Ontology Aided Smart Contract Execution for Unexpected Situations

Farhad Mohsin[1], Xingjian Zhao[1], Zhuo Hong[1], Geeth de Mel[2], Lirong Xia[1], and Oshani Seneviratne[1]

[1] Rensselaer Polytechnic Institute, Troy NY 12180, USA
[2] IBM Research, Hursley Park, Hursley SO212JN, UK

Abstract. We describe an oracle for smart contracts for strengthening the functionality of their execution, thus making them amenable to any future changes that may be critical for sustained use. The oracle is supported by an ontology—specifically, constraints for transactions can be represented by semantic rules, and as such, this ontology-based oracle can help resolve *break glass in case of emergency* type scenarios that require going beyond pre-defined rules in the smart contracts that are already deployed in the distributed ledger platform. The issue of resolving an unexpected situation that was not explicitly considered in the code can be thought of as dynamic changes to attributes of assets and participants involved in a smart contract, which is equivalent to changing rules for a transaction. One way to implement such changes could be by having users vote on different proposals to change the rule. In our case, such a change can be invoked using the ontology and contextual information associated with the smart contract itself. The ontology also helps constrain the voting system itself, imposing a form of access control to allow only valid changes in rules.

Keywords: Blockchain · Smart Contracts · Ontology · Semantic Rules · Unexpected Situations.

1 Introduction

A decentralized application (DApp) does not require an arbitrator due to the implicit trust properties guaranteed by the immutable nature of the distributed ledger framework. Typically, DApps employ smart contracts to execute a particular set of rules in a trusted way. However, smart contracts are usually immutable, and thus requires prior consideration for different unexpected situations that may require change in rules. For example, in a decentralized course selection system spanning multiple schools, there may be specific rules for student enrollment in courses. A professor might employ a smart contract to reduce the class size for the Summer semester, but there could be a capacity of students

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

needing to finish the course during the Summer since they are to start a work placement in the Fall. Therefore, in the absence of a central authority, smarter contracts with flexible structure are needed to manage unexpected situations.

Liu *et al.* [1] suggest smart contracts with a voting mechanism for users to vote to resolve such situations. However, the approach could only deal with a subset of situations by changing some asset attributes related to the transaction that led to the situation. While voting easily gives a valid decentralized way to select new attribute values, a more generalized solution is required to resolve more complex situations.

We propose an ontology-based oracle, where transaction constraints are defined by semantic rules. Since all participant and asset classes have structured—and associated semantic definitions, we can execute complex rules and also modify the rules when necessary. The main contribution of this work is a solution where unexpected situations of complex nature can still be resolved by voting. Aggregating user votes on a new set of proposed rules, we can select a new consistent rule that may resolve the situation.

2 Related Work

There have been attempts to define a formal ontology for blockchain and smart contracts [3, 4]; others have attempted ontology-driven design of blockchain for specific use cases such as provenance tracking [5]. Our work with ontologies is similar to the method for auto-generating smart contracts from domain-specific ontology models and semantic rules by Choudhury *et al.* [2]. In contrast to the aforementioned work, we propose a dynamic ontology-aided oracle, that extends the solution in [1] to adapt smart contracts when faced with unexpected situations.

3 Preliminaries

3.1 Ontologies and Semantic Rules

An ontology contains definitions of concepts, properties, and relationships among them for a particular domain—or system; an ontology also provides a common vocabulary among stakeholders, thus enabling data share and reuse. By incorporating formal rules to such models (e.g., Semantic Web Rule Language (SWRL) [6]) we can increase the model expressivity. Smart contracts essentially enable DApps on blockchains to work based on pre-defined rules, and our proposal is to make the rules be executed based on an ontology.

3.2 Oracles in Blockchain

A smart contract executes completely deterministic transactions and then update the blockchain in an according way. Thus, communicating with an external system becomes problematic as it is not possible to verify all external data in a

deterministic way. Oracles solve this problem by providing a trusted system for information transfer. While an oracle may seem like a centralized component, a mutable ontology-based oracle for holding transaction rules actually aids our solution for decentralized handling of unexpected situations.

4 Ontology for Smart Contract Execution

Though smart contracts are useful in implementing decision rules in decentralized applications, it might be desirable to some to keep intricate and complex logic off-chain for ease of design[7]. Moreover, smart contracts are immutable, which also makes it difficult to implement changes in logic, which can sometimes be expected. Motivated by the barrier to change the logic in smart contracts, we instead take the approach of implementing such logic off-chain on an Oracle, building on a domain-specific ontology.

