PROFESSOR: OK. Welcome back to computational systems biology We have the honor today of having Professor Ron Weiss visit us. As I told you on Tuesday, he's going to talk about synthetic biology.

And Ron is probably from both the department of biological engineering and the Computer Science department and also a founding member of the Synthetic Biology Center at MIT. And now, thank you, Ron.

PROF. RON WEISS: Thank you, Dave. Thanks for inviting me here. Did you mention our background? Dave was actually-- advised me when I first came to our graduate school at MIT. And at the time, I was working on digital video and information retrieval.

And Dave started getting into the business of biology. This is back in the early '90s. And I was like, this is cool stuff but it's really messy. How can you engineer with these molecules?

PROFESSOR: And we've got the answer to that.

PROF. RON WEISS: Yeah. So we'll see. So let's see if the answer is-- it does actually work.

So yeah, so I-- after being a non-believer-- I don't know if non-believer, but just-- I didn't feel I quite had the engineering capabilities. Towards around '96 or so is when I decided to actually make the switch. At the time, I was working-- around '96, I was working at this notion of how can we use what we know in biology to understand how we program computers and especially situations where you have lots and lots of computing elements like-- things like smartDOS.

I don't know if people have heard, but amorphous computing. So back in the mid

'90s or so, this notion that we would be able to embed computation everywhere was kind of an exciting notion. And I thought to myself, where-- how could I get inspired? And I thought, well, biology obviously could serve as great inspiration.

Because that's a situation where you have millions or billions of little computing elements that don't have too much power, kind of interact locally. But they still perform very robust operations. And so I performed a variety of simulations, for example, of embryogenesis and other processes to try to understand what happens in biology and can I again use that to program computers or little tiny computers. And I remember one day, I just decided to flip the arrow and basically rather than trying to use biology to understand or program computers, I decided, let me use what I know in computing to actually try to program biology, OK?

And so now that field is basically called synthetic biology. So I've been in that field-- it's hard to count now, but I guess maybe 18 years or so. And it's been fun and has not been easy.

But I think at least we're starting to make some-- we're making some progress. So I'll try to tell you about some of our efforts there. And I certainly encourage you to ask me questions. So please interrupt me at any point in time.

Any kind of question is fair game. Dave promised me that you guys are a tough crowd. So let's see. And you can always stump. Let's try to have that happen.

So when I look at this, I get excited as an engineer, OK? And I think to myself, wow. This could be really cool to be able to program something like this, again, in the same way that we may program computers. And so this notion of genetic engineering in a direct way, in a way where we can create new DNA, certainly has been around since the '70s or so.

And so this notion has allowed us as a community to create various mechanisms that control what the cells do. So for example, transcriptional regulation, translational-- so being able to regulate things in a cell, be able to create genetically encoded sensors, cell-cell communication mechanisms, synthesis of various interesting molecules-- biofuels, pharmaceuticals, and control physical aspects. And so those capabilities have been around before synthetic biology. But if you were to ask me what is different about synthetic biology, I would say it's really the emphasis on systems level engineering. OK, so this notion that we are not just trying to engineer over expression of this gene or that gene or a couple of genes, but really trying to understand how to create systems of interactions.

So in the same way that systems biology has come to the forefront with this notion that you can't understand a cell by understanding what is the exact purpose of this particular gene-- we always have to think about it within the context of a pathway within the context of the entire organism-- in the same way, when we want to be able to get cells to do interesting things, we have to think about the system as a whole. And to get the sophistication that we need, we need to understand how to connect these various elements, regulatory elements, all these kinds of elements, in reliable, predictable ways, efficient ways and so on to be able to get the cells to be as programmable as computers. So that if you were to ask me what synthetic biology is, that would be my answer.

And so now you know it and you can tell all your friends that it's a completely defined notion and so on. Maybe if yo ask other people, they'll give you slightly different answers. But there you go.

So how do we develop an engineering discipline out of that? OK, so that's really how can we get undergrads to come in and take Synbio 101 where it's really a well defined mechanism and set of methodologies and practices that allow us to do this reliably. OK, and so we often try to get inspired by how other disciplines approach the engineering of complex systems.

And so kind of an obvious one would be of computing or robotics where there's this notion of, for example, bottom up assembly. And so you start with basic devices and think about how to create modules that have specific behaviors in them and then put those modules and integrate those modules to create these autonomous entities such as robots. And we often think about how to create communities of interactions, communities of robots in this case, and so on.

And so that's worked quite well in a variety of different other engineering disciplines. And so we often ask the question, can we import these mechanisms into the world of biology? So can we take basic mechanisms of regulation-- it could be transcriptional, it could be other modes of regulation-- and then wind these things up to create customizable pathways that we then embed into cells? And then we can create programmable communities of bacteria. We can create programmable tissues of mammalian cells and so on.

And so the question is, is this a useful and efficient way to approach things? So for example, how are these different approaches similar? What can we borrow from here that make sense to push on over there?

And I will say when I started working in synthetic biology, most of my efforts were really focused on adapting and implementing these things-- so adapting them from other disciplines and trying to understand how to implement them into the world the biology. But as time has gone by and as we've started to understand and appreciate the cell more and more, we are also quite interested in how these things are different as well. So what makes engineering biological systems a truly unique, new engineering discipline? What would you do in the world of biology that might be different than what you would do with computers or robots or building bridges and cars and planes and so on?

And so that that's become more and more of an important focus in my lab and I think in the community as a whole although not yet everywhere. And you often see situations where people come in from other disciplines and just think, oh, we'll just program it, engineer it, just like we do in computing and so on. And it doesn't just work like that.

So when we approach these tasks of programming the cells, we usually divide things up into modules of sensors, processing, and actuation. So for example, we would want to develop sensors that can detect in live cells levels of microRNA messenger and then proteins and then connect them to synthetic regulatory circuits that we embed in the cells, OK? So it's important that these sensors not just, for example, give us fluorescent readouts, but it's important that these sensors then connect to the regulatory networks that we have in mind and so that these regulatory networks can then integrate multiple pieces of information and make decisions about actuation.

