

Formal Modeling of Smart Contract-based Trading System

Woong Sub Park[†], Hyuk Lee, Jin-Young Choi

School of Cybersecurity, Korea University, Seoul, 02841, Republic of Korea

{wsub, fmlab_hlee, narnia}@korea.ac.kr

Abstract— With the development of blockchain technology, the fields of use of smart contracts are diversifying. Blockchain-based smart contracts are suitable in areas where integrity and transparency must be guaranteed with distributed ledger technology as the core. However, once the system is deployed, it cannot be modified, so it is important to ensure that the system works with the requirements and principles of the smart contract at the design stage. Therefore, in this paper, we aim to show that the system is accurate without contradictions/errors through formal verification using UPPAAL, a formal verification tool for the public descending auction system (Dutch Auction).

Keywords— Blockchain, Smart Contract, Formal Specification, Formal Verification, Model Checking

I. INTRODUCTION

The concept of early smart contract was first proposed by Nick Szabo, which is a contract written in a digital form. The smart contract has the basic principles of observability, verifiability, privacy, and enforcement, etc[1]. However, it was difficult to implement the functions of smart contracts with the technology at the time, so only the concept was proposed. But, with the development of blockchain technology, an environment has been created to meet these principles. It is used in various fields where it is safe from untrusted clients, such as e-voting[2], healthcare[3], and supply chain management[4], and that need to be difficult to tamper with[5]. Privacy, one of the principles that were difficult to implement in the early smart contract was solved by the distributed ledger technology of the blockchain. However, if the smart contract is implemented using it distributed ledger technology, there is also a disadvantage that it cannot be modified. Therefore, errors in the smart contract pose a risk of leading to an unexpected results, such as The Dao Attack[6]. To prevent such accidents from occurring, smart contracts must ensure that the system meet intended functional requirements before deployment.

In this paper, we aim at formal modeling of a smart contract-based auction system using model checking tool UPPAAL[7]. First, check whether the system is modeled correctly through simulation. Then, the properties that must be satisfied in the requirements are expressed in temporal logic, and whether they are satisfied in the model is verified through model checking. And we verify the model using TCTL (Timed Computational Tree Logic) and confirm that the

model satisfies the basic principles and functional requirements of the smart contract before deployment.

II. RELATED WORK AND BACKGROUND KNOWLEDGE

Formal methods refer to methods of specifying, developing, and verifying systems based on repair and logic. Formal methods are largely divided into formal specifications and formal verification. Formal specification can be used to mathematically specify system requirements, designs, and behaviors to eliminate ambiguity. Formal verification is to verify the requirements, design, etc. of the system using repair and logic. Formal verification is largely divided into semantic proof of verification for automata-based specifications and syntactic proof of verification for syntactic rules[8].

A. Related Work

Nehai *et al.*[9] used NuSMV[10] to model a Blockchain Energy Market Place(BEMP) application written in solidity, and CTL to verify its properties. NuSMV is a tool that verifies that a model satisfies the verification properties by making the verification target a logical expression and proving that the logical expression cannot be true. Osterland *et al.* converted the solidity of MetaCoin into the PROMELA model and verified it with SPIN[11]. PROMELA is suitable for distributed systems because it can express multiple simultaneous processes to be represented in the model[12]. This PROMELA model can be used to verify the system's specifications and behavior using the SPIN model checking tool. Andrychowicz *et al.*[13] proposed a general contract model framework based on the verified bitcoin contract by looking for errors using the UPPAAL modeling tool.

In developing system, the cost to detect and correct errors increase exponentially as the development phases progress. Therefore, the purpose of this paper is to use the UPPAAL model checking tool to perform specification and verification before deploying smart contracts. It also seeks to find errors and contradictions of the system are found in the early stages of development and corrected and supplemented.

B. Blockchain and Ethereum

Blockchain is a decentralized, distributed ledger technology. Therefore, the agreement of all nodes participating in the blockchain network can ensure integrity and transparency by validating the same data.

Ethereum is a platform for developing a decentralized applications(DApps). Therefore, once deployed on the Ethereum network, the applications always operates as programmed and has reliable characteristics as it cannot be controlled arbitrarily by a specific organization or individual[14]. One of Ethereum's representative services is that programs written in solidity can be implemented as smart contract functions.

C. Smart Contract

Smart Contract is a kind of digital contract written in Ethereum's programming language. The concept of smart contracts existed even before the development of the blockchain technology, but with the advent of blockchain, a distributed ledger technology, it has developed into an environment suitable for using smart contracts. A smart contract fulfills certain contract terms online, and when the conditions are met, the contract is automatically established according to the contents of the smart contract. Therefore, it is possible to make transactions with untrusted counterparts within the blockchain network without an intermediary.

