/mlst

Tau Language: The Software Synthesis Future [Sponsored] - Ohad Asor

March 23, 2026



Machine Learning Street Talk

Podcast Transcript

Revision: 0d2be958

Topic headings include retention indicators: • above average, • average, • below average (vs. similar YouTube videos)

Interested in licensing ReScript for your podcast? Get in touch with Tim Scarfe.

Contents

Overview 2

Fundamental Limitations of Machine Learning and PAC Learning Theory 3

Transductive Learning and the Three Curses of Machine Learning 5

Language, Reality, and AI System Design 6

Program Synthesis and Formal Verification Approaches 8

Definability, Computability, and the Halting Problem 11

Self-Referential Language and Tarski's Influence 16

Boolean Algebra and Logical Foundations of Tau 17

SAT Solvers, DPLL/CDCL, and the Revival of Logical AI 19

Program Synthesis and Specification in the Tau Language 23

Pointwise Revision and Belief Revision Theory 27

Quantifier Elimination and Normalization 30

User Control and Software Governance via Blockchain 32

Automation, Finance, and the Future of Blockchain Economics 38

References 46

Overview

Mathematician Ohad Asor makes the case that machine learning has fundamental theoretical ceilings – PAC learning guarantees only probabilistic approximate correctness, and beyond a certain problem complexity, neural nets perform no better than coin tosses. His alternative: the Tau language, a formal specification language grounded in Boolean algebra that can refer to its own sentences (something Tarski proved impossible in standard arithmetic, but Asor circumvents by abstracting sentences into Boolean algebra elements).

The conversation moves from the philosophical – Wittgenstein's limits of language, the gap between words and reality, knowledge as justified true belief – to the deeply technical: how SAT solver breakthroughs (DPLL + CDCL) have made logical AI practically feasible after decades of dismissal post-Lighthill, how program synthesis works by proving satisfiability of specifications rather than exhaustive search, and how pointwise revision enables live software updates while preserving consistency.

Asor describes Tau as a software specification language where you write *what* the program should do, not *how*. The system then synthesizes provably correct implementations – potentially in any Turing-complete target language. The interpreter already works; full synthesis was finalized roughly a month before recording.

The second half explores Tau's blockchain architecture: a governance layer where users collectively define and update software rules through logical consensus rather than voting, with applications to automated finance, tax systems, and decentralized decision-making. Asor frames it as addressing a fundamental power asymmetry – giving users genuine control over the software they depend on.

Tim pushes back throughout on the expert systems brittleness problem, the gap between formal specification and messy reality, whether logical and statistical approaches can coexist, and the political risks of a system that truly empowers users.

Fundamental Limitations of Machine Learning and PAC Learning Theory ●

Ohad Asor 00:00:00

Machine learning is a mathematical miracle. It can learn from examples and give correct answers in high probability for examples he didn't see yet. But this is only up to certain probability. The error will never be 0. Accuracy gets better only up to certain difficulty of problems. From a certain size of your logical question, it will do no better than coin toss.

Tim Scarfe 00:00:30

You're talking about the 3 curses of machine learning. There's the optimization curse, the statistical curse, and the approximation curse. You're describing a completely different approach. So rather than doing statistics, we use a logical method to deduce an answer.[1][2]

Ohad Asor 00:00:48

The Tau language is designed to achieve 1 and only 1 goal, which is how to make software to be controlled by its users. Right now, they they have very little control, if any, so new blockchains come and go every day. Tau is the end game of all blockchains. Every new idea that you want to incorporate into the blockchain, well, just say it in Tau, and it will automatically become what you want it to become. You just say it and it will happen. You tell the system, make profit for me, and it will make profit for you, right, of course, whenever possible. But the concept of automatic businessmen, that's that's strong stuff and that's only 1 aspect.

Tim Scarfe 00:01:34

Ohad, welcome to MLST. It's it's such an honor to have you here.

Ohad Asor 00:01:37

Thank you very much. My pleasure and honor. Thank you.

Tim Scarfe 00:01:40

Can you tell me a little bit about your background?

[1] Tau Language — Company
[2] Tau Language GitHub — Company

Ohad Asor 00:01:43

I've been mathematician and software developer my my whole life. I've been interested in various fields of mathematics. But in the last 15 years, mainly in the field of AI at the beginning, machine learning and the theoretical foundations of machine learning. But in the last about 10 years, mainly about the intersection between logic, mathematics, and computer science.

Tim Scarfe 00:02:15

So what was the moment for you when you felt that this logical view of AI was really really important?

Ohad Asor 00:02:23

Well, I I got familiar with the whole idea of mechanized reasoning. And normally, mathematicians view mathematical logic as a very boring subject, and I was no exception. But when I heard how algorithms can kick in and how logic can be mechanized by computers, this became really very interesting.

Tim Scarfe 00:02:51

So I've been I've been going through your Twitter post so hard and in particular you've been talking about just how unreliable these approaches are. I mean, you know, tell me more.[3]

Ohad Asor 00:03:03

Yes. So machine learning is a mathematical miracle. It is very surprising that it can do what it does, which is to learn from examples and give correct answers in high probability for examples he didn't see yet for out of sample instances. But this is only up to certain probability. There is certain probability that the error will be low enough. The error will never be 0 and the probability for 0 error error will never be 1. And that's why it is called PAC learning, probably approximately correct learning. You will probably be approximate, namely low error. But that's it. You will never get guarantees for absolute correct correctness. And sometimes you need guarantees for absolute correctness. And moreover, sometimes you want to communicate to the machine in a way that is not effectively captured by examples. Like now, we are talking, we are not only giving examples to each other. That would be crazy. And indeed, it would be crazy to only give examples and accept and expect to get true intelligence out of it. You need to say your things, not only exemplify. So it is very impressive that machine learning can do what it does by examples, but we are

[3] PAC Learning Framework — Paper

pretty

Transductive Learning and the Three Curses of Machine Learning

Ohad Asor 00:04:50

much at the peak. It will not get much better than that. You need methods that are not machine learning, that will go beyond guessing for examples.

Tim Scarfe 00:05:02

There's always a bit of cognitive dissonance here for me. So we've had, you know, Noam Chomsky on the show, Gary Marcus, Walid Saba, many many famous symbolists, and they've always said reasoning sans guarantees is not reasoning. And OpenAI have been boiling the frog for the last few months with the o series models. What we are seeing, even though these are statistical empirical methods, the accuracy is going up and up and up, and we can build engineering systems to make them better and better and better, and things just seem to be getting better and everyone just seems to be accepting it. Why do we need to have these guarantees?[4]

Ohad Asor 00:05:42

Well, accuracy gets better only up to certain difficulty of problems. So if so so from a certain size of your logical question, it will do no better than coin toss. It will be completely random. So for example, the the boolean satisfiability problem, which you can imagine a set of constraint of the form if this guy comes to the party and this guy does not come to the party, then the third guy will come. So a set of constraints like this. And such solvers can solve things like this with more than thousands of variables. But even o 3 or o 30 or o 300 will not be able from certain let's say hundreds of variables to give better than random answer. It will be coin toss. It's not that in 90% of such instances that's this kind of problem. It will return the correct answer. From certain point it will be random. 1 way to see it is if you are familiar with polynomial interpolation like like Lagrange interpolation, you can take any set of points and find the polynomial that fits this set of points. So if you have 1000 points, it will be a polynomial of degree 1,000. Will you be able to take if you have a time series of something and you fit it with the polynomial to predict the next time point. Well, this will be completely random. This is captured by theoretical

[4] Boolean Satisfiability Problem — Paper

machine learning as infinite VC dimension. If you can fit infinite time points using a polynomial of degree 3, that's very surprising. That's probably something fundamental about the nature of your setting and then you are you have much more confidence that you will be able to predict in good probability and low error the next time points. But if you just fit everything, this is what is called overfitting, you will be random.

Tim Scarfe 00:08:10

So intuitively, I agree with you. But what we have seen really in the last 6 months actually on on Francois Chollet's ARC challenge is people have been leveraging transduction. And that is simply a prediction function where the test data specification is part of the prediction function. So rather than having this inductive model which memorizes all of the training sets for the purpose of generalizing to any new unseen example, This transductive approach embraces the situated complexity in the world. So when a model needs to do a prediction, it incorporates situated knowledge inside the prediction function, and it's a way of kind of embracing the complexity that

Language, Reality, and AI System Design

Tim Scarfe 00:08:57

that's out there. So of course, it's still just a new form of empiricist prediction, but it seems to be far more capable of of making reasonable predictions in given situation.

Ohad Asor 00:09:09

Forgetting how you reach the prediction function, what is your model, what is your training set, you just look at the at the end result and you can see right away the the computational limitations of such prediction functions no matter how you reach them.

Tim Scarfe 00:09:28

Yes. But I am a fan of this local form of prediction. Because if you think about it, machine learning methods as they stand today, they are maximum likelihood estimators. So what they are doing is they are generalizing over these huge statistical distributions, and in every situation they're always giving you the regression to the mean. They're always giving you the general answer, And there is a lot of optimization just using local information to get a more specific and more

predictive answer for you.

