IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 7, July 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.542

International Journal of Innovative Research in Computer and Communication Engineering

|e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 9, Issue 7, July 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0907070 |

Legal Agreements to Smart Contract using Blockchain Technology

Thorlikonda Balasundar, Thota Bharadwaj, Pulivarthi Balakrishna Prasad, Pachhala Nagababu

Assistant Professor, Department of Information Technology, Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

UG Students, Department of Information Technology, Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

ABSTRACT: Complex legal agreements enable many real-world applications, from data sharing systems to financial transactions. However, legal expenses scale with complexity because of the manual processes to draft, revise, and enforce agreements. To reduce such costs, we propose a new framework for lawyers to develop machine readable legal agreements, which are automatically verified and deployed on the Ethereumblockchain. Specifically, our framework introduces domain specific repositories to store human and machine readable legal language, a web interface and Python API to draft legal agreements, correctness checking via formal verification, and a voting system for blockchain based adjudication. Experimental evaluation found that our proposed framework offers an efficient verification system, incurs linear scaling of Ethereum blockchain gas consumption in terms of agreement size, and correctly models 81% of conditions in real world agreements through the domain specific repositories. These results suggest a practical approach for developing verifiable and blockchain compatible legal agreements

I. INTRODUCTION

"Smart contracts" is a term used to describe computer code that automatically executes all or parts of an agreement and is stored on a blockchain-based platform. As discussed further below, the code can either be the sole manifestation of the agreement between the parties or might complement a traditional text-based contract and execute certain provisions, such as transferring funds from Party A to Party B. The code itself is replicated across multiple nodes of a blockchain and, therefore, benefits from the security, permanence and immutability that a blockchain offers. That replication also means that as each new block is added to the blockchain, the code is, in effect, executed. If the parties have indicated, by initiating a transaction, that certain parameters have been met, the code will execute the step triggered by those parameters. If no such transaction has been initiated, the code will not take any steps. Most smart contracts are written in one of the programming languages directly suited for such computer programs, such as Solidity.
At present, the input parameters and the execution steps for a smart contract need to be specific and objective. In other words, if "x" occurs, then execute step "y." Therefore, the actual tasks that smart contracts are performing are fairly rudimentary, such as automatically moving an amount of cryptocurrency from one party's wallet to another when certain criteria are satisfied. As the adoption of blockchain spreads, and as more assets are tokenized or go "on chain," smart contracts will become increasingly complex and capable of handling sophisticated transactions. Indeed, developers already are stringing together multiple transaction steps to form more complex smart contracts. Nonetheless, we are, at the very least, many years away from code being able to determine more subjective legal criteria, such as whether a party satisfied a commercially reasonable efforts standard or whether an indemnification clause should be triggered and the indemnity paid.

Before a compiled smart contract actually can be executed on certain blockchains, an additional step is required, namely, the payment of a transaction fee for the contract to be added to the chain and executed upon. In the case of the Ethereumblockchain, smart contracts are executed on the Ethereum Virtual Machine (EVM), and this payment, made through the ether cryptocurrency, is known as "gas." The more complex the smart contract (based on the transaction steps to be performed), the more gas that must be paid to execute the smart contract. Thus, gas currently acts as an important gate to prevent overly complex or numerous smart contracts from overwhelming the EVM. Smart contracts are presently best suited to execute automatically two types of "transactions" found in many contracts: (1) ensuring the payment of funds upon certain triggering events and (2) imposing financial penalties if certain objective conditions are not satisfied. In each case, human intervention, including through a trusted escrow holder or even the judicial system, is

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 9, Issue 7, July 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0907070 |

not required once the smart contract has been deployed and is operational, thereby reducing the execution and enforcement costs of the contracting process.

As just one example, smart contracts could eliminate the so-called procure-to-pay gaps. When a product arrives and is scanned at a warehouse, a smart contract could immediately trigger requests for the required approvals and, once obtained, immediately transfer funds from the buyer to the seller. Sellers would get paid faster and no longer need to engage in dunning, and buyers would reduce their account payable costs. This could impact working capital requirements and simplify finance operations for both parties. On the enforcement side, a smart contract could be programmed to shut off access to an internet-connected asset if a payment is not received. For example, access to certain content might automatically be denied if payment was not received.

II. EXISTING SYSTEM

Existing projects, such as Ergo, model legal agreements as software programs by embedding machine readable functions inside legal clauses. Ergo provides a repository of clauses that users can import into their agreements, and it supports Coq for formal verification. Unlike Ergo, which focuses on representing the commonly used parts of a legal agreement via a functional programming language, our framework models the entire agreement via combinational logic and thus can formally verify the entire agreement.

III. PROPOSED SYSTEM

When instantiated, the lawyers must provide parameters to populate the text template, a list of parties who can update the state of the action, and a list of independent arbiters who can adjudicate disputes. When deployed as an Ethereumblockchain smart contract, the parties can propose updates to the state of any action. The other parties, and arbiters in the case of a disagreement, vote to decide whether to update the action state. The design of an action was based on concerns raised in that some inputs into programmatic agreements require human judgment. Clauses use deterministic combinational logic to resolve combinations of actions or other clauses (recursively) to a single Boolean output.

