Risk-Free Interest Rates in Decentralized Finance

Abstract

Decentralized Finance (DeFi) aims to use advancements in both computation and cryptography to tackle standard economic problems. It must, therefore, operate within the intersection of constraints required by both the computer science and economic domains. We explore a foundational question at the junction of those fields: is it possible to synthesize variable market-clearing risk-free yield for native tokens via smart contracts? We show using a stylized model representing a large class of existing decentralized consensus algorithms that this is not possible. This places strong bounds on what decentralized financial products can be built and constrains the shape of future developments in DeFi. Among other limitations, our results reveal that markets in DeFi are incomplete.

Keywords: DeFi, cryptocurrency

JEL Codes: G12, E42, E43, E51, E52

1 Introduction

The blockchain concept introduced in Nakamoto (2008) joins ideas about computing and economics. Blockchain-powered smart contracts enable new approaches to solving financial problems, and the technology introduces new trade-offs (Shrier et al., 2016). In this paper, we study whether a financial system can be built atop a native blockchain-powered ecosystem by focusing on a foundation in modern financial engineering – market-clearing risk-free interest rates (Caballero et al., 2017; Gorton, 2017). Derivative finance such as those in Hull (2006) depends heavily on a liquid riskless nominal interest rate.

In particular, the DeFi setting – whose key features include permissionless and trustless participation in the ecosystem – presents two separate but entangled questions. First, is it possible to synthesize a positive variable riskless rate with smart contracts? Second, if such a structure can be built, does it offer sufficient flexibility that the funding market can clear? Unless the answer to both questions posed above is yes, DeFi would not be able to simply copy decades of work from traditional finance (TradFi) without major changes. At first glance, it might seem that a simple solution exists for DeFi systems in the style of Nakamoto (2008). For riskless, non-interest-bearing deposits, simply leave the tokens at rest on the blockchain. This solution is akin to a vault of cash but without the physical storage requirements. Because such deposits are riskless, they provide a lower bound for deposit rates but say nothing about lending rates. For lending rates, a funding market must clear, where willing borrowers and lenders agree to transact at a specified rate. This focus on market structures with sufficient flexibility to clear goes all the way back to the foundations of economics in Smith (1776). In TradFi, this price is discovered in the government bond market. But can an analogous market clearing interest rate be generated from the smart contracting space in DeFi?

Our intention is to investigate whether something reminiscent of TradFi can be constructed on top of this novel foundation. We are not trying to explore incremental engineering improvements attributable to a new distributed ledger design. Therefore, it is important to clarify two points regarding stores of value in general. First, we will consider only a nominal interest rate in some native on-chain token and not focus on real rates. Given the motivation for studying whether a modern financial system can be built in DeFi, this approach is without loss of generality. For example, extant modelling tools for inflation-linked derivatives and real-rate products still rely on an underlying liquid and clearing nominal interest rate (Dam et al., 2018; Brigo and Mercurio, 2006). Therefore, even if one is able to construct a decentralized platform that maintains real value, that is not sufficient to carry over the machinery of financial engineering. Second, and again without loss of generality, we only focus on a unit of value that is in finite supply. The unbounded generation of tokens in an anonymous, permissionless system

means any participant can create an arbitrarily large bank balance at any time. For this reason, we focus on tokens that emerge from a consensus process in the style of Nakamoto (2008): the creation of tokens is by design randomized, decentralized, and constrained. In other words, we consider the "native" tokens of a blockchain such as Bitcoin or Ether. Other tokens generated by smart contracts on these sorts of platforms are not necessarily bound by our results. But all smart contract interactions with fees paid in such native tokens surely are.

Our analyses concern a question of computational feasibility: can we produce a provably-riskless investment contract, with certain properties, in an specific environment. The TradFi frameworks discussed above operate on top of a riskless interest rate in the sense that does not admit "probably riskless" contingent on collecting fees, successful collateral liquidations or other uncertain revenue streams. Per Damodaran (2020), what we require is not a probabilistic argument or an equilibrium condition. Rather, our goal is something akin to proving the correctness of a sorting algorithm: it that works every single time as long as the program is allowed to run to completion (Cormen et al., 2001). This is a critical distinction for derivative frameworks where implicit leverage is unconstrained. For example, if there is some $\epsilon > 0$ chance that fees may become insufficient, a default that becomes a practical problem for a derivative dealer levered on the order of $1/\epsilon$. Similarly, free-market collateral liquidation processes are not guaranteed to raise any particular level of revenue. As discussed below, this reality underpins the structure of backstops in the money market fund space (President's Working Group On Financial Markets, 2020; Menand, 2020).

Using a halting problem reduction methodology, we find that a general Turing-complete smart contract language is incompatible with floating risk-free yield in this sort of environment for native tokens. The result follows from both Turing-completeness and decentralization properties. The former means a smart contract cannot be proven to halt while the latter means allowing an authority to intervene to halt the program is unacceptable as a solution. Although it is possible that some narrower class of a smart contract language can be used to generate useful services, we make no claims on the matter.

We emphasize that the results from this paper are not an equilibrium result in the economics sense. For example, we do not consider designing proper incentives to deter or encourage certain behavior by participants. Instead, we focus on the technical feasibility.[1]

[1]For example, economics research often considers what actions to take when a system is perturbed out of a good equilibrium, perhaps due to some exogenous natural disaster or a change in business cycle conditions. This line of research typically studies what actions should be taken or what sorts of incentive systems can prevent further trouble or nudge the system back to the good equilibrium. However, we are not looking at this sort of analysis.

Incentive systems may work in practice, and may work most of the time. We are concerned with the question of whether, in an idealized world with practical considerations abstracted away, it is even theoretically possible to achieve absolute risklessness of the sort we find in similarly abstracted traditional fiat monetary systems.

A financial system run on an unstoppable decentralized blockchain requires this sort of guarantee because there is no party, by definition, that can step in and arbitrarily reorganize things.[2] When we talk about riskless interest rates in economics we mean something similar to this latter approach. In line with Damodaran (2020) we consider investments as riskless when there is no variance in realized returns relative to expected returns when measured in the currency of interest Incentive systems do not work that way. This distinction is important for much of finance because of leverage: an ϵ chance of failure is a practical problem when you are levered beyond $1/\epsilon$. We see leverage employed in this sort of unbounded fashion in a wide range of financial engineering models starting from Black and Scholes (1973) and running through essentially all of dynamic-replication finance as discussed in surveys including Rebonato (2012); Hull (2006); Pascucci (2010). Leveraged replication is also a core component of strategies used in corporate finance beginning with Modigliani and Miller (1958) and portfolio theory back to Markowitz (1952).

For macroeconomic policy, this is not necessarily a problem; for a financial derivative market-maker or computer scientist it absolutely is. Our contribution is to show what classes of products cannot provably work – what cannot be copied wholesale from TradFi – and to cast recent advances in DeFi through the lens of an economic model that is based on computational theory. We show that smart contacts are not alchemy: they cannot generate yield out of nothing. Until someone designs a smart contract infrastructure more powerful than a Turing machine, it will not be possible to enforce risklessness with only code in a complex decentralized system. Further, the history of financial market crises teaches that events which are widely assumed unlikely ex ante are, at least sometimes, rendered inevitable by mechanisms that only come into clear view ex post. This is discussed in great detail in Reinhart and Rogoff (2009); Aliber and Kindleberger (2015); Garber (2001). Furthermore, when coupled with the well-known Mundell-Fleming impossibility result, an economy without a variable risk-free rate faces substantial macroeconomic policy constraints (Wu and Xia, 2016).