Ohad Asor 00:09:56

If your concept class, the class of prediction functions is all possible polynomials, like we said before that you can fit everything, this class is too big. Its VC dimension is infinite, and there is no surprise that if you do Lagrange interpolation over arbitrary data points, its prediction ability will be 0. It will be completely random. If you take a smaller class, then if you fit your training data, you're not guaranteed to fit it. But if you fit it, then you have a lot more confidence, not guarantee, that you will fit out of sample.

Tim Scarfe 00:10:36

Well, no. This is interesting. You're talking about the 3 curses of machine learning. There's the optimization curse, the statistical curse, and the approximation curse. So we can choose a smaller hypothesis space, and if it's too coarse, we we get approximation error, and we have the statistical error, of course, which is just, you know, fitting a statistical model to this class of functions. And you're describing a completely different approach. So rather than doing statistics, we use a logical method to deduce an answer. But what's the trade off? What do we lose by having a logical methodology instead of a statistical 1?

Ohad Asor 00:11:23

For 1 thing, the information has to be there. So if all your information is examples, that's not enough. You need to actually say the thing. Machine learning is good for precisely the cases when you don't know how to say the thing. For example, face recognition. No 1 can put in words how face look like. You you can see and tell right away but you cannot put it in words for things you cannot define. For this machine learning is very good. But for things that you can define, then go ahead and define them. Why go by example?

Tim Scarfe 00:12:06

Is it possible though that this is a kind of dichotomization? So many people say machine learning is great for things that I can't write code to do. I can't write code to recognize a face or recognize a cat or recognize a digit, and there are many things in the world that we can describe logically using language. But isn't isn't that too much of a separation? Aren't there situations where we could actually decompose face recognition into a program? Or we could combine pattern recognition and logical reasoning into some amalgamated form? Does it make sense to think of them as 2 completely distinct domains?

Ohad Asor 00:12:47

Before speaking about computers, there is a big dichotomy between the world and the things that we say about the world.

Program Synthesis and Formal Verification Approaches

Ohad Asor 00:12:58

The things that we say about the world are just things that we say. The finger pointing to the moon is not the moon. There is a very big difference. So this dichotomy is already there. Now, as human beings, we are obsessed with living in the world of language. The conceptual world, the things that we say about the world, the language is even more central to our life than the real physical world. Here, what are we doing right now? We are talking. Right? And for us, this is actually doing something, but we are only talking. Now there is nothing wrong with that. That's our human nature. We live in the world of concepts, in the world of language.[5]

Tim Scarfe 00:13:51

Well, mean, let let me extend a little bit. So knowledge is a justified true belief. And as you say, what we need to do is reason about knowledge that we actually know to be real. What's happening with the machine learning world at the moment is we are being possibly gaslighted into thinking that reasoning is more general than what the Greeks thought. So that there was a famous Monty Python sketch where they were saying, is someone a witch? And all these people were saying, oh, she's a witch because she got out of bed on this side, and she's got a pointy nose, and she's got funny hair or something like that. And this sounds like reasoning because we're constructing all of these different rules and we're composing them together, but there's no surface contact with reality.

Ohad Asor 00:14:37

You're right. So indeed, this dichotomy between the world and things that we say about the world, they are pretty much disconnected. To demonstrate this, we cannot even define what it means to physically exist. If we cannot define that, then how can we define anything else in the real world? Even to exist, we don't know how to put in words. So there is a very serious dichotomy between

[5] Knowledge as Justified True Belief — Paper

language and the world. However, in our human experience, language is as real as the world if not even more real. So if we were some different kind of creature like aliens that may be very advanced but they are just not obsessed with with language the way we do, then maybe logic wouldn't be interesting. But we are humans. We do reasoning using language, And that's why it is so relevant for us.

Tim Scarfe 00:15:45

You said that language feels very real to us. But there's there's a phenomenal component of real and then there's there's a, you know, like an epistemic component of of real. Certainly many of the things that we talk about in language, we know we know to be true. They are facts. They are things that exist, cities that are placed in a certain country. And then I suppose that there's a spectrum beyond that of things that we just feel to be true.

Ohad Asor 00:16:15

Yes. So of course, as humans, we have some connection to the real world. But when you say when you say I will go to the shop to buy to buy tomatoes. Right? Tomato is only a word. It is not connected to the physical tomato that is there in the shop. Got nothing to do with it. You have no idea what is that physical tomato in the shop. You cannot even point to it until you reach to the shop and you you take it. But until then, you got the idea of tomato. Right? And nothing more than that. Right? And also, I say about myself that I'm a mathematician. Well, It's just a word. In reality, I'm just a guy with glasses and no hair. That's what I am in reality. Mathematician is only a word, but that's what matters to us as human beings. We live in the world of language.

Tim Scarfe 00:17:17

Well, in which case then there is a potential divergence between words we use and what they mean and how they correspond to the real world. So the concept of a tomato, this is an example of abstraction. It's a it's a category which has formed over possibly a constructivist evolution of language use or maybe it's just a fact of the matter about the universe. I mean, did that how did that concept come about?[6]

[6] Wittgenstein on Limits of Language — Person

Ohad Asor 00:17:44

So indeed, language can somehow point to the world but it can never touch it. And to show you it can never touch it, let's go to the question we asked before. Can you define what it means to physically exist? No 1 can. And if to physically exist you cannot define, you cannot put in language, then you cannot even put all the rest. So all you can do is to approximately point but never touch.

Tim Scarfe 00:18:12

So what does it mean then to build AI systems when even by your own description there's little surface contact with reality? Even even in the most formal descriptions of things.

Ohad Asor 00:18:24

What we are really interested is not in reality, but in reality as perceived by us. And we perceive stuff largely by language. And this, we can implement in a computer. Moreover, how are we going to communicate with a computer? Well, by language. Because we are creatures of language. So it's much less about the world as it is. It is much more about human nature. We want computers to serve us as humans given our very peculiar human nature.

Tim Scarfe 00:19:03

Only potential objection with that is I I think of language as a living organism that supervenes on us, it rests on us, it's always adapting and changing and so on. And when you extract a formalism, because natural natural language isn't a formal language. So when we formalize it and put it into an AI system, wouldn't it very quickly diverge from the language that we use and and know?

Ohad Asor 00:19:28

It will and indeed when I say language, I'm not pointing to any specific language. I'm referring to the abstract concept of saying things, of expressing things in a symbolic way like letters or not always not only letters. So if you ask me where is this and that street and I tell you it's there, that's also language. Right? It's signs that describe something whether real or or completely abstract. So I I by no means refer to the specific nature of the specific language. I'm talking about the general idea of abstraction.

Tim Scarfe 00:20:12

There seems to be a relationship between symbol use and abstraction in general. And humans are different from all other animals because we have this declarative labeling ability. So we have this plasticity about how we can point to things and, you know, give them labels and share those labels with with other people. And of course, the use of those labels gives rise to this abstraction. And certainly in machine learning models, they don't do symbols and they don't do abstraction without hacking. We can wire them in this recursive loop and we can give them chain of thoughts and we can kind of make them act as if they are doing symbol type stuff, but it's not it's not very natural. It comes very naturally to us, doesn't it?

Ohad Asor 00:20:58

Yes. So indeed,

Definability, Computability, and the Halting Problem

Ohad Asor 00:21:00

if you want to do logical reasoning, why go so indirectly and encode it in in some linear algebra instead of directly doing logic? Yes. I completely agree with you.

Tim Scarfe 00:21:14

The the big thing that that you're advocating for is verification. Being able to have guarantees.

Ohad Asor 00:21:21

Oh, I'm I think verification is old news. The real deal is a synthesis. In verification, you describe a system and you know describe statements about the system. And then you verify that the system meets those statements. In synthesis, you describe only the statements about the system, only the requirements, and then you automatically generate a system that meets those requirements. 1 way to look at it, the common practice of programmers is to write tests, which are programs that test your main program and make sure that it returns the right answer. So to demonstrate a synthesis, imagine that you write only the test and then the computer automatically synthesizes a program that will make the test pass.[7]

[7] Theories and Applications of Boolean Algebras — Paper

Tim Scarfe 00:22:19

Then there must be some kind of erroneous behavior that my smartphone might do that we want to guard against it doing. And wouldn't our system of I mean, we called them tests, but of course we're talking about epistemic tests. We're not talking about statistical tests. But when we describe the behavior of the smartphone, it needs to make sense. I mean, does it mean for for the for a test to make sense?

Ohad Asor 00:22:43

Yeah. So let's say, I want to make sure that the phone will never send over the network my passwords. Right? Now a test for that which is which you cannot implement in reality the way I'm gonna say it is try all possible inputs for the phone, all infinitely many possible inputs, and check if at 1 point it sends your password over the Internet, and then the test fails. How are you gonna test something over infinite domain? Well, by the magic of mathematics. Right? Sometimes on mathematics you can prove something about all infinitely many numbers. So if you describe this test and synthesis problem in the right mathematical framework, then you will be able to give guarantees about all infinitely many possible inputs.

