WIP: Relational Specification of Smart Contracts

Derek Sorensen

August 14, 2022

Abstract

Smart contracts can exhibit high-level behavior that is desirable to formally verify but that can be difficult to characterize by way of program-level formal specification. In some cases, such behavior may be easier to understand and articulate in relation to another, simpler smart contract. To enable this kind of study, we introduce the notion of a morphisms of smart contracts. Morphisms of smart contracts are well-defined mathematical functions that indicate computational and structural relationships between smart contracts. For complex and high-level contract properties, such as the scope and limits of a contract's upgradeability features and the governance model implemented in a DAO, we show that difficult-to-articulate properties of these contracts can be abstracted and expressed by way of their structural relationships to simpler contracts. Both of these contract classes have been studied from, respectively, software engineering and economic, legal, and game-theoretic perspectives, but never through formal, contract-level specifications as we present here. This work has applications to software development life cycles, formal methods, and verification-led programming.

1 Introduction

Category theory is very effective in characterizing the behavior of mathematical objects by way of morphisms, where different kinds of morphisms indicate different kinds of structural relationships between objects. Through universal properties, a particular object can be completely characterized by how it relates to other objects via morphisms—the mathematical version of, "Tell me who your friends are and I'll tell you who you are." Due to its level of generality, category theory has applications in many areas of mathematics, physics, and computer science.

The problem that we aim to address here is that, in the formal verification of smart contracts, certain desired contract behavior is hard to characterize by way of current formal specification methods. The scope and limits of a smart contract's upgradeability features and the governance properties of DAOs are two examples

of these. Both feature complicated upgrade and at times game-theoretic logic that can be difficult to characterize in a formal specification without giving an example implementation that embodies all the relevant details [2, 8]. Rather than attempt to fully characterize the behavior of these contracts through a formal specification, by using morphisms of contracts we can instead abstract this minimal behavioral structure into a simpler smart contract and prove an invariant relationship between the contracts, indicating that the abstracted structure—and thus adherence to the specification—always holds. The abstracted contract is in some sense a highly expressive or embodied specification.

In order to have direct relevance to executable smart contract code, it is important that the theory developed here be rooted in a formal verification pipeline from which executable code can be extracted which satisfies the formally proven properties. To this end, we adopt the notion of the blockchain and smart contracts as abstracted in ConCert, a verification framework for third-generation blockchains written in Coq [3].

2 Morphisms of Smart Contracts

We propose studying smart contracts category-theoretically, by studying *morphisms of smart contracts*. A morphism of smart contracts is a function between smart contracts, defined mathematically by using the provably faithful and executable abstraction from ConCert in the proof assistant Coq [4]. Fundamentally, a morphism between two smart contracts indicates a structural and computational relationship between the two objects. As we will show, this can be useful in formally proving compliance with a given contract (*e.g.* token) standard. It can also be useful to show that a contract satisfies some minimal architecture requirements that characterize its upgradeability features or that characterize it as a DAO. It is our hope that these methods will let us study, characterize, and formally verify with more rigor and clarity some aspects of contract specification that are hard to express using current methods.

We introduce the following notions:

Morphism of Smart Contracts: On many third-generation blockchains, smart contracts are pure functions. Thus a morphism of smart contracts is a morphism of functions, which can be defined in the usual way of natural transformations and commutative diagrams. The definition of this morphism is nontrivial, however, because smart contracts emit operations, and so as functions smart contracts can exhibit general recursion. The existence of a morphism indicates structural relationships between smart contracts, as in the following examples of morphisms.

(Strong) Equivalence and Weak Equivalence of Smart Contracts: A morphism of contracts $f : \texttt{C} \to \texttt{D}$ which has a two-sided inverse is an equivalence of contracts. Computationally and semantically, two equivalent contracts have the same properties, and an equivalence of contracts induces an equivalence

of corresponding environments. This simply means that deploying `C` vs deploying `D` affect the state of the blockchain in mathematically equivalent ways.

This is a strong form of equivalence that may not fully capture the notion of equivalence; in particular, it does not reflect common intuition that a smart contract can be implemented modularly or in a monolithic contract, and as long as the modular structure is correct, these would be essentially equivalent. We thus introduce a weaker form of equivalence of contracts, which is weak equivalence, which captures a looser, but mathematically sound, notion of equivalence of contracts.

Importantly, equivalent contracts are indistinguishable propositionally.

Monomorphism of Contracts: A monomorphism, the categorification of an injective morphism in the category of sets, indicates a faithful preservation of structure by way of a morphism. In particular, a monomorphism of contracts $\texttt{C} \hookrightarrow \texttt{D}$ indicates that the computational structure of `C` is present in `D` in the form of a *sub contract*. One can use a monomorphism, *e.g.* to show that a particular contract satisfies a standard by showing a monomorphism from the standard into the contract.

Epimorphisms of Contracts: An epimorphism, the categorification of a surjective morphism in the category of sets, indicates a compression of structure onto another contract, in our case characterizing bounds of the contract's structure. One can use an epimorphism, *e.g.* to show that a contract conforms structurally to a skeleton, like a basic upgradeability framework.

3 Contract Upgradeability

Because contracts, once deployed, are immutable, upgradeable contracts are of great interest to most blockchain-based projects. More than needing a framework for fully upgradeable contracts, it is highly relevant to understand and be able to mathematically classify the bounds of upgradeability of a given contract. There are, for example, smart contracts for which it is not desirable to be fully upgradeable, or which need explicit and strict conditions on how an upgrade could be executed and to what extent a contract can be upgraded.

