defining functions. They name the function. They take some inputs, which call other functions, and then they maybe do something-- they have some output that might be an object. It might itself be a function. These can be functions that generate other functions.

And where the probabilities come in is that sometimes these functions call random number of generators basically. If you look carefully, you'll see things like Dirichlet, or uniform draw, or Gaussian, or flip. Right those are primitive random functions that flip a coin, or roll a die, or draw from a Gaussian. And those captured things that are currently unknown.

In a very important sense, the particular language, Church, that you're going to learn here with its sort of stochastic LISP-- basically just functions that call other functions and maybe add in some randomness to that-- is very much analogous to the directed graph of a Bayesian network. In a Bayesian network, you have nodes and arrows.

And the parents of a node, the ones that send arrows to it, are basically the minimal set of variables that if you were going to sample from this model you'd have to sample first in order to then sample the child variable. Because those are the key things it depends on. And you can have a multi-layered Bayesian network that, if you are going to sample from it, it's just you start at the top and you sort of go down.

That's exactly the same thing you have in these probabilistic programs where the defined statements are basically defining a function. And the functions are the nodes, and the other functions that they call as part of the statement are the things that are the nodes that send arrows there.

But the key is, as you can imagine if you've ever-- I mean, all of you have written computer programs-- is that only very simple programs look like directed acyclic graphs. And that's what a Bayesian network is. It's very easy and often necessary to write a program to really capture something causally interesting in the world where it's not a directed acyclic route. There's all sorts of cycles. There's recursion. One thing that a function can do is make a whole other graph.

Or often it might be directed and acyclic, but all the interesting stuff is kind of going on inside what happens when you evaluate one function. So if you were to draw it as a graph, it might look like you could draw a directed acyclic graph, but all the interesting stuff will be going on inside one node or one arrow.

So let me get more specific about the particular kind of programs that we're going to be talking about. In a probabilistic programming language like Church, or in general in this view of the mind, we're interested in being able to build really any kind of thing. Again, there's lots of big dreams here. Like I was saying before, I felt like we had to give up on some dreams, but we've replaced it with even grander ones, like probabilistic modeling engines that can do any computable model.

But in the spirit of trying to scale up from something that we can get traction on, what I've been focusing on in a lot of my work recently and what we've been doing as part of the center, are particular probabilistic programs that we think can capture this very early core of common sense intuitive physics and intuitive psychology in young kids.

It's what I called-- and I remember I mentioned this in the first lecture-- this game engine in your head. So it's programs for graphics engines, physics engines, planning engines, the basic kinds of things you might use to build one of these immersive video games. And we think if you wrap those inside this framework for probabilistic inference, then that's a powerful way to do the kind of common sense seen understanding, whether in these adult versions or in the young kid versions.

Now, to specify this probabilistic programs view, just like with Bayesian networks or these graphical models, we wanted general purpose tools for representing interesting things in the world and for computing the inferences that we want. Again, which means basically observing, say, just like you observe some of the symptoms and you want to compute the likely diseases that best explain the observed symptoms.

Here we talk about observing the outputs of some of these programs, like the image that's the output of a graphics program. And we want to work backwards and make a guess at the world state, the input to the graphics engine that's most likely to have produced the image. That's the analog of getting diseases from symptoms. Or again, that's our explanation right there.

And there are lots of different algorithms for doing this. I'm not going to say too much about them. Tomer will say a little bit more in the afternoon. The main thing I will do is, I will say that the main general purpose algorithms for inference in probabilistic programming language are in the category of slow and slower and really, really slow.

And this is one of the many ways in which there's no magic or no free lunch. Across all of AI and cognitive science, when you build very powerful representations, doing inference with them becomes very hard. It's part of why people often like things like neural networks. They're much weaker representations, but inference can be much faster.

And at the moment, the only totally general purpose algorithms for doing inference with probabilistic programs are slow. But first of all, they're getting faster. People are coming up with-- and I can talk about this offline where that's going-- but also-- and this is what I'll talk about in a sharper way in a second-- there are particular classes of probabilistic programs, in particular, the ones in the game engine in your head.

Like for vision is inverse graphics and maybe some things about physics and psychology too. I mean, again, I'm just thinking of the stuff of like what's going on with it when a kid is playing with some objects around them and thinking about what other people might think about those things.

It's just that setting where we think that you can build sort of in some sense special purpose. I mean, they're still pretty general. But inference algorithms for doing inference in probabilistic programs, getting the causes from the effects that are much, much faster than things that could work on just arbitrary probabilistic programs and that actually often look a lot like neural networks. And in particular, we can directly use, say for example, deep convolutional neural networks to build these recognition programs or basically inference programs that work by pattern recognition in, for example, an inverse graphics approach to vision.

So that's what I'll show you basically now. I'm going to start off by just working through a couple of these arrows. I'm going to first talk about this sort of approach we've done to tackle

both vision as inverse graphics and some intuitive physics on the scene recovered and then say a little bit about the intuitive psychology side.

