Journal of Engineering and Applied Sciences 13 (Special Issue 11): 8716-8721, 2018
ISSN: 1816-949X

Conditional Formalization of Smart Contract Using Semantic Web Rule Language

[1]Sung Hyun Na, [1]Jung Hyun An, [2]Jae Soo Yang and [3]Young B. Park
[1]Department of Computer Engineering Graduate School,
[2]Department of Electronics and Electrical Engineering,
[3]Department of Software Science, Dankook University, Youngin, South Korea

Abstract: As networking technologies evolve, distributed computer technologies such as the cloud are becoming more important. A blockchain is a kind of distributed computer technology in which all nodes have the same ledger to guarantee data integrity. Within a blockchain, smart contract defines the contexts of various items in the form of program code and acts as business logic. However, studies on existing smart contracts do not focus on formalizing rules themselves or managing conditional according to various situations. In this study, we formalize data related to smart contract and propose a method to apply conditional to various situation. This method models data of smart contract using ontology and applies semantic web rule language to conditional and manages rule sets in blockchain network. In this way interoperability of the smart contract type can be ensured and the relationship between the contexts in each transaction can be inferred by linking the ontology reasoner to the blockchain network. We describe the smart contract applying the semantic web rule language and show how to actually operate within the blockchain network through the program example.

Key words: Blockchain, smart contract, ontology, SWRL, ontology, technologies

INTRODUCTION

As the development of network technology and the importance of large-scale data processing are increasing, a distributed computing environment that is configured to interconnect systems and work together is becoming important. The blockchain provides a solution to this problem which is often referred to as the Byzantine general problem (Swan, 2015). Blockchain is a kind of distributed computing technology. In order to solve the Byzantine general problem, a node generates a transaction using a consensus algorithm and each node has the same ledger as the result of the transaction and guarantees the integrity of the data.

Smart contract is a concept proposed by Nick Szabo and consists of contract conditions or rule bases designed to execute business logic within a blockchain (Alliance, 2016). These smart contracts are included in the blockchain as codes, so that, various types of contracts such as financial transactions and real estate contracts are concluded and executed according to prescribed rules. Ethereum, one of the blockchain based platforms can create arbitrary rules, transaction types, etc. for crypto currency ownership by creating a Decentralized Application (DAPP) using Solidity.

Smart contracts are undergoing various studies due to their characteristics that can provide not only crypto currency but also various services. The following research suggests an approach to use smart contracts with increased security. Hawk (Kosba *et al*., 2016) is a system that prevents exposure of privacy contents of user data included in a transaction. It allows users to create smart contracts in an intuitive way without having to implement encryption and the compiler provides the ability to automate the creation of encryption protocols. The following research suggests an approach to domain concept integration by introducing ontology into a blockchain. An ontology-based supply-chain blockchain (Kim and Laskowski, 2018) introduced a blockchain modeled as an ontology in order to guarantee the integrity of the supply chain source and integrated the concepts of the products and defined the components as ontology models. Ontology based smart contract, Kruijff and Weigand (2017) defined the blockchain data as an ontology class and applied it to the semantic web environment in order to reduce the integration and conceptual ambiguity of the smart contract. Existing research focuses on the role of smart contracts and data modeling in the blockchain but it does not focus on how to formalize the conditional itself and manage it according to various states. Therefore, in this study, we propose a smart contract that generates ontology model for business logic and defines the condition of smart contract using Semantic Web Rule Language (SWRL). SWRL

Corresponding Author: Young B. Park, Department of Software Science, Dankook University, Youngin, South Korea

(Horrocks *et al*., 2004) is an OWL (Ontology Web Language) rule markup language that can be used to express logical rules within an ontology. SWRL is defined as a part corresponding to the condition and it is modeled in a format that can interact with other media and the condition can be managed in units of rules according to the state. In addition, we can infer the relationship of smart contracts by associating rule inference framework with blockchain network.

Literature review

Rule based system: Frameworks such as EARTH (Singh and Chana, 2016) and ANGY (Stansfield, 1986) support decision making of expert systems based on rules. EARTH is designed to reduce data center energy costs due to network resource consumption in a cloud computing environment. We consider the energy consumption as a QoS parameter and perform resource management scheduling autonomously through fuzzy logic. ANGY is a rule-based expert system in the field of medical image processing that identifies blood vessels in digital angiography of the chest, ignoring structures that can occur in noise, background contrast changes and meaningless anatomical details. The expert system at ANGY decides on domain knowledge as a rule base. These frameworks show that within the expert system, rules can be used to determine ambiguity about the context.