ARCHITECTURE:

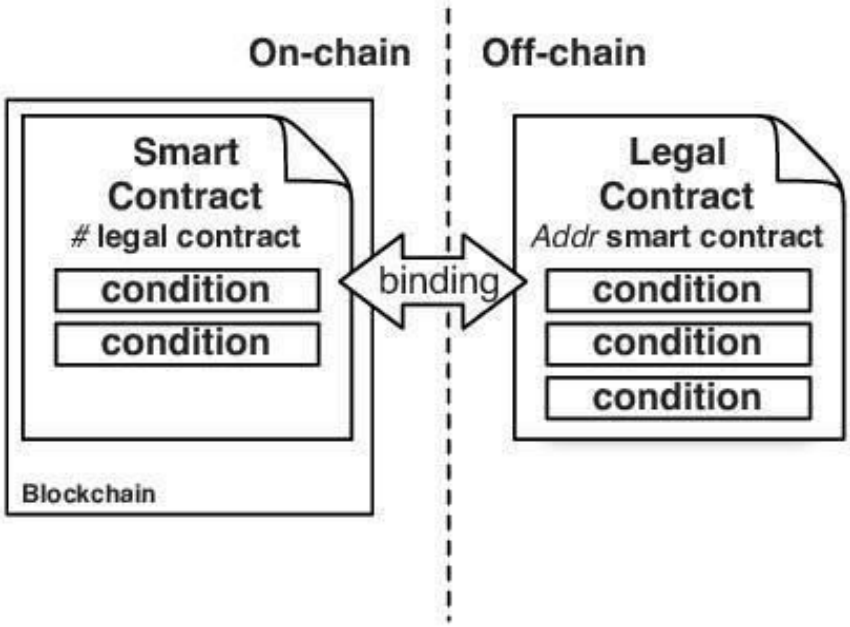

Fig 1: Architecture diagram of legal agreement

|e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 9, Issue 7, July 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0907070 |