D. Model Checking

Model checking is an automated technique that shows whether the model satisfies the properties for a given system's finite-state model and logical properties as a type of formal verification [15]. As shown in Figure 1, model checking is composed of a model that specifies the system, properties of the system, and a model checker that operates by receiving properties and models. To check the model, finite state-based modeling is performed and specific properties are expressed using temporal logic. Then, it verifies whether the specified model and properties are satisfied with the model using the model checker. Model checker can be fully automated and generate counter-examples through the exhaustive state exploration. Therefore, it is possible to implement a system that satisfies the properties through counter-examples generated by performing model checking.

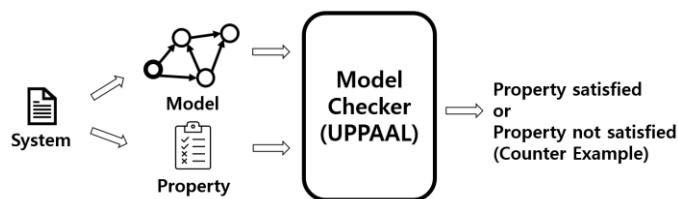


Figure 1. Model Checking

E. UPPAAL

UPPAAL is a model checking tool based on Timed automata. UPPAAL is suitable for real-time system modeling because it uses a data type called a channel and an actual value called a clock [7]. UPPAAL is composed of Editor, Simulator, and Verifier and Editor specifies Timed Automata in template unit. The specified system can be run dynamically by the user directly through the simulator. Verifier can verify the properties of the model that it wants to verify using Timed Computation Tree Logic(TCTL) as shown in Figure 1[16]. As

a result of the properties to be verified through Verifier, there are four types: “be (not) satisfied” and “may (not) be satisfied”.

III. FORMAL MODELING OF AUCTION-BASED TRADING SYSTEM

Smart contracts can be used in various fields. In this paper, we formally model the Dutch auction(open-outcry descending-price auction)system, which is one of the auction-based trading system methods.

A. System Requirements

The functional requirements of the Dutch auction system are as follows.

- 1) Beneficiary can set auction time, initial proposed price, lowest proposed price, deduction amount, and deduction time interval.
- 2) The proposed price will be deducted by the deduction amount after the deduction time interval.
- 3) The auction shall end if:
 - A single bid exists,
 - Pre-set auction end time reached.
- 4) Commodity prices cannot be deducted below the lowest proposed price.
- 5) If the auction is over and a successful bidder exists, the bid price must be paid to beneficiary.
- 6) When the auction begins, the auction must end at the specified end time.

B. Auction-based Trading System

The beneficiary sets the auction time, initial proposed price, lowest proposed price, deduction amount, and deduction time interval and the smart contract. Bidders can check the current proposed price. The initial proposed price is reduced at each deduction time interval specified by the beneficiary. At this time, the price is not deducted below the deduction amount specified by the beneficiary. And if the price that the bidder wants to purchase is greater than or equal to the price of the current proposed price, the bidder bid in the auction. If there are more than two bidders of the same price during the bidding process, the bidder who bids non-deterministically first will be the winner of the auction. Besides, if the auction is not successful, when the auction end time is specified by the beneficiary.

C. Formal Modeling using UPPAAL

Formal modeling is a type of formal specification. Modeling is the most basic step when trying to verify that there are no errors or contradictions in the system through model checking, and if the model is not accurate, formal verification will also result in low accuracy. Modeling was performed using the UPPAAL modeling tool by reflecting the functional requirements in Section III-A.