SWRL (Semantic Web Rule Language): The following studies show approaches for introducing SWRL rules in a distributed context environment. Context inferring in the smart home (Ricquebourg *et al*., 2007) describe context-aware systems based on a service-oriented architecture dedicated to smart home. The approach in the study proposes to build a smart environment that provides context-aware services in particular to provide services to user's needs at the smart home service-oriented architecture layer. We constructed a context model based on OWL ontology and implemented the system through SWRL based inference layer. Reasoning about resources and hierarchical tasks, Eleninus *et al*. (2009) suggests an approach to ensure the same understanding and interoperability of context information in distributed systems in a military training environment. Use ontologies to model systems and requirements and solve problems using automated reasoning. OWL ontology was layered and SWRL was used to ensure organic reasoning between the layers. In conclusion, contexts are conceptually integrated for the understanding of multiple systems of distributed environmental information and they are refined to common rules to ensure interoperability between systems.

Data formatting of smart contract using ontology: This chapter introduces the process of blockchain network used in this study and introduces the method of applying SWRL to smart contract. The proposed method requires two systems. First, a blockchain network consists of a validator that verifies a transaction, a rule that corresponds to business logic and a ledger that stores integrity data. The second is the client program. The user requests the transaction through the client, the blockchain network receives the request, checks the SWRL rule and stores the contents. It then returns the results of the transaction to the client. If a rule change or an upper rule change occurs according to an administrator's request in the rule check section it is detected and the rule is changed. The process is shown in Fig. 1. The system consists of the Java programming language and uses one of the blockchain open source hyperledger fabric (Cachin, 2016).

In Fig. 1, the transaction processing requested by the user is detailed as follows. The sequence consists of a transaction request, a transaction confirmation, a smart contract action and a transaction completion. First, the user enters his requirements through the client. The requirement is changed in transaction form and sent to validator. When the validator receives the request it uses the consensus algorithm to perform validation. This is called the transaction verification step. When the transaction is verified it is the smart contract operation phase. smart contract condition is the business logic defined by SWRL. Smart contracts are defined as ontology models and rules and handle actions according to transaction type. After processing the action, the data is stored in the ledger. When the data is saved, the process proceeds to the transaction completion step. The transaction completion step sends the transaction result to the client.

In this study, we propose a smart contract applying ontology and SWRL to formalize the rules of smart contracts in the process of transactions and manage them according to the state. Smart contract generates many kinds of domain knowledge according to the business model to be applied and the domain knowledge must share a common understanding of the information structure of a person or software. Based on the ontology development guide (Noy and McGuinness, 2001) we define the ontology model using the items necessary for smart contract to operate.