Here's an example of the kind of specific domain we've studied. It's like our Atari setting. It's a kind of video game inspired by the real game Jenga. Jenga's this cool game you play with wooden blocks. You start off with a very, very, very nicely stacked up thing and you take turns removing the blocks. And the player who removes the block that makes the whole thing fall over is the one who loses.

And it really exercises this part of your brain that we've been studying here, which is an ability to reason about stability and support I very briefly went over this, but this is something that is one of the classic case studies of infant object knowledge. Looking at how basically these concepts develop in some really interesting ways over the first year of life.

Though what we're doing here is building models and testing them primarily with adults. It is part of what we're trying to do in our brains, minds, and machines research program here, collaboration with Liz and others, to actually test these ideas in experiments with infants. But what I'll show you is just kind of think of it as like infant-inspired adult intuitive physics where we build and test the models in an easier way, and then we're taking it down to kids going forward.

So the kind of experiment we can do with adults is show them these configurations of blocks and say, for example, how stable under gravity is one of these towers or configurations? So like everything else, you can make a judgment on a scale of zero to 10 or one to seven. And probably most people would agree that the ones in the upper left are relatively stable, meaning if you just sort of run gravity on it it's not going to fall over. Whereas the ones in the lower right are much more likely to fall under gravity. Fair enough? That's what people say. OK.

So that's the kind of thing we'd like to be able to explain as well as many other judgments you could make about this simple, but not that simple world of objects. And again, you can see how in principle this could very nicely interface with what Demis was talking about. He talked about their ambition to do the SHRDLU task, which was this ability to basically have a system that can take in instructions and language and manipulate objects and blocks world.

They are very far from that. Everybody's really far from having a general purpose system that can do that in any way like a human does. But we think we're building some of the common

sense knowledge about the physical world that would be necessary to get something like that to work or to explain how kids play with blocks, play with each other, talk to each other while they're playing with blocks and so on.

So the first step is the vision part. In this picture here, it's that blue graphics arrow. Here's another way into it. We want to be able to take a 2D image and work backwards to the world state, the kind of world state that can support physical reasoning. Again, remember these buzzwords-- explaining the mind with generative models that are causal and compositional.

We want a description of the world which supports causal reasoning of the sort that physics is doing, like forces interacting with each other. So it's got to have things that can exert force and can suffer forces. It's got to have mass in some form.

It's got to be compositional because you've got to be able to pick up a block and take it away. Or if I have these blocks over here and these blocks over here and I want to put these ones on top of there, the world state has to be able to support any number of objects in any configuration and to literally compose a representation of a world of objects that are composed together to make bigger things.

So really the only way we know how to do that is something like what's sometimes in engineering called a CAD model or computer-aided design. But it's basically a representation of three-dimensional objects, often with something like a mesh or a grid of key points with their masses and springs for stiffness, something like that.

Here my only picture of the world state looks an awful lot like the image, only it's in black and white instead of color. But the difference is that the thing on the bottom is actually an image. Whereas the thing on the top is just a 2D projection of a 3D model. I'll show you that one. Here's a few others. So I'll go back and forth between these.

Notice how it kind of looks like the blocks are moving around. So what's actually going on is these are samples from the Bayesian posterior in an inverse graphics system. We put a prior on world states, which is basically a prior on what we think the world is made out of. We think there's these Jenga blocks basically.

And then the likelihood, which is that that forward model is the probability of seeing a particular 2D image given a 3D configuration of blocks. And going back to the thing you had, it's basically deterministic with a little bit of noise. It's deterministic. It just follows the rules of OpenGL

graphics. Basically says objects have surfaces. They're not transparent. You can't see through them. That's an extra complication if you wanted to have that.

And basically the image is formed by taking the closest surface of the closest object and bouncing a ray of light off of it, which really just means taking its color and scaling it by intensity. It's a very simple shadow model.

So that's the causal model. And then we can add a little bit of uncertainty like, for example, maybe we can't-- there's a little bit of noise in the sensor data. So you can be uncertain about exactly the low level image features. And then when you run one of these probabilistic programs in reverse to make a guess of what configuration of blocks is most likely to have produced that image, there is a little bit of posterior uncertainty that inherits from the fact that you can't perfectly localize those objects in the world.

So again, what you see here are three or four samples from the posterior-- the distribution over best guesses of the world state of 3D objects that were most likely to have rendered into that 2D image. And any one of those is now an actionable representation for physical manipulation or reasoning. OK?

And how we actually compute that, again, I'm not going to go into right now. I'll go into something like it in a minute. But at least in its most basic form, it involves some rather unfortunately slow random search process through the space of blocks models.

Here's another example. This is another configuration there-- another image. And here is a few samples again from the posterior. And hopefully when you see these things moving around, whether it's this one or the one before, you see them move a little bit, but most of them look very similar. You'd be hard pressed to tell the difference if you looked away for a second between any one of those. Which one are you actually seeing?

And that's exactly the point. The uncertainty you see there is meant to capture basically the uncertainty you have in a single glance at an image like that. You can't perfectly tell where the blocks are. So basically any one of these configurations up here is about equally good. And we think your intuitive physics, your sort of common sense core intuitive physics that even babies have, is operating over one or a few samples like that.

