[SQUEAKING] [RUSTLING] [CLICKING]

ROBERT M. TOWNSEND: OK. So today is this lecture on tokenized and programmable assets. We're going to talk about programming asset exchanges in advance related to the concept of a dynamic ledger. That's in contrast to what we have mostly today, legacy systems, where there are trade fails.

Despite the advantages of programmability of assets there are problems, nevertheless. But, in turn, there are new solutions emerging to handle those lingering problems. In particular, we're going to dig a little deeper into the meaning of tokenization and conditionality, then turn to having multiple blockchains and face the issue of interoperability, which is related to having blockchains coexisting with legacy systems. And then think about exchange and contracting platforms as proposed by various international agencies and countries.

And central bank innovations, I'm going to leave for the reading list. There are some starred articles having to do with the Swiss National Bank and Brazil. But that would, I decided, take too much of the class time. I want to focus on the remaining items.

So asset exchanges programmed in advance-- introduce this concept of a dynamic ledger, where ownership and transfers take place over time, and in principle, conditioned on states of the world. An example will be featured. As is the case in so many of these lectures, we start with a simple example, which is trade agreements for asset lending and then atomic swaps.

This whole thing about trade, is it separate from settlement? These things take place at different points in real time. Is there a sense in which trade and settlement can be instantaneous?

So I want to start with-- because we're going to talk about programmed assets-- with the Ethereum slides, two of them that I showed last time and then a little bit more detail. So Ethereum is the premier or often cited blockchain that allows contracts. There are two types of accounts on Ethereum-- externally owned accounts like Bitcoin, where we have addresses and balances, and balances get transferred across the households or clients or nodes.

But we also have contract accounts. And the contract accounts are nodes on Ethereum, and they contain code and potentially data. So the contract accounts receive "transactions," quote unquote, and update their state. And they can also send transactions and messages and so on. But a contract node can only initiate transactions as a response to another transaction.

But this I smile at because effectively, the code is not a person. It's a passive response to incoming messages, and then can send outgoing messages.

There is a coin associated with Ethereum, Ether, in particular, which is like a fee you pay to execute contracts, and it is intended largely as a way to economize on the number of users who want to do something on the Ethereum platform at the same time. It's like a congestion fee. And in your mind, you can imagine setting it to zero, if you like, to simplify things.

So these are the pictures of the normal user accounts with addresses and balances. Currency, for simplicity, would be Ether. As you'll see, it could be other objects, other coins. And the code and storage are added to these contract accounts.

So the transaction structure-- you have a sender. You have a recipient. You have the currency, say, Ether. That, we're used to. We've been doing this for a while with Bitcoin and ledgers and so on. Then you have the data part, part of the contract with these values and so on, and potentially the gas, the ether that you need for computation.

So all transactions in Ethereum have a sender, a recipient, and a currency field, although that field could be empty. And it's an option to have or not the data part and the contract part and the gas part.

So Ethereum smart contracts-- I often get asked, what is a smart contract anyway? I think you can't really understand what that means without understanding these next two slides because it has to way has to do with the way the code is executed.

A smart contract computes on an Ethereum virtual machine. What is a virtual machine? It has CPU, memory, disk storage, and network interface. And it performs tasks just like a physical machine would. But it's called a virtual machine because it's not physical. It allows administrators to change, in fact, if they wish, the resources the machine is using, based on workload intensity.

Another way to put this is that a single server is allowed to operate many separate machines, each with their own resources, like partitioning your computer or having multiple computers in a network.

The EVM is Turing-complete software. Turing-complete means it's any if/then statement can be executed in the code. Not all codes are Turing-complete.

It's also software, meaning what? That it's not hardware. It emulates hardware. But the software is creating the memory, the disk storage, and the network interface. So anyone can execute code in this trustless ecosystem.

And when smart contracts are executed on the Ethereum Virtual Machine, every node is executing the contract code. It's like every node participant is compiling the contract, which ensures there's consistency across the network. Remember the distributed ledger. It's as if it were a single common ledger, but then distributed to all the participants.

The virtual machine, again, is isolated from the rest of the physical machine. So execution failures in EVM cannot harm the host computer. And again, EVM has its native currency, Ether, which is used to pay for transaction fees.

So smart contracts are self-executing contracts with the terms of the agreement written into the code. They run on the EVM. They're automatically enforced. They execute the terms of the contract when predetermined conditions are met.