So how do we turn on specific proteins that will then influence that particular cell or even the environment in a programmable fashion as dictated by the levels of particular sensors as well as by other mechanisms or, for example, from looking at historical information that the cell itself has processed as well? And so this is, I would say, represents the paradigm for most of the things that do take place in synthetic biology. And so why do we want to do this?

It's not the program the next version of the iOS or iPhone or something like that. even though that initially that was one of the things that was discussed, it's not just for the sake of computation, but really for the sake of specific applications. So for example, if we have really slow logic gates that work on the order of hours or even days, that might be fine if the application, for example, is a tissue engineering application, OK?

And so in synthetic biology, initial emphasis is really been on what can we do with microbial, let's say, communities or individuals, for example, for synthesis of high value compounds? I mentioned bioenergy, environmental applications as well. So that, I would say, was most of the emphasis there.

But over the last few years, there's been a growing interest in health-related applications. And so my lab in particular looks at mostly health related applications. And so I'll give you examples of those today, OK? And those include things involved with cancer, diabetes, in tissues by design.

And in order to do this, in order to have this programmability, you want to think about scales. So you want to think about how much DNA does it take to do X? And to a large extent, that controls the sophistication. It really is an important defining element what we do is the scale of the DNA that we can actually engineer reliably quickly, efficiently, predictably, in high throughput fashion, and in inexpensive ways.

So we would start with things on the order of genes where I would say that that's really the basic elements. I would say that a single gene that you overexposes or a few genes than you inducibly express, I wouldn't count that as synthetic biology. But when it gets kind of interesting for synthetic biology is when we have this circuitry, where we now embed interactions that didn't previously exist in that particular cell context.

And so most of synthetic biology has been really at this level right here-- actually, mostly from here to here in terms of the scale of the DNA and now trying to go beyond that-- more along the lines can we create something that's 20,000 bases, 50,000 bases of DNA? OK, is this something that a graduate student can come into the lab and say, I want to design something that will take 20,000 to 50,000 bases? Is this a reasonable thing to consider?

OK, and then the question would be, what kind of power does that provide to you? What things can you do with that? Beyond that, people have explored this notion of minimal life and even full genome rewrites.

I would say at this point, this is-- what some people clump that in with synthetic biology, which is fine. We don't have, at the moment, really good ways of being able to engineer minimal life from scratch or even in a really fundamentally different way. So most of the efforts on minimal life would be take an organism and try to figure out what to knock out, right, as opposed to saying, I'm going to engineer this new minimal organism.

And I'm going to define what reactions to put in there from scratch. And I'm going to create a whole bunch of new ones that didn't exist before. OK, in the future, will we be able to do this? Hopefully, OK?

Not quite yet-- this is really where the action of right now. And again, driving force for this is how inexpensive is DNA synthesis. And so we're following some kind of Moore's law with respect to dropping costs in terms of DNA synthesis.

And this is one of the enabling-- I don't know if it's-- it's not the only enabling technology. But is one of the most important enabling technologies is the fact that it is less and less expensive to be able to order longer and longer sequences of DNA. And so this notion that, for example, you'll be able to design something that's, again, that's 20,000 to 50,000 bases of DNA and just go online and order to that and have your advisor willingly pay for that-- not at the level of 20,000 to 50,000 bases yet. But that's going to change.

And that's going to get to the point where those really become available to everyone. And I think that's going to fundamentally change how we do business in biological engineering and how we do, I would say, almost everything in biology as a whole. So if you have-- even if you don't care about engineering new biological functions but you want to understand biological systems and your adviser told you, well, just design a whole bunch of circuits that will allow you to regulate things in arbitrary ways to learn something about the underlying networks that control a natural systems, again, I think that that fundamentally changes what kinds of questions you will ask.

OK, so I'll talk about basic design. I'll talk about scalability. So how do we go from these basic elements to bigger and bigger things?

And then I'll talk about some recent things that we're doing where we're building this foundation. But we think that this foundation then can matter. I think this foundation can change how you approach things that really don't have the greatest of solutions.

Now, they really change the paradigm, for example, for cancer, for this notion of building tissues by design on chips and for diabetes and so on. OK, so we start with parts. So just about everything that we do, we define what are the basic parts that are available in our toolbox.

And so these will be transcriptional regulatory parts. We do things at the translational level. We do things also at the protein-protein level. One of the things we often do, not always, is engineer cell-cell interactions. Could be by means of cell-

cell communication.

We often want to find out what's going on in the cell. So just like when we program-- where we create a new software, new computer program, we have debugging outputs that tell us what the program is doing. Usually, the way we do this is with fluorescent protein. It could be with dyes too.

So they tell us, you know, here's how your circuit is behaving. Here's how the cell might be behaving. And another set of parts would be ones where we want to be able to create sensors and actuators inside the cell.

What are specific biomarker levels? How can we affect what the cell is doing? For example, one that gets used a lot is kill the cell is one of the favorite actuators that people are using. Another one would be, let's say, tell this stem cell to differentiate into a different cell type.

That would be another kind of actuator. A different one might make the cell-- make this high value compound that would be relevant for some application. And so right now, if you're looking for parts, they actually used to be stored in Stata up until-- or big libraries of synthetic biology parts were stored in Stata that up until about two years or so. So I don't know how many people know about iGEM.

Any folks know about iGEM? So iGEM was started at MIT, was headquartered, as I mentioned again here, in Stata. There are these couple of big freezer that were on the, I think, the fourth floor here.

And they stored 5,000 to 10,000 parts that word commonly used by synthetic biology folks. OK, so now they moved over closer to Cambridge brewing company. And they're not affiliated directly with MIT anymore and they have 15,000 parts or so available.

So if you want to get started in synthetic biology, this is one quick way to do that. You can contact iGEM headquarters and say, please send me 1,000 parts, OK? And as long as you're credible and not from one of those blacklisted countries, then they typically will send it to you.

So that's a good way to get started, OK? So what are these parts? So this is actually going back to my Ph.D. here. This is one of the parts that I characterized.