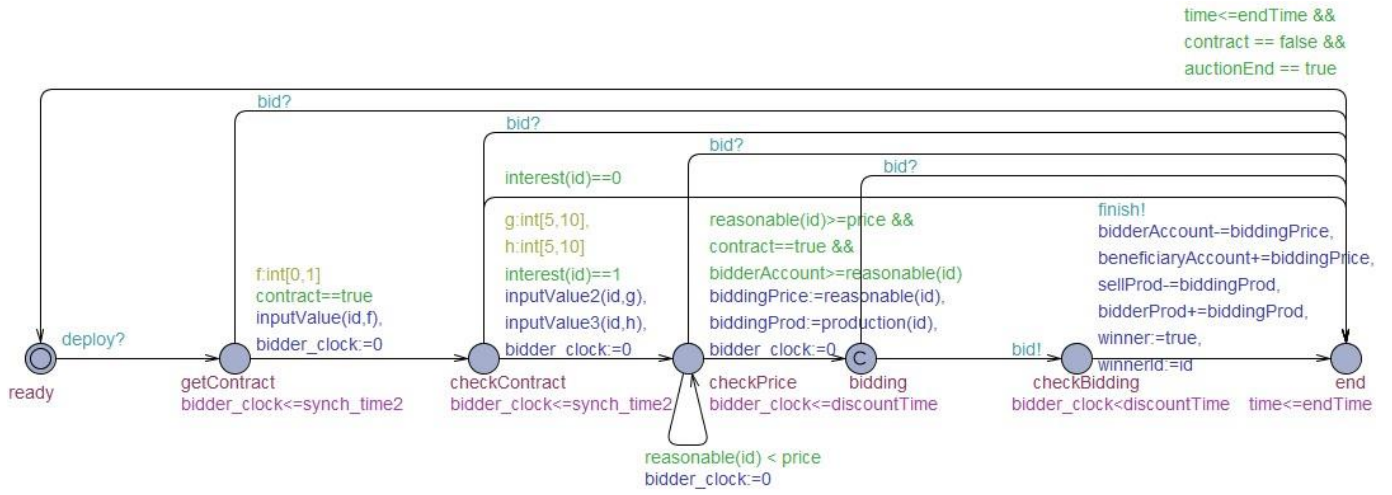


Figure 2. Bidder Template

The transition conditions of ready, register and end status of Beneficiary template are as follows.

1-1) As shown in Figure 4. the beneficiary sets the auction time, initial proposed price, lowest proposed price, deduction amount, and deduction time interval. Then, it is synchronized with the Contract template through the deploy channel and then transitions to the register state.

1-2) In the Register state, when synchronization is made through invalid and finish channels from the Contract template, it transitions to the end state. When the auction ends, it transitions to the initial ready state.

The transition conditions of ready, getContract, checkContract, checkPrice, bidding, checkBidding and end of the Bidder template are as follows.

2-1) When synchronized through the deploy channel of the Beneficiary template, it goes into the getContract state.

2-2) The bidder decides whether to participate or not, and then if the contract has been deployed, the bidder transfers to checkContract state.

2-3) If the bidder has an intention to participate in the auction, the bidder decides the price and quantity bidder wants to purchase, and transitions to the checkPrice state, and if there is no intention to participate, it transitions to the end state.

2-4) If the current proposed price of the commodity is less than or equal to the price bidder wants to purchase, the bidder transitions to bid state. At this time, the bidder must have more than the amount to purchase.

2-5) The bidder is transferred to the checkBidding state and then, synchronized with the Contract template through the bid channel.

2-6) Bidders are synchronized with Contract and Bidder templates through the finish channel. At this time, the contract is converted to the end state after delivering the

price paid by the bidder to the beneficiary and delivering the beneficiary's commodity to the bidder.

2-7) When the contract ends and the auction ends, it transitions to the ready state.

The transition conditions of the state deactivate, active and chContract of the Contract template are as follows.

3-1) It is synchronized through the deploy channel of the Beneficiary template and transitions to the active state.

3-2) If the clock in the Contract template equals the deduction time interval and the current proposed price is above the lowest proposed price, the deduction amount specified by the beneficiary is reduced from the current proposed price of the commodity.

3-3) In the active state, when the current proposed price is equal to the deduction amount and the auction end time is reached, it is synchronized with the Beneficiary template through the invalid channel and the contract is terminated.

3-4) In the active state, when the auction end time comes, it is synchronized with the Bidder template through an invalid channel and the contract is terminated.

3-5) In the active state, if the price, the deduction amount, and deduction time interval is incorrect, it is synchronized with the Beneficiary template through the invalid channel and the contract is terminated.

3-6) When synchronized through the bid channel in the Bidder template, it goes into the chContract state, and when synchronized through the finish channel, that is, when the auction ends and the contract ends, it goes into a deactivate state.

D. Simulate Trading Scenarios

As defined in III-A, modeling was carried out using the UPPAAL model checking tool. It is impossible to check the simulation by entering all input values. Therefore, the simulation proceeded by setting the correct input value and the

incorrect input value. An incorrect input value in this model means an abnormal input value for which an auction is not held. The correct input values were put into the model, and the design was confirmed through simulation.

In the case of setting an input value that does not proceed with an auction, it is as follows.

- 1) The beneficiary has no commodity to sell.
- 2) The bidder does not have the price of the commodity to purchase.
- 3) The beneficiary set the deduction amount, deduction time interval, and commodity price as negative numbers.

In this case, as shown in Figure 2 and Figure 3 the contract was deployed, but it was confirmed through simulation that the contract was terminated because it was an incorrect input value.