Tim Scarfe 00:23:43

Devil's advocate on that though. We see an interesting phenomenon with, you know, on on Instagram and Twitter and and so on. There are all of these constraints. So you're you're not allowed to to say you know, you're not allowed to refer to women's for example. And what happens is people can get around the filters by using all of these new language terms to describe female as as an example. Isn't it the same with a password that I could encode the password in an infinite number of ways and wouldn't that be able to circumvent the system? That's right. You could,

Ohad Asor 00:24:20

but it was only an example. Right? So let's say that you don't send the password in clear text as it is. Right? Even this, how are you gonna do that?

Tim Scarfe 00:24:33

But but doesn't that just create an infinite regress though that we wouldn't be able to describe the uncountably many ways that people could send a password?

Ohad Asor 00:24:42

So I I could give you an idea which I'm not saying it's practical, but 1 way is that the password is stored somewhere and you can track where it is stored and see how it how it transforms and whether it transforms into 1 to man or many to 1 function. And if it is many to 1 then, okay, how many? And then see how the scrambled password goes over the network and whether it is possible to recover the password by whatever definition of possibility of recovering that you give to the system. So yeah, that's 1 wild way to to implement such a thing, but it is not the point of the example. The point of the example is that you want to say something about all possible configurations of the system, all possible inputs, which are basically infinitely many. And to be able to get a guarantee for that, you will need very specific mathematical frameworks. So it is not going to be a test like normal programmers do. Or let's say this way. It could be a test like normal programmers do, but that runs to infinity. That tests all possible inputs, so it's gonna run forever. And then you are gonna ask the halting problem for this specific test. Will it ever find a bug? So you will need to write this test in a language in which the halting problem will be decidable.[8]

Tim Scarfe 00:26:26

Yeah. The the only thing I I can't get my head around is I mean, just coming back to that example before. So someone might come up with a new term for women's. So it might be calcium cannons or something, you know. So you can invent anything. And any reasonable language user would be able to perform inference in that situation and say, that that that's what they mean. But wouldn't we have a similar issue with, I can't write code to do facial recognition because people's faces are changing all the time. There's always an exception that that makes the rule. So what we're doing is is we're using a logical framework to describe an amorphous living thing, and it needs to adapt continuously.

Ohad Asor 00:27:06

Yes. Well, so there are 3 moving parts here. There is the computer world, there is the human world, and there is the physical world. I know how to treat the computer world. How to treat the human world or the physical world, that's beyond me. I can give you guarantees about what the computer is gonna do. So for example, you want to start your own bank and you you need a software for your bank and you wanna say no balance can be below 0 unless it is authorized. Right? You you really want you really want this. You really need this. Right? How are you gonna do it? Are you gonna ask all 3 to give you your bank software? Let's be real. Even the biggest LLM proponents will never imagine doing

[8] The Halting Problem — Paper

such a thing. Right? You really cannot trust this. So what are you going to do? You're going to do traditional programming, normal programming? Well, that's what people really do. But then how can you make sure that the balance really is is really never negative? Well, you will need to go over the whole places in the code where balance is updated implicitly or explicitly and then make sure that it is not negative without authorization. But in the synthesis for requirements way, you simply just put the sentence. Balance is greater than greater or equal than 0. And that's it.

Tim Scarfe 00:28:49

That makes sense. I mean, isn't there always an exception though? So that, you know, let's say next Friday there's this really interesting case where a banker from Zurich, he has a special arrangement with the bank. So he's allowed to go 0 under these conditions, but no 1 else is allowed to. And I can imagine a world where we can decompose it into pockets of regularity, Where there are situations that can be quite rigidly defined, and then the rest of it is just chaos. So I I can see specific situations where we could build systems like this, but it seems like sometimes we need to have the flexibility as well.

Ohad Asor 00:29:28

Yes. So now you are going a step forward from writing software into how software changes. And that's of course also very important, no less important. And indeed, the synthesis framework that we work on on the Tau project heavily involves how software is being updated.

Tim Scarfe 00:29:53

Yes. Maybe we should get to that in in a little while. But coming back to a few interesting questions here. So you haven't been influenced by Godel, just Tarski?[9]

Ohad Asor 00:30:03

No. Also not really by Tarski. I mean, what does it mean to be influenced? No. I you know, when I open a book mathematics, that's something that I do regularly, right? Of course, I'm influenced from the things written and I write, but

Tim Scarfe 00:30:18

Would it be fair to say that you think Tarski is 1 of the best logicians of the twentieth century?

[9] Alfred Tarski — Person

Ohad Asor 00:30:26

Everyone will agree to that statement. Yes.

Tim Scarfe 00:30:28

In in which case, why why would you not say you are influenced by him?

Ohad Asor 00:30:33

Well, I cannot say that he changed my course of action. It did turn out that our course, the the big Tarski and the small me had some similarities in the course of actions, but I discovered this only in retrospect.

Tim Scarfe 00:30:53

Right. So your your journey into this world, was it very much self guided? And then and then afterwards you contextualized it?

Ohad Asor 00:31:04

Yes. I'm always very original and not always right and not always bright, but always very original.

Tim Scarfe 00:31:14

Is that a better approach than, you know, looking at what's out there and being influenced? Do do you think it's a better approach just to do it from first principles?

Ohad Asor 00:31:25

Well, I I always survey the literature. I never try to to do things from first principles if I can find sources that can help me and continue from there. Sometimes there is no choice like when I needed to have a language that can refer to other sentences in the same language, there is absolutely nothing out there. I had to to do it from scratch.

Tim Scarfe 00:31:54

So you've

Self-Referential Language and Tarski's Influence

Tim Scarfe 00:31:55

said also that pure logic based AI is essential for safety. What gives you the conviction to say that?

Ohad Asor 00:32:02

That's not exactly what I said. I did say that a language that can refer to sentences in the same language is essential for safe AI. That's the more key point. Right? Because also common logical AI, common logical framework which are not machine learning. Also they cannot do safe AI without this component.

Tim Scarfe 00:32:26

That makes sense. So there is a specific form of logical AI in which a language can refer to its own sentences.

Ohad Asor 00:32:33

Yes. And I'm the only 1 to discover this. Yes.

Tim Scarfe 00:32:36

So that specific form of logical AI is essential for AI safety?

Ohad Asor 00:32:41

Yes. It's specific in logic in general. There is no logical language that can do it. Yes. This is achieved by abstracting sentences into boolean algebras.

Tim Scarfe 00:32:53

I think the core of what we're talking about is in the 19 eighties, there was this notion of the expert systems knowledge engineering bottleneck. And the folks back then, they they were building these systems, and they were describing the world using a series of rules and logical language and and so on. And the problem is, just like it's very difficult using code to do facial recognition, These folks started building these these ontological frameworks to describe, you know, domain based systems in in the enterprise. And they found that a form of brittleness emerged, that they always needed to put more and more and more special cases in. How is your system different from that?

Ohad Asor 00:33:36

That's very simple. I have no intention to describe the whole real world. I have intention to make computers do what you want what you want them to do. To accurately describe the world, well, that's too much, and it's also not very necessary. We want to describe the things that

Boolean Algebra and Logical Foundations of Tau

Ohad Asor 00:34:00

we are interested at to achieve our human goals.

Tim Scarfe 00:34:07

Okay. So does does that mean then that there is a class of domain based systems like safely flying a plane that we could describe using a logical framework? But there might be other classes of things like, for example, ensuring that people don't post racist remarks on a social media site. That might be beyond the capability. Where do you draw the line?

Ohad Asor 00:34:32

Programming languages can express anything. Anything that computer can do. Well, and then you could ask me, if they can do anything, then please write me a program that detects racist statements in social media. Because I just said that it can do anything. Well, if I promised anything, then you can come and tell me, okay, please do the anything. Well, that that's exactly the dissonance. Right? You can do anything as long as you know what what you want. You can put in an accurate description. You can give a definition of what is that you want. Then you can achieve it. But if you don't if you cannot come up with a definition, then maybe choose machine learning. And you can give examples, and you will get a less accurate system, but it's better than nothing.

Tim Scarfe 00:35:29

But what is what is the bright line between things which we can describe in language and things that we can't?