So this notion of an inverter-- so digital logic convert. So I assume people here-- everybody is familiar with logic gates. Is that true? OK, raise your hand if you are familiar with it.

I just want to see- oh. Just trying to calibrate-- and again, ask me questions. So this notion that you have a single input, single output device that works on binary values that has-- basically inverts the signal. So you have zero on the input.

You have one on the output. One in the input, you have zero in the output. And so one of the ways in which you can implement this in a biological system is just use transcriptional repression.

OK, so if you have no repressor present, then you have a high level of output protein. If you have a repressor present, it represses the production of the alpha protein. And so in theory, you should be able to use this as a digital logic gate.

OK, and so that-- sounds-- looks pretty simple here. But for my Ph.D., it took me about three years to do something like this just to give you an indication. Now it's a lot faster. Now you can do this in-- you can do many of those in a day. So there has been progress.

Here's another one of those gates that I used for my Ph.D. And so this is now not just a repressor, but a repressor that can be inactivated by a small molecule, OK? And so the way it works is you have this repressor that works as before. And then when a small molecule comes in, it prevents a repressor from binding the promoter. And as a result of that, even if the repressor is present, you can have activation of the output protein, OK?

So this is what's called a not x or y or it implements the implies logic function. How many people use the implies logic function to do anything? OK.

So it's not a commonly used logic function. And you won't find it-- there's no logic gate that does the implies logic function in a typical computer. But this is a useful logic function that we can implement in cells.

And it allows external control of gene expression, OK? And so this is a simple way-- and so once you can do that as a user, essentially you can interact with the cells and modify what's going on inside the cell, OK, using a pretty simple looking mechanism that predates synthetic biology, if you will. But I don't know if it was called the implies logic function before.

So anyway, so then logic gates-- can we build logic circuits? This is where I would say synthetic biology starts kicking in. And so this is one of the first logic circuits that we built.

And the question was, OK-- looks nice to have this logic gate representation. In biology, does this make any sense at all? Can you really do digital logic inside cells?

Can you take noisy biological components and actually implement reliable digital computation in cells? And it wasn't an obvious thing, I would say. Is it 100% obvious now?

In some situations, I think we can claim that we can build digital logic that's reasonably reliable. So in this particular case, I'm showing you this implies logic function that allows us to have small molecule induction of a cascade of not logic gates or transcriptional repressors. And so the nice thing about this in particular is the fact that this is the input output steady state is that as the circuit so goes from blue to black to this yellow color here is as the circuit gets longer, as a cascade gets longer, it actually becomes more digital.

It actually becomes more step-like, OK? More on off. So we're going from this blue input output function to this yellow. So now we have over 1,000 fold change in the output in response to two to four fold change in the input.

OK, and then we have good noise margins, good signal restorations, all these good things that we need to have for the creation of larger and larger reliable digital circuits. So the basis of digital computation is that you have-- and the reason why you can actually create computers is that you can have logic gates that do signal restoration-- that the output is a better representation of the digital meaning then the input. So as the signal , propagates this analog signal could be voltage.

But it could be protein concentrations. As it traverses through the logic gates, it needs to actually become cleaner in order for us to be able to have reliable digital computation. So people have figured out how to do this with electronics a long time ago.

We figured out how to do it with synthetic biology, let's say, 10 to 15 years ago. And nature has figured out how to do this billions of years ago, OK? So things like cooperativity-- so I assume you've looked a little bit on cooperatively in, let's say, gene regulation.

OK, so that is a situation where you get a nonlinear response in a system that biology has figured out is a useful mechanism so that signals that come in actually result in some kind of actual digital behavior. So you get non-linear signal processing in these regulatory elements. And at the end of, let's say, a signal transduction cascade, the output is either high or low.

There's no-- for the most part, there's no in between. The transition between high and low is super fast. OK, so in a sense, that's creating digital or discrete outputs.

And that's really critical for many situations-- certainly in synthetic biology, but many situations in biology as well. So one example would be, let's say, stem cell differentiation. You want the cells to be able to make a discrete decision. Should I make-- should I become a kidney cell or a liver cell or a muscle cell and so on. So those are discrete decisions that have to be made by the cells. And so the cells have come up-- or nature's come up with mechanisms to guarantee that. And so we've now figured out how to do that ourselves in a synthetic fashion as well.

It's important to note that when we engineer these systems, we don't just think about digital behavior. So we spent an equal amount of time perhaps thinking about how to implement things that have transient properties or things that have more kind of analog behavior to them. And that's absolutely critical to be able to program cells to do whatever we want.

So this is an example where we have engineered cell-cell communication where sender cells make a small diffusible molecule which then goes to receiver cells. OK, so now the receiver cells don't just have an on response, but rather they have a pulse, OK? So a signal travels from the sender to the receiver cells.

And the cells, what we engineer them to do is have a pulse response. And the idea is to have GFP go up-- a Green Fluorescent Protein go up-- and then go down. And so to be able to do that, we engineered a feed forward motif where we have binding of the small molecule to this activator which activates two things simultaneously-- a green fluorescent protein and a repressor which then represses the green fluorescent protein.

And then-- so the idea is that the green fluorescent protein goes up. And then eventually, the repressor builds up to sufficient levels to repress the green fluorescent protein. So again, one of those simple looking motifs.

This is about three years to actually make that happen around the 2004 frame. Looks simple. If you study a naturally occurring system that has this motif, you say, oh yeah-- no problem. Yeah, we have this feed forward motif.

And obviously, you can do this kind of information processing function. Let's move on to another motif. You actually try to build this in a lab in a new organism, I'm not sure if it can drive you insane. But it is not trivial to actually make it work.

It's much easier now than it was 10 years ago. But you still have to pay attention to a lot of things-- rate constants, threshold matching, and so on to actually make it happen. But eventually, after looking-- creating-- so this is our first attempt at this was this blue line right here. So a completely flat line, OK?

Input comes in, nothing happens. So I would say that pretty much typifies synthetic biology maybe up until today. You build something. You think it's going to work. It doesn't work.