Contracts can be reached with these transactions called messages, as I was saying on the first slide. And a message can change the state of the data within a contract. A message could create a new contract or transfer the Ether.

Contracts are written in a computer code language, which, in this case, is Solidity. And again, it's Turing-complete. And then compiled down to bytecode, which is intermediary code that bridges the gap between this high-level source code and the low-level machine code. And machine code is just bits, 0s and 1s, which is the way the computers electronically do things.

So we will come back to all of this. But first, the example, and this comes from a paper with Lee and Martin, the "Optimal Design of Tokenized Markets." So the idea is that a distributed ledger-based settlement system is a technological advancement or solution, sometimes called a token system. And there's a lot of hype nowadays about tokenized assets. So these slides are intended to clarify the meaning of these words.

Tokenized financial assets on the distributed ledger technology-- to anticipate a bit, that's kind of a loaded sentence. You could have assets that are native to the blockchain that originate as part of a smart contract, or you can have preexisting assets which are escrowed, and then they have a representation in the code on the blockchain. So the word "tokenize" could be-- depending on how the speaker is using it, he could mean one or the other of these things, and I'll come back to that later.

The key thing is the programmability of the assets, as in the smart contract and Ethereum, enables traders to commit to settlement because you're locked in. Once you enter into the contract, that's it. The code is going to do its thing as written. So you commit to, quote, "settlement" at the time you enter into the contract. And that's the sense in which trade and settlement are collapsed.

Now, let me just say, even then, it's a little bit confusing. Because in the previous lectures, we've been talking about real time, and who has what goods at which point in time, and potentially even at different locations. So they're agreeing to who is going to have the asset and at what date. But those transfers, quote, "temporary ownership of the asset" is executed under the smart contract.

So it doesn't mean that the assets are actually transferred when we talk about collapsing trade and settlement. It means there's no more uncertainty about what's going to happen. The settlement part is all preordained in the code. And so this has the potential of eradicating trade fails.

So here's the example. There are three risk-neutral traders, agents A, B, and C. There's one single long-lived asset, and it's owned initially by agent A. And the traders have period-dependent payoffs on holding the asset. So this is like asset borrowing and lending.

There's also another good. You've got to pay to have temporary ownership of the asset. So there's an underlying monetary asset. When we talk about prices of contracts, we're talking about the agreement as to who is going to hold the asset, but also the amount that you pay at the time. But the contract will specify the amount of the payment in advance.

So we're going to distinguish trading stages from settlement stages or asset transfer stages. So the T equal M1 or M2, this is all initial trading stages. And again, we're going to partition this. So they're not all together agreeing to everything. Trader B is going to be the intermediary.

So there's two sequential bilateral meetings that can happen in this trading stage. And in the settlement stage, that's when the asset is transferred between the traders according to what they have agreed to in the trading stage, unless there are fails.

So in the trading stage, M1 and M2, A meets B, and B meets C. B is the intermediary, the broker, you might say-- B for broker. And to make it interesting, the order of these meetings is random. It could be 50/50 in terms of who's meeting with B first. And that's only known to B. So A and C never meet. All the pairwise meetings are with B. And so B is the, quote, "intermediary" between A and C.

So up to two trades can successfully occur in principle in the trading stage, the transfer of the asset from A to B, and the transfer of the asset from B to C in some way. Those asset transfers and the corresponding monetary payment is happening in the dates 1, 2, and 3.

So for example, if B borrows the asset from A in period 1, and B agrees to pay a price, P, that means that A transfers the asset to B at the beginning of period 1, and B transfers the asset back to A, in this case, at the end of date 3. The intermediate date, 2, has to do with what C is or isn't doing in the process.

So in the token system this bargaining over prices and asset transfers and the programming that backs it up is occurring simultaneously as if immediately settled. So the asset's programmed during the meetings of this trading stage, and the transfer instructions are self-executing. Assets are transferred without any further trade or action. And no trader is able to prevent the program transfer from occurring.

And we're going to begin taking the point of view of this paper, that this requires the trader to be the current holder of the asset at the time that the contract specifies that it's going to be transferred. That doesn't mean you have to own the asset initially. You could contract to get the asset. B can contract to get the asset from A. Then B has the ability in this trading period to make contracts with C. Both parties know this condition, that A must own the asset to lend the asset onwards, and B knows that, too.