Ohad Asor 00:35:35

I can understand this question in 3 flavors. The whole field of mathematical logic speaks about which mathematical structures you can define in language. And here you can have a simple countable, uncountable argument because a language will contain sentences. Each sentence is a finite sequence of symbols from finite alphabet. So there are always only countably many sentences and mathematical realities go way beyond the countable cardinality. So of course, not every mathematical structure is definable. In fact, very few of them are. Second flavor is to speak about what is computable. And then we have indeed computability theory, complexity theory, and we can speak about what is computable and what is not computable. We know for example that the halting problem is not computable and there are computable numbers. There are of course only countably many computable numbers. The real numbers are uncountable. In other words, almost all numbers almost all real numbers are uncomputable. And if you choose a real number at random, then in probability 1, it will be uncomputable. And the third flavor is what in the real world we can we can define in language. And this goes back to what I told you before. If we cannot even define what it means to physically exist, then the answer is nothing. Nothing in the real world you can define. Maybe you can approximate. Maybe you can define a diff maybe you can give a definition that is good enough for another human being to understand what you mean. Like if you tell someone, get me tomatoes from the shop. They can do that. But it's not a definition of the physical tomato in the shop.

Tim Scarfe 00:37:52

Yes. So so you're saying that subject to the constraint of definability, computability, and language, there exists a subset of problems which are amenable to this kind of logical description.

Ohad Asor 00:38:05

Yes. And of the most practical interest is most probably to define what computers should do, to define programs by logical means. So going back to the bank example, you want to say the balance should never be negative. And you want to just say this, balance greater or equal to 0 and that's all. You don't want to go over the whole control flow of the program and check whether balance can be 0. We just want to put this constraint and be guaranteed that it will be met.

Tim Scarfe 00:38:43

So another thing you've said is that in the last 20 years or so there have been advances in, you know, computation and and and, you know, the kinds of things we can do with computers that make this more plausible, make make it more realistic. Can you tell me about that?

Ohad Asor 00:38:59

Yes. So the big AI winter in the seventies was all about logical AI. It came from the Lighthill debate. It's available on YouTube. I recommend everyone to watch it. And you can see mister Lighthill lecturing and recommending to the British government that because of the complexity of logical reasoning, they should defund AI, and in particular logical AI. And it was on the time that the class of problems called NP complete problems where even the most basic logical questions are at least NP complete hard, were believed to be absolutely intractable. That even small NP complete problems will never be able to be solved.

SAT Solvers, DPLL/CDCL, and the Revival of Logical AI

Ohad Asor 00:40:00

There was, up until today, no theoretical proof of that. It was a belief that turned out to be wrong. Now we have an empirical proof, not a mathematical proof, that even very large such problems can be solved very fast. And this in the last 20, 30 years, the field of SAT solvers suddenly started to surprise all researchers showing that what they believed cannot be done can be done quite easily. And since then, they there is like a Moore law of such solvers. They just become better and better and they continue being so. They still improve. So we have empirical proof that logical reasoning is feasible in practice in contrast to what was thought before. And that's why the time to invest and build and research logical AI is right now. Now that we know that we were wrong in abandoning abandoning it, It was not because it was not good, not promising. It was just thought to be very difficult.

Tim Scarfe 00:41:19

I mean, was a similar thing with with deep learning that everyone wrote it off in the 19 eighties and then suddenly we have all of these GPUs and, you know, some algorithmic tweaks and data and so on and and it's become tractable. So you're talking about constraint satisfaction problems. What are the key improvements that have made them tractable?[10]

Ohad Asor 00:41:42

Well, it's it's some boring stuff. It's a combination of an algorithm that is called DPLL with another algorithm that is called the CDCL that eventually turned out for reasons no 1 knows why to solve SAT very fast. It's basically a bunch of quite trivial heuristics that no 1 thought that they are going to solve difficult problems, but it just turned out that they did.

Tim Scarfe 00:42:12

Do you see a potential hybrid between deep learning systems and logical based systems, or are you advocating for purely using logical based AI for specific scenarios?

Ohad Asor 00:42:27

I advocate for purely using logic, and whenever statistical reasoning or machine learning reasoning is required to express it in the logic. After all, if you open a machine machine learning book, it is written in logic, in something equivalent to logic. Programs that do machine learning are programs. They have their own logic. So I advocate for implementing machine learning inside logic.

Tim Scarfe 00:42:58

What sorry. What does it mean to to implement machine learning inside logic?

Ohad Asor 00:43:03

So any machine learning algorithm, you can describe it in logical languages. Right? And then you get the benefit that this logical description lives inside the broader logical description. So it's just part of the whole thing that you do. And you can augment it with also guessing and machine learning capabilities.

[10] DPLL and CDCL SAT Solving Algorithms — Paper

Tim Scarfe 00:43:30

That makes sense. So pragmatically you're saying we can combine the 2 modalities by almost creating a compiler, if you like. So we describe what the machine learning problem is using the logical language, and then we generate or synthesize the machine learning framework, but still within the construct of logical

Ohad Asor 00:43:51

AI. Yes. If all you want to do is only machine learning, then do machine learning. But it never comes in a vacuum. Alright? So let's imagine our bank software. It is mostly not about machine learning. It is mostly not about guessing. But sometimes you want to do some guessing, then alright, express this method inside your logic.

Tim Scarfe 00:44:20

Okay. So that means we could use the logical framework as a form of verification that could be used in line with a machine learning algorithm?

Ohad Asor 00:44:30

Logic is a form of description. What you do with the description, whether you verify or synthesize, that's a different question. Now, normally, machine learning does not come in a vacuum. You want to describe some system where machine learning is part of it. It's not all of it. In logic, you can do that. You can describe your whole system including the machine learning part.

Tim Scarfe 00:44:55

Makes sense. But what you're saying though is that the logical language is the source of truth. It's very similar to there were these object relational mappers for building databases, and what they did is is they might say code is the way we describe database models. So we don't want anyone building databases directly. There are no SQL statements. No one's talking to the database server. We now describe database access code using, let's say, c c sharp or whatever, and we abstract away the database. So you're kind of saying we abstract away all of the machinations of the software. All we need to do is describe it with logical language. Describe the requirements, the the what and not the how. I you describe the test that the software has to pass. You don't need to describe the software itself. Now this to me seems like a galaxy brain idea, which is that we can describe declaratively what the software does. Why is there not an infinite space of possible implementations that could be synthesized from those requirements?

Ohad Asor 00:46:02

There's an infinite space. Normally, there will be infinitely many programs that meet your specification. And if you are fine with it, with the back end eventually choosing 1 arbitrarily, then that's good. If you are not fine with it, then you have to put more constraints on in your spec.

Tim Scarfe 00:46:21

So are you describing a workflow where we create the requirements, we we synthesize 1 of the infinite number of possible implementations, we observe the behavior, and we already know that the program is correct because the whole point of of your system is that it's correct given the requirements. At what point will we realize that we need to make a change?

Ohad Asor 00:46:45

Well, either that you had a bug in your specification, you wanted to specify something but you didn't, or reality changes and you want to adapt to reality.

Tim Scarfe 00:47:00

And I suppose that might happen in 1 of 2 ways. We might just realize that our specification has changed, or we might see from the behavior that we didn't get the specification right in the first place. Yes, of course. So can you explain in simple terms how it is possible to create a program automatically from a set of requirements? It seems like so many people are trying to do program synthesis, and the naive way of doing program synthesis is you just have this exhaustive exponential search over the space of programs. I'm guessing that's not what you do.

Ohad Asor 00:47:39

No. Because the space is infinite, so of course no exhaustive search can cover all options. So for this you need a mathematical framework

Program Synthesis and Specification in the Tau Language

●

Ohad Asor 00:47:50

where you can give statements about the infinitely many objects in in in your framework quite easily. So let's say an example from a different field of mathematics is if you have a polynomial over the complex numbers, if it's not constant, then you know that it has a 0. That's the fundamental theorem of algebra. You don't need to check all infinitely many complex numbers. You are guaranteed that it has a 0. And a similar thing happens in this in the mathematical framework of the Tau language. You say statements about infinitely many sentences And by relatively simple checks, you can get a guarantee that something exists. And you can even get an algorithm of how to obtain 1 example of this thing. Let's give a very simple example that is not about the Tau language but a very simplistic example. Suppose you have to output a number and you said the number has to be bigger than 10. There are infinitely many numbers bigger than 10. How are you gonna choose? Well, that's not so hard. Just because the search space is infinite, it doesn't mean it's impossible. Right? So of course in the Tau language it's way more sophisticated than that, but that's 1 example.[11]

Tim Scarfe 00:49:24

How did you overcome Tarski's undefinability of truth in defining your language?

Ohad Asor 00:49:29

Yes. So I identified the real life necessity of having a language to speak about its own sentences. Beforehand, I don't think anyone identified the practical need of it. They saw it as a philosophical gadget. Tarski proved that in a certain setting it is really impossible, in a certain very broad setting. But because of the practical necessity of it, I tried by all means to find a way how to do it in a way that is good enough in practice. And the way I found is to abstract sentences to be only Boolean algebra elements. So we forget about the structure of the sentence, we look at it only as a boolean algebra element, and then it all works.

[11] Tarski Undefinability Theorem — Paper

Tim Scarfe 00:50:19

Maybe we should just for the audience here introduce boolean algebra and also Tarski's undefinability of truth theorem.