IV. RESULTS

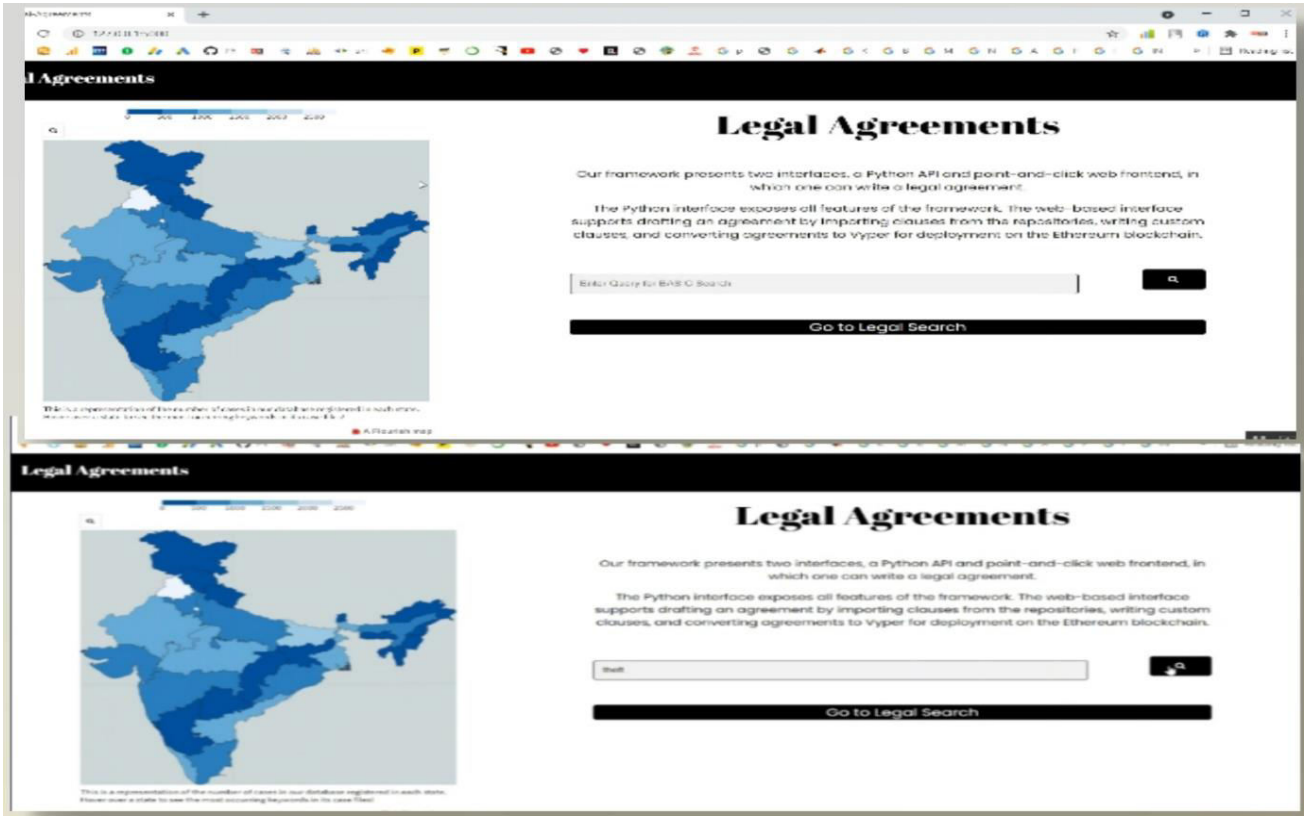

Fig 2: Interface of Legal Agreement

Fig 3:Interaface of legal Agreement

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542

|| Volume 9, Issue 7, July 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0907070 |

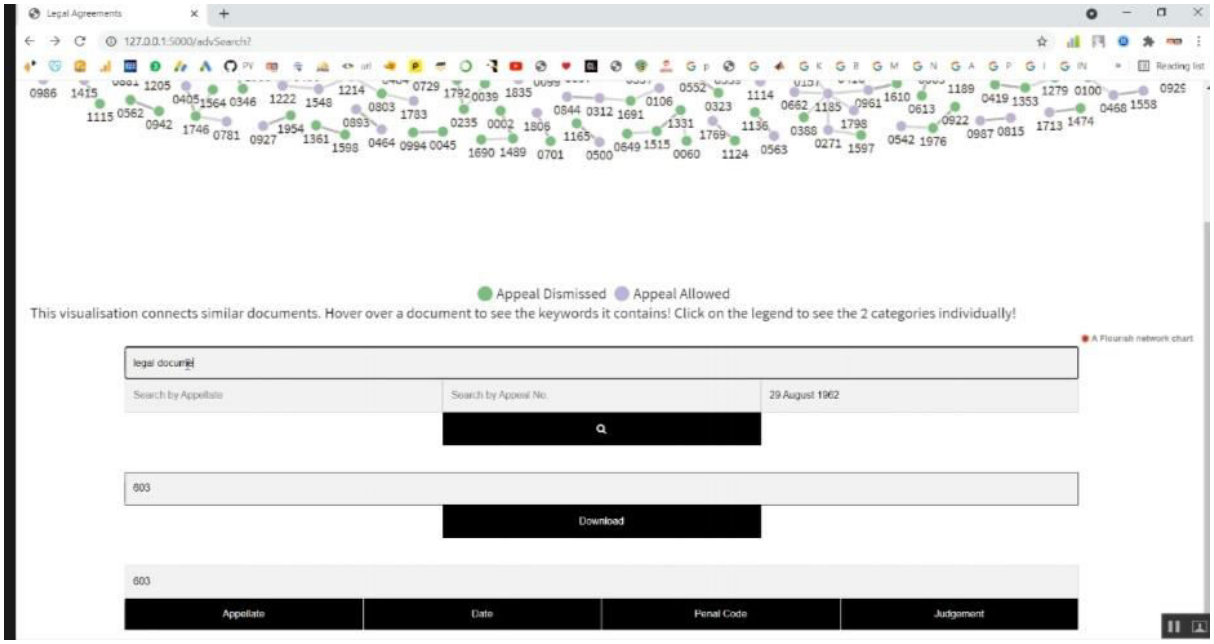

Fig3: Interface of Legal Agreement

V. CONCLUSION

Our framework enables lawyers to use a web interface to write legal agreements that can be automatically verified via formal verification and deployed on the Ethereumblockchain for lower adjudication costs when disputes arise. Storing legal language in repositories enables reuse of human and machine readable legal language across agreements. By having both a web interface and a Python API, our framework is accessible to both legal professionals and software developers alike. We aim to reduce the manual work involved with drafting, checking, and enforcing complex agreements. The evaluation indicates that the framework, through combinational logic, can accurately model 81% of legal constructs found in real-world agreements. The formal verification runtime and blockchain gas costs for generated agreements scale linearly with agreement complexity. These results suggest a practical approach for developing verifiable and blockchain compatible legal agreements

REFERENCES

1. Bourke v. Dun & Bradstreet Corp, 59 F.3d 1032 (7th Cir. 1998). [Online]. Available: https://casetext.com/case/bourke-v-the-dun-bradstreet-corp
2. Baybank v. Vermont National Bank, 118 F.3d 30 (1st Cir. 1997). [Online]. Available: https://caselaw.findlaw.com/us-1st-circuit/1057693.html [3] Lease abstraction. Kira Systems. [Online]. Available: https://kirasystems.com/solutions/lease-abstraction/
3. Garoufallou, S. Virkus, R. Siatri, and D. Koutsomiha, Eds., Toward a Metadata Framework for Sharing Sensitive and Closed Data: An Analysis of Data Sharing Agreement Attributes. Cham: Springer International Publishing, 2017. [Online]. Available: http://cci.drexel.edu/media/19189/ metadata-for-sharing-closed-data-grabus-greenberg-56-mtsr2017.pdf

INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 7.542

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details