So here's the timeline. In principle, I should have had some initial date, zero, where A holds the asset. A is the original holder of the asset. But then over time in dates 1, 2, and 3, the value of holding the asset is high, medium, or low or something else in between.

So A no longer really wants-- A could hold it but gets a low return from continuing to hold it. B has a high value for it at date 1. And C has a high value for it at date 2. And then, in principle, A wants it back at date 3. So it's a temporary lending of the asset.

However, there are these other things that are happening. The value that B places for the asset at date 2 is random. And it could be medium with a certain probability lambda sub B or 0 otherwise. And likewise, the value that C places on the asset at date 3 could be high with a certain probability, but 0 otherwise.

And these random variables, m tilde and h tilde, are not known initially. They're random variables that get realized at dates 2 and 3, and they're private to the agents. So the target here, given the parameters, is to have B acquire the asset from A, hold it with high utility at date 1. Pass the asset to C, who holds it at date 2.

And then now watch yourself here because it's a bilateral agreement. The agreement between A and B means that the asset has to go back to that-- let me start over. C and B have this contract between dates 2 and 3. And C, if things go, quote, "well," would pass the asset back to B And then B is executing the final leg of its contract at date 3, passing it back to A. So the asset is respecting the bilateral contracts that they have entered into.

So this is the underlying environment. And then we're going to consider legacy systems that allow trade fails and compare that to the token system. So I'll show you how the current financial system operates. We'll talk about the magnitude of these trade fails, including cascades of fails.

So in the same underlying economic environment, we can think about settlement in the legacy system. So these traders have entered into the contracts, in this case, the contract between A and B dates 1 and 3, and the contract between B and C, dates 2 and 3. At date 3, that h tilde is realized for C. And C can decide whether or not to return the asset to B.

Remember, there is this trade agreement between B and C. And C could fail on it, because it has a high value for the asset, and refuse to return it to B. And at date 2, for that matter, when m tilde, which could be medium or 0, is realized, B is the one deciding whether or not to transfer the asset to C.

So let me concentrate first on this, the legacy system in the environment. There's two contracts here between A and B spanning dates 1 and 3, and B and C spanning dates 2 and 3. If things go along the no failure path, then B here will make a deal with A and settle it in real time. And then B has the asset. B will settle with C in real time and pass the asset. And then C will settle back with B, so that B can fulfill the final leg of the trade with A.

But working in reverse, C could settle with B, but C could fail on the contract and not pass the asset back to B. So that would be a failure on that branch. Even if B had settled with C initially to pass the asset, it doesn't. It fails to return it. Or B could fail on its contract with C, then doesn't give C the asset all along that they had agreed to.

Or back here, B can fail to execute the agreement that it's made with A, in which case, of course, the asset's not going to get transferred to C. And so both of these contracts are failing. So this is like a cascade of failures.

This is a bit redundant, but it's kind of useful, which is just to consider in isolation the contract between A and B as if there were no contract with C. And this is something that wouldn't happen in the equilibrium necessarily. But this is a two-period contract between A and B So C is just out of it, and A and B agree to do something under that contract.

So there's a lot going on here. The main thing to take away is the possibility of failures because it's the legacy system-- failures to acquire the asset and pay money for it, failures to return the asset under the contract. And it's a dynamic program, so you have to be both forward-looking in the sense of anticipating what could happen and assess where you are in the current timeline relative to your expectations about the future.

Do we see trade fails in practice? Yes, absolutely. The slides are going to highlight what happened in the great financial crisis with failures. The settlement fails basically were $400 billion in one day. This is a trillion dollar market.

People are aware of the failures. The Fed tried to introduce a charge for failing, an additional price to be paid. But the settlement, the failures, continue to occur. It's not just around the great financial crisis, although the diagram coming in a minute is going to feature that. It's an institution and technological failure of our current legacy systems.

Individual traders submit instructions to their corresponding bank. These instructions have to do with the trades that they've entered into to say to buy securities or sell them. And then it comes time to execute the contracts, even just at the end of the day, and situations have changed. Incentives to settle may have broken down, and the trade fails.

So here's a picture of the fails in the great financial crisis, reaching almost a trillion-- $10,000, $500 billion. And when this slide was created many years ago, the Fed was saying that the increase in liquidity in the system actually helped mitigate subsequent trade fails. But I really ought to update this diagram and show you that trade fails have continued in the US.