Ohad Asor 00:50:26

Yeah. So Tarski's undefinability of truth is if you take the language of arithmetic of natural numbers with addition and multiplication, and of course any language that contains this language. So this language or any bigger language. And you add to the language the truth predicate. Now what's the truth predicate? And the truth predicate is something that takes a number, after all it's after all it's a theory of numbers. And this number is some encoding of another sentence in this language, in the language of arithmetic with the truth predicate. This kind of encoding of sentences is called a Gedel number. So the truth predicate will get a Gedel number of some sentence and will tell you true or false whether this statement is true or not. And Tarski doesn't ask you to implement this truth predicate. He only assumes that it exists in your language. And then from this assumption, he was able to recover a contradiction. So a language like this will be inconsistent. You will be able to prove every statement and its negation. Now for Boolean algebra. Boolean algebra, the the the I will give 2 ways to define Boolean algebras. 1 way is a Boolean algebra is a bunch of sentences where the operations defined between the sentences are and or not. That's the only operations. Another definition is a Boolean algebra is a bunch of sets where the only thing you can do with these sets is union intersection and complementation. Most importantly, you don't have the membership predicate. You can't look at the elements inside the set. You have only union intersection complementation, and I forgot, equality. Also in the sentence definition, you also have equality which would mean semantic equality whether the sentences mean the same thing, not necessarily written in the same way. That's a Boolean algebra. So the theory of Boolean algebra will be quantifiers for all and exist, and then and or not an equality.[12]

Tim Scarfe 00:52:50

That's right. How do you compute whether 2 Boolean algebras mean the same thing?

[12] Boolean Algebra Foundations — Paper

Ohad Asor 00:52:54

You don't. You just take this for granted. So if you want to work with the language of Boolean algebra over some Boolean algebra of choice, well, you need to implement this Boolean algebra of choice. You need to give me the end or not inequality, and if you give me that, then I can take a sentence in the language of Boolean algebra and tell you whether it's true or false in your Boolean algebra.

Tim Scarfe 00:53:21

So you've defined I mean, you've invented the tau language, and it overcomes Tarski's undefinability theorem by talking about how it can refer to its own sentences. And you've abstracted sentences into these Boolean algebra elements. Can you talk about some of the fundamental building blocks here? So there's simple Boolean functions, Boolean functions, point point wise revision. How do all of these things operate in the Tau language?

Ohad Asor 00:53:51

Yeah. So the Tau language is the standard theory of Boolean algebra with certain extensions. So first extension is in this language you can write any Boolean algebra element because in the standard theory of Boolean algebra, the only thing you can refer to directly is the 0 and 1. But in this extended theory of Boolean algebra you can refer to any element. That's first extension. Second extension is that the Boolean algebra it is talking about is nothing but the Boolean algebra of tau sentences of itself. The third extension is the temporal dimension. So when we speak about the Boolean algebra elements, we speak about how they evolve with time. So you can say this statement now compared to the statement 5 steps ago. And it is not only temporarily, there is also a distinction between inputs and outputs. And this thinks the temporal dimension and the inputs and outputs are necessary in order to make this language a software specification language. This distinction between inputs and outputs is a very strong property because in the Tau language you can prove that at each point in time, for all input exist output that meets the specification. And this is not something you can do in other specification languages. You can do that only if your inputs and outputs come from a fixed finite domain. But here the domain is infinite. The domain is all tau sentences. So that would be another extension. And another extension which you mentioned is point wise revision, is about how to update Tau programs. If you want to change your specification in a live software, how to incorporate only the change and keep all the rest intact.

Tim Scarfe 00:56:05

Okay. So so so the first thing is we've created an algebra that can, for any point in time, for any input, guarantee that the output matches the specification.

Ohad Asor 00:56:18

It can well, it can check. Right? So you have your specification. Maybe your specification does not guarantee for all input exist output that meets the specification. Well, we can check your spec. And that's what we mean when we said the spec is satisfiable. It means that for all input excess output at each point of time. If your spec is satisfiable, then we can synthesize a program that meets your spec.

Tim Scarfe 00:56:49

And how do you take that step? How do you synthesize a program from a boolean algebra?

Ohad Asor 00:56:55

So this is very complex. So at the beginning, we had I knew only how to do interpreter, not a compiler, So I could execute your specification, but I couldn't put on the table a program for you to run. But I could still execute it. And this is also not trivial at all. In fact, the full synthesis algorithm I finalized only about a month ago. And now I know how to put a whole program on the table.

Tim Scarfe 00:57:29

Can you tell me about that? So does does your system support, but I'm just imagining here, are you possibly generating some c plus plus code and then compiling it? Yes. So so there's almost like a transpilation process. Yes. And in principle, it doesn't have to be c plus plus it could be any Turing complete language.

Ohad Asor 00:57:51

Yeah. That's right. Yes.

Tim Scarfe 00:57:52

And have have you gone with c plus plus I mean, what what what guided your decision?

Ohad Asor 00:57:57

Well, in the company we work with c plus plus I'm a fan of c plus plus so it will be in c plus plus Right now we have implemented the interpreter which can execute a specification, but we didn't implement yet this synthesis.

Tim Scarfe 00:58:12

Is there a thing where efficiency isn't being respected? You know, like for example, I train a deep learning model to recognize a face and it works, but it's hideously inefficient. It's it's just taken some galaxy brain weird. It's generated some weird circuit. Is there a thing here that when you generate a program, it it it's not necessarily the most parsimonious program?

Ohad Asor 00:58:38

Yes. It's a big thing. It's a big difficulty. And it will be an ongoing effort. I do know how to identify cases in which things can be executed fast. And moreover, the the user, the 1 who writes the specification, can write it in a form that it will be executed fast. There is also the possibility of using SATO SMT solvers, although it is not clear at this point that it will make things better. And there is of course the ongoing optimization effort. But yes, I guess it will come down to mainly identifying the easy cases and synthesize functions that don't need to do logical solving. You do the logical solving beforehand and you output a straightforward function that just takes the input and gives you the output.

Tim Scarfe 00:59:47

Is there a relationship between the complexity of the requirements and the complexity of the generated program? Yes, of course.

Ohad Asor 00:59:59

Not

Pointwise Revision and Belief Revision Theory

Ohad Asor 01:00:00

in all cases. Sometimes you can take something very simple and write it in a very complex specification. But if you speak asymptotically, as the size of the specification goes to infinity and in the worst case, will the complexity program of the program will go to infinity? The answer is of course, yes.

Tim Scarfe 01:00:23

So there's there's a huge skill component I suppose to defining the requirements in such a way to comp you know, to collapse as much complexity as possible before you compile the program?

Ohad Asor 01:00:35

Yes. But it is shadowed by the very strong simplification algorithms that can be implemented.

Tim Scarfe 01:00:46

Would the simplification algorithms be implemented at the compiler level or at the requirements level? What I mean by that is you might be able to identify certain patterns of bad design. So I I place these requirements into the Tau language and you might rewrite those requirements to be more efficient, but then perhaps they are less intelligible to the person who wrote the requirements. What's the trade off there?

Ohad Asor 01:01:15

Oh, yes. So when we normalize or simplify formal statements, they might become less intelligible. Sometimes yes, sometimes no. Right? If you take a very very big speck and you normalize it into 1 line, that's very nice, but it will not always be the case. So yes, human readability and explainability is will be an ongoing effort.

Tim Scarfe 01:01:48

So I suppose that sketches out a future where we should think of the Tau language as being a kind of interface. So at at the interface point, needs to be maximally legible to humans. And then there might be some intermediate minification or normalization. And then there'd be another step of of compilation where it'd be optimized again.

Ohad Asor 01:02:10

That's how it works. Yes.

Tim Scarfe 01:02:12

Yes. Very interesting. 1 of the really cool things that you've done is this concept of point wise revision and that allows you to kind of change the system while respecting as much of the old specification as possible. How does that work?[13]

Ohad Asor 01:02:28

Yes. So generally the revision problem is unsolvable. There is a whole field called belief revision that speaks about this impossibility. So for example, if you say if your old knowledge base says a implies c or b implies c, and now you want to revise it with the new knowledge that says not c. So now, yeah, you have any subset of the following options. Either you delete the law a implies c, or you delete b implies c, or you say not a, or you say not b. Right? There are many ways. There is no way to optimally choose 1. So revision is really impossible. But in a certain setting, it is possible in a very clean and optimal way. In the setting of the Tau language, which is a software specification language, if we focus on what matters. And what matters is what output should the program output now at each point in time. If you focus on that, then you can perform the optimal revision. You simply write a tau formula that says, if exist an output that satisfies both the old and new specification, then choose it. And if such output does not exist, then choose 1 that satisfies only the new specification, and that's all there is to it. Pointwise revision is 1 of those those things that when you see the answer, it looks very easy. But before you see the answer, it is really very hard. So let's say you have a a big program with a lot of moving parts, with a lot of features, and you want to change only 1 small thing in specific small component out of many, do you need to write the whole software from scratch? Well, no. You want to write only this change. And indeed in using point wise revision, you write only the thing that you want to be new and the rest will remain intact.