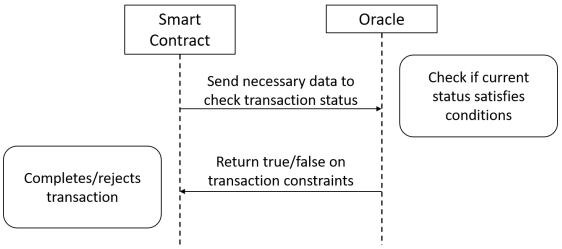

Fig. 1. Executing transaction aided by ontology

As depicted by Figure 1, the main idea is as follows: the blockchain itself will act as a verifiable data structure, but most of the logic for each transaction will be performed off-chain in this model. This is achieved by implementing an oracle with a domain-specific ontology which holds all relations, and the rules of transactions with respect to the semantic concepts found in the said ontology. Each time a transaction is attempted by a user or a smart contract on the network, a query will be made to the oracle about present constraints for the particular transaction. The oracle will be provided with information about the participants and assets involved in a given transaction, and an instance will be created. The semantic rule defined for that transaction will then verify whether this transaction will be completed or not.

4.1 Example: Decentralized Course Selection Problem

We demonstrate our model with the example case of a decentralized course selection system. A registrar's office usually acts as a governing body for the case of students' course selection. But in a decentralized system, there would be no

such system. If we implement a smart contract for this, conditions like prerequisites for a student joining a course or capacity for a course may be defined by semantic rules. Figure 2 shows a section of the ontology on a decentralized course selection (dcs) namespace. A basic transaction for the smart contract is *AddCourse*, which a student uses to enroll in a course, if they fulfill necessary prerequisites. Now, we can have a SWRL rule like the following to define this

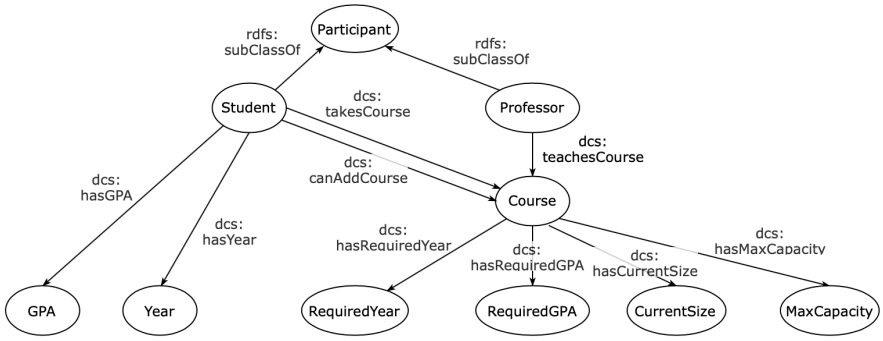

Fig. 2. Decentralized Course Selection Ontology

condition.
$Student(?s) \wedge hasGPA(?s, ?g) \wedge hasYear(?s, ?y) \wedge Course(?c) \wedge hasRequiredGPA(?c, ?rg) \wedge$
$hasRequiredYear(?c, ?ry) \wedge hasMaxCapacity(?c, ?mc) \wedge hasCurrentSize(?c, ?curr) \wedge$
$swrlb : greaterThanOrEqual(?g, ?rg) \wedge swrlb : greaterThanOrEqual(?y, ?ry) \wedge$
$swrlb : lesserThan(?curr, ?mc) \rightarrow canAddCourse(?s, ?c)$

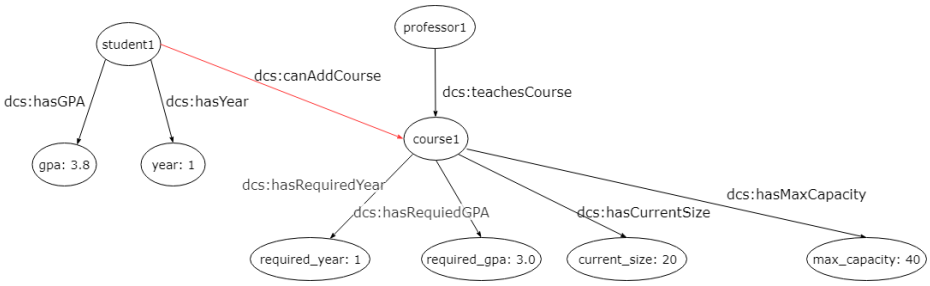

Fig. 3. Examples of classes as instances in the decentralized course selection ontology

When trying to execute an addCourse transaction, the smart contract would query to the oracle about whether the conditions meet. The oracle in turn queries the necessary data from the blockchain and receive data in triples. For the instance in Figure 3, the oracle would receive triples: $student1 : hasGPA : 3.8$,

$student1$: $hasYear$: 1, $course1$: $hasRequiredGPA$: 3.0 etc. With these triples, the query can be evaluated and a True/False value returned to the smart contract for the AddCourse transaction.

In a case when a student does not have sufficient GPA, the professor may consider their grade point in some specific course and modify the rules with an alternate prerequisite. This could have been unforeseen while writing the contract. In our solution, this can be resolved by updating the SWRL rule in the oracle. Since changing the condition for a transaction shall impact other participants than just the one professor and one student, any such change should be made in a fair manner. To ensure this, we implement a voting system to update the rules.