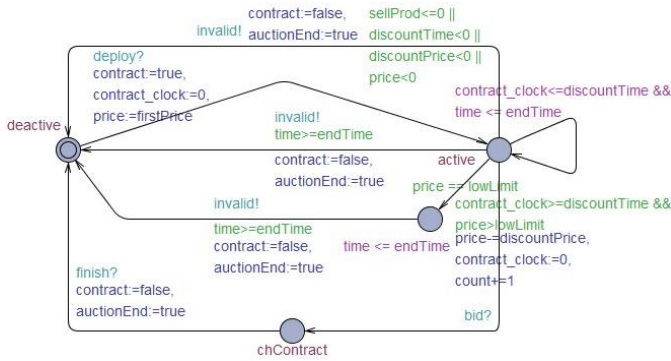


Figure 3. Contract Template

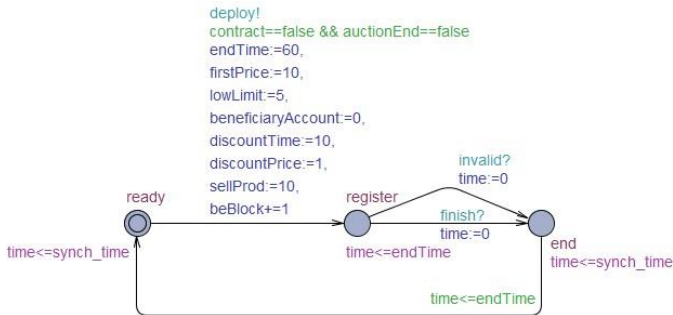


Figure 4. Beneficiary Template

E. Property Expression Using TCTL

TCTL is a mechanism that can express constant boundaries for delays between state transitions to model finite-state systems[17], and which is accepted by UPPAAL. The temporal operator used in UPPAAL is as follows.

- E - Exists a path (“E” in UPPAAL)
- A - For all path (“A” in UPPAAL)
- G - All states in a path (“[]” in UPPAAL)
- F - Some state in a path (“◇” in UPPAAL)

The properties derived from the functional requirements are expressed in TCTL as follows

- **Property 1:** The lowest proposed price must be reduced correctly at the specified time.
 $\rightarrow A[] \text{ discountTime} == \text{Contract.contract_clock} \text{ imply } \text{price} == \text{firstPrice} - (\text{discountPrice} * \text{Contract.count})$
- **Property 2:** Bidding cannot be made by two people at the same time.
 $\rightarrow A[] \text{ forall } (i : \text{id_b}) \text{ forall } (j : \text{id_b}) \text{ Bidder}(i).\text{bidding} \ \&\& \ \text{Bidder}(j).\text{bidding} \text{ imply } i == j$
- **Property 3:** The price of commodity cannot be deducted below the lowest proposed price.
 $\rightarrow A[] \text{ price} \geq \text{lowLimit}$
- **Property 4:** If the auction is over and there is a winner, the correct price must be delivered.
 $\rightarrow A \diamond \text{ winner} == \text{true} \text{ imply } (\text{beneficiaryAccount} + \text{Bidder}(\text{winnerId}).\text{biddingPrice} == \text{beneficiaryAccount} + \text{Bidder}(\text{winnerId}).\text{reasonable}(\text{winnerId}))$
- **Property 5:** When the auction begins, the auction must end at the specified end time.
 $\rightarrow A \diamond \text{ Contract.active} \text{ imply } \text{Contract.deactive}$

The UPPAAL model checker was used to confirm that the model satisfies the above properties derived from the functional requirements. Property 1 states that if the deduction time interval specified by the beneficiary and the clock of the contract template becomes the same, the current proposed price is the same as the price reduced by the deduction amount from the initial proposed price. In this property, the 'count' variable represents the number of times the amount has been decreased, as shown in Figure 3. And when the clock of the contract template becomes equal to the deduction time interval, it is initialized to 0. Property 2 states that if the bidder in the bidding state is bidder(i) and bidder(j), it means that i and j are the same, indicating that it cannot exist in the bidding state at the same time. Verified property 3 is a safety property, and it has been shown that the initial proposed price does not deducted below the lowest proposed price in all states of all routes, and the verified property 4 is a safety property. It means that the bid is delivered to the beneficiary. At this time, in the above property, it was assumed that the beneficiary initial asset is 0. The verified property 5 is the liveness property, indicating that the contract will be terminated someday when the contract is deployed.