Now in separate work that is not really-- I think of it as really about common sense, but it's one of the things we've been doing in our group and in CBMM where are these ideas best make

contact with the rest of what people are doing here and where we can really test interesting neural hypotheses potentially and understand the interplay between these generative models for explanation and the more sort of neural-network-type models for pattern recognition.

We've been really pushing on this idea vision as inverse graphics. So I'll tell you a little bit about that because it's quite interesting for CBMM. But I want to make sure to only do this for about five minutes and then go back to the how this gets used for more the intuitive physics and planning stuff.

So this is this an example from a paper by Tejas Kulkarni who's one of our grad students. And it's joint work with a few other really smart people such as Vikash Mansinghka who's a research scientist at MIT and Pushmeet Kohli who's at Microsoft Research. And it was a computer vision paper, pure computer vision paper from the summer, where he was developing a specific kind of probabilistic programming language, but a general one for doing this kind of vision as inverse graphics where you could give a number of different models.

Here I'll show you one for faces, another one for bodies, another one for generic objects. But basically you can pretty easily specify a graphics model that when you run it in the forward direction generates random images of objects in a certain class. And then you can run it in the reverse direction to do scene parsing to go from the image to the underlying scene.

So here's an example of this in faces where the graphics model-- it's really very directly based on work that Thomas Vetter, who was a former student or post-doc of Tommy's actually, so kind of a early ancestor of CBMM, built. And his group in Basel, Switzerland, where it's a simple but still pretty nice graphics model for making face images.

There's a model of the shape of the face, which again, it's like a CAD model. It's a mesh surface description. Pretty fine-grained structure of the 2D surface of the face in 3D. And there is about 400 dimensions to characterize the possible shapes of faces. And there's another 400 dimensions to characterize the texture, which is like the skin, the beard, the eyes, the color, and surface properties that get mapped on top of the mesh.

And then there's a little bit more graphic stuff, which is generic, not specific to faces. That stuff is all specific to faces. But then there is a simple lighting model. So you basically have a point light source somewhere out there and you shine the light on the face. It can produce shadows, of course, but not very complicated ones.

And then there's a viewpoint camera thing. So you put the light source somewhere and you put a camera somewhere specifying the viewpoint. And the combination of these, shape, texture, lighting, and camera, give you a complete graphic specification. It produces an image of a particular face lit from a particular direction and viewed from some particular viewpoint and distance.

And what you see on the right are random samples from this probabilistic program, this generative model. So you can just write this program and press Go, Go, Go, Go, Go, and every time you run it, you get a new face viewed from a new direction and lighting condition. So that's the prior.

Now, what about inference? Well, the idea of vision as inverse graphics is to say take a real image of a face like that one and see if you can produce from your graphics model something that looks like that. So, for example, here in the lower left is an example of a face that was produced from the graphics model that hopefully most of you agree looks kind of like that. Maybe not exactly the same, but kind of enough.

And in building this system-- this system, by the way, is called Picture. That's that first word of the paper title, too, the Kulkarni, et al. paper. There were a few neat things that had to be done. One of the things that had to be done was to come up with various ways to say what does it mean for the output of the graphics engine to look like the image.

In the case of faces, actually matching up pixels is not completely crazy. But for most vision problems, it's going to be unrealistic and unnecessary to build a graphics engine that's pixel-level realistic. And so you might, for example, want to have something where the graphics engine hypothesis is matched to the image with something like some kind of features. Like it could be convolutional neural network features. That's one way to use, for example, neural networks to make something like this work well.

And Jojen just showed me a paper by some other folks from Darmstadt, which is doing what looks like a very interesting similar kind of thing.

Let me show what inference looks like in this model and then say what I think is an even more interesting way to use convolutional. And that's from another recent paper we've been looking at. So here is, if you watch this, this is one observed face. And what you're seeing over here is just a trace of the system kind of searching through the space of traces of the graphics program. Basically trying out random faces that might look like that face there. It's using a kind

of MCMC inference. It's very similar to what you're going to see from Tomer in the tutorial.

It basically starts off with a random face and takes a bunch of small random steps that are biased towards making the image look more and more like the actual observed image. And at the end, you have something which looks almost identical to the observed face.

The key, right, though, is that though the observed face is literally just a 2D image, the thing you're seeing on the right is a projection of a 3D model of a face. And it's one that supports a lot of causal action.

So here just to show you on a more interesting sort of high-resolution set of face images, the ones on the left are observed images. And then we fit this model. And then we can rotate it around and change the lighting. If we had parameters that control the expression-- there's no real expression parameters here-- that wouldn't be too hard to put in. You could make us happy or sad.

But you can see-- hopefully what you can see is that the recovered model supports fairly reasonable generalization to other viewpoints and lighting conditions. It's the sort of thing that should make for more robust face recognition. Although that's not the main focus of what we're trying to use it here. I just want to emphasize there's all sorts of things that would be useful if you had an actual 3D model of the face you could get from a single image.