[2]There are examples of blockchains that decide to pause block production to address problems from time to time such as Terra (Ponciano, 2022) and Solana (Nelson, 2022). If a group is able to decide to take a blockchain offline, it is not really decentralized – it is clearly under the control of that group and inconsistent with the ideas of Nakamoto (2008). Here we analyze the properties of the ideal type of DeFi that proponents want to construct and not the compromised system in use today.

Related Literature

The extant economics research studying blockchain technology characterizes "impossibility theorems" which broadly outline what is and is not possible (e.g,. Abadi and Brunnermeier (2018)), studies the incentive compatibility of token adoption (Cong and He, 2019; Cong et al., 2021b) or coordination/consensus formation (Biais et al., 2019; Cong et al., 2021a). For example, Malinova and Park (2018) study when tokenization beats equity as a financing source for a platform company. However, research on automation and financial technology on market structures has been around since Black (1971a,b), who study whether exchanges can be automated. More recently, this question has been revisited by O'Neill (2021) in the context of an automated market maker based on recent developments in financial technology. Related to our work, Huberman et al. (2019) studies whether and how a cryptocurrency like Bitcoin can be used as a viable monetary device to facilitate the trade of goods and services. We build upon this interpretation by studying whether a class of Turing-complete DeFi protocols can support a risk-free rate upon which a financial system can be established.

On the computational side, related to the trustless nature of different consensus protocols, Primavera (2020) discusses the potential governance issues that arise in decentralized autonomous organizations (DAOs). More recent research like Wohrer and Zdun (2018), Krupa et al. (2021), and Qin et al. (2021) explore security patterns and concerns in existing systems, and Jansen et al. (2020) questions whether smart contracting languages need to be Turing-complete.

2 DeFi Protocols & Blockchain: A Background

The application of distributed databases in financial contexts raises the question of whether it can bring about a new wave of innovation (PWC, 2019). DeFi is a new paradigm for financial system design where parties need not trust each other, and transactions are finalized on blockchains via some variant of Nakamoto Consensus. DeFi services operate in public via smart contracts with all transactions recorded in the open (Fang et al., 2021).

We specifically focus on decentralized blockchain systems employing incentives in the style of Nakamoto (2008). Pre-programmed, immutable, money supply was an explicit design goal of the protocol from the beginning:

> If we wanted to trust someone to actively manage the money supply to peg it to something, the rules could have been programmed for that...Instead of the supply changing to keep the value the same, the supply is predetermined

and the value changes Nakamoto (2009).

As we will see later, for a decentralized environment, if permissionless access to print money supply is available, then no guarantees around supply can be made in general. Before presenting the model, we introduce vocabulary to connect computer science concepts to the economic mechanisms discussed later.

A token k is a fungible, reproducible, and freely transferable unit of account. Correspondingly, a token set is a set of different fungible tokens $k = 0, ..., L$ which are not fungible with each other. Then, we define a wallet balance as:

Definition 1 (Wallet Balance)**.** *A* wallet balance *is a tuple of quantities of tokens from a given token set. We write this as* $\{w^0, w^1, \dots\}$.

A wallet balance is *valid* if $\forall_k \; w^k \geq 0$. The time dimension, for a blockchain, is discretely indexed $t = 1, ...$ We can define a wallet balance at a particular point in time as a collection of wallet balances indexed by time t. Then we write the wallet balance for token $k = 1, ..., L$ at time t:

$$W_{i,t} = \{W_{i,t}^1 \; W_{i,t}^2 \; ...W_{i,t}^L\}$$

where i denotes a wallet address, L is the number of tokens, and t denotes discrete time which we will link to a specific block of transactions below.

Definition 2 (Transaction)**.** *A* token transaction *is a 5-tuple consisting of a time, token, sending wallet index, receiving wallet index, and quantity. We write this as* $(t, k, w_{sender}, w_{receiver}, n)$.

This represents the transfer of n k-tokens from sender to receiver at time t. We follow a simple algorithm when applying a set of transactions:

function ApplyTransactions($Transactions, t$)
  for all $address, k$ **do**
    $W^k_{address,t} \leftarrow W^k_{address,t-1}$
  end for
  for all $r \in Transactions$ **do**
    $W^{r_k}_{r_{recv},t} \leftarrow W^{r_k}_{r_{recv},t} + r_n$
    $W^{r_k}_{r_{send},t} \leftarrow W^{r_k}_{r_{send},t} - r_n$
  end for
end function.

Definition 3 (Block)**.** *A* block of transactions *is a set of token transactions that occur*

at the same time t. A block is said to be valid iff all impacted wallets are valid at $t-1$ and t following the application of the transactions.

Definition 4 (Block History)**.** *A* block history *is an ordered set of blocks of transactions where all are valid. We say a wallet history is consistent with a block history if $\forall t$ the application of the blocks at time t generates the wallet changes from $t-1$ to t. We write this as $\mathbb{B} = \{B_0, B_1, \dots\}$.*

Given the definitions above, we call a block history a blockchain when there is sufficient information within the blocks to establish with a high degree of certainty in what real-world order they were produced. The most common mechanism used to achieve this property is encoding a hash of the $t-1$ block within the t block (Merkle, 1980).

Definition 5 (Decentralized Blockchain)**.** *A blockchain is decentralized when the process of appending new blocks is open to all, and there is more than one participant.*

Definition 6 (System)**.** *A* system *is a $(\mathbb{W}, \mathbb{B})$ tuple, where $\mathbb{W} = \{W_{i,t}\}$ $\forall i = 1, ..., N$. A system is valid when all wallet balances are valid and consistent when $\forall_{k,i} W^k_{i,0} = 0$ and $\forall t \geq 0$ the application of all transactions in B_t gives us W_t from W_{t-1}.*

2.1 Trustlessness & Decentralization

Since we are working in the intersection of economics and computer science, it is important to clarify what we mean by "trustless." When we are talking about the time transformation of assets – fiat, crypto or anything else – it is always within a system. For an economic system, trustlessness exists within the confines of that system. Local currency government bonds are considered riskless even though there are myriad cases in history of countries losing wars and their bonds becoming worthless. Similarly, it is possible for a decentralized blockchain to stop running, experience catastrophic bugs, or to somehow stop. Smart contracts will keep running so long as the system is around. We do not count the system shutting down as "risky."[3]

All smart contracts run on a public computation platform where we can read the code, run it ourselves, and monitor all transactions in plain view. Further, if there is a conflict within the system – say a block of code has more than one valid interpretation – the system itself contains a conflict resolution mechanism which we can observe in action. Trustless means that we can verify everything for ourselves and safely rely on our own

[3]Consider a system of continuously-collateralized contracts where the collateral defaults iff the system for enforcing contracts falls away. Now assume continuous liquidity into physical commodities for the equity in your trading account so long as the system continues. Your equity would be preserved, in physical form, even after the system went down. The interest rate paid on collateral, and for discounting collateralized future payments, is riskless in some basic Lockean sense.

computations.

This style of system is consistent with the Bitcoin whitepaper where "nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone" (Nakamoto, 2008). Anyone can enter and exit as they please, anonymously. This property is the reason a "token printer" smart contract does not fix the risk-free problem. If we restrict access to certain nodes, then the system is centralized and not DeFi. If we allow access to all but cap supply, then someone will default: we just cannot know in advance who. That renders liabilities built atop the facility risky. If we allow unfettered access then anyone can appear, claim an arbitrary number of tokens, and vanish. That is not an economic system focused on the consequences of scarcity (Samuelson and Nordhaus, 2001).