So "why", quote, do these things fail? In a little more detail from Garbade, there's miscommunication possible. A buyer and a seller may not have a common understanding of the trade they thought they entered into, or they may fail to communicate well with their custodian, or they communicated incorrectly with their custodian.

So another reason for failure is system failure, as in 1985 with the Bank of New York or 9/11. There was an initial surge of failures, massive disruption. But it didn't go away even several days subsequently because there were insufficient incentives to resolve the fails. Again, the seller may not have the requisite securities in its commercial bank account. But that's OK, as long as they're able to acquire the securities by borrowing in order to deliver them.

With cascading failures, you have a-- it says sellers, but should be buyer. Buyer's failure to pay the money leads to lack of liquidity in the chain, so the recipient would be-- recipient of the money doesn't have it in term, they can't execute other purchases that they were supposed to make.

It's an odd typo because in Wall Street we, meaning me as an economist, I often think of borrowing and lending money with securities as collateral. But their language is borrowing and lending securities with money as collateral. It's all intertwined.

AUDIENCE: What happens after these failures? So basically, if the settlement was not made, then just the other parties have a loss?

ROBERT M. TOWNSEND: They can-- yeah.

AUDIENCE: [INAUDIBLE].

ROBERT M. TOWNSEND: Well, they could try to adjudicate the deal. There are long-term relationships going on. So you say, OK. Maybe not. That maybe at a lower price, some exposed bargaining is going on.

But the thing about incentives-- maybe that was a little vague. But I mentioned this before. And I always want to look at the State Street over there. So State Street is one of the largest, of course, bonded banks in the US. And they make trades for their clients. The clients are trading on the Stock Exchange and so on.

And then at the, I think it's 3:00 in the afternoon, they begin to settle the trades, and it takes a couple of hours. But obviously, prices have moved during the day. So things that you agreed to do at 10:00 in the morning may no longer-- you may have regrets if you could have gotten a lower price and so on.

So in principle, the programmed trade on the dynamic ledgers would not allow fails because you have to execute. But other problems can emerge, namely, holdup problems. It's related to the commitment problem.

So B, say, has agreed to lend the asset to C at date 2, and B's valuation of the asset is high, but not at date 1, but not subsequently. Why is B acquiring the asset? B can acquire the asset because it likes holding it for date 1. But B is also anticipating, as a broker, that it can transfer the asset to C. And these meetings are happening asynchronously, not simultaneously.

So B is like a market maker-- commits with A to one side of the chain in advance and anticipating its deal with C. In fact, B may pay more than its own underlying valuation of the asset in order to obtain the asset for C, and C being willing to pay a high price for it.

Now, the problem is that if C knows that B has the asset, and so far this is what we're assuming that B can enter into that contract with C only because B has already entered into the contract with A and C knows that, then there is this problem that C could say, oh, I got you. You've already got the asset. You don't like it much, and I'm not going to pay you much for it, kind of pushing right down to the margin at that time.

So the legacy system does have some advantages. There's a complete decoupling of trade and settlement. Traders can enter into any contract they want without explicit proof that they can fulfill it. And B, therefore, in the legacy system can hide from C whether or not he's met A. And hence, is or is not in control of the asset.

So there is an interplay between trade and settlement here. Trading occurs asynchronously. But in the legacy system, there's no transparency over the history of previous trades. So taken in isolation, everything is opaque. Incentive-based settlement is potentially suboptimal, but they either settle or not what they've said they would do based on their underlying individual shock. So there's flexibility ex-post to do as they wish. So this decoupling of trade and settlement actually facilitates the transfer of ownership.

Again, there's no longer a holdup problem because C doesn't know that B has the asset. B is entering into the agreement with C, but C has no way of knowing in the legacy system whether A and B have already met or not, that maybe C and B is the first meeting. And when C and B make this deal, then A will turn around and get the asset from B. Sorry, B will turn around. So there's an advantage in that way.

So this diagram is revealing for parameter values. In particular, these probabilities of realizations of valuation, lambda B and lambda C. Doesn't matter much if you remember or not what they are. They're just unobserved random variables of valuation. This diagram maps out when the legacy system would dominate and when the token system would dominate.

