Smart contract model for complex reality transaction

Tianyu Feng, Xiao Yu, Yueting Chai and Yi Liu
National Engineering Laboratory for E-commerce Technologies, Tsinghua University, Beijing, China

Received 11 March 2019
Revised 3 June 2019
Accepted 20 June 2019

Abstract

Purpose – The application of smart contract can greatly reduce transaction costs and improve transaction efficiency. The existing smart contract are expensive, single application scenario and inefficient. This paper aims to propose a new smart contract model to solve these problems.

Design/methodology/approach – By investigating the research history, models and platforms, this paper summarizes the shortcomings of existing smart contracts. Based on the content and architecture of traditional contract, a smart contract model with wider application scope is designed.

Findings – In this paper, several models are used to describe the operation mechanism of smart contracts. To facilitate computer execution, a decomposition method is proposed, which divides smart contracts into several sub-contracts. Then, the advantages and deployment methods of smart contract are discussed. On this basis, a specific example is given to illustrate how the application of smart contract will change our life.

Originality/value – Smart contract is gradually applied to more fields. In this paper, the structure and operation mechanism of smart contract system in reality are given, which will be beneficial to the application of smart contract to more complex systems.

Keywords Evolution, Modelling, Blockchain, Smart contract

Paper type Research paper

1. Introduction

Smart contract is the program deployed in a distributed network that can acquire outside information and update the internal state automatically. Szabo proposed the concept of smart contract as early as 1993, but it was not widely promoted because of the lack of digital systems and technologies that could support programming. After the emergence of Ethereum, a built-in Turing-complete scripting language, the popularity of smart contracts began to rise rapidly. In theory, Ethereum can create any transaction type and application through the "smart contract" mechanism (Gao *et al.*, 2016). Currently, smart contracts are mainly used in blockchain or distributed ledgers. The smart contracts used by platforms such as Ethereum (Buterin, 2014) and Hyperledger Fabric (Androulaki *et al.*, 2018) have some deviation from the initial definition of smart contracts.

Smart contracts can realize automation, intelligence, efficiency and credibility of transactions. There are still some problems in existing smart contract, such as the low degree of intelligence and the inefficiency of consensus mechanism. To apply smart contract

International Journal of Crowd Science
Vol. 3 No. 2, 2019
pp. 184-197
Emerald Publishing Limited
2398-7294
DOI 10.1108/IJCS-03-2019-0010

to real life, this paper designs smart contract model which is more intelligent, better structured and easy for computer to understand. It will be conducive to the application of smart contract to more fields.

Through the research history, existing shortcomings of smart contracts, traditional contracts and practical needs, the future goals of smart contracts are proposed. In Section 2, we will focus on the history of smart contracts. In Section 3, the key point is to combine the shortcomings of smart contract and traditional contract, to carry out the demand analysis of smart contract and introduce the intelligent method. In Section 4, we describe the ultimate goal of our smart contract model through execution model, structure model and state model. In Section 5, taking the milk industry as an example illustrate that the application of smart contract can change the existing production and consumption structure and the transaction mode.

2. History of research

2.1 The history of smart contract research

Majority of smart contract procedures are based on blockchain technology. Existing smart contracts control digital currencies principal. Whereas they were found having defects and deficiencies in the course of their operations, leading to more serious consequences, such as "The DAO" and Ethereum Parity wallet incident, which caused a large number of digital currencies to be stolen, causing a large loss. If these program bug cannot be processed, smart contracts will be difficult to manipulate real assets.

2.1.1 Intelligence of smart contract. How to make smart contracts more intelligent is one of the focuses of current researchers. The ultimate goal of smart contracts is to use computers instead of human beings to bring more efficient and accurate transaction efficiency.

Frantz and Nowostawski (2016) proposed a semi-automatical method for translation of human-readable contracts into computational language equivalents. Idelberger *et al.* (2016) discuss the advantages of law and technology in a logic-based smart contract platform, and propose logical and procedural approaches that complement each other. Raskin (2016) studied smart contracts from a legal perspective in the literature published in 2016, explaining how smart contracts operate in current contract law. Norta (2016) designed a smart contract application layer for decentralized autonomous organizations for the problem of false claims that are difficult to resolve in the event of conflicts in traditional contract machines and the current smart contract deployment at the protocol level.

2.1.2 Efficiency of contract implementation. The efficiency of smart contracts is not high enough. For large-scale applications, it needs to increase processing speed.