2.2 Smart Contract Turing Completeness

One of the core results in the theory of computation is the so-called Halting Problem. Turing (1937) proves that, for a general purpose programming language, it is not possible to write a separate program to verify if an arbitrary block of code terminates in a finite amount of time. To be precise, this result applies only to "Turing-complete" programming languages. This is a technical condition the details of which are beyond the scope of the discussion here, but in practice most real software is written in such languages. Even seemingly simple languages like *awk* are Turing-complete and this document was prepared in LaTeX which is also Turing-complete.

Smart contracts are computer programs that run on, and interact with, a blockchain ecosystem. When we "run" a smart contract, we execute an algorithm and implement all the transactions it returns. This idea dates back to Szabo (1998). An early example system was built by Wittenberger and Germany (2002) using Scheme (Sussman and Steele, 1975). Scheme is a teaching language for lambda calculus, a formal model for Turing-complete computation (Turing, 1937). That example was an exploratory system, but we see this degree of generality in smart contracts in real-world use as well. A discussion of the largest smart contact platforms is contained in The Block (2022).

When it comes to blockchain smart contracting languages, Turing-completeness is near-ubiquitous. Ethereum and the Ethereum Virtual Machine (EVM) were designed as Turing-complete (Buterin, 2014; Dannen, 2017). Both the Binance Smart Chain and Fantom are EVM-compatible (Binance Smart Chain; Fantom) and Turing-complete. Solana, Avalanche and Terra all use Rust as their smart contract language (Yakovenko, 2017; Avalanche; Kereiakes et al., 2019) which is also Turing-complete (Wohrer and Zdun, 2018; Gramlich, 2020; Bandara et al., 2021). While not in as widespread use as those

systems, Bitcoin's scripting language is also Turing-complete (Nakamoto, 2008; Wright, 2019), as is Cardano's Plutus (Cardano Team).

As explored in Chepurnoy et al. (2018), it is difficult to build a non-Turing-complete smart contract environment, but there are examples. For instance, Algorand employs a restricted model by design (Micali, 2016). Such systems maintain a small fraction of contracts in real use per The Block (2022).

This condition is also violated when we are performing computation on a platform with finite storage or a cost for computation. Initially, we will work with a model that has access to unbounded resources. Later we will see that imposing a finite resource limit does not provide a way out of our impossibility theorem; instead, it simply changes the way in which risky contracts fail.

3 Model

We work with a stylized blockchain model. The intention is not to capture every conceivable feature of blockchains or consensus algorithms. Rather, we aim simply to show how, even with a simple and clear model, there is a strong connection between computational decidability and risk. After introducing this model, we will write some simple smart contracts that exploit this connection. We will then illustrate how fundamental impossibility results prevent us for solving seemingly-simple problems.

We consider a valid, consistent system where blocks form a decentralized blockchain. This property is necessary for decentralization or trustlessness – the core innovation of blockchain. For simplicity, we only consider the case of one particular token, and drop the subscript where there is no confusion.

3.1 Risk-free Rate

It is important to clarify what we mean by a riskless loan: there is zero chance a loan denominated in the local currency will not be repaid (Damodaran, 2020). This is riskless in the same way a local-currency government bond is deemed to be riskless. A default is not a necessity but a policy choice. This is the standard definition of "riskless interest rate," defined as a debt instrument which preserves its value in all states of the world.

Definition 7 (Risk-free Rate)**.** *A rate of return $r_{t,t+m}$ is an m-period return (defined in blocks) offered by a wallet such that for a transfer into the wallet worth $W^k_{i,t}$, we get $\mathbb{E}(W^k_{i,t+m}) = W^k_{i,t+m} = W^k_{i,t} \times (1 + r_{t,t+m})$ almost surely.*

This definition of a risk-free rate allows a rate to still be considered risk-free even if

there are probability measure zero events which can generate an *ex post return not exactly equal to the* ex ante reported return. We adopt this definition as analogous to those in fiat currencies whereby, in theory, any government may default on its liabilities, but market participants may place probability zero over that event and therefore continue to treat it as a risk-free rate.[4]

In a decentralized system, a loan is riskless if some smart contract can enter into loans that provably will always be repaid. Doing this in a trustless environment means there exists some smart contract that validates risklessness. This algorithm must be open-sourced and publicly observable. We take as given that off-chain systems cannot be trustless. Similarly, we do not accept decentralized token collateral as sufficient to eliminate risk.[5]

3.2 Node Payoffs

We consider the payoffs of a node validator expending resources to reach network consensus. We also consider the deployment of potential smart contracts to create a risk-free product, and begin with the payoff to a single node when serving as either a proof-of-work miner or proof-of-stake validator. In general, decentralized consensus mechanisms feature probabilistic rewards for an individual node in order to incentivize decentralized participation, trading off the cost of verification and networking (Catalini and Gans, 2016). A node corresponds to a wallet; we use the term interchangeably in this section.

Cast in the language of proof-of-work or proof-of-stake, the algorithm hands a fixed quantity of tokens to a randomly-selected address with each block.

If the network has N nodes, $i = 1, 2, ..., N$ the payoff to a node is:

$$W_{i,t} = \begin{cases} W_{i,t-1} & \text{with probability} \quad 1-p \\ W_{i,t-1} + R & \text{with probability} \quad p, \end{cases}$$

where R is the number of tokens paid for participating in achieving network consensus and p is a probability which may be some function $p(K_{i,t})$ such that $p'(K) > 0$ for node i's resource. Exactly what K represents in practice depends on the consensus mechanism.

[4] Academic researchers typically use the short-term Treasury rate with the implicit assumption that the probability of the United States government defaulting in the short term is zero, which also is a result of a strategic complementarity in He et al. (2016). Practitioners often use overnight index swap rates, the rate for a period of time to borrow central bank reserves (Hull and White, 2013).

[5] Consider a collateralized loan that is backing some assets. Such liabilities suffer from two problems. First, you may need to trust the underlying collateral protocol. Alternatively, you could rely entirely on liquidations for protection, but there is a non-zero probability that the collateral value falls below the balance of the loan. Further, relying on risk mitigation via regulation is not a viable solution for a decentralized system.

For example, in proof-of-work consensus, K represents the hashing power while in proof-of-stake consensus, K represents the amount of tokens staked.

The expected value of the wallet's value at time t is $\mathbb{E}[W_{i,t}] = W_{a,t-1} + pR$. We can see that $Pr(W^k_{i,t} > W^k_{i,t-1}) = p$. As discussed later, this basic setup has strong implications.

3.3 Smart Contracts

We call a given smart contract decentralized if it runs on a decentralized platform and has no direct connection to off-chain activities (Tikhomirov (2018)). Such a contract can enter into trustless relationships and *in principle* issue risk-free liabilities. Here, we mainly consider the case of Turing-complete smart contracts (Turing, 1937).

Our discussion here is limited to permissionless smart contracts. If someone creates a token and manages it through smart contracts only they can control – if they effectively erect a perimeter around which code can interact with their token – then they can construct provably risk-free liabilities. But this is precisely because they are running a permissioned system which will not execute arbitrary smart contract code. They may be running on a decentralized platform but the token itself is centralized.

A smart contract is an algorithm that runs during a block and generates a collection of transactions. These are complex transactions that contain, rather than simple transfer details, algorithms which generate a set of such transfer details. Without loss of generality, we consider that smart contracts are run in order before transactions are applied and that the resulting transactions are applied after all pending non-smart-contract-derived transactions are completed.