Both can happen. The token system can improve on efficiency, but only if that holdup problem goes away. And the commitment problem is-- the limited commitment problem is, indeed, a big problem.

AUDIENCE: What's the interpretation of lambda being close to 1? It means what random variable becomes likelier to be high? Like M tilde or what is it?

ROBERT M. TOWNSEND: So lambda-- what's--

AUDIENCE: That's that M tilde is high.

ROBERT M. TOWNSEND: Yeah.

AUDIENCE: OK. So if both agree that the asset's going to be high value, commitment has positive value. Otherwise, it has none.

ROBERT M. TOWNSEND: Good.

AUDIENCE: Thank you, sir.

ROBERT M. TOWNSEND: OK. So if one system doesn't dominate the other, are there solutions? One solution is direct centralized trading, and the second has to do with encryption. So what we featured so far is this intermediated broker-dealer trading. But we ran into this problem of asynchronicity and information.

If we combine trade and settlement together and think about the design of the financial system, we could jump to the idea of having direct trading. In effect, having the buyer and seller be directly matched with each other if they can come to an agreement rather than having to go through B.

So in the same notation, but with this alternative trading arrangement, at the initial trading date, t equal m1, B and C simultaneously post ultimatum offers for ownership of the asset at dates 1 and 2, respectively. And at the second date, trading date, A chooses whether or not to accept the offers. So again, probably worthwhile to go back.

So B wants the asset, say, at date one. Makes a new deal with A. And then likewise, A over here can make a deal with C for C to have the asset at date 2. And it doesn't go through B-- two bilateral arrangements that A is making with B and C, which does respect this timeline in terms of if they come to a deal, who is holding the asset for what dates. But we're going to call it direct trading.

And just for the sake of argument, we could imagine that there's some cost for direct trading. But certainly, if you set that cost to 0, direct trading dominates. There's no holdup problem anymore. You have all the advantages of full commitment to the agreement you've entered into. So now maybe the hype surrounding tokenization and programmed asset trades is coming into its own in the direct trading system, more centralized, in some sense, in the sense of making bids.

But let's come back to tokenization again and think about fungible and nonfungible tokens on Ethereum Virtual Machine. And then finally, I'll come back to this tokenization with pre-existing assets.

So what is ERC, Ethereum Request for Comment? So this was a standard that got created in Ethereum for fungible tokens created, again, on the Ethereum blockchain. A fungible token is one that's exchangeable with another token. Whereas, other standards ERC-721, pertains to nonfungible tokens. Data, for example, is not a fungible token, but it's part of the Ethereum contract.

So this ERC-20 standards allow you to create smart contract enabled tokens that can be used with various products and services. They are a representation of the asset. The right ownership access could be cryptocurrency or anything else that can potentially be transferred. The additional capabilities under the ERC-20 standard has to do with this approve and transfer function.

Let's suppose asset A is on this Ethereum blockchain as a token, and A is the initial owner of it. And then with Nicholas Zhang, we return to the Lee Martin Townsend paper I was just expositing. A owns the asset initially, in the sense that it's a smart contract node on the Ethereum ledger.

This is not, quote, "owned by A as a participant," in the sense of the legacy system of carrying it around and deciding what to do with it. It's entirely represented by the code. And at the moment, A is the owner of it.

So my emphasis now on the fact that it's a smart contract has to do with it's a proven transfer function. And, in particular, when A and B meet, and they do say meet first, they can agree on the conditions for B to potentially sell the asset to C at a later stage. But B doesn't want to commit to it. Why? Because of that holdup problem.

B has the right to sell it to C and take it from A, effectively. So A approves B to transfer one unit of the asset when B chooses to do so. But B, of course, could say, I don't choose to do it.

So when B and C meet later, if they reach a deal, B can acquire the asset to A and then pass it to C But if they don't reach a deal, if B and C don't reach a deal, the contract says, OK, we're just leaving the asset in A's account, and we do nothing. And A can revoke the approval later, or there could be a timer that the contract will expire. There are many options.

So this completely eliminates the holdup problem because it's a conditional approve and transfer function. They're committing in advance to what they would do without loading all the bargaining advantage to C. If C chooses to play hardball, B says, fine. I don't have the asset. You can't hold me up. I don't even own it.