4.2 Voting Mechanism

We give a sketch of the voting mechanism here, Figure 4 depicting the basic flow. The voting system itself would be explicitly guided by the ontology as it would not be tractable to allow all users to vote on all proposal or make all attributes changeable by voting. A form of access control is implemented on the voting system, again utilizing the semantic rules with respect to the ontology so that concerns like *who could invoke a startVote transaction*, *how proposals are gathered* or *how proposals are aggregated to a single desirable outcome* can be addressed properly. The voting mechanism may initiate transactions that request new proposals and votes from users.

5 Implementation Challenges and Future Work

We present specific implementation considerations, drawing examples from the course selection scenario when necessary.

- The ontology will help maintain the extent to which rules and attributes are changeable. For example, while attributes such as *course.MaxCapacity* may be changeable, *student.GPA* cannot be changed, overriding proper rules for updating student grades, by voting.
- For privacy concerns, the oracle would receive data necessary for forming instances for each transaction and never store a complete knowledge graph.
- The oracle consists of both an ontology and a reasoner that helps respond to queries. Updates on the rules should only occur from the smart contract, each transfer of data will also need to ensure that the oracle has not been externally tampered, using a mechanism like change signatures.

Having some rules off-chain may pose a modicum of new security threats that we would like to investigate. Also, we understand that the implementation may also lead to growth in transaction completion time, so a complexity analysis is also due. Another point of interest would be different ways to aggregate propose rules based on different voting rules.

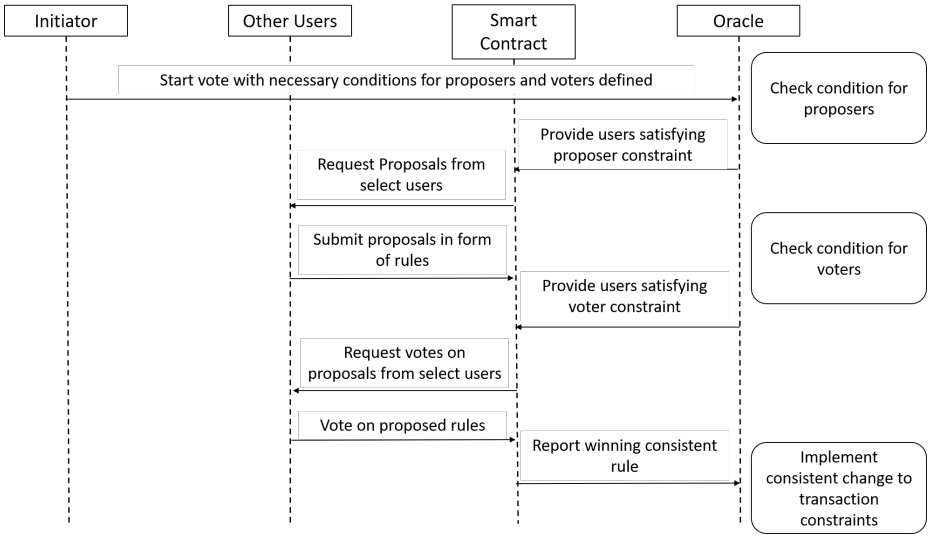

Fig. 4. Implementing the voting mechanism to update rules of a transaction

6 Conclusion

Our proposal of an ontology-based oracle not only makes execution of complex logic in smart contracts easier, but coupled with the voting system, also allows for a way to update the contracts. Empowered with the ontology, we could make use of the data that from update requests for the rules. The voting data may be leveraged towards learning preferences of voters, and proper utilization of this may lead towards creating smarter contracts able to auto-generate proposals for voting or have intelligent proxies for voters to speed up the updating process.

References

1. Liu S., Mohsin F., Xia L., Seneviratne O.: Strengthening Smart Contracts to Handle Unexpected Situations. In: 2019 Proceedings of IEEE International Conference on Decentralized Applications and Infrastructures, CA, USA (2019)
2. Choudhury O., Rudolph N., Sylla I., Fairoza N., Das A.: Auto-Generation of Smart Contracts from Domain-Specific Ontologies and Semantic Rules. In: 2018 Proceedings of IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data, Halifax, NS, Canada, 2018, pp. 963-970.
3. de Kruijff J., Weigand H.: Understanding the Blockchain Using Enterprise Ontology. In: Dubois E., Pohl K. (eds) Advanced Information Systems Engineering. CAiSE 2017. Lecture Notes in Computer Science, vol 10253. Springer, Cham
4. de Kruijff J., Weigand H.: Ontologies for Commitment-Based Smart Contracts. In: Panetto H. et al. (eds) On the Move to Meaningful Internet Systems. OTM 2017 Conferences. OTM 2017. Lecture Notes in Computer Science, vol 10574. Springer, Cham

5. Kim, HM, Laskowski, M.: Toward an ontologydriven blockchain design for supplychain provenance. Intell Sys Acc Fin Mgmt. 2018; vol 25: pp 18 27. https://doi.org/10.1002/isaf.1424
6. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M., 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 21(79), pp.1-31.
7. Desrosiers L., Olivieri R., 2018. Extend your blockchain smart contracts with off-chain logic. https://developer.ibm.com/articles/cl-extend-blockchain-smart-contracts-trusted-oracle/