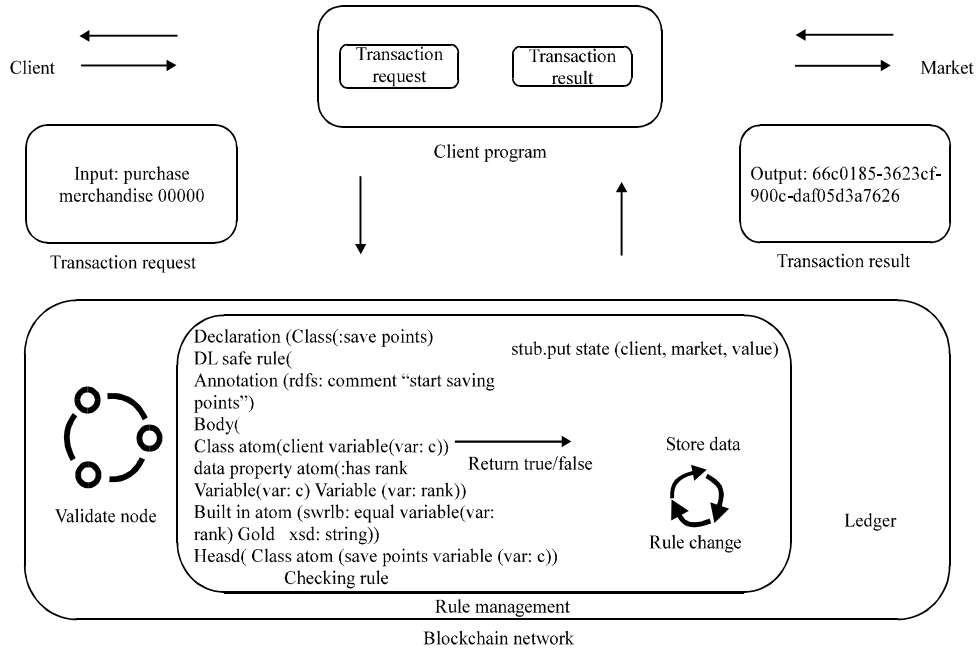

Fig. 1: Blockchain service process

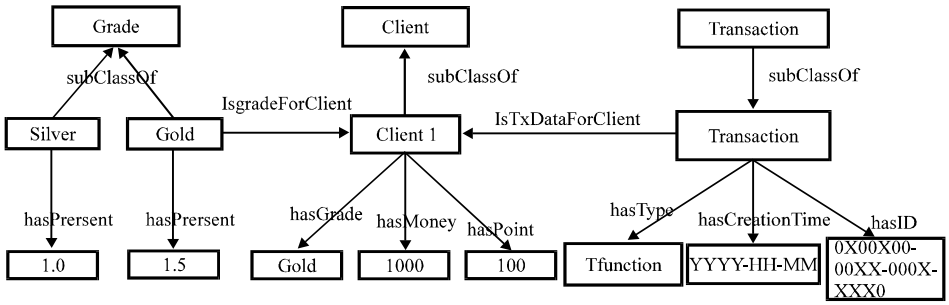

Fig. 2: Ontology model graph

First, we applied the scenario of point accumulation to apply SWRL as an example as shown in Fig. 2. In this scenario when a user purchases an item, the user accumulates a certain percentage of the amount of the item in points according to the member. The ontology expresses the expression that the data belongs to the class as dataproperty and uses objectproperty to express the relation of the class or data. The generated graph is divided into client, transaction and grade classes. In the client class, dataproperty has a hasgrade to indicate the membership level, a hasmoney to indicate the amount and a haspoint to indicate like mileage. Next, the transaction class is created as a result of the user's transaction request. The corresponding transaction class will have an objectproperty for each of the requested users.

Express the logical expression based on the defined ontology model. If the transaction type requested by X is earnpoint, check whether X has a relationship to the point accumulation and whether the property X corresponds to the membership level is equal to the membership level. If the condition is satisfied, points corresponding to the number of times the point accumulation rate P% for the membership level are multiplied by the item purchase amount M are accumulated. The contents are as shown in 1:

$$\begin{aligned} &\text{isEarnPoint}(?X):\text{hasEarnPoint}(?X)\Pi \\ &\text{hasGrade}(?X)\ni \text{swrlb:equal}(\text{Membership level}) \\ &\rightarrow \text{EarnPoint}(P\%*M,?X) \end{aligned} \quad (1)$$

We define the condition of smart contract as ontology model and logical expression and introduce ontology that can formalize domain knowledge and define SWRL rule that operates in actual code by defining rule based operation.

```
Declaration(Class(:EarnPointForGoldMember)
DLSafeRule(
  Annotation(rdfs:comment "Start saving point.")
  Body(
    ClassAtom(:Client Variable(var:c))
    DataPropertyAtom(:hasGrade Variable(var:c) Variable(var:grade))
    BuiltInAtom(swrlb:equal Variable(var: grade) "Gold"^^xsd:string)
  )
  Head(
    ClassAtom(:SavePoints Variable(var:c))
  )
)
```

Fig. 3: Point accumulation SWRL rule expression

```
RuleManager ruleManager = new RuleManager();
ruleManager.loadOntology();

if(ruleManager.reasoningRule(Client, "EarnPointForGoldMember")){
  pState = "EarnPointGold";
}else if(ruleManager.reasoningRule(Client, "EarnPointForSilverMember")){
  pState = "EarnPointSilver";
}else{
  pState = "EarnPointNormal";
}
pAction(State);
```

Fig. 4: Action code for each state according to rule checking

```
{"jsonrpc":"2.0","result":{"status":"OK","message":"61c841d9-13c2-4a5f-8780-0dc30d18cdee"},"id":1}
{"jsonrpc":"2.0","result":{"status":"OK","message":"{"Id":"Client1","Grade":"Gold","Money":10000,"Point":100}"},"id":2}
{"jsonrpc":"2.0","result":{"status":"OK","message":"66c01c85-3b23-44cf-900c-daf05d3a7626"},"id":3}
{"jsonrpc":"2.0","result":{"status":"OK","message":"{"Id":"Client1","Grade":"Gold","Money":5000,"Point":175}"},"id":4}
```