Or here's the same kind of idea now for a body pose system. So now, the image we're going to assume has a person in it somewhere doing something. Remember back to that challenge I gave at the beginning about finding the bodies in a complex scene like the airplane full of computer vision researchers where you found the right hand or the left toe.

So in order to do that, we think you have to have something like an actual 3D model of a body. What you see on the lower left is a bunch of samples from this. So we basically just took a kind of interesting 3D stick figure skeleton model and just put some knobs on it. You can tweak it around. You can put some simple probability models to get a prior. And these are just random samples of random body positions.

And the idea of the system is to kind of search through that space of body positions until you find one, which then when you project it from a certain camera angle looks like the body you're seeing.

So here is an example of this in action. This is some guy-- I guess Usain Bolt. Some kind of interesting slightly unusual pose as he's about to break the finish line maybe. And here is the system in action. So it starts off from a random position and, again, sort of takes some takes a bunch of random steps moving around in 3D space until it finds a configuration, which when you project it into the image looks like what you see there.

Now, notice a key difference when I say looks like-- it doesn't look like it at the pixel level like the face did. It's only matching at the level of these basically enhanced edge statistics which you see here. So this is an example of building a model that's not a photorealistic render. The graphics model is not trying to match the image. It's trying to match this. Or it could be, for example, some intermediate level of convonet features.

And we think this is very powerful. Because more generally while we might have a really detailed model of facial appearance, for bodies, we don't have a good clothing model. We're not trying to model the skin. We're just trying to model just enough to solve the problem we're interested in. And again, this is reflective of a much more broad theme in this idea of intelligence as explanation, modeling the causal structure of the world.

We don't expect, even in science, but certainly not in our intuitive theories, to model the causal structure of the world at full detail. And a way that either I am always misunderstood or always fail to communicate-- it's my fault really-- is I say, oh, we have these rich models of the world. People often think that means that somehow the complete thing. Like if I say we have a physics engine in our head, it means we have all of physics. Of if I say we have a graphics engine, we have all of every possible thing.

This isn't Pixar. We're not trying to make a beautiful movie, except maybe for faces. We're just trying to capture just the key parts, just the key causal parts of the way things move in the world as physical objects and the way images are formed that at the right level of abstraction that matters for us allows us to do what we need to do.

This is just an example of our system solving some pretty challenging body pose recognition problems in 3D, cases which are problematic even for the best of standard computer vision systems. Either because it's a weird pose, like these weird sports figures, or because the body is heavily occluded. But I think, again, these are problems which people solve effortlessly. And I think something like this is on the track of what we want to do. You can apply the same kind of thing to more generic objects like this, but I'm not going to go into the details.

The last thing I want to say about vision before getting back to common sense for a few minutes-- and in some sense, maybe this is the most important slide for the broader CBMM, brains, minds, and machines thing. Because this is the clearest thing I can point to to the thing I've been saying all along since the beginning of the morning about how we want to look for ways to combine the generative model view and the pattern recognition view.

So the generative model is what you see on the left here. It's the arrows going down. It's exactly just the face graphics engine, the same thing I showed you. The thing on the right with the arrows going up is a convonet. Basically it's a out-of-the-box, cafe-style, convolutional neural net with some fully connected layers on the top. And then there's a few other dashed arrows which represent linear decoders from layers of that model to other things, which are basically parts of the generative model.

And the idea here-- this is work due to Ilker Yildirim, who some of you might have met. He was here the other day. He's one of our CBMM postdocs, but also joint with Tejas and with Winrich who you saw before. It's to try to in several senses combine the best of these perspectives, to say, look, if we want to recognize anything or perceive the structure of the world richly, I think it needs to be something like this inverse graphics or inverting a graphics program.

But you saw how slow it was. You saw how it took a couple of seconds at least on our computer just for faces to search through the space of faces to come up with a convincing hypothesis. That's way too slow. It doesn't take that long. We know a lot about exactly how long it takes you from Winrich, and Nancy's, and many other people's work.

So how can vision in this case, or really much more generally, be so rich in terms of the model it builds, yet so fast? Well, here's a proposal, which is to take the things that are good at being fast like the pattern recognizers, deep ones, and train them to solve the hard inference problem or at least to do most of the work. It's an idea which is very heavily inspired by an older idea of Geoff Hinton's sometimes called the Helmholtz machine.

Here the idea in common with Hinton is to have a generative model and a recognition model where the recognition model is a neural network and it's trained to invert the generative the model. Namely, it's trained to map not from sense data to task output, but from sense data to the hidden deep causes of the generative model, which then, when you want to use this to act to plan what you're going to do, you plan on the model.

To make an analogy to say the DeepMind video game player, this would be like having a

system which, in contrast to the DeepQ network, which mapped from pixel images to joystick commands, this would be like learning a network that maps from pixel images to the game state, to the objects, the sprites that are moving around, the score, and so on, and then plans on that. And I think that's much more like what people do is that.

Here just in the limited case of faces, what are we doing, right? So what we've got here is we take this convolutional neural network. We train it in ways that you can read about in the paper. It's very easy kind of training to basically make predictions, to make guesses about all the latent variables, the shape, the texture, the lighting, the camera angle.