Dickerson *et al.* (2017) proposed a new approach based on adaptive software memory technology that allows miners and verifiers to execute smart contracts in parallel, proving that multicore architectures can increase the throughput of smart contract processing. Cong and He (2018) deemed that smart contracts and distributed ledgers can save costs and analyze the impact of blockchain technology on business. The birth of block chain guarantees the security of smart contract, but the existing consensus mechanism is inefficient; thus, smart contract cannot be put into use on a large scale.

2.1.3 The application of smart contract. The ultimate goal of all research on smart contracts aims at applying them to reality. At present, there are many studies for real life in smart contracts.

Brandstätt focuses on the efficiency characteristics of smart contracts. Based on the challenges of distributed generation and smart grids to the distribution network, regulatory reforms are needed. Smart contracts solve this problem well. Besides, network capacity and

security issues can be settled appropriately as well. The article lists three examples for the application of smart contracts in postponing network investment (Brandstätt *et al.*, 2011). Nugent *et al.* (2016) apply smart contracts to complex data management in medical clinical trials, avoiding tampering with data due to other factors, improving test credibility and improving health service levels. Christidis considers applying blockchain and smart contract technology to the Internet of Things to solve the problem of facilitating sharing and services and automatically addressing password verification. Deployment of the system can save a great number of time and cost. In the deployment process, we need to consider the impact of the privacy of digital assets on the expectation of network transactions (Miller and Bentov, 2016).

2.1.4 The model of smart contract. Models are very important in scientific research. It can reveal the shape, characteristics, essence and development law of the object of study in an abstract form. But now, there are relatively few theoretical studies on the smart contract model.

The smart contract in Ethereum can input value and information, and change its value and status after entering (Buterin, 2014). Richard (2015) thinks that the model of smart contract "smart contract is a computer program running on copiable and shared books, which can process information, receive, store, and send value". The smart contracts of platforms such as Tron and Ethereum use such logic. Eze *et al.* (2017) proposed a triple smart contract model based on blockchain technology. The model is decomposed into technical models, legal models and business models for processing. To support this mechanism, the assets are divided into four categories, as shown in Figure 1.

2.2 Platform

Different smart contract platforms adopt different blockchain types. For example, Ethereum mainly uses the public chain, while Hyperledger Fabric adopts the alliance chain. Ethereum is the first platform to truly support smart contract programming. Unlike UTXO, which is used by Bitcoin, it uses the Account model, so the state can be directly reflected in the latest block without backtracking. The contract code runs on the EVM (Buterin, 2014).

Hyperledger Fabric proposes a blockchain solution consisting of peers, chains, channels, and consensus services. Each peer can have multiple account books and participate in multiple chains at the same time. The peers communicate via a channel connection (Androulaki *et al.*, 2018).

Corda uses the Kotlin programming language and runs in the JVM. It uses the UTXO model for the account, but does not use the blockchain for record transactions, does not require mining, and has no built-in tokens. By Notary verification method, transactions in Corda will not be broadcast to the entire network, only between the participants and the notary. This makes Corda fast and low-cost, but it also brings poor scalability.

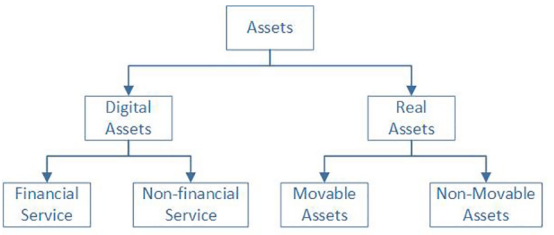

Figure 1.
Assets classfication

Tron's original intention was to achieve a truly decentralized internet, allowing users to take ownership and disposition of the data they created, and to solve the serious problem of Internet centralization at this stage. Compared to a centralized Internet architecture, Tron is able to: data freedom, decentralized infrastructure and personal digital asset issuance. Tron uses the UTXO model to build blockchains, and to be more intuitive, Tron's designers abstracted the concept of accounts and cached account information locally, which is convenient for users.

Also, there are some platforms used to validate the logic of contract. The formal verification tool CertiK turns the smart contract into a mathematical model and verifies it through logical operations to prove the security of the smart contract. The whole verification process divides a comprehensive problem into several small problems that are easy to prove (Ronghui *et al.*, 2018). After proving all minor problems, recombine them again while ensuring end-to-end correctness. Chengdu Blockchain Security has developed a VaaS blockchain formal verification platform to provide formal verification services for Eos. GrantPassmore released ImandraContracts at the 2nd Global Blockchain Developer Summit to provide formal verification of smart contracts for Ethereum. The MATRIX platform advocates the integration of blockchain and smart contracts, using scripting language to illustrate contractual intent, transforming into contract code through neural network-based code generation technology, and optimizing and transforming code against network and analog performance evaluation methods until Generate contract codes that meet security requirements (Steve, 2018).