Definition 8 (Risk-Free Contract Liability)**.** *A given smart contract liability is risk-free if it will never generate an invalid wallet.*

The closest analogue to a smart contract in the human-operated world of regulated finance are the "passive" investment vehicles as discussed in Fisch et al. (2019). These are rules-based platforms where the operators exercise minimal discretion and react mechanically to investor requests. Financial Stability Board (2021) and Hanson et al. (2015) discuss a range of options to remove discretion and default risk from money market funds. We will use this as an example to illustrate how a smart contract designed to execute this kind of mechanical transactions may behave to provide more intuition for

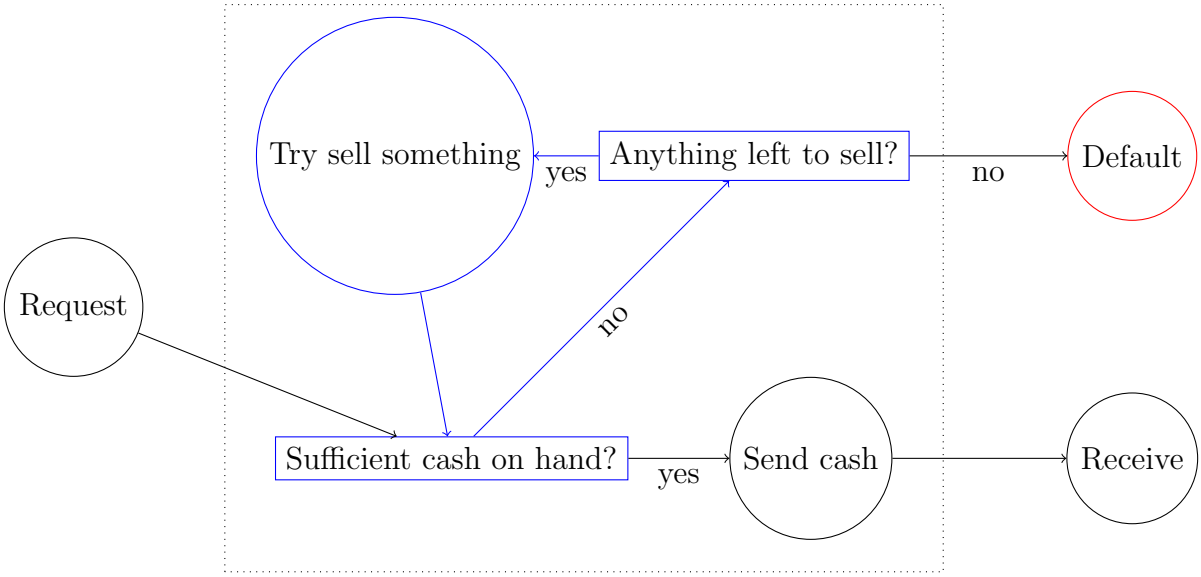

Figure 1: Withdrawal requests must be satisfied within m blocks to meet our definition of risk-free. In this simplest case we can get stuck in a non-terminating loop, in blue, if we are unable to raise sufficient cash via repeated sale attempts. If we sell everything and still do not have enough cash to satisfy a request we default. Neither outcome is consistent with the product being "risk-free."

our results.[6]

4 Model Solution

The standard technique in computer science to show a problem is undecidable is to "reduce it to halting." This is to say, we show how if we could solve problem X we can re-purpose that tool to solve the halting problem. And then, as the halting problem can never be solved, we know problem X also can never be solved. This is discussed in detail in Moret (1998); Jones (2002); Savage (1998). Church and Turing did the hard work in the 1930s showing certain decision problems were undecidable outright. Ever since that time, we can rely on simpler and easier-to-produce reductions that build atop their results. A range of proofs of this result are available in those references.

When, in computer science, we say something is undecidable, we mean that no program can be written to answer the question (Savage, 1998). This does not mean the question has any particular answer – it means we cannot know the answer without running

[6]In both cited articles, the situation where many fund managers are trapped in a workflow reminiscent of Figure 1, with no route to the good-end state, is deemed a crisis. The policy options explored lean on central authorities and look like different versions of the red arrow from Figure 2. Those reports are not idle speculation: they were published by the FSB and IMF which exist to work with Central Banks and Finance Ministries. The goal of those proposals is to simultaneously reduce discretion and default risk in what is ultimately a human-operated system. As we are discussing a purely automated system which admits no discretion, we aspire to stronger bounds on what is possible.

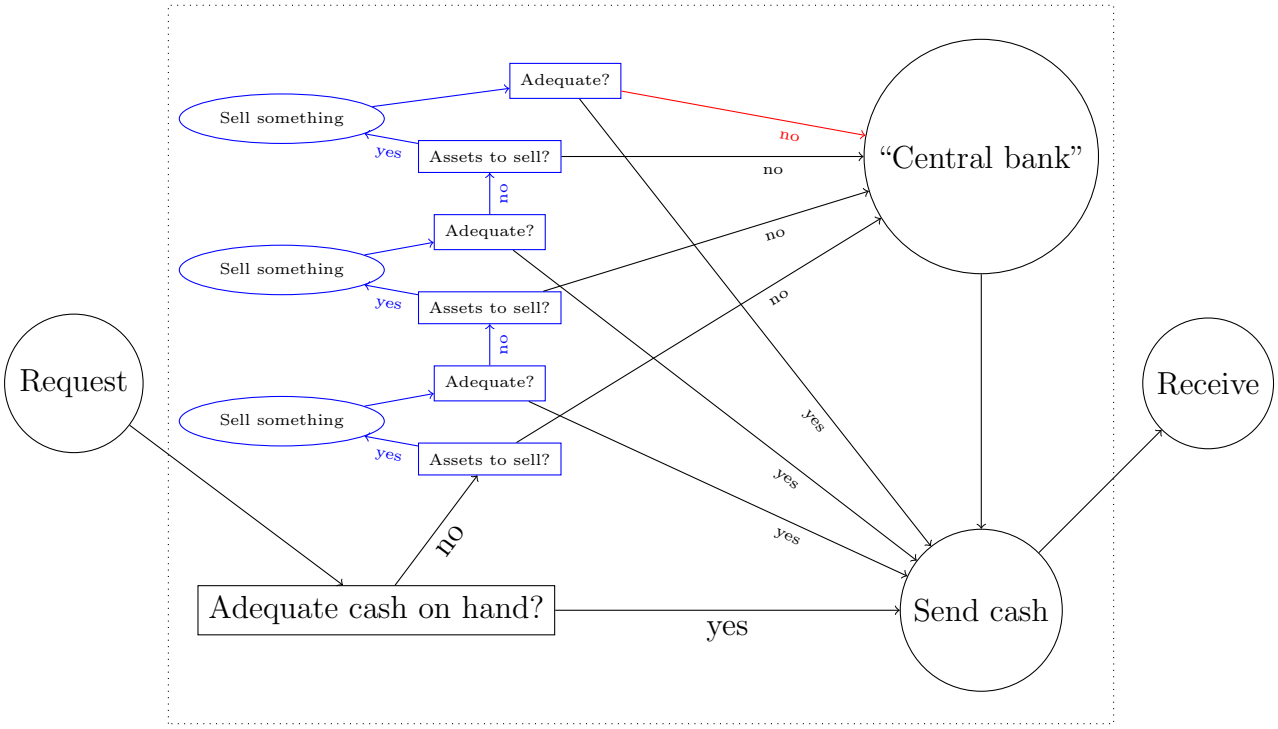

Figure 2: With a trusted lender of last resort there are no infinite loops. Here there are at most three attempts to raise more cash. The edge shown in red breaks the loop and accesses "extraordinary liquidity assistance." We cannot, in general, identify these loops in advance and insert that kind of state transition in a Turing-complete system.

the program and awaiting its successful termination. And, as one of the unanswerable questions is "will an arbitrary program halt within finite time?" we may need to wait forever. In finance, "risk" pertains to the variation in returns (Fama, 1968; Sharpe, 1964; Lintner). So any system exhibiting undecidability is, in an economic sense, "risky." This is precisely how most investments work: you buy something, wait and find out at the end whether you get your money back or not. Yes, the underlying economic activities financed by the investment occur on earth, possibly at a known address, and in a manner consist with the laws of physics. However, nobody pretends they can forecast with certainty whether an arbitrary investment will be paid back or with what return.