So this approval allows the token holder to authorize a designated smart contract to spend a previously defined quantity of the token from the balance. Note the language here. The holder is empowering the smart contract. We're back to those initial slides where these messages are being sent to the contract node.

This ERC-20 approved method enables the token holders to selectively allow spending for specific purposes or transactions, enhancing the flexibility of the token use. In fact, in other settings, people like or dislike a lot this idea of programmed money. You're putting conditionality onto the object. You can only say withdraw so much of it because we're really not sure as regulators that we want unlimited amounts of this stuff around.

But in this case, the same approval function in the ERC-20 standard is being used to direct future-- to controls, consistent with what they've agreed to in the contract, the future flows of the asset. So this is a function, f, a signature approval function that has the address and the amount. And it's an approval function, so it's yes, I approve, or no I don't.

So the spender is the address of the smart contract. It's not the agent. It's the smart contract who's the spender. And the amount is the amount of the token that's permitted to be used on behalf of the token holder.

So preview of coming attractions, and this is premature, but I'll mention it, nevertheless. These tables, 2 and 3, which we're here, A is the initial holder, and then a deal made in the smart contract between A and B. What the participants are seeing potentially is gobbledygook. They're just seeing encrypted values.

So this is done under homomorphic encryption, which intuitively is an isomorphic space to the underlying true value space that respects things like addition and multiplication. So you can do things in the encrypted space with the code. And the code doesn't really, quote, "know" it's not even a person. But if you decipher the underlying function, you can encrypt the amount, have the function operate on the encrypted values, and then decipher it back to the underlying function, f. So you achieve equivalent results.

And again, I apologize, in a way, that this is premature. But since I mentioned those values are private, you may be thinking that there's some information leakage going on. But we have the power of homomorphic encryption to completely obscure the state of-- there is a true underlying state, but we're able to partition with encryption to allow subparties of contracts to know things that others don't know and cannot decipher. And we have a whole lecture on encryption, which is hardly enough. But I'll go through this in more detail in a couple of lectures.

So let me come back to this interface with legacy assets, where the digital assets are not native. There's a lot of questions that would need to be considered, including rules for issuance of tokenized assets, redemption, access to them, the underlying funds and claims, transfer mechanisms, privacy, interoperability and so on. And you have to make choices. Depending on those choices, there are implications for safety and efficiency of the arrangement.

For example, this comes up with these digital dollars, the stablecoins in circle and so on are supposed to be completely backed by treasuries or underlying liquidity. But if that's not part of the blockchain, then you have to take that on faith.

But there's simpler backing, which is you could take, say, an account at the Federal Reserve that a bank has in its reserve account and put it in escrow, and issue a corresponding amount of the tokenized asset. And then you have to trust the person holding it in escrow to not mess with it in the interim, so as to guarantee, if you trust it, the one-to-one convertibility of the underlying asset with its tokenized version. So this is written by the BIS. There needs to be clarity on who has a claim to what and what the backing relates to and so on-- CPMI.

So now I want to come to interoperability. And the blockchain technology as we see it today is typically fragmented into different blockchains, not just one. And in general, they can't communicate or exchange assets across these chains. Likewise, crypto exchanges are not operating on the blockchain at all, typically. The exchanges maintain liquidity pools on various of the blockchains and exchange balances across the agents.

So there, it's a bit like agent B in the example in the sense of having these intermediaries. It's not decentralized. Some decentralized exchanges exist, but they require some kind of conversion between an intermediate currency and the consensus of the intermediate blockchain.

So let me take you through some slides on that. The centralized exchanges, like FTX was, you have the two separate blockchains-- one for Bitcoin, one for Ethereum, and there's trade going on. But that's going on in these centralized off-chain exchanges.

There are some decentralized exchanges in existence, which is like creating a third chain, an intermediate chain. They no longer operate like banks. They don't require trust. But the state of the art is that they're computationally expensive to run. But advances are being made in the design of these DEXes to allow them to scale up and run more efficiently.

ChainLink is an example of something having to do with external information. There's an oracle that gives information to on-chain nodes, but you have to trust the source. Or you gather information from multiple sources, like multiple oracles, and then perform some kind of consensus operation. And these oracles are limited to data retrieval. There's no mechanism for transmission of data. This is like consulting FX markets to get an exchange rate and then bringing that into the blockchain.