IV. CONCLUSIONS

Smart contracts are systems that it is appropriate to specify and verify using UPPAAL model checking tools that can use time properties. A smart contract-based Dutch auction trading system was modeled and verified, and timing constraints were placed on each template and state from occurring in the modeling process. Also, by specifying the time constraint as a range, the price decreases at the correct time. Various scenarios in which the auction ends were supplemented through simulation, and as a result of the verification using TCTL, all properties derived from functional requirements were confirmed to be satisfied.

Smart contracts are based on distributed ledger technology, so once deployed they cannot be modified. Therefore, it is important to find, correct, and supplement system errors in the

early stages of development through formal specification and verification. The smart contract developed through this process can more satisfy functional requirements and principles, so it will be easier to apply and use in various fields.

With future research, we will implement the smart contract function using Solidity from the verified model, and specify and verify the implementation to show that the functional requirements and principles are satisfied.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2018R1A2B6009122).

REFERENCES

- [1] Szabo, Nick. "Formalizing and securing relationships on public networks." First Monday (1997).
- [2] Shahzad, Basit, and Jon Crowcroft. "Trustworthy electronic voting using adjusted blockchain technology." *IEEE Access* 7 (2019): 24477-24488.
- [3] Griggs, Kristen N., et al. "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring." *Journal of medical systems* 42.7 (2018): 130.
- [4] Cruz, Jason Paul, Yuichi Kaji, and Naoto Yanai. "RBAC-SC: Role-based access control using smart contract." *IEEE Access* 6 (2018): 12240-12251.
- [5] Xia, Q. I., et al. "MeDShare: Trust-less medical data sharing among cloud service providers via blockchain." *IEEE Access* 5 (2017): 14757-14767.
- [6] WIRED. 2016. A \$50 MILLION HACK JUST SHOWED THAT THE DAO WAS ALL TOO HUMAN. (2016). <https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/>
- [7] Larsen, Kim G., Paul Pettersson, and Wang Yi. "UPPAAL in a nutshell." *International journal on software tools for technology transfer* 1.1-2 (1997): 134-152.
- [8] Clarke, Edmund M., and Jeannette M. Wing. "Formal methods: State of the art and future directions." *ACM Computing Surveys (CSUR)* 28.4 (1996): 626-643.
- [9] Nehai, Zeinab, Pierre-Yves Piriou, and Frederic Daumas. "Model-checking of smart contracts." 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2018.
- [10] Cimatti, Alessandro, et al. "Nusmv 2: An opensource tool for symbolic model checking." *International Conference on Computer Aided Verification*. Springer, Berlin, Heidelberg, 2002.
- [11] Holzmann, Gerard J. "The model checker SPIN." *IEEE Transactions on software engineering* 23.5 (1997): 279-295.
- [12] Osterland, Thomas, and Thomas Rose. "Model checking smart contracts for Ethereum." *Pervasive and Mobile Computing* 63 (2020): 101129.
- [13] Andrychowicz, Marcin, et al. "Modeling bitcoin contracts by timed automata." *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, Cham, 2014.
- [14] [Ethereum.org/Ethereum.org/ko/what-is-ethereum/](https://ethereum.org/ko/what-is-ethereum/)
- [15] Clarke, Edmund M., and E. Allen Emerson. "Design and synthesis of synchronization skeletons using branching time temporal logic." *Workshop on Logic of Programs*. Springer, Berlin, Heidelberg, 1981.
- [16] Bouyer, Patricia. "Model-checking timed temporal logics." *Electronic notes in theoretical computer science* 231 (2009): 323-341.
- [17] Alur, Rajeev, Costas Courcoubetis, and David Dill. "Model-checking for real-time systems." [1990] *Proceedings. Fifth Annual IEEE Symposium on Logic in Computer Science*. IEEE, 1990.



Woong Sub Park received B.S. degree from the School of Computer Engineering, Dongguk University, Korea in 2020. He is currently a M.S. candidate in School of Cybersecurity, Korea University. His current research interests are in formal modeling and secure software engineering.



Hyuk Lee received the B.S. degree from the University of Technology Sydney, Sydney, Australia, in 2006, and M.S. degree in 2009 and the Ph.D. degree in 2019 from the Korea University, Seoul, Korea. He is currently a research professor with the Graduate School of Information Security, Korea University, Seoul, Korea. His current research interests are in formal methods, constraint problem solving, and secure software engineering.



Jin-Young Choi received the B.S. degree from Seoul National University, Seoul, Korea, in 1982, received the M.S. degree from Drexel University, Philadelphia, PA, in 1986, and the Ph.D. degree from the University of Pennsylvania, Philadelphia, in 1993. He is currently a Professor with the Graduate School of Information Security, Korea University, Seoul, Korea. His current research interests are in real-time computing, formal methods, programming languages, process algebras, security, and secure software engineering.