Tim Scarfe 01:04:47

But I suppose 1 of the advantages of the logical based approach is global consistency, which is this idea that here's the system and this is what it does in all situations. And when you start placing I'm gonna I'm gonna use the word local, you know, which is to say, in this time, in this situation, you do this, otherwise, you do that. Doesn't that, if anything, create illegibility? It makes it harder for people to understand as a whole what the thing does.

[13] Belief Revision Theory — Paper

Ohad Asor 01:05:19

Yes. And a partial remedy is the normalization process. So for example, in point wise revision, I said if exist an output that satisfy both. Okay. So the keyword is here is exist. That's in logic. This is called a logical quantifier. And in atomless Boolean algebra, we have a quantifier elimination. So we know how to make this exist, disappear. And that's only 1 example. So during the normalization process, everything will be crunched together, all redundancies will be removed. So it will be more intelligible. I cannot tell you that[14]

Quantifier Elimination and Normalization

Ohad Asor 01:06:00

it will be fully intelligible.

Tim Scarfe 01:06:03

But then then there's the thing that you are essentially rewriting the rules. Doesn't that then mean that the, you know, the humans that created the specification, it's now being transformed into this slightly alien form that they no longer

Ohad Asor 01:06:19

recognize? Yes. It can happen. It happens all the time in in standard programming, and yes, it is something to mitigate. There there is no magic solution. No 1 can mathematically define what it means to be intelligible. If you could define it, then we could implement it, but no 1 can define it. Right?

Tim Scarfe 01:06:44

Isn't isn't it fascinating though? Because in in normal software programming, there's this cognitive interface between the software engineers and what the computer does. And we come up with these high level abstractions. We might use the mediator pattern or the observer pattern or all of these different design patterns. And weirdly, the way the code actually works on the computer doesn't resemble those abstract patterns very much at all. It's just a cognitive interface. But it would be very strange if my Python interpreter rewrote my code for me and said, actually, this is the way I'm gonna represent your code. Because what that means is my my the provenance, my mental journey of getting to that place has now been scrubbed, and I can't then make subsequent steps in that cognitive

[14] Quantifier Elimination in Boolean Algebra — Paper

space.

Ohad Asor 01:07:32

Python interpreter does rewrite the code for you. It just doesn't show it to you, but it does.

Tim Scarfe 01:07:37

Oh, I see. So you're talking about rewriting the code, but still maintaining the rules as the developer created them.

Ohad Asor 01:07:45

Okay. It's your choice. Right? Now, the way Python rewrites your code, it's always horrible. You never want it in this way. But the way we rewrite your Tau specification, sometimes it's even better than how you wrote it initially.

Tim Scarfe 01:08:02

Yes. I suppose what I'm getting to with this, always I mean, certainly if if if you think about how a c plus plus compiler does optimizations to make algorithms run faster. The optimizations always seem very alien to us. But are are you making the argument that optimizations in this language space, they actually seem even better to us than the original thing we came up with?

Ohad Asor 01:08:29

Can be. Not always. Can be. Sometimes. So 1 example I already gave you, which is quantifier elimination. Yes. So all the for all and exist that you will write, you can make them disappear.

Tim Scarfe 01:08:42

How do you see the Tau language being incrementally adopted existing software development workflow? So let's imagine we're not gonna throw everything out and start again. We're going to incrementally adopt it inside systems, you know, subsystems. How might that look like?

Ohad Asor 01:09:00

Tau language is designed to achieve 1 and only 1 goal, which is how to make software to be controlled by its users. Software is there for 1 and only 1 thing, which is the users. Right? How can users control the behavior of the software and change it with time? Right now they they have very little control, if any. If you want any software to be controlled by the users, you will need to use the Tau technology. You will have no other choice. You have no other choice. And it all comes to what we spoke about software specification and point wise revision that the user can say, I want this to hold and then the whole old specification will remain intact except this part that the user wants to change which will takes take place. So this whole thing again was built for only 1 reason, To make software controlled by the users. And this is something that you want in virtually any software. And there is no other way, only the Tau way.

Tim Scarfe 01:10:19

This makes a lot of sense. I suppose that there's a bit of an ideological here, which is that you're a fan of decentralization and a fan of allowing

User Control and Software Governance via Blockchain

Tim Scarfe 01:10:30

just the people who use software to control it essentially. So at the moment, we have these centralized teams of developers and they decide what features to build into software. And you're saying in the future we'll have software which is actually written by the users that that use the software.

Ohad Asor 01:10:51

Yes. And and moreover, it's not only about decentralization. I am the user. I am the boss of my computer. I decide what runs off on my computer. Why should other people decide for me? I want this power. Right?

Tim Scarfe 01:11:07

Okay. But how could that work? So 1 way of doing that is you write your own software that runs on your computer. But if we broaden out a little bit, there could be software which you share, you know, you're a user and thousands of other people are are users, and you collectively improve that software. So you decide as a collective what the best new features are to add to the software, and there's some kind of coherence mechanism if you like, and new requirements get

added over time. How does that work?[15]

Ohad Asor 01:11:43

That's right. So indeed, that's another step which is also very fundamental to the whole TAU project. A software that is collectively being controlled by all of its users. And in particular a peer to peer network. Net blockchain network that is controlled by all its users because what is the alternative? The alternative is what we already have today. A blockchain network, an economy that is controlled by a small group of developers. How can that have wide enough shoulders to support a real economy? Well, it just doesn't. I'm not saying that the other extreme is is the way to go. I'm not saying that it should be controlled by all of its users in a completely equal manner. Maybe not. Maybe it should be some kind of meritocracy. I don't know. And for this reason, we invested a lot of effort in how to formalize laws of changing the laws. How to be able to change not only the program, but also the way that program is being changed. And only in this way that the governance mechanism governs also itself. There is a way to change the governance mechanism itself. Only in this way you can have a blockchain network that can reliably hold economy in the wider sense that lives in society. It's not just a coin in a vacuum. There is a whole society, a whole market, a whole economy around currency. There's 2 components to this. The first component is using the blockchain to create a decentralized and coherent or consensus based approach to writing software. So it's not that Tau uses blockchain. It's that blockchain uses Tau. Tau doesn't need blockchain, but the blockchain needs Tau. The blockchain need to be redefined over time by the users. And like any other software, you cannot have a software that is defined in a sound way by its users without the Tau technology.

Tim Scarfe 01:14:13

So you're saying use the Tau language to define the blockchain? Yes. Well, so I'm trying to understand that. Okay. So we we we use the Tau language to define the blockchain, and there's some kind of consensus algorithm which itself could be defined with the Tau language. And what is the currency? You know, blockchain has a coin, has a currency. How how is how is that related to everything we've just been discussing?

Ohad Asor 01:14:42

Well, so when we say software controlled by its users in general. Right? The software has to do something. Right? It's not only controlled by its users, but it has to do something. Right? And this thing is also controlled by the users. And in

[15] Tau Net Blockchain Platform — Company

blockchain, so that's what it does. It does cryptocurrency in a way that is controlled by the users. So if you want to change how the crypto cryptocurrency behaves or if you want to make other things, you can do it. Alright. So for example, smart contracts, right, Just 1 aspect of users controlling the system. So as a user controlling the blockchain, you say, I want this behavior to take place and it will take place.

Tim Scarfe 01:15:37

Okay. But I'm I'm just trying to understand here. So before we were talking about I'm using the Tau language. I create specifications for software which is synthesized. And now we're talking about using the Tau language to essentially, a as a collective in the blockchain network, decide how the blockchain network is transacted. But is there an intersection between those 2? I mean, what are we doing here? Are are we are we designing new ways of controlling blockchains? Or are we using the Tau language to create new types of software?

Ohad Asor 01:16:16

To create new types of software in particular, this blockchain software. Yes. 1 is a special case of the other.

Tim Scarfe 01:16:23

Okay. But but you're primarily focused on the the blockchain flavor of this?

Ohad Asor 01:16:28

So new blockchains come and go every day. Because they supposedly offer this feature or that feature that they were supposedly not offered before. Where is it gonna end? There are thousands of blockchains. Where is it gonna end? Well, it's gonna end with a blockchain that however you wanna change it, it will change on the fly. And that's why I wrote an article on Twitter that Tau is the endgame of all blockchains. It is indeed. Every new idea that you want to incorporate into the blockchain, well, just say it in Tau and it will automatically become what you want it to become. So 1 use case that we speak about is a trading knowledge. So you could say in in the very same way that you say anything about the software, controlling the software as a user. So 1 of your statements would be if someone sends me an answer to this question, and you will have to define what it means to be an answer to a certain question, then they will get I will give them coins. Right? You just say it and it will happen. Right? Or another concept is the automatic businessman. You just say, this is the assets that I have. This is the things that I want. This is the deals I'm willing to to take part in. And all of this you

don't have to say even explicitly. You can say it implicitly. And then you tell you tell the system, make profit for me. Software that does what the user wants. Of course, users are gonna tell it, make profit for me. And it will make profit for you, right, of course whenever possible. But the end, the concept of automatic businessmen, that's that's strong stuff and that's only 1 aspect.