LayerZero is a decentralized messaging system that does not require an intermediate blockchain. It enables currency exchange in arbitrary messages between two arbitrary chains. So we have chain A, chain B, somehow each linked to layer zero. And the system will allow a message to be delivered from chain A to chain B only when something got committed to on chain A. So it does get a little bit complicated.

Hopefully, this picture conveys a little bit more. You have chain A, chain B, and they're both part of LayerZero ecosystem. So each of these blockchains has a LayerZero node, you might say. And then also, you've got these relayers and oracles. And within each chain, you've got communicators and validators.

So this thing works on something, again, related to encryption, zero knowledge proofs where you can prove that something happened on one blockchain by a series of coded operations without actually seeing directly what happened there. So all these things are trying to emulate the strength of having one centralized-- one blockchain doing all the operations where you can control the states and monitor the transfers.

So this exchange in contracting platform was something the IMF envisions, which is to set aside the domestic systems of each of the countries. But nevertheless, set up a blockchain for foreign exchange transactions, XC being exchange and contracting platform, maybe with an emphasis on the contracting part. And I'll say a couple more words about that on the next slide.

The Bank for International Settlement has now also envisioned something they call a unified ledger even within countries. And we'll review that carefully. And then I'll come to what the Federal Reserve Board wrote having to do with this coherence guarantee, which is going to take us right back to essentially the way Ethereum operates.

So exchanging contracting platform for cross-border payments-- the current state of the system, it's slow, it's expensive, and it's risky. You do have these broker dealers who are intermediating the system, and they are relying on trusted relationships the clients have with them and vice versa, trying to deal with the fact that there's not a common settlement asset, and there's not common rules for governance. That's the current system. I mean, let me just put that in plain English.

We're dealing here with multiple countries. They're not like states in the US, or even countries in the EU. There's no central governance unit. So that has to be agreed to as part of a platform. And likewise, what do we mean when we say there's one money, when we have a fiat money for each of the countries?

So this multilateral platform that we envisioned allows the fiat money of each central bank, the reserves, to be tokenized in the sense that I was describing earlier, and the tokenized objects or what exists on the blockchain. Then you can write smart contracts that allow for risk sharing and other financial contracting, and of course, simple spot exchanges of one currency for the other. So we're leveraging the technological innovations that we've been talking about-- distributed ledgers and smart contracts and encryption-- to gain efficiency.

The BIS now is envisioning a unified ledger, and they meant within countries, not just across countries. So let's review this BIS proposal. First of all, the tokenization, which are rules for what the asset can and cannot do, as in a smart contract. And hopefully, that's quite clear from the lecture today so far-- information about where the asset is, where it comes from, who owns it.

So tokenization means recording and mirroring the claims on real or financial assets that exist on traditional ledgers. That's this meaning of tokenization which is not native. But once they're tokenized, you have the database and the rules and logic of the smart contract.

The tokenization has its advantages. It combines messaging, reconciliation, and settlement in a single operation. I hope that that is clear now from the example that we've gone through in the lecture. It improves efficiency, and with a commitment we're unlocking new features that enables atomic settlement, transfers that occur reciprocally, simultaneously, or transfers where one is a precondition for the other.

It enables contingent performance of actions, as in that transferability and assignment that I was going through and other things as well. It certainly allows for multiple parties to be doing things with each other simultaneously, not just bilaterally. And it expands the universe of contractual outcomes so that more productive use can be made of resources which would otherwise be lost. And if you are on one blockchain, then you can do this.

But the questions are, what happens when information comes from outside, as in that oracle example? What happens if, as in real life, there are other ledgers? Who is the custodian of these non-native assets? Who is synchronizing the asset movements across the different ledgers?

So at this point, the BIS paper starts to talk about tokenizing deposits, meaning commercial bank liabilities, and then makes three claims about why tokenized deposits are better than, say, asset-backed stablecoins. First claim-- tokenized deposits help preserve the singleness of money. This is the way they think about things.

Second, payments in tokenized deposits settle in wholesale central bank digital currency, which ensures finality because they are claims on a central bank. By using its own balance sheet as the ultimate means of settlement, the central bank provides the means for ensuring the finality of the wholesale payment.

Third claim-- tokenized deposits ensure that banks could continue to offer credit and liquidity in a flexible way. And then, in contrast, tokenized stablecoins represent a transferable claim on the issuer akin to a digital bearer instrument. Stablecoins are tradable. Their prices deviate from par, undermining the singleness of money. And then subsidiary claim-- stablecoins do not allow elastic liquidity.