Fig. 5: Client program result console

Implementation of conditional formatting of smart contract using semantic web rule language: The SWRL rule to be used in the smart contract code is a functional syntax and is implemented in the ontology rule set using the ontology model and the logical expression defined above. The scenarios in the smart contract using the SWRL rule are as follows. Configure client 1's dataproperty as shown in Fig. 2. The membership level is gold, 10000 money and 100 points. Client 1's transaction request assumes a state scenario in which the client program requests point accumulation on a blockchain network after purchasing an item of money 5000. The scenario is run through a rule that checks the conditions of the gold membership level in the created ruleset. SWRL in Fig. 3 is defined based on the logical expression 1 and it must satisfy the following factors. The client class requesting point accumulation has a dataproperty of hasgrade corresponding to the membership level and the data is a rule for checking whether the string is the same as gold which is one of the membership levels corresponding to membership level of the logical expression. If the rule is true, the action that returns 75 points and earns 75 points is executed.

According to the SWRL rule shown in Fig. 3, the rule is checked in the actual code and the action is performed according to the rule. An ontology that defines rules through rulemanager is loaded by loadontology method. Then we do rule inference using reasoningrule. The current code is divided into gold, silver and normal to deduce the rule. The p state value is changed according to the result of rule inference and an action appropriate to the current state is executed through the p action method. The Java code that operates according to the SWRL rule is shown in Fig. 4.

Based on this, we apply the SWRL rule and code to smart contract and execute the program to show the actual operation screen. Initialize client1's dataproperty based on Fig. 2. One transaction occurs due to initialization and T×ID and client1 information are output in Json format. After initialization, client 1 generates one transaction

```
2018-05-10 14:51:53,569 INFO  (AbstractChaincode:38) Function_Purchase Merchandise() run
2018-05-10 14:51:53,569 INFO  (Chaincode:72) In Purchase Merchandise
Client1 >> Purchase Merchandise
Load Ontology
<http://com.github.aites/ontologies/SWRLSmartContract.owl#Client1>
Client1 >> EarnPointGold
End
2018-05-10 14:51:54,258 INFO  (Chaincode:159) Result >> Client1: {"Id":"Client1","Grade":"Gold","Money":5000,"Point":175}}
2018-05-10 14:51:54,258 INFO  (Chaincode:177) Purchase Merchandise complete
```

Fig. 6: Smart contract program result console

```
Declaration(NamedIndividual(:Client1))
ClassAssertion(:Client :Client1)
DataPropertyAssertion(:hasGrade :Client1 "Gold"^^xsd:string)
DataPropertyAssertion(:hasMoney :Client1 "5000"^^xsd:integer)
DataPropertyAssertion(:hasPoint :Client1 "175"^^xsd:integer)
```

Fig. 7: Client1 ontology data created as a result of the transaction

```
Declaration(NamedIndividual(:Transaction1))
ClassAssertion(:Transaction :Transaction1)
DataPropertyAssertion(:hasTransactionType : Transaction1 "PurchaseMerchandise"^^xsd:string)
DataPropertyAssertion(:hasTransactionCreationTime : Transaction1 "2018-05-10-
14:51:53,569"^^xsd:string)
DataPropertyAssertion(:hasTransactionID : Transaction1 " 66c01c85-3b23-44cf-900c-
daf05d3a7626"^^xsd:string)
ObjectPropertyAssertion(:isTxDataForClient : Transaction1 :Client1)
```

Fig. 8: Created ontology transaction data

requesting purchase of a specific item at a price of 5000 for purchase. The blockchain network that received the request verifies it according to the purchase request process and accumulates points according to the rules. Fig. 5 shows the result window according to the purchase request. After purchasing the item, 5000 moneys have been reduced and 75 points have been added.

Smart contract checks if it is suitable for SWRL when it receives a transaction request. Read the ontology and check whether the transaction is suitable for the rule through the reasoner. Figure 6 shows the operation window of SWRL rule and code contract based on Fig. 4. It is a program result window in which purchases are made according to the purchase request of client 1 and points are accumulated according to SWRL.

When the smart contract completes the rule check it records a record in the blockchain. In addition, the current state of client1 is recorded in order to infer the transaction information from the reasoner as shown in Fig. 7.