Tim Scarfe 01:18:27

Okay. Because many of the audience might not know much about blockchain technology in general. So just to make it clear that the technology is, I suppose, the most efficient form of structured financialization. So you can create a market. I'm sure many folks have used smart contracts before, so you have certain guarantees. You can say, here's a piece of software. It runs on the blockchain with certain guarantees, and it's decentralized, so we have this web of trust. And and you're saying with the Tau language, we can actually adapt the blockchain. We can make it do what we want to do while retaining the guarantees that we had before with blockchain technology.

Ohad Asor 01:19:05

Yes. Yes. So to contrast Tau with other blockchains that support smart contract smart contracts would be mainly in 2 ways. So if you can have smart contracts that let's say are updatable, you can replace the contract with another contract. It is still not the case that the code of the blockchain itself is a contract. And in Tau indeed they are the same level. They are the same thing. Second difference is to be in existing smart existing smart contract languages such as programming languages. There are not specification languages that can speak about other sentences in the same language. So in the Tau language, you could have a contract that says, the balance can never be 0. You don't have to go over the whole flow. And you can also say, if some other contract implies, implies as a logical word, implies certain reality, then do this and that. So that would be 2 major differences between the smart contract aspect of Tau compared to traditional blockchains.[16]

Tim Scarfe 01:20:24

So very broadly, are you saying it's a

[16] Tau Whitepaper — Paper

Ohad Asor 01:20:27

type of meta blockchain? Yes. But the Tau is its own meta language because it speaks about its own sentences. Tau is its own meta language. Now in meta language where usually people speak about meta language, the meta language so there is the meta language and there is the object language. The meta language can do basically anything it wants with object language. But Tau is its own meta language restricted to Boolean algebra. That's the only thing it can do with itself. So yeah, the Tau blockchain is its own meta blockchain.

Tim Scarfe 01:21:13

This is making a lot of sense. So you're saying the reason why we have this proliferation of blockchains is because they are essentially handcrafted. So people decide I'm going to code these smart contracts. They write them with code, and there's a kind of brittleness. And you're talking about a blockchain that has not only a meta programming and amorphous capability, but that behavior is controlled by the users.

Ohad Asor 01:21:43

Yes. That's right. Yes.

Tim Scarfe 01:21:45

So that's that's what you've built?

Ohad Asor 01:21:48

Still not ready, but almost ready.

Tim Scarfe 01:21:51

When you're gonna roll this out?

Ohad Asor 01:21:53

It's a good question. I don't want to to give any promises. I was too optimistic in the past. I thought that it will be already ready by now. Unfortunately, it's not ready yet, but the hard parts are behind us. I'm sure you can appreciate that the hard part is to implement the engine of the Tau language, and this is ready to a large extent. There is still more still more work on it, but also the hard part of this hard part is over. So yeah, I I don't want to give any specific time estimation, but it's coming.

Tim Scarfe 01:22:32

Do you see this as the single blockchain to rule them all or do you think there'll be many tower blockchains?

Ohad Asor 01:22:39

I think that so because it is controlled by the users, the users will also set the incentives. And I expect the users to disincentivize forking and incentivizing keeping it all at the same place.

Tim Scarfe 01:22:57

Okay. So when when you launch this new blockchain, people will be able to buy tokens?

Ohad Asor 01:23:03

Yeah. People are able to buy tokens already right now. The Agora's token, AGRS. But when we launch the system, the users will be able to control all aspects of the blockchain and the token.

Tim Scarfe 01:23:16

Okay. So people can already buy it. So the the blockchain's been launched, but it doesn't have the new software running on it?

Ohad Asor 01:23:21

No. The blockchain is not launched. It is currently a temporary token implemented over Ethereum that will be swapped to the full token when the blockchain is ready. So what people can buy right now is the temporary token that will be swapped.

Tim Scarfe 01:23:39

What do you think about the angle of collective intelligence? So do you think it's possible using a system like Tau when implemented in in the blockchain that it might be able to solve problems that individual brains can't solve. So there's there's a kind of collective work angle if if you like to solving big problems.

Ohad Asor 01:23:56

Yes. Just like software development.

Automation, Finance, and the Future of Blockchain Economics ●

Ohad Asor 01:24:02

You know the famous quote that what 1 developer can do in 1 month, 2 developers can do in 2 months. Because development doesn't scale, but there is more brainpower. So imagine many thousands, millions of developers in a way that can scale. They just say the requirement and the system crunches it all together. So of course, we are going to get a software of scale and quality never seen before.

Tim Scarfe 01:24:39

What happens when people just vehemently disagree with each other?

Ohad Asor 01:24:44

It all depends on the laws of changing the laws. Right? Which is yet another thing to be decided over the system.

Tim Scarfe 01:24:52

So do you think that there could be a social benefit to a system like this? So for example, right now we have echo chambers and people are not really aware of their logical contradictions. Do you think a system like this could actually help us get closer to various truths in the world? What it more directly brings us to achieve is

Ohad Asor 01:25:13

a computer that does what we want it to do. Right? So computers already do stuff that we want them to do, but not everything and not perfectly. It will take software to the next level of satisfying the users. What does it mean to be a good software? It means to do what the user wants. That's what it means. And to be a better software, it means to do more of what the user wants. What would be the best software? The best software will be a software that the user tells it what to do and it will do it. So by this definition, which is not so crazy, by this definition, Tau is the best software. It will just do what the user wanted to do.

Tim Scarfe 01:26:02

You know, Steve Jobs famously said that people would want faster horses and people don't know what they want. What do you think about that?

Ohad Asor 01:26:14

It is true. So indeed the definition that I gave is a bit simplistic, but it is already a big deal.

Tim Scarfe 01:26:22

But but I suppose maybe we're not giving the users enough credit. So maybe if the users actually had interactive control over the requirements that generated the software, maybe initially they would start building faster horses, but they would learn pretty quickly as Apple learned what would be a better course of action.

Ohad Asor 01:26:44

Indeed. So the collaborative approach of Tau is not only that people put requirements and are combined together. It is also about a setting of discussion. People having discussion about what software should do and indeed 1 of my old definitions of Tau is, which is still relevant, is Tau is a discussion about Tau. And indeed, on Tau, people discuss what Tau should be like and Tau becomes the consensus of the discussion. So in a social setting, people discuss what they they want to happen. And then the system builds what we call an opinion map, which is a mapping of all opinions in the discussion, which is a special case of which, which contradicts which, and so on. And the consensus of the discussion becomes the software update. And even what it means to be consensus, that's also definable. That's the laws of changing the laws.

Tim Scarfe 01:27:44

Do you think that'd be a learning curve for users of the system to define requirements using this logical language?

Ohad Asor 01:27:52

Yes. Yes. Of course. Definitely at the beginning, you will have to learn the language. Now it's not such a complicated language. It's quite a minimalistic language, but just because you control the language doesn't mean it's easy. Many programmers control certain programming language. It's still very, very hard to build software. Studying the language is the easy part. Yes, there will be learning curve and we will try to make it easier over time by using so called controlled natural language and by better interactive tools that help you to express yourself. So this is going to be an ongoing effort.

Tim Scarfe 01:28:35

It's interesting what you said about language. So you're almost saying that the the actual domain understanding is harder than this the syntax or the the technical understanding of writing in the Tao language.

Ohad Asor 01:28:46

Yes. Just because I speak a little bit English doesn't mean I'm Shakespeare. Right?

Tim Scarfe 01:28:53

And do you think there might be cul de sacs? So are are you worried about the system becoming stuck because people initially thought that this thing would be a good idea, and then it transpired that something else would be a good idea. But now because of the way the system is designed, we can't we can't change it.

Ohad Asor 01:29:11

So indeed, Tau is all about change. It's designed to change. Now you can change it in a way that prevents other changes. There is nothing to do against that. If I give you the power to change, then you also have the power to avoid change. By the way, you also have the power to never avoid change. This you can also do.

Tim Scarfe 01:29:36

Yeah. I'm just thinking about some I mean, some religions for example, they had error correction built into them, which meant that they just stayed the same and didn't really change for millennia. And other religions were were far more dynamic and and they're splintering and and mutating all over the place. So do you think that it's a bit of a lottery that depending on the trajectory of change that the system might be quite sclerotic versus if the the chips fell down slightly different, it might be, you know, a very different outcome.

Ohad Asor 01:30:08

Yes. It's like suppose, let's imagine that a new country is formed and this country is starts with the democracy. And I ask you what would be the laws of this country in 20, 50, hundred years from now? Well, no 1 knows. Right?

Tim Scarfe 01:30:27

But then what if people said we need to have some kind of a veto? You know, what what what if the system became so entrenched and it didn't serve the the desires of the users anymore, but unfortunately it just has become locked in because of, you know, previous decisions.