And then you stop crying after a little while. But then you have to think about how do I fix this. And so this iterative design debug cycle is absolutely critical.

So what you normally do is you create computational models that tell you how different rate constants in the system affect the behavior of your circuit, OK? For example, you could do sensitivity analysis. Which rate constants have the most influence on the performance of your system? And so we did some sensitivity analysis here and learned that, for example, the degradation of this repressor makes-- one of the things rate constants makes the biggest difference is on the performance of the system or its affinity to the binding site on its respective promoter. Yes.

AUDIENCE: Just knowing the sheer [INAUDIBLE] entire circuit, it started off with the [INAUDIBLE] constant?

PROF. RON WEISS: No. I wish it was. Because that would make life a lot easier.

And we are trying to get better at that. So we're trying to-- so here's maybe two ways of thinking about that. One challenge would be somebody comes in, gives you DNA sequence, and you have to predict the rate constants. OK, so I would term that person an adversary, not your friend.

It's just too hard to do that. Now, an easier task would be give your adversary or friend limited choices and say in the freezer, I have these DNA sequences that consist, let's say, of specific promoters, specific ribosome binding sites, specific proteins with specific degradation tags on the proteins, OK? And that's-- you're allowing that adversary or friend to only use those elements in the design of a circuit.

And then they come back to you and they say, now predict what the circuit will do. You still-- it still doesn't work yet. But I think-- but I would say that's how we would phrase the challenge, OK?

Stick to things that we know and allow us to even characterize those things ahead of time. What we have that-- unfortunately, I don't have that here. But what we have done-- so people can get a kind of a general characterization. So they can say it'll be roughly this input output behavior.

And when I say roughly, the errors could be on the order of five to 10-fold. That's approximately what's been published so far. Now, five to tenfold depending on your perspective could be great because it's biology or could suck if you're an engineer. It just depends on if you're trying to do something like get green fluorescent protein to turn on and off, tenfold is probably great if you're doing this in a Petri dish.

If you're trying to create a cancer classifier circuit you put into humans to kill cancer cells but not harm healthy cells, tenfold is probably not great I wouldn't take a circuit like that into me, especially if that circuit controls, for example, the production of a killer protein, which I'll try to show you a circuit that does that. We recently have been able-- we're in the process of submitting a paper about this-- been able to show that if you have really good characterization of regulatory elements such as these repressor devices-- and you have to do a lot more characterization and you do the-- we can get within 20% on average on predicting the behavior in mammalian cells, actually.

And so as an engineer, I would be happy with 10%, 20% percent for many, many applications. So I think we've gotten better at it. But it's not quite perfect yet.

One of the things about that approach is that we don't necessarily know the rate constant for everything. What we do know, however, is a very detailed behavior, both steady state and dynamic behavior, of a repressor promoter pair. So we don't know, for example, what's the binding affinity or what's the rate constant for the repressor binding the promoter, what's the rate constant for RNA polymerase binding that promoter, what's the exact translation rate or transcription right too.

But we do know what's the input output behavior. And that's actually been enough to get really good predictions. But I would say these kinds of predictions are one of the most important aspects and challenges and bottlenecks of synthetic biology. So those include-- again, the challenges include how fast can you build DNA. But wouldn't it be nice if you can actually predict what the DNA does?

So it's just as important, if not more. And also having-- so those are two of the important challenges. I'd say another one would be-- OK, so if you can predict things how many parts do you have in your freezer that you can actually put together and they're well-characterized actually now build the circuits? Probably three of the most important challenges.

And so if you go back to what we'll call the post generator, this is a loop tape of bacteria that now respond to the pulse. So sender cells that then secreted the small molecule then went into receiver cells. And they light up.

So one of the things to note here is that it works. The other thing to note here is that it's not perfect, right? And so the amount of heterogeneity here I think is quite astounding.

So if you take the average behavior, it's actually quite predictable. But if you now start looking at the distribution in the response, it's staggering. And we quantified that.

And so we quantified what's the distribution in terms of the fluorescence levels that-- the peak and also the buildup and so on. And we then correlated that also-- we created the stochastic simulations that then correlate reasonably well with the system. So we can get simulations to generally correspond with what we're seeing at the population level.

But I do want to bring up this point that when you think about engineering biological systems, don't try to figure out how to engineer a single cell to do something reliably, OK? So you always want to think about kind of statistical engineering. You want to think about, I'm going to create a circuit. And when I put this circuit into a population of cells, this is the distribution of behaviors that I'm going to get.

Because if you're trying to depend on any individual cell giving you exactly the behavior that you're looking for, it is just-- it's going to fail. So you have to really think about distributions. And that I think changes things a little bit.

So that's not normally the way you think about-- maybe that's the way Microsoft thinks about programming. So if 90% of the time, the computer doesn't crash, that's pretty good. Probably Bill Gates agrees with that, right?

But that's not what we want. That's not what we typically do with software. So to kind of further think about this in terms of populations, we program something else which was a pattern formation.

So now we have the desire to create senders and receivers where the senders send the same message to the receivers. Now we have a longer feed forward motif. OK, so this feed forward motif has two branches.

And so these two branches actually have a different impact on the final output. One has-- there's two repressors meaning that input comes in. It activates a fluorescent protein and another one represses the fluorescent protein.

OK, so it's an incoherent feed forward motif. And so what we use that here is not for post generation, but rather to define a range of concentrations that would turn on the final output, OK? So it would be activated-- the range of concentration would be activated starting with this branch right here and then ultimately repressed by this. So this defines the low threshold and this defines the high threshold.

So under the low threshold, nothing gets activated. Whenever you have just the right amount, it activates this which represses this which allows this fluorescent protein to get turned on. OK, so this branch right here is more sensitive.

So it defines when this thing goes up, when the response goes up. And then this is less sensitive. So this defines under high concentrations when the output goes down.

So we basically have a non monotonic response to the input, which is low then high then low. So that's the design that we had in mind. And the idea is that whenever you put, let's say, receiver cells everywhere in a Petri dish and you put senders in the middle, then the communication signal basically builds up.