This is in the paper. This is the way, at the time of writing, the BIS was viewing these things. But the critique is that they have now merged the traditional legacy accounting point of view of money with this new technology. They're basically saying, look, we understand commercial bank deposits. We understand central bank reserves. Let's try to adopt an aspect of the new system by allowing only tokenized deposits rather than more general representation of assets.

What, in particular, does the BIS mean by a unified ledger? Trying to bring together various central banks, CBDCs, bring together tokenized deposits and other tokenized assets on a shared programmable platform. So this is the larger vision, bringing to the same venue tokenized assets, tokenized deposits, and CBDCs. But what do they mean by this venue?

They seem to suggest there's a role for private platforms and for central banks. And they mentioned applied programming interfaces, but it's really open for interpretation. It could be a unified ledger, after all, on a public platform, which could connect the various distributed ledger technologies to an existing central bank infrastructure. For example, within a country and its central bank and its reserves, convert the settlement system of real-time gross settlement onto a blockchain. That is an option.

So then CBDC would mean existing central bank reserves. It's now a technologically improved system. Still via a public platform, it could be there are new central bank distributed ledgers with bridges back to the private and public tokens. Or put more succinctly, here central bank digital currency would mean tokenized central bank reserves or tokenized Treasury bills.

Or in principle, it could be done, or it could have the interpretation, that when they say unified ledger, they mean the ledgers are unified via regulations and partnerships, that existing assets are tokenized with custodians and reconciled by someone, either a public or a private entity. For example, in that regulated liability network, the tokenized commercial bank deposits.

So in that case, the tokenized money could then be transported to a public distributed ledger platform, or continue to exist on the private regulated DLT platforms. And in this latter case, there's no CBDC at all.

So I realize that my tone is conveying a criticism of the BIS paper for being unclear. And indeed, that is the content of these slides. But you could also interpret it as these are all decisions that individual countries and the international community are going to have to make about whether or not to use these new technologies and how the system should be configured.

It's indeed pretty hard to have these conversations without clear definitions of things like tokenization and smart contracts and understanding legacy systems, and that was the content today. So these open questions are, could a retail CBDC be used as a wholesale CBDC, one unified CBDC?

Should Fed or Treasury tokens-- should they tokenize native objects to be used on the private platforms-- Federal Reserve, CBDC-- which is what Switzerland did, actually. It allowed a representation of its reserves on a private sector platform and conducted an open market operation there.

Should the Fed regulate private platforms or provide a public one, like this regulated liability network? Recent US policy guidance suggests doing everything with private blockchains, stablecoins, in particular, and is negative about CBDCs. So that is also stimulating a lot of conversations these days, both in the US and in other countries.

The main thing is to recognize the power of programmability, that programmed money is a unified, coherent product that encapsulates both the storage of the digital value and the programmability of that value. This coherence-- this is from the Federal Reserve Board, US. This coherence guarantees refers to a mechanism for guaranteeing that the technical components of programmable money are inseparable, and that those components are consistently functional so that the product is stable and coherent.

They could be nominally separable, as traditional technologies are, and implement two core technical components, namely, the storage of the digital value and the programmability of that value. But the coherence guarantee must ensure they are not separable as far as the overall product is concerned. Without that guarantee that you get with Ethereum and maybe other ways, the individual components are not novel at all.

Digital money has been around for many years, as has the ability to write computer code and software for commercial banks and financial institutions. With the guarantee, programmable money is a new product category sitting alongside of the more traditional money products of central bank deposits, demand deposits, and nonbank money and cash.

But this benefit of the coherence guarantee is not necessarily present in other systems that can rely on services and potentially be of other providers, and can be altered by those providers as they wish.

So that's the state of the domestic and international financial system. There was a lot in this lecture. We went through programmability, first and foremost. Carried that in through examples. Kept coming back to it, as if there, without saying so, one blockchain. Then we get into the issue of how powerful the tokenized assets are with conditionality.

But then recognize that we have multiple blockchains. We have the legacy system coexisting with these blockchains. And we can talk the talk of taking advantage of the power of the blockchain. But the devil's in the details in terms of exactly how to design the system and what regulations would be appropriate.