3. Demand analysis of smart contract

3.1 From traditional contract to smart contract

The basic contents of traditional contracts include: subject matter, contract participants, quantity and quality, price or remuneration, contract duration, mode of performance, liability for breach of contract, dispute settlement method and additional clauses (The State Council, 1999). Participants, contract content (transaction content, mode of performance, time limit, etc.), liability for breach of contract and dispute settlement method are the necessary contents of a contract.

A contract needs to go through several stages from preparation to the end of its life cycle: preparation before signing, contract examination and approval, contract signing, contract review, contract performance and contract filing. When disputes arise over the content of the contract, the parties usually consult first. If the negotiation is unsuccessful, the relevant departments shall apply to supervise the implementation of the dispute settlement method in accordance with the contract; if there is a serious breach of contract, legal means shall be used to resolve it.

Traditional contracts are manually signed, and their implementation needs the protection of third-party organizations. Once disputes arise, they can only be judged afterwards, which will cause delays in the post-project. Traditional contracts are signed according to contract law, mostly in paper version, which brings difficulties in management and contract progress monitoring. Although electronic contract has appeared, it only facilitates circulation and synchronization, and has not fundamentally solved the problems of timeliness and monitoring (Table I).

Traditional contracts have great inspiration for logic completeness of smart contracts. The design of smart contract not only needs to consider the participants and contract content, but also needs to incorporate breach of contract liability and penalty and dispute resolution into the contract implementation process to eliminate the drawbacks of traditional contracts as far as possible.

Schedule	Shortcomings
Signing	Ambiguity Inequality caused by information asymmetry
Execution	Delay High default rate
Dispute	Conflict of different contract contents Difficult in adjudication
Default	Low default cost High resistance
Security	Easily tampered with Difficult to preserve High management cost

Table I.
Shortcomings of traditional contracts

3.2 Deficiencies of existing smart contract

3.2.1 The intelligence of smart contracts is not enough. The current smart contracts can only perform simple digital currency verification and exchange. The contracts are all man-made and not truly intelligent. Security issues and asset theft are not automatically recognized, resulting in significant losses and security breaches that can be discovered. The intelligence of contract generation can be realized by applying speech recognition, natural language processing, transaction model recognition and security check based on semantic analysis to the contract generation process. In the process of implementation, digital assets are used as the underlying support, which is convenient for smart contracts to call.

3.2.2 Scalability, compatibility and portability are poor. At present, all contract deployment and execution can only be on a specific blockchain, and cross-chain collaboration confuses the researchers. As the transaction data is stored in the contract, the new features are difficult to expand. All contracts are designed for specific needs and are only valid for one or a type of transaction. If there are subtle differences, a new contract is required. Thus, we can establish standardized, pluggable interfaces for all contracts and unify contract specifications and rules. Use cross-chain technology to solve the problem of isolated islands in the chain.

3.2.3 Contracts may conflict with the realism laws. At present, the design of smart contracts has not considered the constraints of real laws. The legality of many digital currencies, including Bitcoin, has not been recognized in most countries. When manipulating real assets, you need to consider the law from the beginning of the contract. After the contract is converted to machine language, it is also necessary to ensure that the contract still meets the relevant requirements of the law. These can be solved by introducing computational law to ensure that the contract design meets the contractual standards set by law and is tested after the contract is generated.

3.2.4 It is difficult to guarantee the fairness and credibility. The fairness of the contract means that the contract will not appear to be beneficial to one of the parties, nor will there be a contract that violates the principle of market transactions, and no one can obtain an undue advantage through the contract. Validation by formal methods confirms that the contract does not present a vulnerability that is clearly beneficial to one party. Calculate whether the legal analysis of the contract is fair. Is there a contractual content that violates the contract law?

3.2.5 Access control. Access control issues will arise when multiple contracts are executed simultaneously. If you can't get a good solution, syncing and concurrency can lead to chaos, which can lead to confusion of contract logic data, resulting in significant economic

and property damage. The order of execution can be confirmed by time stamping, and the concurrent system inspection mechanism is used for verification.

3.2.6 Smart contracts are difficult to upgrade. In the smart contract project represented by Ethereum, all the data are stored in the contract. Once the contract is upgraded, all the data stored in the previous contract will be cleared (Buterin, 2014). This is not in line with the current software's idea of resolving vulnerabilities and adding content through upgrades. Therefore, we can divide the contract into data contracts and logical contracts, and only upgrade the logical contracts to ensure the credibility of the data contracts. It is also possible to create a new contract at the time of the upgrade and use the existing contract content as a new block for the new contract.