There's a chemical gradient. Each cell interprets the chemical gradient and then decides whether to make a fluorescent protein. And then only-- because there's this steady chemical gradient due to diffusion and decay of the signal, then you would get some kind of a bullseye pattern.

So that was the hope at least. And so to give you again a timescale, so it took me about I think three hours on a plane to make the slide. It took us about three weeks to create the computational model. And again, for whatever reason, three years was the magic number to create the actual functional circuit.

So that was an older version of PowerPoint. But I haven't tried it on the new. But anyways, so we created this.

This is a computational model. We actually used a computational model to predict how changes in rate constants would affect this band detect, the region where we're actually responding to the signal. And we used that to engineer different responses.

And so we created eventually three different responses input versus output. And we put different fluorescent proteins on them-- a red fluorescent protein and a green fluorescent protein.

This is the experimental set up over here. And after 16 hours of waiting, this is basically what we got. So we got a lot of bacteria to make all kinds of patterns.

And we were very happy about this. We danced around in the lab a little bit-- you know, yay! So this was fun on those rare occasions where things actually work.

So we said, let's have some more fun. And so we put senders in other configurations. And so we have programmable patterns of bacterial communities. So I'm not sure of is this useful.

I'm not sure by itself besides having some fun with it. But one of the things we're using this for right now-- and depending on time, I may get to that later-- is this notion of embedding these circuits in mammalian stem cells or actually also in human IPS cells so that we engineer these human IPS cells to communicate with one another to make decisions. And then those decisions actually lead to differentiation patterns, right?

So you can imagine in principle, if you can create three dimensional versions of these and use those to cause the cells to make differentiation decisions so that red would mean make neurons, green would mean make muscle, different colors-- yellow would mean make bone and so on. So in principle, you might be able to create tissues by design.

OK, so that's something that we are working on actively in the lab right now. So we don't quite have a working heart in a Petri dish yet. We won't for a little while. But we taking some baby steps along the way.

And so we have been able to get cell-cell communication to work. We've been able to get programmed stem cell differentiation to work. And hopefully, I'll be able to show you some images that we have of some recent examples where we take human IPS and actually created these embryonic liver buds that have lots and lots of interesting-- and actually all the cell types that are known to exist in the embryonic liver.

So there are some progress along the way. Now, we don't anticipate to replace your liver, you know, any time soon. So don't destroy it. So actually one near term application that we're specifically looking at is if we can take-- imagine taking your own fiberglass, de-differentiating them into human IPS cells-- those are your human IPS cells-- and then differentiate them into, like, this liver-like environment and put that in a Petri dish and then test out the effect of drugs on your mini liver.

OK, so maybe it's a good idea to test drugs on things that resemble human tissue as opposed to some random mouse that may or may not be as correlated with what the drug would actually do to actual human cells. And if we actually even do it in the patient specific manner, I think that really changes the way drug development would actually work. And that's something that I think within the next few years could become a reality.

They're talking about the next-- beginning to do that within the next one to three years in a laboratory setting. So I think that is near term and realistic. So one of the things that we did notice is that when we engineer the small systems, its intuition works quite well.

So I can look at this circuit design and say, if I modify this, this is what's going to happen. If I modify this, that's what's going to happen. So you can use intuition and it works reasonably well.

And what happened in I would say the first 8 to 10 years of synthetic biology, every paper would have a computational design. But most of those would build something. And in order to publish, we also tacked on a computational model that correlates really well with the experimental results.

And we are just as guilty of doing that as anyone else, OK? So it wasn't critical to have a computational model to create something successfully in the lab. And I think that that is changing.

So we have examples right now of designs where-- I'm not sure if I'll get to that today, but we have published on that-- designs where it involves about 20 to 25 components. And it's related to a diabetes system the retired engineer where we can have intuition about it. But our intuition doesn't work great anymore, guys.

So we might have some intuition about the system. But the computational analysis would all of a sudden shed light and provide insight that is very difficult to get this by drawing this thing on a blackboard. OK, and so I think that computational design tools are becoming absolutely essential. They can provide insight into system behavior that you can't get just using intuition alone.

But in another aspect of computational design is one where imagine being able to specify want this behavior. And then the computational design tool says, here's

1,000 different versions of circuits that you should build and test. OK, and so it's still difficult for a human to generate easily 1,000 different versions of a circuit to build and test.

It is becoming easy to actually build-- I wouldn't say-- maybe easy is a strong word-- feasible to generate 1,000 versions of a particular circuit. I'll give you an example. Very recently in my lab, one of the graduate students has come up with a framework that-- he is generating 200 versions of a circuit in three hours, OK?

And they're pretty much gotten to the point where they're all correct. In three hours-- so that really, I think, changes what you do in synthetic biology. And so again, having that be connected to a computational design tool that tells you which ones to build would be rather useful.

And so specifically recognizing that-- this is a collaboration with some folks at BBN. And actually Jake Beal was one of my former graduate student colleagues. So he was in the same lab as me. And then Doug Densmore is from Boston University.

And so the notion here is that this is what we want synthetic biology to look like. OK, so if you're trying-- or maybe all of biological engineering. But we'll start with synthetic biology. So if you want to program biology, should you really care what the ribosome binding site is for the lambda repressor?

You know, hopefully not, right? What you should do, just like when you program your simulations in MATLAB, you don't think about, well, here's the shift register in this Intel Pentium chip. And this is how it's working to simulate this ODE over here.

That's just not the level at which your program. So you think really at a high level. And then you have compilers and lots of infrastructure that takes care of everything in between.

And so maybe someday in the future, the graduate students maybe five to 10 years will look back at synthetic biology graduate students now and would just have a lot of pity for them and, oh my god. You actually had to know what elements you were using in circuit and actually build them by hand? You know, wow. And so here's a notion that we start with a high level description.

And by the way, this is now-- there's a website that you can go through right now and get a free account and then type in a high level program. And it will actually create a low level genetic circuit representation. It will also give you MATLAB simulation files of this. And so, in this course I'm teaching to undergrads right now that many of them didn't even hold pipettes before they started the course, one of the first things that we taught them was this tool.