Information about the transaction as well as information about the user is also recorded in the ontology. The transaction information generated according to the user's request is shown in Fig. 8. This confirms the relationship between client1 and the corresponding transaction.

We implemented smart contract using SWRL rule using blockchain open source. Applying ontology and SWRL to formalize the data resulting from transactions and formalize the rule format of smart contract. As a result, we confirmed that the smart contract defined in the SWRL rule through the above method can automate the business logic processing of transactions in the blockchain.

CONCLUSION

In this study, we propose a smart contract using ontology and SWRL. We define the condition of smart contract as SWRL rule and show that smart contract works in block chain by linking rule inference framework with blockchain network. The SWRL rule format is used to formalize the format of the smart contract and it is easy to manage it with the ruleset. In addition it is possible to deduce a related relation through a reasoning framework which is a rule inference framework. In the current phase, only smart contracts which are business logic operating within a blockchain are defined as SWRL. However, it is possible to secure the generality of the ontology for the blockchain through the relationship between the blockchain structure managing the domain data that is the basis of the smart contract and the ontology model.

RECOMMENDATIONS

In future research, we will study blockchain ontology automatically from metadata of blockchain and user-friendly generation of SWRL for convenient creation of conditions in blockchain environment.

ACKNOWELDGEMENTS

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the "Employment Contract based Master's Degree Program for Information Security" supervised by the KISA(Korea Internet and Security Agency) (H2101-18-1001).

This research was supported by The Leading Human Resource Training Program of Regional Neo industry through the National Research Foundation of Korea(NRF) funded by the Ministry of Science and ICT (No.NRF-2016H1D5A1909989).

This research was supported by the MSIT (Ministry of Science, ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2017-0-01628) supervised by the IITP (Institute for Information and communications Technology Promotion).

REFERENCES

Alliance, S.C., 2016. Smart contracts: 12 use cases for business and beyond. Chamber Digital Commerce, USA. http://digitalchamber.org/assets/smart-contracts-12-use-cases-for-business-and-b%20eyond.pdf

Cachin, C., 2016. Architecture of the hyperledger blockchain fabric. Proceedings of the Workshop on Distributed Cryptocurrencies and Consensus Ledgers, July 25-29, 2016, PODC, Chicago, Illinois, USA., pp: 1-4.

Elenius, D., D. Martin, R. Ford and G. Denker, 2009. Reasoning about resources and hierarchical tasks using OWL and SWRL. Proceedings of the International Conference on Semantic Web, October 25-29, 2009, Springer, Berlin, Germany, ISBN:978-3-642-04929-3, pp: 795-810.

Horrocks, I., P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

Kim, H.M. and M. Laskowski, 2018. Toward an ontology-driven blockchain design for supply-chain provenance. Intell. Syst. Accounting Finance Manage., 25: 18-27.

Kosba, A., A. Miller, E. Shi, Z. Wen and C. Papamanthou, 2016. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. Proceedings of the 2016 IEEE International Symposium on Security and Privacy (SP), May 22-26, 2016, IEEE, San Jose, California, USA., ISBN:978-1-5090-0825-4, pp: 839-858.

Kruijff, D.J. and H. Weigand, 2017. Ontologies for commitment-based smart contracts. Proceedings OTM Confederated International Conferences on the Move to Meaningful Internet Systems, October 23-27, 2017, Springer, Berlin, Germany, ISBN:978-3-319-69458-0, pp: 383-398.

Noy, N.F. and D.L. McGuinness, 2001. Ontology development 101: A guide to creating your first ontology. J. Ontol., 17: 1-25.

Ricquebourg, V., D. Durand, D. Menga, B. Marhic and L. Delahoche *et al.*, 2007. Context inferring in the smart home: An SWRL approach. Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops AINAW'07. Vol. 2, May 21-23, 2007, IEEE, Niagara Falls, Canada, pp: 290-295.

Singh, S. and I. Chana, 2016. EARTH: Energy-aware autonomic resource scheduling in cloud computing. J. Intell. Fuzzy Syst., 30: 1581-1600.

Stansfield, S.A., 1986. ANGY: A rule-based expert system for automatic segmentation of coronary vessels from digital subtracted angiograms. IEEE. Trans. Pattern Anal. Mach. Intell., 8: 188-199.

Swan, M., 2015. Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc., Sebastopol, California, USA., ISBN:978-1-491-92049-7, Pages: 128.