3.3 The goal of smart contract

Based on the above information, we hope that the smart contract can automatically generate contracts for real-world application scenarios, and adjust according to the user's individual needs. After the contract is signed, the contract can be automatically executed. When a certain condition is reached, the contract is automatically executed according to the preset input and rules. It does not need external transaction triggering, automatically changes the contract status, and records the new status to the blockchain to prevent human tampering and avoid individual person or organize monopoly information. In addition to this, it is necessary to ensure the supervision of all auto-executed contracts. The system needs to solve the problem of automatically executed transactions and to prioritize the transactions with less resources.

In view of the above problems, with reference to the shortage of traditional contract and current smart contracts, combined with the current research status, the objectives of smart contract are: intelligent, consistent, efficient, compatible, convenient and observable.

3.3.1 Intelligence. Intelligence means that you can automatically convert requirements from natural language to machine language, automatically match according to requirements, generate required contracts and interact with real-world assets. It can be divided into the following sections: natural language requirements are converted to machine language requirements; machine language requirements are converted to machine language contracts; and machine language contracts are converted to standardized contracts, contract verification and contract execution. In the end, it can be guaranteed that the contract can be completed automatically without any manual intervention from generation to running to logout.

3.3.2 Consistency. The consistency of smart contracts has two levels of meaning: the consistency of the state of the system, consistent with the real law. Consistent system status means that all assets and contract execution in the smart contract system are in the same state, ensuring that there is no "double flower problem". Consistent with the real law indicate that all the clauses in the smart contract cannot violate the law, and all of its actions should meet the relevant requirements of the law, but some links may be different from the contract. The contract needs to include the needs of the parties and related laws, all of which are clearly stated and there is no ambiguity.

3.3.3 Efficiency. Efficient means that the smart contract is executed at a sufficient speed, the operating cost is lower than the existing substitutes, and the problem solving in the transaction is more efficient than the existing contract.

3.3.4 Compatible. Compatible refers to the mutual call between the contracts, the call interface between the contract and the asset is mature, and a similar situation can be conveniently used.

3.3.5 Convenience. Convenience indicates that the contract can be used in most cases simply. The technical barriers are low and no special training or teaching is required.

3.3.6 Observable. Access control needs to guarantee the following two functions: during the execution of the smart contract, it is necessary to monitor the execution status of all contracts to ensure that the loopholes of the contract can be discovered in time; ensure that all contracts can be effectively supervised, after the transaction problem occurs. It has the ability to perform third party verification.

3.4 Design concept of smart contract

At present, Ethereum's smart contract uses a layered design to separate the data contract from the controller contract, so as to avoid the operation command update affecting the data. The perfect combination of smart contracts and token systems can accomplish a range of value transfers. The smart contract is executed uniformly at all nodes, and the determined output is guaranteed according to the determined input and the determined code, and is also the guarantee of the state consistency of all nodes. Finally, smart contracts are called by external triggers, there are no timed calls.

Smart contract design needs to take into account the standard content of the contract content and parameter selection, the contract storage retrieval transmission, and also the operating costs. The implementation of smart contracts requires blockchain as a support, so the size and structure of the blockchain that needs to be considered in its design. Traditional smart contract designs include token, authorization, oracle, randomness, poll, time constraint, termination, math and forkcheck. Under the current blockchain architecture, the operational efficiency, security and decentralization of smart contracts are difficult to be met in the same network. Therefore, the current smart contract design mostly selects the appropriate blockchain according to different scenarios.

3.5 The intelligence of smart contract

The intelligence of smart contracts is the problem we have to focus on. In Section 3, it is mentioned that the smart links have to be intelligent. This chapter shows that through the rational model and design, the intelligence of the contract can be realized.

The current technology is still difficult to completely remove the artificial intelligence, and sometimes encounter unsolvable errors that require the intervention of the maintainer. We can divide the intelligence of smart contracts into the following aspects: intelligent transformation of natural language and machine language; intelligence of contract generation and decomposition; and intelligentization of contract verification.

The conversion between natural language and machine language means that the natural language requirements proposed by the contract participants can be automatically converted into machine language, and the contract can be converted between natural language and machine language.

The intelligent generation of contract means that system can automatically generate machine language contracts according to the requirements of machine language and the basic requirements of computational law; the intelligentization of contract decomposition means that the system language requirements can be changed into multiple independent operation according to the contract split standard. In the next section, the intelligence of contract decomposition will be explained in detail.