So before telling them, for example, this is the way the lac repressor works by DNA looping, we said, there are these things called repressors. And when there's more of them, there's less output. Now let's design with that.

This is, I think, heresy to biology as a whole probably. I don't know that-- this is the first time that we've actually tried to do that. And I think it's actually worked out OK.

But teach them enough so that they can move forward. And then, yes, let's simultaneously be teaching them about the underlying biology and mechanisms as much as they need to know. But what can they do if they just know that there's this thing called a transcriptional repressor?

And then the bio compiler will figure out everything that needs to happen. And so we're actually-- so they did a whole bunch of designs. And starting next week, we're going to be testing them out.

So we will find out whether that was a useful way to teach biological engineering. But I think so. I think it is.

Because they seem to understand what design means, OK? Because that's a thing that we focus on-- design as kind of a first class object. OK, so what you do is you write code that looks like this.

Has anybody programmed in code that looks like this? It should be-- so Lisp. OK, one person only? Any computational?

OK, we usually get one or two people. But I was hoping for more here. So anyways, for the people that should be ashamed of themselves and haven't programmed in lisp, this is a simple program.

This is, if the input is high, it produces cyan fluorescent protein. Or else, it will produce a yellow fluorescent protein. And so the bio compiler then automatically takes that and first translates that into a data flow.

So you have a data flow-- and I'll actually go through an example of that-- and then creates an abstract gene circuit and then looks at essentially what you have in the freezer and says, well, this is the actual DNA sequence that would implement this. And it creates robot instructions to assemble this so that God forbid you would have to actually touch a pipette to build this. And then we have a robot, liquid handling robot, that does most of the assembly.

Now, this pipeline is not fully end to end yet. So it's not-- if you came to my lab right now, the still-- I could tell you that this works but then I would be slightly lying. Is-- mostly works.

But it's actually-- there are companies right now that will go from this level-- about this level-- not this level, but this level. And actually, this is mostly automated. So-- to the point where the only thing that's not automated is somebody, not necessary your op, but maybe a technician goes from a robot and takes a plate and puts it in another robot. And, like, everything else is essentially automated in DNA assembly

So those companies have, I guess, more money than us. But eventually, this is the way DNA assembly will happen. So we're collaborating with some people.

I'm doing this with microfluidics. The problem with this robot-- it's $150,000. So we can't put it on everyone's bench yet.

But we are collaborating with Lincoln Labs to have microfluidic devices that would cost around $3,000 that would do this. And we've already demonstrated with the microfluidic devices we can do DNA assembly of large circuits. So this is not that far away that everyone will have microfluidics. They program here.

And they get the DNA assembled. And then they realize it doesn't work. But at least there'd be a lot faster to realize that things don't work, which is good.

So let's look at how this compiler, bio compiler, works to see that it's not magic. So this is saying green if not IPTG. So IPTG is a small molecule that we have a sensor for. And then the information gets routed to an inverter which then gets routed to activate a green fluorescent protein.

So you can take a program specified like this and automatically convert it into a data flow graph. And then the data flow can be translated-- again, this is automated and this is also automated in a rather simple way-- to an actual portions of the gene circuit. So IPTG sensor is simply this motif.

So you have a repressor that responds to IPTG. And then it basically inactivates the repressor. So more IPTG, more output. And so this is the IPTG sensor box. And this is how you implement this.

Not gate-- so I showed you how a not gate already works. So a repressor represses the output. Green fluorescent protein, you just have to have an activator that activates a green fluorescent protein.

So every data flow box can automatically be converted into a small motif. And then what we're missing is the glue. And so the glue is just these transcriptional activators and repressors.

And so once you put them in there, then this is a circuit. And so this is an automatic way to go from here to here. And it can do rather complex circuits and logic.

It can't do everything yet. But it can do an interesting set of things. OK, so this is-- again, it's not completely finished in the sense that I haven't told you what A is. I haven't told you what B is.

So there's a whole bunch of things that still are-- we have-- either published on or still need to be designed. But there is, for example, tool called matchmaker which will decide what's a good protein A, what's a good protein B. So things like that-- so those things already exist.

Anybody look at this and figure out why this is not perfect? So we have an input that represses a repressor, which means that more input, more activator-- sorry. More input, more protein here, which represses B, which activates GFP. So the compiler spits that out automatically.

But you may not want to build this right away. Any ideas why? I'm sure John would.

AUDIENCE: So basically, you have this A can directly repress object, so you activate B?

PROF. RON WEISS: So this is what it-- the first version of B. But A represses B. B activates GFP.

So why not just hook A up to regulate GFP directly? So this seems like a non-optimal solution. And so you can get the compiler to figure that out too.

And so what you do is you say copy propagation. So these are actually just tools that are available. These are mechanisms that are part of normal compilers-- normal software compilers.

So they do something called copy propogation-- means that A, if it sees that A-- the compiler sees A regulates B, A represses B, and B activates A. So you can just say, well, let A just directly regulate this. And then the compiler realizes, well, B doesn't do anything.

Let's get rid of it. And then the promoter doesn't do anything. Let's get rid of it. And now the compiler figured this out.

And that's using just basic compiler technology. OK, so it's able to do that. So the difference in your life between four and three may not be huge. But the difference in your life between 15 and five could be the difference between getting your Ph.D or going insane, dropping out, and then starting a company and becoming a billionaire.

OK, so-- but this is what the compiler can do. And that does make a difference. And it may be able to come up with optimizations that you wouldn't be able to really easily come up with or even at all under a reasonable amount of time. So the compiler can do combinatorial logic.

It can do state. There's also some aspects that can do spatial things as well. And again, this is available online right now.

I'm going to-- maybe I should skip a few things. So very quickly, I'll tell you. So in the lab-- so it's nice [INAUDIBLE] But if you can't do anything in the lab, then nobody believes you in the world of biology or synthetic biology.