And then you take those guesses, and they start off that Markov chain. So instead of starting off at a random graphics hypothesis, you start off at a pretty good one and then refine it a little bit. What you can see here in these blue and red curves is the blue curve is the course of inference for the model I showed you before, where you start off at a random guess, and after, I don't know, 100 iterations of MCMC, you improve and you kind of get there.

Whereas the red curve is what you see if you start off with the guess of this recognition model. And you can see that you start off sort of in some sense almost as good as you're ever going to get, and then you refine it. Well, it might look like you we're just were refining it a little bit. But this is a kind of a double log scale. It's a log plot of log probability. So what looks like a little bit there on the red curve is actually a lot-- I mean perceptually.

You can see it here where if you take-- on the top I'm showing observed input faces. On the bottom I'm showing the result of this full inverse graphics thing. And they should look almost identical. So the full model is able to basically perfectly invert this and come up with a face that really does look like the one on the top.

The ones in the middle are the best guess you get from this neural network that's been trained to approximately invert the generative model. And what you can see is on first glance it should look pretty good. But if you pay a little bit of attention, you can see differences. Like hopefully you can see this person is not actually that person in a way that this is much more convincingly. Or this person-- this one is pretty good, but I think this one-- I think it's pretty easy to say, yeah, this isn't quite the same person as that one. Do you guys agree? We've done some experiments to verify this.

But hopefully they should look pretty similar, and that's the point. How do you combine the

best of these computational paradigms? How can perception more generally be so rich and so fast? Well, quite possibly like this. It even actually might provide some insight into the neural circuitry that Winrich and Doris Tsao and others have mapped out.

We think that this recognition model that's trained to invert the graphics model can provide a really nice account of some of Winrich's data like you saw before. But I will not go into the details because in maybe five to 10 minutes I want to get back to physics and psychology.

So physics-- and there won't be any more neural networks. Because that's about as much-- I mean, I think we'd like to take those ways of integrating the best of these approaches and apply them to these more general cases. But that's about as far as we can get.

Here what I want to just give you a taste of at least is how we're using ideas just purely from probabilistic programs to capture more of this common sense physics and psychology. So let's say we can solve this problem by making a good guess of the 3D world state from the image very quickly inverting this graphics engine.

Now, we can start to do some physical reasoning, a la Craik's mental model of in the head of the physical world, where we now take a physics engine, which is-- here again we're using the kind of physics engines that game physics-- like very simple-- again, I don't have time to go into the details. Although Tomer has written a very nice paper with, well, with himself. But he's nicely put my name and Liz's on it-- about sort of trying to introduce some of the basic game engine concepts to cognitive scientists. So hopefully we'll be able to show you that soon too. Or you can read about them.

Basically it's that these physics engines are just doing again a very quick, fast, approximate implementation of certain aspects of Newtonian mechanics. Sufficient that if you run it a few times steps with a configuration of objects like that you might get something like what you see over there on the right. That's an example of running this approximate Newtonian physics forward a few time steps.

Here's another sample from this model, another kind of mental stimulation. We take a slightly different guess of the world state, and we run that forward a few time steps, and you see something else happens. Nothing here is claimed to be accurate in the ground truth way. Neither one of these is exactly the right configuration of blocks. And you run this thing forward, and it only approximately captures the way blocks really bounce off each other. It's a hard problem to actually totally realistically simulate.

But our point is that you don't really have to. You just have to make a reasonable guess of the position of the blocks and a reasonable guess of what's going to happen a few time steps in the future to predict what you need to know and common sense, which is that, wow, that's going to fall over. I better do something about it.

And that's what our experiment taps into. We give people a whole bunch of stimuli like the ones I showed you and ask them, on some graded scale, how likely do you think it is to fall over? And what you see here-- this is again one of those plots that always are the same where on the y-axis are the average human judgments now of-- it's an estimate of how unstable the tower is. It's both the probability that it will fall, but also how much of the tower will fall. So it's like the expected proportion of the tower that's going to fall over under gravity.

And along the x-axis is the model prediction, which is just the average of a few samples from what I showed you. You just take a few guesses of the world state, run it forward a few time steps, count up the proportion of blocks it fell, and average that. And what you can see is that does a really nice job of predicting people's stability intuitions.

I'll just point to an interesting comparison. Because it does come into where. Where does the probability come in in these probabilistic programs? Well, here's one very noticeable way. So if you look down there on the lower right, you'll see a smaller version of a similar plot. It's plotting now the results of-- it says ground truth physics, but that's a little misleading maybe. It's just a noiseless physics engine.

So we take the same physics model, but we get rid of any of the state uncertainties. So we tell it the true position of the blocks, and we give it the true physics. Whereas our probabilistic physics engine allows for some uncertainty in exactly which forces are doing what.

But here we say we're just going to model gravity, friction, collisions as best we can. And we're going to get the state of the blocks perfectly. And because it's noiseless, you notice that-- so those crosses over there are crosses because they're arrow bars, both across people and model simulations. Now they're just vertical lines. There's no arrow bars in the model simulation because it's deterministic.