So, in this style, let us assume the existence of a "riskless oracle" that can determine if a given piece of smart contract code running on a decentralized blockchain will ever default. Now consider the following two functions. The first function inserts defaults before return statements in an arbitrary piece of smart contract code:

function ModifyContract(*Contract*, *debtor*)
  for all lines where *Contract* can terminate **do**
    InsertCodeBefore($b \leftarrow$ CurrentBalance(*debtor*))
    InsertCodeBefore(TransferFrom(*debtor*, $b + 1$, *anywhere*))

end for
return *Contract*.

The second function composes that with our oracle to solve the halting problem:

function SolveHalting(*Contract*)
return RisklessOracle(ModifyContract(*Contract*, x)).

The function SolveHalting tells us whether a given piece of smart contract code terminates. If our language is Turing-complete this oracle cannot be written.[7] This tells us that, for an arbitrary piece of smart contract code, we can never prove risklessness. Formally speaking, the problem is undecidable, not unsolvable. When we say this problem is undecidable we mean that in a computer science sense: we cannot mechanically work out whether a given contract will pay us back or not. In economics we call that risk.

Now, some individual block of code can be proven riskless. But, on a decentralized blockchain, to be "some individual block of code" means you can never call any external functions. Code outside your exclusive control can be anything at all. This is an incredibly strong restriction and we will expand on this point below.

We know from the design of our blockchain model that any subset of nodes N that excludes at least 1 node receives block rewards of $\sum_{i \in N} p_i < 1$. So any contracts with access only to those node's resources are not guaranteed to receive any block reward during any period of time. It may have a good chance of receiving some rewards – and that chance may increase with time – but Pr(chance) < 1.

5 Implications for the Ecosystem: Foiling Attempts At End-Running Impossibility

If we wish to achieve Pr(chance) $= 1$, we must try calling external functions. Let us consider a protocol like Uniswap (Adams et al., 2020). This is a well-known decentralized exchange (DeX) platform. Among Uniswap's well known features is that the code cannot be upgraded. When the team wishes to add new features, or change things, they must release a new version of the protocol with new keys and new wallets.

This is as immutable as a DeX can get: the code cannot be changed. However, as we will see, even this is not sufficient to allow proofs of risklessness. The reason

[7] If ModifyContract is fine, then it must be impossible to write RisklessOracle. This is again the standard approach to building halting reductions. First present clearly-working code around some black-box function. Show the composition of the given code and black-box function can solve halting. Then conclude the black-box function cannot be written.

is because Uniswap is locked down – but it is locked down into a state where anyone can interact with it. Uniswap accepts all properly formatted, and funded, incoming requests. This means the state of Uniswap in the future depends on arbitrary other smart contract state on the blockchain. Furthermore, anyone can publish anything. Once we touch Uniswap, we are interacting with unknown future state. An alternative, equivalent, perspective is presented in Sultanik et al. (2022) where those authors point out that permissionless Turing-complete systems can never be immutable because a Turing machine can simulate any other Turing machine given the right inputs. We can see everything on a public blockchain, but handling arbitrary smart contract code and state is going to require an automated procedure that certainly looks like it might be subject to halting impossibility.

We will now explore several common approaches that attempt to escape this impossibility. The purpose of this section is to show how the halting result corners us. It is possible to change the nature of the uncertainty but we cannot excise it entirely.

5.1 Printing More Money

Perhaps the first idea one has is to simply print more tokens as required to satisfy our liabilities. The first problem is that this is plainly not admitted by our blockchain model. By definition, in our framework, the only way to receive all the token rewards is to operate a centralized system. Hence risk-free yield in centralized systems is possible. This solution is, however, antithetical to the purpose of cryptocurrency. We quoted the Bitcoin whitepaper above to this effect, and in the post announcing Bitcoin to the world, Satoshi Nakamoto wrote:

> The root problem with conventional currency is all the trust that's required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust Nakamoto (2009).

What we wish to investigate here is whether these sorts of decentralized systems admit new solutions. Let us consider a decentralized facility to print more tokens. This service is either free or chargeable. If it is free, anyone can have any amount of tokens they like at any time. Again this approach is not admitted by our model and our impossibility result does not apply. However, this is not really an economic system in the traditional sense (Samuelson and Nordhaus, 2001).

If the contract is chargeable, we then immediately run into another problem: how do we know the party which needs to print more tokens can afford the fee? We can rewrite the halting reduction functions above to convert halting into this problem instead.

Printing tokens for a fee merely transforms the problem from "we are unable to pay our liabilities" into "we are unable to pay the fee to print tokens to cover our liabilities." We can rearrange our prior halting reduction as follows:

function PrintTokensFee($n, wallet$)
  if WalletBalance($wallet$) $< n \times perTokenFee$ **then**
    Terminate Fail
  end if
  if $currentSupply + n \leq supplyCap$ **then**
    $currentSupply \leftarrow currentSupply + n$
    DebitWallet($wallet, n \times perTokenFee$)
    return n
  else
    Terminate Fail
function ModifyContractFee($Contract, debtor$)
  for all lines where Contract can terminate **do**
    InsertCodeBefore($b \leftarrow$ CurrentBalance($debtor$))
    InsertCodeBefore($printableTokens \leftarrow \frac{b}{perTokenFee}$)
    InsertCodeBefore(PrintTokens($printableTokens + 1$))
  end for
  return $Contract$
function SolveHalting($Contract$)
  return RisklessOracle(ModifyContractFee($Contract, x$)).

There is a generalization of this kind of transformation in the computer science literature: Rice's Theorem (Savage, 1998). This tells us that any non-trivial property of a general purpose programming language is as undecidable as halting. Essentially Rice tells us that we can convert nearly all properties of arbitrary computer programs into halting via the sort of state-tracking we find above.

The problem is the "arbitrary" part not the code part. We can prove lots of properties about a specific piece of code. However, we can mechanically prove very little about arbitrary code. The smart contracts on a decentralized platform are arbitrary as anyone can add new ones, or modify old ones, at any time.

5.2 Capped Money Supply

Perhaps we can solve the infinite supply problems of a fee-free token printing contract by capping supply? Unfortunately this runs in to a similar Rice problem.

function PrintTokens($n, wallet$)

if $currentSupply + n \leq supplyCap$ **then**
$currentSupply \leftarrow currentSupply + n$
return n
else
Terminate Fail

If we make this function free to call we can again build a halting detector.

function ModifyContractCap($Contract, debtor$)
InsertCodeAtBeginning($supplyCap \leftarrow X$)
for all lines where Contract can terminate **do**
InsertCodeBefore(PrintTokens($X + 1$))
end for
return $Contract$
function SolveHalting($Contract$)
return CapOracle(ModifyContractCap($Contract, x$)).

So long as you can track the condition and then use it to terminate, you will always have this issue. The standard proofs of Rice generalize this concept in a way that blocks all the exits.

5.3 Paying Yield From Reserves

If we are stuck with Pr(chance) < 1, can we try to backstop some positive rate of interest with reserves? Recall that riskless zero-rate deposits are trivial on a blockchain. Let us maintain a treasury with tokens in it and use that to cover interest payments as required. Again we end up with a halting problem. Consider this function:

function ForceDefault
$reserveBalance \leftarrow$ ReadReserveBalance(...)
$maxLoanSize \leftarrow \frac{reserveBalance}{riskFreeYield}$
RiskFreeDeposit($maxLoanSize + 1$).