The study of those properties of a smart contract which are permanent over time is the study of invariants. However, it can be hard to specify formal invariants via low-level contract properties that guarantee the high-level, desired contract behavior. Take for example the specification of the Diamond standard, which is a robust and flexible upgrade standard for Ethereum smart contracts [7]. It is not clear that the specification itself, despite providing standard interfaces and blocks of code, fully captures the desired, high-level modular behavior of a system of upgradeable contracts. Indeed, it supplements the textual specification with diagrams,

analogies to diamonds, and example implementations to fully communicate the nature of the standard in such a way that the specification makes sufficient intuitive sense that a person can implement it.

Importantly, the standard comes with reference implementations which, in contrast to the informal specification, can capture the full computational meaning of the standard because they are fully implemented. Subtle details, such as details on data handling and message-passing, are expressed because they are implemented.

Even if the standard itself can be formally verified to be correct and safe, small changes in implementation can result in fatal errors in unexpected ways. It would be useful, then, for an implementation of the Diamond standard to show that it satisfies the same structural properties as the standard itself. That is, that its structure can be formally related to that of a correct implementation of the contract standard in such a way that key properties are preserved.

This can be done by use of morphisms, in particular by an epimorphism from an implementation onto the standard that shows that the implementation conforms structurally to the (potentially provably safe) standard, where the standard is expressed as a specific smart contract instance.

4 Decentralized Autonomous Organizations

DAOs are another example of complex, high-level contract behavior that can be hard to specify and study rigorously. In many ways similar to upgradeable contracts—because DAO contracts can change over time in response to governance—the distinguishing feature of DAOs is that they also have a governance model which dictates how and when certain contract parameters can be changed, or certain functions executed [6]. DAOs often define a set of procedures and incentives so that the organization can be governed effectively. This, in essence, defines an economic game encoded into the smart contract [5].

Low-level specifications, written as lists of properties of a prospective implementation, do not formally capture the logic of DAOs. They may give details of an implementation of the implicit game, but they do not formally define the game being played in the specification. Indeed, for DAOs of great economic interest and impact, research has been done to *discover* and characterize the economic game in play [1, 2].

Because of this, a formal specification of a DAO's governance model may be better expressed via a reference implementation of the basic or abstracted governance structure. We show how the governance model and logic implicit to a DAO can be characterized by implementing a generic contract which describes the game being played and showing a structural relation, again by way of morphisms of contracts, to a DAO that seeks to implement that game. In particular, we characterize this by a monomorphism from the game into the contract whose image is a *proper subcontract*.

5 Application to Software Development Life Cycle

The work here has applications to the software development life cycle, both in testing and verification, but also in the intuitive and natural way that smart contracts are developed in practice. This is because the theory presented here is abstracted sufficiently so that it can be written and reasoned about on white boards or on paper, but also expressive enough that it can describe low-level details of a smart contract.

For both upgradeable smart contracts and DAOs, design often starts at a high level. In the case of upgradeable contracts, developers may first define the skeleton architecture which allows for upgrades, and then populate it with the details of a project. For DAOs, it can make sense to start with the governance model and then move into the details. In both of these cases, if the basic structure can be abstracted and encoded in a smart contract, they can later formalize that structure and prove the relevant properties of the finalized smart contract by way of relation through morphisms of contracts.

It is our hope that theories of this nature can help encourage verification-led programming in a natural and intuitive way.

6 Conclusion

We have shown some preliminary ways in which understanding smart contracts in the context of category theory can be useful to understand, describe, and rigorously characterize contract behavior in ways that might be difficult to do with the current tools of formal specification. The extent to which such a study could both bring formal methods earlier on in the standard software development life cycle and advance our understanding of smart contract behavior is yet unclear and requires further research.

Future work includes (1) understanding further properties of the category of smart contract and how they translate into contract behavior and specifications, and (2) studying the category of *blockchains*, drawing on the same abstract framework and defining morphisms analogously. We hope that efforts in the latter will shed light on multi-chain contracts and inter-chain bridges, two instances of blockchain-based software development that have shown themselves to be particularly error-prone.

References

[1] Guillermo Angeris, Alex Evans, and Tarun Chitra. When does the tail wag the dog? curvature and market making. *arXiv preprint arXiv:2012.08040*, 2020.

[2] Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra. An analysis of uniswap markets. *arXiv preprint arXiv:1911.03380*, 2019.

[3] Danil Annenkov, Jakob Botsch Nielsen, and Bas Spitters. Concert: a smart contract certification framework in coq. *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, Jan 2020. URL: http://dx.doi.org/10.1145/3372885.3373829, doi:10.1145/3372885.3373829.

[4] Mikkel Milo Danil Annenkov and Bas Spitters. Code extraction from coq to ml-like languages. *ML'21 at ICFP'21*, 2021. URL: https://icfp21.sigplan.org/details/mlfamilyworkshop-2021-papers/8/Code-Extraction-from-Coq-to-ML-like-languages.

[5] Kedar Iyer and Chris Dannen. Crypto-economics and game theory. In *Building Games with Ethereum Smart Contracts*, pages 129–141. Springer, 2018.

[6] Ido Gershtein Michael Zargham Eyal Eithcowich Joshua Z. Tan, Isaac Patka and Sam Furter. Eip working paper: Decentralized autonomous organizations., 2022. URL: https://daostar.one/EIP.

[7] Nick Mudge. Eip-2535: Diamonds, multi-facet proxy, Feb 2020. URL: https://eips.ethereum.org/EIPS/eip-2535.

[8] Shuai Wang, Wenwen Ding, Juanjuan Li, Yong Yuan, Liwei Ouyang, and Fei-Yue Wang. Decentralized autonomous organizations: concept, model, and applications. *IEEE Transactions on Computational Social Systems*, 6(5):870–878, 2019.