It's graded because there's the proportion of the tower that falls over. But what you see is the model is a lot worse. It scatters much more. The correlation dropped from around 0.9 to around 0.6 in terms of correlation of model with people's judgments.

And you have some cases like this red dot here-- that corresponds to this stimulus-- which goes from being a really nice model fit. This is one which people judged to be very unstable, and so does the probabilistic physics engine. But actually it's not unstable at all. It's actually perfectly stable. The blocks are actually just perfectly balanced so that it doesn't fall. Although I'm sure everybody looks at that and finds that hard to believe.

So this is nice. This is a kind of physics illusion. There are real world versions of this out on the beaches not too far from here. It's a fun thing to do to stack up objects in ways that are surprisingly stable. We would say a surprise because your intuitive physics has certain irreducible noise.

What we're suggesting here is that your physical intuitions-- you're always in some sense making a guess that's sensitive to the uncertainty about where things might be and what forces might be active on the world. And it's very hard to see these as deterministic physics, even when you know that that's exactly what's going on and that it is stable.

Let me say just a little bit about planning. So how might you use this kind of model to build some model of this core intuitive psychology? And I don't mean here all of theory of mind. Next week, we'll hear a lot more. Like Rebecca Saxe will be down here. We'll hear a lot more about much richer kinds of reasoning about other people's mental states that adults and older children can do.

But here we're talking about, just as we were talking about what I was calling core intuitive physics, again inspired by Liz's work of just you know what objects do right here on the table top around us over short time scales, the core theory of mind, something that even very young babies can do in some form, or at least young children. There's controversy over exactly what age kids can be able to do this sort of thing.

But in some form I think before language, it's the kind of thing that when you're starting to learn verbs, the earliest language is kind of mentalistic and builds on this knowledge. And take the red and blue ball chasing scene that you saw, remember from Tomer. That was 13-month-olds. So there's definitely some form of kind of interpretation of beliefs and desires in some protoform that you can see even in infants of around one year of age.

And it's exactly that kind of thing also. Remember that, if you saw John Leonard's talk yesterday-- he was the robotics guy who talked about self-driving cars and how there's certain

gaps in what they can do despite the all the publicity, like the can't turn left basically in an unrestricted intersection. Because there's a certain kind of theory of mind in street scenes when cars could be coming and people could be crossing or all those things about the police officers.

Part of why this is so exciting to me and why I love that talk is because this is, I think, that same common sense knowledge that if we can really figure out how to capture this reasoning about beliefs and desires in the limited context where desires are people moving around in space around us and the beliefs are who can see who and who can see who can see who-- in driving, the art of making eye contact with other drivers or pedestrians is seeing that they can see you or that they can see what you can see and that they can see you seeing you them.

It doesn't have to be super deeply recursive, but it's a couple of layers deep. We don't have to think about it consciously, but we have to be able to do it. So that's the kind of core belief desire theory of mind reasoning. And here's how we've tried to capture this with probabilistic programs.

This is work that Chris Baker started doing a few years ago. And a lot of it joint Rebecca Saxe and also some of it with Julian Jara-Ettinger and some of it with Tomer. So there's a whole bunch of us who've been working on versions of this, but I'll just show you one or two examples.

Again, the key programs here are not graphics or physics engines, but planning engines and perception engines. So very simple kinds of robotics programs, far too simple in this form to build a self-driving car or a humanoid robot, but maybe the kind of thing that in game robots like the zombie or the security guard in Quake or something might do something like this.

So planning basically just means it's a little bit more than sort of shortest path planning. But it's basically like find a sequence of actions in a simple world like moving around a 2D environment that maximizes your long run expected reward.

So there's a kind of utility theory, or what Laura Schulz calls a naive utility calculus, here. A calculation of costs and benefits where in a sense you get a big reward, a good positive utility for getting to your goal and a small cost for each action you take. And under that view, then in some sense-- and some actions might be costly than others, something that Tomer is looking at in infants and something that Julian Jara-Ettinger has looked at in older kids, this understanding of that.

But this sort of basic cost-benefit trade off that is going on whenever you move around an environment and decide, well, is it worthwhile to go all the way over there, or, well, I know I like the coffee up at Pie in the Sky better than the coffee in the dining hall here at Swope. But to think about, am I going to be to my lecture? Am I going to be late to Nancy's lecture? Those are different costs-- both costs. It's that kind of calculation.

So here let me get more concrete. So here's an example of an experiment that Chris did a few years ago where, again, it's like what you saw what the Heider and Simmel, the squares and the triangles and circles or the south gate and chibra, the red and blue balls chasing each other. Very simple stuff.

Here you see an agent. It's like an overhead view of a room, 2D environment from the top. The agents moving along some path. There are three possible goals, A, B, or C. And then there's maybe some obstacles or constraints like a wall like like you saw in those movies. Maybe the wall has a hole that he can pass through. Maybe it doesn't.