This generates a deposit request that cannot be satisfied out of reserves. This may appear sufficient, but can we prevent anyone from having that large a balance? In a decentralized permissionless system, how are we going to achieve that exactly? It gets worse:

function ForceDefault($time$)
$reserveBalance \leftarrow$ ReadReserveBalance(...)
$maxLoanSize \leftarrow \frac{reserveBalance}{time \times riskFreeYield}$
RiskFreeDeposit($maxLoanSize + 1, time$).

By setting the length of the deposit arbitrarily long, we can force a default with an

arbitrarily low loan balance. Furthermore, there is a simple "attack" against this system.

function ReservesAttack
  $attackBalance \leftarrow 0$
  for $attackBalance < maxBalanace + 1$ **do**
    $attackBalance \leftarrow attackBalance +$ ProbabilisticBlockchainStuff$(\dots)$
  end for
  RiskFreeDeposit$(attackBalance)$.

For the sort of blockchain model we presented above, we know what will happen: no matter how high $maxBalance$ is, there is an escalating probability of getting there as we iterate over the loop. That means that there is some non-zero probability of default in each time period. This is risk. We do not need to force a default at time t to show the system is risky. We merely need to show the probability of default is non-zero. That alone is enough to establish the variance of returns is also non-zero.

If the mining/validation process stops handing out rewards at some point, that does not help either. Then this attack does not necessarily work, but it *might* work – which in and of itself represents risk. Furthermore, the initial "halting detector" scheme clearly works when no new tokens are created.

Shifting parameters, or schemes, only moves this risk around the probability space. Rice tells us that we cannot escape Turing through this kind of transformation.

5.4 Trust & Centralization

One simple solution is to lock-down the ability to publish new smart contract code and only allow trusted parties to modify things. This is clearly not a decentralized system, but it is one in which provable risklessness is possible.

Each new smart contract, and all updates to old ones, would need to pass through a vetting process. And risklessness would then depend on trusting that vetting process. If an infinite loop, or default, made it through the process somehow then the system would still generate defaults.

This means, in the same way we cannot prove properties about arbitrary code, we cannot build a foolproof vetting process. To the extent the vetting relies entirely on well studied tools and formally grounded methods it will be fine. But mistakes can occur and there is no generic proof checking process for arbitrary statements.

5.5 Gas Fees

So far, we have not discussed computational costs. In practice, executing smart contracts incurs computational costs which are passed on to users as "gas" fees. The introduction of gas fees bounds the cost that a smart contract may have, meaning that a Turing-complete language running on a gas-fee-limited platform is not truly Turing-complete. For example, in Ethereum transactions, one specifies a maximum amount of gas to be spent. If a transaction uses less than this limit, the excess amount is refunded. But if a transaction hits the cap and fails, the gas is retained by the miner who processed the transaction rather than refunded.

At face value, it seems that gas fees may resolve the computational loop issue shown above. However, this does not solve the problem, because Rice's Theorem tells us that we cannot pre-screen for this problem (Savage, 1998). So long as the underlying smart contract language is Turing-complete, we cannot identify such problems via any method more efficient than running programs and waiting to see if they fail for lack of gas.

Further, a contract that halts when it runs out of gas is also a contract with a non-zero probability of spending the entire balance checking if it might lose money some other way.

5.6 Macroeconomic Implications

We discuss the practical implications of our results for token supply policy design and discuss whether the results apply to central bank digital currencies (CBDCs).

5.6.1 Monetary Policy

In our framework, we consider tokens emitted by mining or validation processes. These are the so-called "native tokens" of a blockchain, created in such a way that future money supply is known in advance. Bitcoin and Ethereum are the best-known examples of such tokens. So far, we investigated whether smart contracts could be used to marshal such tokens into traditional risk-free liabilities and proved they could not.

The essential macroeconomic implication of our finding is that no lever quite like TradFi "monetary policy" can exist for these kinds of native tokens. This significantly reduces what of monetary economics we can borrow (Lucas and Stokey, 1983; Lucas, 1996). And if we then apply Mundell-Fleming's Trilemma the implications are even broader as decentralization also constrains the policy space. Exchange controls are not feasible as there is, by definition, no trusted party in charge to impose constraints on them. Similarly, the exchange rate must be free floating; one can try to design a smart contract to stabilize the rate but it will, in turn, be subject to these impossibility results

around reliability. The lack of a trusted central authority also implies there are no arbiters implementing system-wide tax or labor policies. Finally, there is nothing like a fiscal policy in a decentralized world without a central government.

This only leaves trade policy: participants can attempt to influence the balance of payments or exchange rate (token prices). And these predictions are entirely consistent with what we see in the crypto economy where we find a near-endless stream of participants behaving like government officials on a trade mission. Just substitute "invest in our economy" with "invest in our ecosystem" and the connection is clear. Furthermore there is little, if any, debate about monetary policy. To the extent participants talk about interest rates it is in the service of the "trade ministry" side of things. "Our protocol offers high yields" is not a request to implement tighter monetary policy – it is a marketing claim. The same applies for "we can offer lower-cost loans."

To some extent this is by design. Nakamoto (2009) makes plain that the goal is to build a system without human-controlled monetary policy. That is fine. But the solution – decentralization – drags along an unintended market incompleteness that prevents the emergence of even an endogenous government-bond-like risk-free product. A crypto economy, and finance within such an economy, is significantly different from a traditional one. We now know many desirable products, policies and techniques will not carry over.

5.6.2 Central Bank Digital Currencies

The limitations that we discuss do not apply to CBDCs since such a product, by definition, has a monopoly issuer (i.e., is centralized) and depends on off-chain activities. In addition, CBDCs do not have pre-programmed monetary policy by the nature of their underlying fiat currency.

To the extent a CBDC is transferred on a decentralized blockchain with fees paid in native tokens some element of incompleteness would apply. For example: a CBDC-backed government bond running on the Ethereum blockchain would have difficulty paying coupons if the gas price rose dramatically against the value of the coupon. Again this sort of problem – and whatever solutions for it may or may not exist – is totally foreign to TradFi.

6 Conclusion

This paper evaluates the use of Turing-complete languages for smart contract implementation. We show that generating decentralized, fixed-monetary-policy, risk-free assets is

not merely an engineering problem for a reasonably generic class of blockchain consensus algorithms. We must acknowledge that such designs involve trade-offs which introduce market incompleteness (Arrow and Debreu, 1954).[8]

These results apply to this class of decentralized, permissionless native tokens only. This includes the major Layer-1 tokens such as Bitcoin and Ethereum. It does not apply to centralized tokens or activities on permissioned platforms. In those cases someone – a company or developer or contract owner – controls access and we are no longer talking about uncompromised DeFi. Rather, we are simply constructing centralized financial products, with gatekeepers, distinguished only by the use of blockchain and token primitives. The underlying software may be different but the high-level system properties will resemble traditional finance rather than something new.

This leaves two choices for building products. First, we can choose to operate in a restricted environment. We can compromise on the decentralization, allow arbitrary token-printing, or employ a simpler smart contract language. Whether these approaches can yield a long-term-viable system with novel properties is an open question. Second, we can attempt to reconstruct the scaffolding of finance absent risk-free market interest rates and work with whatever forms emerge. For cash trading and simple short-term derivatives, approximations regarding risklessness and funding rates may be sufficient in practice. Additional future work may also enable connections to extant financial engineering models which clearly delineate the space of possible products. As a start, works such as Angeris et al. (2021), Black (1972), Margrabe (1978), and Kan et al. (2016) offer methods to apply traditional financial engineering absent risk-free instruments.