Ohad Asor 01:30:45

It can be. Yes. But they have logical AI helping them to have guarantees of what can and cannot happen. They have monetary incentive to keep the system. It is a collective effort. It's not just a mistake of 1 person. Right? This is the risk you should accept when you want things to change, but this risk is mitigated by all these things.

Tim Scarfe 01:31:15

I suppose it's quite interesting because in democratic capitalism, Noam Chomsky famously said in Manufacturing Consent that we have uninformed consumers making irrational purchasing decisions. Now, course, in TAO, you can say that we actually eliminate the the deception that people can actually see why things are happening in a certain way. But I suppose the main philosophy though is that users know what's best for them. There are people who say we need to have experts and we need to protect people from themselves. What do you think?

Ohad Asor 01:31:52

Well, I I would guess that many users will delegate their vote quote unquote. There is no voting in Tau. There is no need for voting. But delegate their voice to experts that they trust and maybe Tau will end up as a form of meritocracy.

Tim Scarfe 01:32:13

Given the learning curve, do you think that we might use something like language models or some syntactic sugar or some way to make it easier for people to program, you know, rules into the system?

Ohad Asor 01:32:26

Yes, and indeed 1 consideration that we have is to use LLMs to translate from natural language to the Tau language. And, yeah, we are not working on it right now, that's definitely something on our list.

Tim Scarfe 01:32:45

What do you think governments will think about this? Do do you think they might try to subvert a system which is so democratic that the users are in control? I don't know. Is it something that concerns you?

Ohad Asor 01:33:00

No. Well, there there there is nothing too scary in the system. Right? It's blockchain that is controlled by the users. Why should it worry anyone? Well, what would be the existential threat for such a system?

Tim Scarfe 01:33:18

I mean, how how is it protected from interference from other actors?

Ohad Asor 01:33:23

As it will be deployed, it is not protected by interference of other actors, but the users will have to set rules. Right? And we will start with test net, not with main net. So we will start with a blockchain with fake coins that the users will set the rules. And because it is only Testnet, there is no harm in restarting it. So we will let the users play with the rules and over the system come up with some initial rule set. And if they reach a deadlock, we can just push a button and restart it from scratch. And this would be a very useful sandbox for the users to decide how the governance should be like, and then this can be implemented over the mainnet.

Tim Scarfe 01:34:16

Well, first of all, do you do you agree that the philosophy of the system is that right now there is a power asymmetry? So that there are folks that take away our agency And let's just say agency means the ability to control the future or our future. And a system like this is emancipating in the sense that it gives users more agency. But usually when people get too much power and agency, other people want to take it away again.

Ohad Asor 01:34:49

For 1 thing it it can be, but then it will be a choice of users to give up their agency. For another thing, there is a whole other dimension to all this, which I guess is much bigger, which is how program is being collectively defined and updated. You can do it of course in traditional coding, but it's like stone age. Right? It's it's very very limited and very very difficult. But when you just say what you want to say and it automatically becomes a computational reality, that's a different level.

Tim Scarfe 01:35:29

Doing that, you're talking to people who know nothing about the blockchain whatsoever. Why why should they be interested in the blockchain?

Ohad Asor 01:35:35

It's the future of economy. Right? If you think how traditional economy works with the paper money and printing money and interest rates and all the politics and all the dirty stuff. Yeah. That's crazy.

Tim Scarfe 01:35:52

So just help me understand more about that. So so the way let's say we have the Bank of England and they set the interest rates and they can print new money with quantitative easing and there's all of these different levers and and, you know, actions going on and we have no control over it. And you're saying that the blockchain is a new type of financialization where we just that while there's more transparency, we just understand how it works better. Yes. That's 1 aspect of it. Yes. And then what are what other aspects are there?

Ohad Asor 01:36:22

So let's say automation. Right? If you want some automatic process, if you go to your bank up, which automatic financial processes can you do? We're very limited. Right? But on blockchain, you can do virtually anything.

Tim Scarfe 01:36:38

So you mean having custom software that runs in the blockchain that does things that you want it to do? Yes. And that might be I don't know. When when someone sends me some money, I want to automatically move it somewhere or I want to run this logic program and depending on the output of the program, I might want to notify someone or do something.

Ohad Asor 01:37:01

You want to write a program that pay your taxes automatically. Right? It does the whole accounting, the whole calculation, calculate how much need to be transferred to the tax authority and does it automatically. Right? Why we are not there yet? It's 2025.

Tim Scarfe 01:37:22

I I can attest to that. So just getting my accountant access to my, you know, v like, we have a VAT system in in The UK and the amount of paperwork required, and then I had to get him validated, they had to send him an email. He had to send a special code back, and just paying my tax is incredibly difficult. They tried to set up a direct debit, and then it didn't work, and then it bounced, and then they fined me, and then I've had to appeal the fine, all this kind of stuff. It's a nightmare. So so you're saying that by having this structured interface, I could plug my accountant in there, I could pay my taxes automatically, the whole thing would just run, and and then we could almost have a marketplace of accountants maybe. So, you know, I could I don't like this accountant, so I'm gonna plug this accountant in, and it would just work.

Ohad Asor 01:38:06

Well, you don't need accountants at all if it's automatic. Right? I mean, it's easy to see that the system is in the stone age. Right? As you just described. In 2025, you expect everything to be automatic, everything instant, everything accurate, but it's not like this. The monetary system is 1 of many things in life that is a century back. Right? If not 2 centuries back. Right? There is so much room for advancement and that's to your question. That's another thing why blockchain is interesting.

Tim Scarfe 01:38:49

Isn't this 1 of those things where we could automate it, but weirdly we don't? Like, for example, in large corporations, there are advisory committees. There are, you know, gating systems. You can't check code in without release control. And my bank, for example, every time I pay my staff, they're phoning me up there. They're blocking the transaction. They're saying, who is this person? Did they send you an invoice? Do do you do you know them? And it's the same conversation every single time. And maybe they're just so worried that if we automated this system, it would just spiral into chaos and there'd be lots of fraud. Is is is that the reason why they've put all of these manual checks in?

Ohad Asor 01:39:29

I don't know, but I I don't know. Like you said before when you asked if the users control the blockchain, will it spiral into deadlocks and problems like this? That's exactly what happens in real legal life. Right? That there are so many laws. How many new laws pass every year and how many old laws are deleted every year? Right? It's like this. Right? That's already happening in a system without logical AI. Right? In a system that doesn't care about your opinion.

Tim Scarfe 01:40:08

But in in a way, what you're arguing against is bureaucracy. So there are so many processes in the business world where you get gatekeepers. And are you saying that a big part of that is just because it creates a job for people and people like to interfere and and whatnot?

Ohad Asor 01:40:28

The only thing I advocate I get advocate against is illogicalism. Right? I'm saying don't be illogical and ill being illogical you find in LLM, you find in social structures, you find everywhere. And to bring logic to our life, we can mechanize it with computers, we can fix I don't want to say all of this, but a large part of it.

Tim Scarfe 01:41:02

Amazing. Well, Ahad, thank you so much for joining us today. It's been an honor.

Ohad Asor 01:41:06

Thank you, Tim. My honor and my pleasure.

References

[1] **Tau Language**
https://tau.ai/tau-language/
Company

[2] **Tau Language GitHub**
https://github.com/IDNI/tau-lang
Company

[3] **PAC Learning Framework**
https://en.wikipedia.org/wiki/Probably_approximately_correct_learning
Paper

[4] **Boolean Satisfiability Problem**
https://en.wikipedia.org/wiki/Boolean_satisfiability_problem
Paper

[5] **Knowledge as Justified True Belief**
https://plato.stanford.edu/entries/epistemology/
Paper

[6] **Wittgenstein on Limits of Language**
https://plato.stanford.edu/entries/wittgenstein/
Person

[7] **Theories and Applications of Boolean Algebras**
https://tau.net/Theories-and-Applications-of-Boolean-Algebras-0.29.pdf
Paper

[8] **The Halting Problem**
https://plato.stanford.edu/entries/turing-machine/#HaltProb
Paper

[9] **Alfred Tarski**
https://plato.stanford.edu/entries/tarski/
Person

[10] **DPLL and CDCL SAT Solving Algorithms**
https://www.cs.princeton.edu/~zkincaid/courses/fall18/readings/SATHandbook-CDCL.pdf
Paper

[11] **Tarski Undefinability Theorem**
https://plato.stanford.edu/entries/tarski-truth/
Paper

[12] **Boolean Algebra Foundations**

https://plato.stanford.edu/entries/boolalg-math/
Paper

[13] **Belief Revision Theory**
https://plato.stanford.edu/entries/logic-belief-revision/
Paper

[14] **Quantifier Elimination in Boolean Algebra**
https://people.math.wisc.edu/~hkeisler/random.pdf
Paper

[15] **Tau Net Blockchain Platform**
https://tau.net/
Company

[16] **Tau Whitepaper**
https://tau.net/Whitepaper.pdf
Paper