4. Modeling

In this part, we divide smart contracts model into execution model, structure model and state model, and propose a classification method of smart contracts. The execution model is

used to explain the operation mechanism of each contract in the smart contract system. To facilitate computer execution of contracts, a method of dividing each contract into several sub-contracts is proposed. We assume that all contracts are error-free, there is no ambiguity, and the contract is unique in the split result. Because different asset types have diverse calling methods and detection methods, different types of contracts control different types of assets to protect the contract's control over assets. The state model of the contract represents the whole life cycle of a smart contract from deployment and execution to final termination. When the contract is in different states, the control content that needs to be accessed is different, and the control state of its external interface is different under different states. It is worth to know that the contract state here refers to the entire contract.

4.1 Execution model

The execution of smart contracts can be divided into four stages: signing, deploying, executing and ending. The execution model of smart contracts is shown in Figure 2. After the user makes a request, the contract text is automatically generated. After the two parties sign the confirmation, the contract needs to be verified by the verification tool to be deployed to the contract executor on the blockchain to start running. If it is a new contract, it is saved in the contract event library as a backup. During the operation of the contract, various digital assets are manipulated according to the contract content. If there is a breach of contract, the penalty is imposed according to the contract, and the credit situation is updated; if there is a dispute, the third party (usually the authority) is involved in the forensics.

4.1.1 Structural model. Analogous to the relationship between the main contract and the subordinate contract existing in the traditional contract, we divide the model design of the

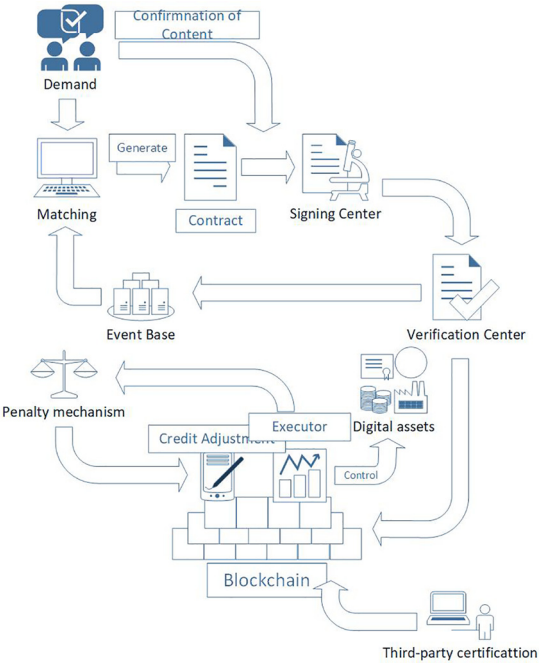

Figure 2.
Execution model of smart contract

smart contract into control contract and based contract, in which the based contract can be divided into logical contract and data contract in the end. The method of contract splitting will be instructed in the next section. The logical contract does not store the state, focus on the running function and running state; the data contract provides the data structure definition; stores the interface for reading and writing data; saves the assets, status, user and other data to the blockchain; and can be from the blockchain. Read the data.

The control contract determines which base contracts are called based on the contractual content signed by both parties. The control contract controls the mutual calling and execution order between different logical contracts; the logical contract acquires the data after accessing the data contract, returns the data to the data contract after processing, and is saved by the data contract; the data contract is responsible for storing and handling the data.

As shown in Figure 3, a logical contract and a data contract form a simple contract function implementation, but data contracts and logical contracts may be one-to-one calls, one-to-many calls, many-to-one calls, and many-to-many calls. Therefore, it is necessary to consider the operating mechanism and conflict handling mechanism under different calling conditions. Data contracts can sometimes need to be read and written by multiple logical logic contracts, so we need to add read and write controls – control which addresses can access the data, in what order.

4.2 The method of contract splitting

To facilitate the execution of the contract by the computer, the contract needs to be split. To facilitate this split, we divide the contract into: a root contract, molecular contracts and atomic contracts. The atomic contract is the smallest functional unit and can perform certain functions.

The actual operation of the smart contract system is the atomic contract, which can be called between each other. A molecular contract is a collection of complete logical operations, data storage, and status updates. The root contract consists of several molecular contracts and atomic contracts, which can complete the content specified in the traditional contract, as shown in Figure 4.