So we can build big things. So here, you can-- there's a library of promoters and genes that we have available. And you can decide, I want to build a new circuit has these promoter gene pairs. And within five days, you can create in the lab that circuit. And we've been able to demonstrate things that are 61, 64 Kb that you build in five days.

And you build them efficiently. And then the undergrads that we're teaching also have been able to-- again, these are people to barely knew what pipette are in the beginning a semester can now efficiently build large circuits. So that's become an easy to use technology.

And then I mention this notion that you can build this one at a time. But this very recent development where you could build 200 versions of these at a time. OK, so you can build them.

We can then-- again, I'll skip this part. You could build them. You can put them into mammalian cells. These things work.

And we have lots and lots of parts-- regulatory parts. And so let me skip this particular example. So this is-- the two sentence explanation is, OK, you build lots of parts. You can build modules. And then you put modules together.

And guess what? This is biology. So these modules actually affect each other, potentially in undesirable ways. So they can place things like load.

So whenever you have a module that works really well-- maybe I will show you one slide. But I won't show you how we solved it. But one slide-- so this is a circuit. Any idea what this circuit might do?

So this is a regulatory circuit. They have an activator activating itself and a repressor that represses the activator. Have you seen this motif?

So it's a-- some people are whispering to themselves, doing this. So this is a relaxation oscillator. When you do it by yourself, it works great. So these are simulations.

But then if you connect it-- so it's nice to have an oscillator in a cell, again, if you want to have blinking bacteria or mammalian cells which is, again, fun. But then you typically want to connect it to something. So you spent three years building an oscillator, got it to work.

And now your adviser says, OK, let's just get a paper on this. But before we get the paper, I want you to connect it to something meaningful. Because we want to go for a high impact journal.

And then you connect it to something. And you realize that it doesn't work anymore. I think you're really pissed off at-- you will be pissed off at your adviser. I don't know if your adviser would want to be mean to you.

But anyways, so that's a real problem in biology or synthetic biology is that these things have impacts on each other that, first of all, are going to be undesirable but because of unique aspects, for example, of the substrate. Now, they're not as unique as you might think because these load issues also come to play in electronic circuits. And so in electronics circuits, these things have been solved decades ago with these notions of load drivers.

So if you have a module that has, for example, high fanout and it controls many things, then guess what? Those things that it's trying to control, even though the arrows are pointing one direction, they actually have an upstream effect too. So those modules that you think you're-- the downstream modules that you're just controlling, they actually have an impact on the upstream.

So what you do-- in electronics, you just build a load driver. And it basically takes care of things. So now there's no kind of parasitic effect.

And so we've demonstrated that we can also build-- so this is a notion of retroactivity. And this is work with Domitilla Del Vecchio here. And we've demonstrated that we can-- so I wont' go into the details here-- so we can actually solve this.

So this is a real problem even in simple circuits experimentally. And then it uses this cool notion of timescale separation to solve it. But we can take these things that are highly affected to go from black to red and put a load driver in and then it fixes it.

So this should hopefully lead to the generation of much more predictable circuit construction targeting one of the real challenges in scaling going from simple toy modules to large scale systems. So I'll skip that and then move on to another example-- oh, sorry, an example of an application. So in this particular case-- again, besides turning GFP on and off, what can we do with synthetic biology that we can't necessarily do without synthetic biology?

And so one of the most important challenges in cancer therapeutics is specificity-- perhaps the most important. There are other things that are important such as delivery of a therapeutic agent. But as you improve specificity that it can actually change how you do delivery of a therapeutic agent.

So if you have a therapeutic agent that's much more specific and has no side effects, you can deliver lots and lots more. So these things are highly related. So imagine a therapeutic agent that recognizes something on a cell surface and then says, this is a tumor cell. Kill it.

So there are actually a lot of efforts ongoing that have this particular approach. So whether it's small molecules, whether these vesicles that contain various cell surface-- various molecules that bind to cell surface receptors, a really hot area right now is these engineered killer T cells. So this notion that you can actually engineer your immune system by placing various receptors on these killer T cells which then go and then bind tumor cells by recognizing something on the cell surface and then kill those.

Sounds great, maybe, depending on your perspective, or not so great, depending on your perspective. So what's happened with those is they're really sometimes great at eliminating your tumors. But then the side effects can be horrendous.

And so what happens is that those cell surface markers that exist on the tumor cells-- guess what? They're also present on healthy cells as well. And so the side effects for those killer T cells approaches have been, I mean, just terrible in various patients. So one-- this should not be a shock. One marker is typically not enough to distinguish a cancer cell versus a healthy cell.

Seems pretty obvious. So actually, what they've been doing with these engineered killer T cells is now just saying, OK, has to be this cell surface receptors, but also cannot be this. So it's like, and biomarker one and not-- you know, biomarker one, and not biomarker two.

OK, so they're starting to engineer more logic into these, more multi-input logic into these things. And so they haven't done any clinical trials. But I saw a couple weeks ago where they've actually made progress in Petri dishes.

So we recognized this as an issue several years ago. And this is work with Coby Benson. And so really, all the information that you want is actually inside the cell to make a highly precise decision about whether this particular cell is cancerous or not.

OK, so the idea is when the therapeutic agent, does the computation by integrating multiple pieces of the sensory information and then decides whether to express a killer protein or not. Now, even if this is not the cure for cancer-- can't really guarantee that, right? But even if this is not, just this notion of being able to create multi input circuits to go into cells and analyze in live cells real time what's going on in the cell with various molecules that might be interesting-- that has applications I would again say just about anywhere you could imagine in biology.

So one of the things, for example, we're looking at-- I mentioned this notion of organs on a chip or programmed organs on a chip. So one of the things we're looking at now is placing these types of sensory circuits into cells within this organ on a chip. And so the idea would be expose cells to these drug candidates. And then the cells light up in different colors to tell you how they're responding to it.

So for example, certain color green would mean just fine. Green with red and yellow would indicate that some apoptotic pathway is being expressed or this drug is affecting this proliferation path or this pathway or that pathway. And so these mechanisms can tell you in real time in single cells-- also specially-- what kind of impact there is to, let's say, the drug candidate you're looking at.