[8]For a more formal treatment of the consequences of this incompleteness, see Pascucci (2010). Section 1.2.5 illustrates how the lack of risk-free discounting mis-specifies a sample problem. Then, as we are unable to adopt Hypothesis 10.19, we cannot get to Theorem 10.50. The results of Chapter 10 remain out of reach and we are left with under-specified problems.

References

J. Abadi and M. K. Brunnermeier. Blockchain Economics. *Working Paper*, pages 1–53, 2018. ISSN 1556-5068. doi: 10.3386/w25407.

H. Adams, N. Zinsmeister, and D. Robinson. Uniswap v2 core, 2020. URL https://uniswap.org/whitepaper.pdf.

R. Aliber and C. Kindleberger. *Manias, Panics, and Crashes: A History of Financial Crises*. 01 2015. ISBN 978-1-137-52575-8. doi: 10.1007/978-1-137-52574-1.

G. Angeris, A. Evans, and T. Chitra. Replicating monotonic payoffs without oracles. 2021. doi: 10.48550/arXiv.2103.14769.

K. J. Arrow and G. Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, 22(3):265–290, 1954. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1907353.

Avalanche. What is avalanche. URL https://docs.avax.network/.

E. Bandara, X. Liang, P. Foytik, S. Shetty, N. Ranasinghe, K. D. Zoysa, and W. K. Ng. SaaS - Microservices-Based Scalable Smart Contract Architecture. *Communications in Computer and Information Science*, 1364(February):228–243, 2021. ISSN 18650937. doi: 10.1007/978-981-16-0422-5_16.

B. Biais, C. Bisière, M. Bouvard, and C. Casamatta. Blockchains, Coordination, and Forks. *AEA Papers and Proceedings*, 109:88–92, 2019. ISSN 2574-0768. doi: 10.1257/pandp.20191018.

Binance Smart Chain. Binance smart chain. URL https://docs.binance.org/faq/bsc/bsc.html.

F. Black. Towards a Fully Automated Exchange, Part I. *Financial Analysts Journal*, 27: 29–34, 1971a.

F. Black. Toward a Fully Automated Stock Exchange, Part II. *Financial Analysts Journal*, (November-December):25–87, 1971b.

F. Black. Capital market equilibrium with restricted borrowing. *The Journal of Business*, 45(3):444–455, 1972. ISSN 00219398, 15375374. URL http://www.jstor.org/stable/2351499.

F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973. ISSN 00223808, 1537534X. URL http://www.jstor.org/stable/1831029.

. Brigo, Damiano and F. Mercurio. *Interest rate models - theory and practise : with smile, inflation and credit / Damiano Brigo, Fabio Mercurio.* Springer finance. Springer, New York, 2nd ed. edition, 2006. ISBN 3540221492.

V. Buterin. Ethereum White Paper: A Next Generation Smart Contract Decentralized Application Platform. *Etherum*, (January):1–36, 2014. URL https://github.com/ethereum/wiki/wiki/White-Paper.

R. J. Caballero, E. Farhi, and P. O. Gourinchas. The safe assets shortage conundrum.

Journal of Economic Perspectives, 31(3):29–46, 2017. ISSN 08953309. doi: 10.1257/jep.31.3.29.

Cardano Team. Plutus. URL https://developers.cardano.org/docs/smart-contracts/plutus.

C. Catalini and J. S. Gans. Some Simple Economics of the Blockchain. *SSRN Electronic Journal*, 2016. doi: 10.2139/ssrn.2874598.

A. Chepurnoy, V. Kharin, and D. Meshkov. Self-reproducing coins as universal turing machine. In J. Garcia-Alfaro, J. Herrera-Joancomartí, G. Livraga, and R. Rios, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 57–64, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00305-0.

L. W. Cong and Z. He. Blockchain disruption and smart contracts. *The Review of Financial Studies*, 32(5):1754–1797, 2019.

L. W. Cong, Z. He, and J. Li. Decentralized Mining in Centralized Pools. *Review of Financial Studies*, 34(3):1191–1235, 2021a. ISSN 14657368. doi: 10.1093/rfs/hhaa040.

L. W. Cong, Y. Li, and N. Wang. Tokenomics: Dynamic adoption and valuation. *The Review of Financial Studies*, 34(3):1105–1155, 2021b.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001. ISBN 0262032937. URL http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen/dp/0262032937%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0262032937.

H. Dam, A. Macrina, D. Skovmand, and D. Sloth. Rational models for inflation-linked derivatives, 2018. URL https://arxiv.org/abs/1801.08804.

A. Damodaran. What is the riskfree rate? A Search for the Basic Building Block. *Journal of New Finance*, 1(3), 2020. doi: 10.46671/2521-2486.1010.

C. Dannen. *Introducing Ethereum and solidity*. 2017.

E. F. Fama. Risk, return and equilibrium: Some clarifying comments. *The Journal of Finance*, 23(1):29–40, 1968. doi: https://doi.org/10.1111/j.1540-6261.1968.tb02996.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1968.tb02996.x.

L. Fang, B. Hor, E. Azmi, and K. Win Win. *How to DeFi: Advanced*. Tea Spoon Publishing, 2021.

Fantom. Fantom. URL https://fantom.foundation/.

Financial Stability Board. Policy proposals to enhance money market fund resilience, October 2021. URL https://www.fsb.org/wp-content/uploads/P111021-2.pdf.

J. Fisch, A. Hamdani, and S. D. Solomon. The new titans of wall street: A theoretical framework for passive investors. *University of Pennsylvania Law Review*, 2019.

P. M. Garber. *Famous First Bubbles: The Fundamentals of Early Manias*. The MIT Press, 2001. ISBN 978-0262571531.

G. Gorton. The history and economics of safe assets. *Annual Review of Economics*, 9: 547–586, 2017. ISSN 19411391. doi: 10.1146/annurev-economics-033017-125810.

B. Gramlich. Smart Contract Languages: A Thorough Comparison. *Working Paper*, (October), 2020. doi: 10.13140/RG.2.2.22479.92326.

S. G. Hanson, D. S. Scharfstein, and A. Sunderam. An Evaluation of Money Market Fund Reform Proposals. *IMF Economic Review*, 63(4):984–1023, November 2015. URL https://ideas.repec.org/a/pal/imfecr/v63y2015i4p984-1023.html.

Z. He, A. Krishnamurthy, and K. Milbradt. What makes US government bonds safe assets? *American Economic Review: Papers Proceedings*, 106(5):519–523, 2016. ISSN 00028282. doi: 10.1257/aer.p20161109.

G. Huberman, J. D. Leshno, and C. Moallemi. An Economist's Perspective on the Bitcoin Payment System. *AEA Papers and Proceedings*, 109:93–96, 2019. ISSN 2574-0768. doi: 10.1257/pandp.20191019.

J. Hull. *Options, futures, and other derivatives*. Pearson Prentice Hall, Upper Saddle River, NJ [u.a.], 6. ed., pearson internat. ed edition, 2006. ISBN 978-0-13-197705-1. URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+563580607&sourceid=fbw_bibsonomy.

J. Hull and A. White. Credit Derivatives. In *Handbook of the Economics of Finance SET*, volume 2, pages 1363–1396. Elsevier B.V., 2013. ISBN 9780444594068. doi: 10.1016/B978-0-44-459406-8.00020-2. URL http://dx.doi.org/10.1016/B978-0-44-459406-8.00020-2.