There are two main types of atomic contracts: the contract to complete a complete logical operation, the contract to complete the data operation and the status update. Based on these

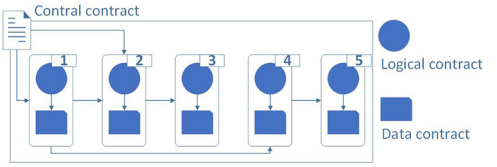

Figure 3.
Structural model of smart contract

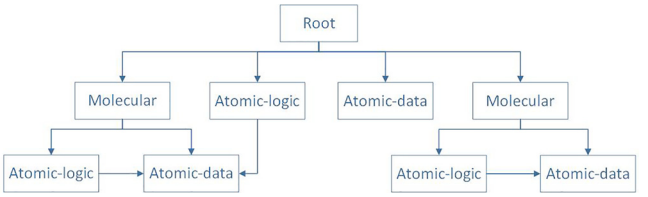

Figure 4.
The contract decomposition tree

two types of atomic contracts, we can complete the various operations required by the contract.

To standardize the interface of atomic contracts, we need to divide the types of atomic contracts. The logical contract is divided into digital contract and actual contract according to whether it needs to be directly connected with the real world. It can also be divided into interactive contract and inspection contract according to whether the external contract or asset is operated. Finally, the logical contract can be divided into digital assets contracts, contracts for manipulating real assets, contracts for detecting the state of digital assets and contracts for detecting the state of real assets, the classification criteria for atomic contracts are shown in Figure 5.

The operation of a smart contract is to realize the processing of assets, to control the different assets, and to set different parameters for the assets. Assets can be divided into digital assets and real assets. Digital assets can be divided into financial service assets and non-financial assets. Real assets can be divided into movable assets and non-mobile assets. The assets controlled by smart contracts can be divided into the above four categories, and the classification method is shown in Figure 1 (Eze *et al.*, 2017).

To standardize the contract, it is necessary to ensure that the results of the splitting of each contract performing the same function are consistent. This requires mathematical methods to prove it, and it is what we need to do in the future.

4.3 State model

According to the possible situations during the contract operation, we divide the contract into five states: deployment, execution, failure, default and cancellation, as shown in Figure 6. Inside, deployment is pre-run state. The system needs to ensure that no deadlock is found in the contract.

4.3.1 Deployment. The contract needs to be verified before deployment, and the post-deployment contract monitors the external conditions through the interface. Once triggered, the contract begins to execute.

4.3.2 Execution. During the execution phase, multiple steps are executed sequentially or in parallel according to the contract content, and there is a default condition or an execution failure, and these operations are performed according to a predetermined script.

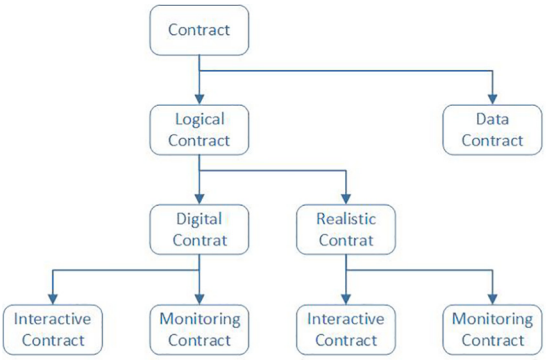

Figure 5.
Classification criteria for atomic contracts

4.3.3 Failure. Failures in the contract may have logical errors, data processing errors, hardware failures, etc. When encountering an error, it needs to be dealt with in time according to the existing error handling mechanism to avoid causing greater losses.

4.3.4 Default. If one party defaults, it may affect the other party to continue to perform the treaty. Therefore, in case of default, it needs to be resolved according to the pre-default situation. If it has not been fulfilled as of the completion date (or the step is seriously delayed, the subsequent steps cannot be completed in time), the defaulting party will need to be punished according to the terms of the breach. To ensure that the breach of contract processing will not be controversial, a more complete default processing logic is needed.

4.3.5. Logout. Once the contract is fulfilled or expires, it needs to be cancelled. Besides, the contract data will be stored in database.

5. Application

The application of smart contract can not only reduce cost and conflict probability, improve efficiency and execution rate, but also change the original production-circulation-consumption mode and transaction structure. Through literature research and factual investigation in factories, we summarize the shortcomings of current milk trading mode, and analyze the changes brought about by the use of intelligent contracts to the current trading structure and trading mode. We also use milk as an example to illustrate how the application of smart contracts can change our life.