So again, this is something that I think would have an impact today if it was available, but realistically making prototypes within the next year to three years.

OK, so we started with looking at HeLa cancer cells as an example. And so we did some bioinformatics. And we saw that based on microRNA profiles that were available to us, this would be a bio program that would distinguish HeLa cells from all other cell types. And so some of you are familiar with this way of annotating logic.

Pretty ugly if you're not used it, but it's simple. This just says, these microRNAs have to be low and these microRNAs have to be high. And that's a HeLa cancer cell.

Because that's the basic logic in this. So it's a rather simple logic statement. What we would argue is that every cell type would have a logic statement that would be true of that cell type and not true of other cells. It's almost by definition.

There's something different about that cell than other cell types. Now, it gets interesting when you start talking about heterogeneous population. So for example, and that's-- I won't get into details about that.

But there is heterogeneity. There's heterogeneity in tumors. So the cool thing you can do-- this is a six input and gate. So if you have heterogeneity, where you can do is you can have a six input and gate with an or operation.

So you can identify this sub population of the tumor has this microRNA profile. And this cell population has a different microRNA profile. And all you have to do is create a new logic circuit and just combine them as like a cocktail drug as an or gate. So this is a really general approach that I think should be relevant, again, for any kind of population.

And you can also set the thresholds and have all these fun things that you can do with it. So I think the question really is, is that microRNA expression profile sufficiently small so that it can be encoded on a circuit that you can deliver into cells? And can you do it reliably?

That really is the challenge. OK, so the idea is that you have a therapeutic agent, goes into a cell does the computation, and then decides whether to make a killer protein or not. OK, so how does this actually work? So how do we implement an and gate with inverted inputs?

OK, so that's one portion of the circuit. So it's actually-- for microRNA, it's actually pretty easy. So what you do is you put microRNA target sites on the gene of interest or the output gene. And so the idea is that the only way that this gene is being expressed, this output protein is expressed, is when this is low and this is low and this is low.

So in principle, this is pretty easy to do. OK, so that's how we have a three input and gate with inverted inputs. And now we can add logic to that. So now if we want to make sure that the output is high also when this input as high as well.

And so what we do-- so remember the first circuit that I showed you was this cascade. And the cascade was a bunch of repressors that would then cascade that did the not not operation. So when you think about the not not operation, it doesn't seem like a very useful operation.

But this here, the not not operation, is actually useful. So it can convert a microRNA sensor into something that can be integrated with other sensors to create the four input logic function. So this microRNA has to be high in order to repress this repressor which then represses the final output.

So the only time that the final output can be high is when this is high and these three are low, OK? And then you can continue. And you can add another one.

This is slightly more complex in the sense that now, essentially the sum of these microRNAs has to be high in order for this branch to allow expression here. So it's slightly more complex in the sense that it's not just-- it's like a plus-- again, like a plus operation, which we found to be actually the relevant operation here. And so the only time the output high is when this microRNA is high, combination of these two is high, and these are low.

Anybody see a potential problem here? A potential crosstalk? So everybody here understand the circuit?

Raise your hand if you actually understand the circuit. OK, that's not a lot of people. OK, maybe I should go back for a second. Everybody-- raise your hand if you understand this.

OK, so if microRNA is high, they repress-- they result in degradation of the RNA. So all of these have to be low in order for this to be high. OK, now if we had this, this RNA-- let's focus just on this maybe to simplify.

So this microRNA represses-- this is a repressor which represses this, OK? So if this microRNA is low, then this repressor as high. And as a result of that, this repressor represses this promoter regardless.

So it doesn't matter what these are. If this is low, this repressor is high. And then there's no way that this output can be high.

Now, if this is microRNA is high, then this becomes low, allowing this to potentially be high. Doesn't guarantee it, but it has a potential of doing. OK, now raise your hand if you understand that. OK, more.

OK, cool. That's progress. By the way, this was not a trivial circuit. There's a lot of connections here.

And then we connect the same-- now we have the same repressor. So this microRNA-- these set of microRNAs repress the same repressor which is now repressing this. So why is it the case that you need-- let's just assume this is one microRNA, this is another micro.

Why do you need both microRNAs to allow high output? And then I will not let anyone leave until you answer the question. And I can sit down.

So why-- and then we'll stop there. And everybody will believe me that the cure for cancer, you got it figure it out. Otherwise-- so there's-- oh, thank you. Letting-- you will be the hero.

AUDIENCE: So if the microRNAs are not present, then you're going to get an expression of Lacks E. So--

PROF. RON WEISS: Lack I.

AUDIENCE: Yeah, like eyes. So only if both the microRNAs are high and present do you repress both of the lack I's which then allow for the expression of the--

PROF. RON WEISS: Exactly. And what happens if one of them is present and the other one is not? So if this microRNA is present and this one is not?

AUDIENCE: Then the lack I still gets expressed because the other one is still not repressed.

PROF. RON WEISS: Exactly. So the only way that you can actually have no expression of lack I is when this is high and this is high. So this is essentially like a two input and gate.

These two have to be high to allow this promoter to potentially be high. And in the other three logic cases where one of those is low, lack I is expressed and hence does not allow the output protein to be expressed, OK? Yeah.

AUDIENCE: So why do you have the partition? All three are acting the same way.

PROF. RON WEISS: Ah, great question. So what happens if microRNA 21, 17, and 38 were all here? So that's a great question.

So that's a question, right? So if you put microRNA 21 here, 17, and 38? What is the logic function here? Yeah.

AUDIENCE: Then you only need one of them in order to degrade the RNA. And then you won't get the same logic.

PROF. RON WEISS: Great. So then it becomes essentially an or. This will be an or operation of the microRNAs where what we want is an and operation on the microRNA. So that by having two separate paths that have the same repressor where they converge on the same repressor, we achieve the and operation. Where we have them on the same repressor, they do the or operation.

That's a great question. OK, so this works. And maybe I'll say thank you. I'll stop there.