M. Jansen, F. Hdhili, R. Gouiaa, and Z. Qasem. Do smart contract languages need to be turing complete? *Advances in Intelligent Systems and Computing*, 1010(January): 19–26, 2020. ISSN 21945365. doi: 10.1007/978-3-030-23813-1_3.

N. D. Jones. *Computability and Complexity from a Programming Perspective*, pages 79–135. Springer Netherlands, Dordrecht, 2002. ISBN 978-94-010-0413-8. doi: 10.1007/978-94-010-0413-8_4. URL https://doi.org/10.1007/978-94-010-0413-8_4.

R. Kan, X. Wang, and G. Zhou. Optimal portfolio selection with and without risk-free asset. *SSRN Electronic Journal*, 01 2016. doi: 10.2139/ssrn.2729429.

E. Kereiakes, D. Kwon, M. D. Maggio, and N. Platias. Terra money: Stability and adoption, April 2019.

T. Krupa, M. Ries, I. Kotuliak, K. Košál, and R. Bencel. Security issues of smart contracts in ethereum platforms. *Conference of Open Innovation Association, FRUCT*, 2021-Janua, 2021. ISSN 23057254. doi: 10.23919/FRUCT50888.2021.9347617.

J. Lintner. The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *The review of economics and statistics.*, 47(1). ISSN 0034-6535.

R. E. J. Lucas. Nobel Lecture: Monetary Neutrality. *The Journal of Political Economy*, 104(4):661–682, 1996.

R. E. J. Lucas and N. L. Stokey. Optimal Fiscal and Monetary Policy in an Economy Without Capital. *Journal of Monetary Economics*, 12:55–93, 1983.

K. Malinova and A. Park. Tokenomics: When Tokens Beat Equity. *SSRN Electronic Journal*, 2018. doi: 10.2139/ssrn.3286825.

W. Margrabe. The value of an option to exchange one asset for another. *The Journal of Finance*, 33(1):177–186, 1978. ISSN 00221082, 15406261. URL http://www.jstor.org/stable/2326358.

H. Markowitz. Portfolio selection*. *The Journal of Finance*, 7(1):77–91, 1952. doi: https://doi.org/10.1111/j.1540-6261.1952.tb01525.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1952.tb01525.x.

L. Menand. Unappropriated dollars: The fed's ad hoc lending facilities and the rules that govern them, 2020. URL https://ecgi.global/sites/default/files/working_papers/documents/menandfinal.pdf.

R. C. Merkle. Protocols for public key cryptosystems. In *1980 IEEE Symposium on Security and Privacy*, pages 122–122, 1980. doi: 10.1109/SP.1980.10006.

S. Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016. URL http://arxiv.org/abs/1607.01341.

F. Modigliani and M. Miller. The cost of capital, corporation finance and the theory of investment. *The American Economic Review*, 48(3):261–297, 1958. ISSN 0002-8282.

B. M. E. B. M. E. Moret. *The theory of computation / Bernard M. Moret.* Addison-Wesley, Reading, Mass, 1998. ISBN 0201258285.

S. Nakamoto. A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260, 2008.

S. Nakamoto. Bitcoin open source implementation of p2p currency, 2009. URL http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source.

D. Nelson. Solana halted by bug linked to certain cold storage transactions, June 2022. URL https://www.coindesk.com/tech/2022/06/02/solana-halted-by-bug-linked-to-certain-cold-storage-transactions/.

P. O'Neill. Can Markets be Fully Automated ? Evidence From an Automated Market Maker. *Working Paper*, 2021.

A. Pascucci. *PDE and Martingale Methods in Option Pricing.* Springer-Verlag, Milan, Italy, 2010.

J. Ponciano. Terra blockchain halted to 'prevent attacks' after luna token crashes nearly 100 URL https://www.forbes.com/sites/jonathanponciano/2022/05/12/terra-blockchain-halted-to-prevent-attacks-after-luna-token-crashes-nearly-100-ove

O. President's Working Group On Financial Markets, FDIC. Overview of recent events and potential reform options for money market funds, December 2020.

d. F. Primavera. Blockchain Technology and Decentralized Governance : The Pitfalls of a Trustless Dream. In *Decentralized Thriving: Governance and Community on the Web 3.0, 2019.*, pages 1–9, 2020.

PWC. Capital Markets in 2030. Technical report, 2019. URL https://www.pwc.com/gx/en/audit-services/capital-market/publications/capital-markets-2030.pdf.

K. Qin, L. Zhou, B. Livshits, and A. Gervais. Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12674 LNCS:3–32, 2021. ISSN 16113349. doi: 10.1007/978-3-662-64322-8_1.

R. Rebonato. *Modern Pricing of Interest-Rate Derivatives : The LIBOR Market Model and Beyond / Riccardo Rebonato.* Princeton University Press, Princeton, NJ, course book edition, 2012. ISBN 0-691-08973-6.

C. M. Reinhart and K. S. Rogoff. *This Time Is Different: Eight Centuries of Financial Folly.* Number 8973 in Economics Books. Princeton University Press, 2009. URL https://ideas.repec.org/b/pup/pbooks/8973.html.

P. A. Samuelson and W. D. Nordhaus. *Economics.* McGraw Hill, New York, 2001.

J. E. Savage. *Models of computation*, volume 136. Addison-Wesley Reading, MA, 1998.

W. F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk*. *The Journal of Finance*, 19(3):425–442, 1964. doi: https://doi.org/10.1111/j.1540-6261.1964.tb02865.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1964.tb02865.x.

D. Shrier, D. Sharma, and A. Pentland. Blockchain and Financial Services: The Fifth Horizon of Networked Innovation - Massachusetts Institute of Technology. (April): 1–10, 2016. URL https://cdn.www.getsmarter.com/career-advice/wp-content/uploads/2016/12/mit_blockchain_and_fin_services_report.pdf.

A. Smith. *An inquiry into the nature and causes of the wealth of nations / by Adam Smith.* Printed for W. Strahan and T. Cadell London, 1776.

E. Sultanik, A. Remie, F. Manzano, T. Brunson, S. Moelius, E. Kilmer, M. Myers, T. Amir, and S. S. and. Are blockchains decentralized? unintended centralities in distributed ledgers, June 2022.

G. J. Sussman and G. L. Steele. Scheme: An interpreter for extended lambda calculus. Technical Report AI Memo No. 349, Massachusetts Institute of Technology, Cambridge, UK, December 1975.

N. Szabo. Secure property titles with owner authority, 1998.

The Block. Digital asset outlook 2022, 2022.

S. Tikhomirov. Ethereum: State of knowledge and research perspectives. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10723 LNCS:206–221, 2018. ISSN 16113349. doi: 10.1007/978-3-319-75650-9_14.

A. M. Turing. Computability and λ-Definability. *The Journal of Symbolic Logic*, 2(4): 153–163, 1937.

J. Wittenberger and Germany. Askemos-a distributed settlement. 2002.

M. Wohrer and U. Zdun. Smart contracts: Security patterns in the ethereum ecosystem and solidity. *2018 IEEE 1st International Workshop on Blockchain Oriented*

Software Engineering, IWBOSE 2018 - Proceedings, 2018-Janua:2–8, 2018. doi: 10.1109/IWBOSE.2018.8327565.

C. S. Wright. A proof of turing completeness in bitcoin script. In *IntelliSys*, 2019.

J. C. Wu and F. D. Xia. Measuring the macroeconomic impact of monetary policy at the zero lower bound. *Journal of Money, Credit and Banking*, 48(2-3):253–291, 2016.

A. Yakovenko. Solana, 2017. URL https://solana.com/solana-whitepaper.pdf.