5.1 Current schema

As shown in Figure 7(a), the current mode of production is that the manufacturer adjusts the production volume according to the comprehensive transportation cost, the production capacity of each factory and the existing inventory after aggregating and forecasting the demand according to the sales feedback from supermarket counters. The circulation cost of products is mainly from manufacturer's inventory to distributors (mainly supermarkets), but the implicit cost of customers to supermarkets should also be considered. Consumption completion is ultimately marked by the payment of price for the goods purchased by the customer and the acquisition of ownership of the goods.

The main transaction is between manufacturers, supermarkets and customers. The transaction cost between

The manufacturer and the customer is mainly the labor cost and transportation cost of the salesman. Transaction costs between manufacturers and supermarkets are mainly composed of inventory and transportation. The transaction cost between the supermarkets and the customers is mainly inventory cost. In the existing supply mode, the supply is often greater than the demand, resulting in a certain degree of waste of resources. In addition, there are a lot of costs for inventory management and supermarket counter management.

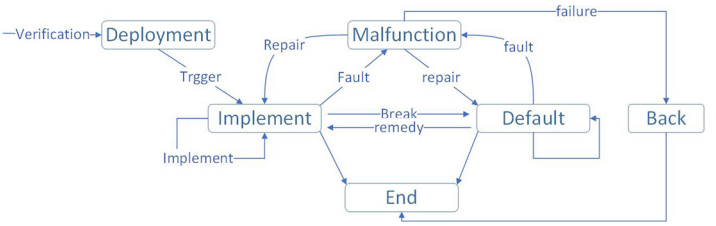

Figure 6.
The running state of contracts

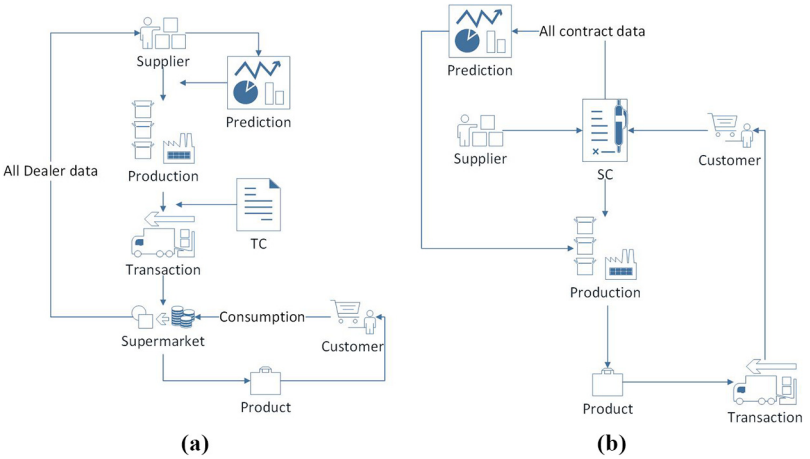

Figure 7.
Transaction structure evolution model

5.2 Schema with smart contract

In Figure 8, the hollow square represents the production node, the hollow triangle represents the sales node, the solid circle represents the customer, the red arrow represents the supply to the customer, and the blue arrow represents the supply to the consumer node. The application of smart contract will *affect the transaction structure*. It can greatly reduce intermediate nodes and improve the efficiency of resource allocation, as shown in Figure 8(a) to (c).

As shown in Figure 7(b), the application of smart contract can also *change the production, transaction and consumption mode of the whole industry*. The whole life cycle of the product is controlled by the contract content. According to the actual demand of customers, all the signed contracts are reorganized into new contracts according to the quantity or area, rational resource allocation, and the contracts control the production, circulation and transaction of products. The application of smart contract in milk industry will reduce the intermediate nodes to a greater extent, and bring about the transfer of resources from storage to logistics.

5.3 Other changes

At present, there are still some problems in product quality control. Once a customer in a supermarket takes a bag of milk (we assume that this bag of milk is M1 and its warranty date is D1) and does not want it, he puts it on the shelf of nuts. Supermarket employees did

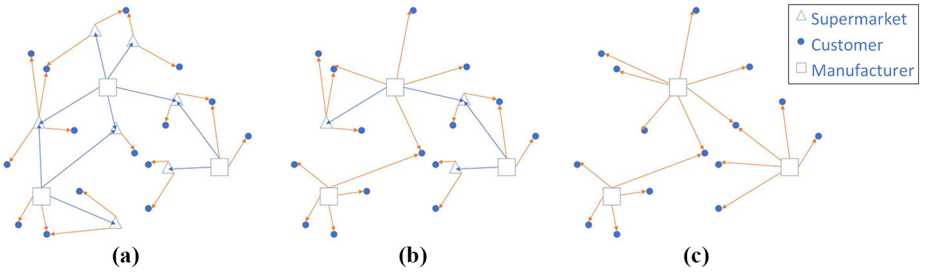

Figure 8.
Production-circulation-consumption evolution model

not return the milk to the exclusive container in time. Another customer finds M1 in D2 (we assume M1 was taken out of supermarket in D2, and D2 > D1) and he buys it. Food safety accidents occur, which is difficult to investigate. Whose responsibility is it?