And across different trials of the experiment, just like in the physics stuff we vary all the block configurations and so on, here we vary where the goals are. We vary whether the wall has a hole or not. We vary the agent's path. On different trials, we also stop it at different points. Because we're trying to see as you watch this agent move around, action unfolds over time. How do your guesses about his goal change over time?

And what you see-- so these are just examples of a few of the scenes. And here what you see are examples of the data. Again, the y-axis is the average human judgment. Red, blue, and green is color coded to the goal. They're just asked, how likely do you think each of those three things is his goal? And then here the x-axis is time. So these are time steps that we ask at different points along the trajectory.

And what you can see is that people are making various systematic kinds of judgments. Sometimes they're not sure whether his goal is A or B, but they know it's not C. And then after a little while or some key stat happens, and now they're quite sure it's A and not B. Or they could change their mind.

Here people were pretty sure it was either green or red but not blue. And then there comes a point where it's surely not green, but it might be blue or red. Oh no, then it's red. Here they were pretty sure it was green. Then no, definitely not green. And now, I think it's red. It was

probably never blue. OK.

And the really striking thing to us is how closely, you can match those judgments with this very simple probabilistic planning program run in reverse. So we take, again, this simple planning program that just says basically just kind of get as efficiently as possible to your goal.

I don't know what your goal is though. I observe your actions that result from an efficient plan, and I want to work backwards to say, what do I think your goal is, your desire, the rewarding state? And just doing that just basically perfectly predicts people's data. I mean, of all the mathematical models of behavior I've ever had a hand in building, this one works the best. It's really quite striking. To me it was striking because I came in thinking this would be a very high-level, weird, flaky, hard-to-model thing.

Here's just one more example of one of these things, which actually puts beliefs in there, not just desires. So it's a key part of intuitive psychology that we do joint inference over beliefs and desires. In this one here, we assume that you, the subject, the agent who's moving around, all of us have shared full knowledge of the world. So we know where the objects are. We know where the holes are. There's none of this false belief, like you think something is there when it isn't.

Now, here's some later work that Chris did, what we call the food truck studies, where here we add in some uncertainty about beliefs in addition to desires. And it's easiest just to explain with this one example up there in the upper left.

So here, and this, like a lot of university campuses, lunch is best found at food trucks, which can park in different spots around campus. Here the two yellow squares show the two parking spots on this part of campus. And there are several different trucks that can come and park in different places on different days. There's a Korean truck. That's k. There's a Lebanese truck. That's l. There's also other trucks like a Mexican truck.

But there's only two spots. So if the Korean won parks there and the Lebanese one parks there, the Mexican has to go somewhere else or can't come there today. And on some days the trucks park in different places. Or a spot could also be unoccupied. The trucks could be elsewhere.

So look at what happens on this day. Our friendly grad student, Harold, comes out from his office here. And importantly, the way we model interesting notions of evolving belief is that now

we've got that perception and inference arrow there. So Harold forms his belief about what's where based on what he can see. And it's just the simplest perception model, just line-of-sight access. We assume he can kind of see anything that's unobstructed in his line of sight.

So that means that when he comes out here, he can see that there is the Korean truck here. But you can't see-- this is a wall or a building. He can't see what's on the other side of that. OK, so what does he do? Well, he walks down here. He goes past the Korean truck, goes around the other side of the building. Now at this point, his line of sight gives him-- he can see that there is a Lebanese truck there. He turns around, and he goes back to the Korean truck. So the question for you is, what is his favorite truck? Is it Korean, Lebanese, or Mexican?

AUDIENCE: Mexican.

PROFESSOR: Mexican, yeah, it doesn't sound very hard to figure that out. But it's quite interesting because the Mexican one isn't even in the scene. The most basic kind of goal recognition-- and this, again, cuts right to the heart of the difference between recognition and explanation. There's been a lot of progress in machine vision systems for action understanding, action recognition, and so on.

And they do things like, for example, they take video. And the best cue that somebody wants something is if they reach for it or move towards it. And that's certainly what was going on here. In all of these scenes, your best inference about what the guy's goal is is which thing is he moving towards. And it's just subtle to parse out the relative degrees of confidence when there's a complex environment with constraints.

But in every case, by the end it's clear he's going for one thing, and the thing he is moving towards is the thing he wants. But here you have no trouble realizing that his goal is something that isn't even present in the scene. Yet he's still moving towards it. In a sense, he's moving towards his mental representation of it. He's moving towards the Mexican truck in his mind's model. And that's him explaining the data he sees.

For some reason, he must have had maybe a prior belief that the Mexican truck would be there. So he formed a plan to go there. And in fact, we can ask people not only which truck does he like-- it's his Mexican truck. That's what people say, and here is the model.

But we also asked them a belief inference. We say, prior to setting out, what did Harold think was on the other side? What was parked in the other spot that he couldn't see? Did he think it

was Lebanese, Mexican, or neither? And we ask a degree of belief. So you could say he had no idea. But interestingly, people say he probably thought it was Mexican. Because how else could you explain what he's doing?