At this stage, the application of smart contract can ensure that when the goods are scanned at the supermarket checkout office, warnings can be issued once the products expire.

6. Conclusion

Through the comparative study of existing smart contracts and their platforms, we summarize the shortcomings of smart contracts, and prospects its future development and research directions. This paper makes a comprehensive summary of the shortcomings of traditional contracts and existing smart contracts, and proposes solutions to these shortcomings. Based on this, the needs of smart contracts are analyzed in combination with reality. This paper presents a smart contract model for computer understanding and execution. According to the nature and needs of smart contracts, smart contract models with more perfect structure is established. The models are mainly divided into several aspects: atomic model, contract structure model, contract state model and contract execution model. These models illustrate the implementation of smart contracts form different aspect.

Starting from the shortcomings of traditional contracts and existing smart contracts, the nature and definition of smart contracts has been further improved. For the convenience of being executed by computer, we proposed a decomposition method of Smart Contract. Finally, we take milk for instance to illustrate how the application of Smart Contract will change our life.

To ensure the proper operation of the contract, formal validation is required before deployment. It can not only greatly reduce the probability of errors, but also reduce the loss caused by errors. To benefit the large-scale application of smart contracts, we need an efficient consensus mechanism. In the future, we will focus on the formal verification and consensus mechanism of smart contracts.

References

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A. and Muralidharan, S. (2018), "Hyperledger fabric: a distributed operating system for permissioned blockchains", in *Proceedings of the Thirteenth EuroSys Conference*, *ACM*, p. 30.

Brandstätt, C., Brunekreeft, G. and Friedrichsen, N. (2011), "Improving investment coordination in electricity networks through smart contracts", Bremen Energy, Working Papers.

Buterin, V. (2014), "Ethereum: a next-generation smart contract and decentralized application platform", available at: www.weusecoins.com/assets/pdf/library/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

Cong, L.W. and He, Z. (2018), *Blockchain Disruption and Smart Contracts*, Social Science Electronic Publishing.

Dickerson, T., Gazzillo, P., Herlihy, M. and Koskinen, E. (2017), "Adding concurrency to smart contracts", *Symposium on Principles of Distributed Computing*. *ACM*, pp. 303-312.

Eze, P., Eziokwu, T. and Okpara, C. (2017), "A triplicate smart contract model using blockchain technology", *Circulation in Computer Science)*, No. 1, pp. 1-10.

Frantz, C.K. and Nowostawski, M. (2016), "From institutions to code: towards automated generation of smart contracts", *IEEE International Workshops on Foundations and Applications of Self* Systems*. *IEEE*.

Gao, H., *et al.* (2016), *Block Chain and New Economy: Digital Money 2.0 Era*, Electronic Industry Press, Beijing.

Idelberger, F., Governatori, G., Riveret, R. and Sartor, G. (2016), *Evaluation of Logic-Based Smart Contracts for Blockchain Systems*, Rule Technologies. Research, Tools, and Applications.

Miller, A. and Bentov, I. (2016), "Zero-collateral lotteries in bitcoin and ethereum", in *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, p. 2566339.

Norta, A. (2016), "Designing a Smart-Contract application layer for transacting decentralized autonomous organizations", *International Conference on Advances in Computing and Data Sciences*. *Springer*, *Singapore*.

Nugent, T., Upton, D. and Cimpoesu, M. (2016), "Improving data transparency in clinical trials using blockchain smart contracts", *F1000Research*, Vol. 5, pp. 2541.

Raskin, M. (2016), *The Law and Legality of smart contracts*, Social Science Electronic Publishing.

Richard, G. (2015), "A simple model for smart contract", available at: https://gendal.me/2015/02/10/a-simple-model-for-smart-contracts/

Ronghui, G. Zhong, S. and Vilhelm, S. (2018), "Certik whitepaper", available at: www.certik.org/

Steve, D. (2018)M, "ATRIX", available at: www.matrix.io/

The State Council (1999), "Contract law of the people's Republic of China", *Chinese Patents and Trademarks*, Vol. 3, pp. 80-103.

Corresponding author
Xiao Yu can be contacted at: yuxiao84@mail.tsinghua.edu.cn