So I mean, if I had to point to just one example of cognition as explanation, it's this. The only sensible way, and it's a very intuitive and compelling way to explain why did he go the way he did and then turn around just when he did and wind up just where he did, is this set of six instances basically. That his favorite is Mexican, his second favorite is Korean-- that's also important-- his least favorite is Lebanese.

And he thought that Mexican was there, which is why it was worthwhile to go and check. At least, he thought it was likely. He wasn't sure, right? Notice it's not very high. But it it's more likely than the other possibilities. Because, of course, if he was quite sure it was Lebanese, well, he wouldn't have bothered to go around there. And in fact, you do see that.

So you have ones-- I guess I don't have them here. But there are scenes where he just goes straight here. And then that's consistent with him thinking possibly it was Lebanese. And if he thought nothing was there, well, again, he wouldn't have gone to check. And again, this model is extremely quantitatively predictive of people's judgments about both desires and beliefs.

You can read in some of Battaglia's papers ways in which you take the very same physics engine and use it for all these different tasks, including sort of slightly weird ones like these tasks. If you bump the table, are you more likely to knock off red blocks or yellow blocks? Not a task you ever got any end-to-end training on, right? But an example of the compositionality of your model and your task.

Somebody asked me this during lunch, and I think it is a key point to make about compositionality. One of the key ways in which compositionality works in this view of the mind, as opposed to the pattern recognition view or the way, let's say, like a DeepQ network works--

AUDIENCE: You mean the [INAUDIBLE].

PROFESSOR: Just ways of getting a very flexible repertoire of inferences from composing pieces without having to train specifically for it. It's that if you have a physics engine, you can simulate the physical world. You can answer questions that you've never gotten any training at all to solve.

So in this experiment here, we ask people, if you bump the table hard enough to knock some of the blocks onto the floor, is it more likely to be red or yellow blocks? Unlike questions of will

this tower fall over, which we've made a lot of judgments of that sort. You've never made that kind of judgment before.

It's a slightly weird one. But you have no trouble making it. And for many different configurations of blocks, you make various grade adjustments, and the model captures it perfectly with no extra stuff put in. You just you just take the same model, and you ask it a different question.

So if our dream is to build AI systems that can answer questions, for example, which a lot of people's dream is, I think there's really no compelling alternative to something like this. That you build a model that you can ask all the questions of that you'd want to ask.

And in this limited domain, again, it's just our Atari. In this limited domain of reasoning about the physics of blocks, it's really pretty cool what this physics engine is able to do with many kinds of questions. It can reason about things with different masses. It can make guesses about the masses. You can make fun of the objects bigger or smaller. You can attach constraints like fences to the table.

And the same model, without any fundamental change, can answer all these questions. So it doesn't have to be retrained. Because there's basically no training. It's just reasoning.

If we want to understand how learning works, we first have to understand what's learned. I think right now, we're only at the point where we're starting to really have a sense of what are these mental models of the physical world and intentional action-- these probabilistic programs that even young children are using to reason about the world. And then it's a separate question how those are built up through some combination of scientific discovery sorts of processes and evolution.

So here's the story, and I've told most of what I want to tell you. But the rest you'll get to hear-- some of it you'll get to hear next week from both our developmental colleagues and from me and Tomer. More on the computational side. But actually the most interesting part we just don't know yet. So we hope you will actually write that next chapter of this story.

But here's the outlines of where we currently see things. We think that we have a good target for what is really the core of human intelligence, what makes us so smart in terms of these ideas of both what we start with, this common sense core physics and psychology, and how those things grow. What are the learning mechanisms that I've just justified.

Again, more next week on the sort of science-like mechanisms of hypothesis formation, experiment testing, play, exploration that you can use to build these intuitive theories, much like scientists build their scientific theories. And that we're starting on the engineering side to have tools to capture this, both to capture the knowledge and how it might grow through the use of probabilistic programs and things that sometimes go by the name of program induction or program synthesis.

Or if you like hierarchical Bayes on programs that generate other programs where the search for a good program is like the inference of a program that best explains the data as generated from a prior that's a higher level program.

If you go to the tutorial from Tomer you'll actually see building blocks. You can write Church programs that will do something like that, and we will see more of that next time. But the key is that we have a language now which keeps building the different ingredients that we think we need.

On the one hand, we've gone from thinking that we need something like probabilistic generative models, which many people will agree with, to recognizing that not only do they have to be generative, they have to be causal and compositional. And they have to have this fine-grained compositional structure needed to capture the real stuff of the world. Not graphs, but something more like equations that capture graphics or physics or planning.

Of course, that's not all. I mean, as I tried to gesture at, we need also ways to make these things work very, very quickly. There might be a place in this picture for something like neural networks or some kind of alternative pro-and-con approach based on pattern recognition.

But these are just a number of the ways which I think we need to think about going forward. We need to take the idea of both the brain as a pattern ignition engine seriously and the idea of the brain as a modeling or explanation engine seriously.

We're excited because we now have tools to model modeling engines and maybe to model how pattern recognition engines and modeling engines might interact. But really, again, the great challenges here are really very much in our future. Not the unforeseeable future, but the foreseeable one. So help us work on it. Thanks.