ABSTRACT

Blockchain and Smart Contract development is challenging due to the niche expertise required and the field’s relative infancy. Our tool, LATTE, visually represents and simplifies the concepts in Ethereum Smart Contracts. Graphical objects and diagrams are used to represent code and logic, which will allow quicker prototyping of smart contracts, and encourage involvement from new users and developers. LATTE also allows easy deployment of smart contract code on to the blockchain with the click of a button. We also examine the performance and gas usage of smart contracts generated using LATTE and compare it to reference smart contracts.

LATTE OVERVIEW

- Novel, simpler way to create and deploy Solidity smart contracts for use on the Ethereum blockchain.
- A key goal is to maximise user friendliness, so concepts are simplified and explained through extensive use of tooltips at every step. User interface is clean and minimal to prevent user confusion.
- Drag and drop interface to define smart contract logic makes LATTE easy to learn and use compared to writing Solidity code.

BACKGROUND

- Blockchain was first conceptualised in 2008, so it is still a relatively new concept.
- Since it is a new technology, the blockchain ecosystem is fragmented and fast shifting. It is challenging for people to pick up the technology.
- User-friendly visual interfaces are important for widespread public development of blockchain. Most blockchain development currently requires use of command line tools that are intimidating to new users.

SUPPORTED FEATURES

- Variable assignment
- Basic variable types (integer, string, address, boolean)
- Mapping variable type and nested mapping
- Logical if/else and while loop constructs
- Events and Structs (known as “Entities” in LATTE)
- Require statements
- Transfer function

UNSUPPORTED FEATURES

- For loops - can replace with while loops
- Bit operations/Assembly – complex, unlikely to be required by user.
- Arrays - can replace with mapping of index to contents of array.
- Modifiers – for code readability, unlikely to be required by user.
- Enumerators - can replace with integers to represent constants.
- Self Destruct - code optimisation, unlikely to be required by user.
- Libraries - code reuse/optimisation, unlikely to be required by user.

RESULTS

- Out of 8 reference contracts used, we were able to implement half of them fully. LATTE does not realise contracts with hashing, encoding, self-destruct or assembly functions as target audience consists of novice users who are unlikely to require these complex functions.
- LATTE generated contracts and reference contracts’ functionality and output are the same.
- For simple contracts, generated contracts were efficient, having performance deviation up to 6%, but LATTE struggled with the complex Voting contract, having up to 80% drop in performance compared to the reference contract.

<table>
<thead>
<tr>
<th>Contract and Function Name</th>
<th>Reference Contract (gas)</th>
<th>LATTE (gas)</th>
<th>Performance Difference (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Auction Deposit</td>
<td>400000</td>
<td>422800</td>
<td>-5.7</td>
</tr>
<tr>
<td>Open Auction Constructor</td>
<td>440866</td>
<td>463667</td>
<td>-5.17186628136441</td>
</tr>
<tr>
<td>Open Auction Bid</td>
<td>63208</td>
<td>63543</td>
<td>-0.529996203012277</td>
</tr>
<tr>
<td>Safe Remote Purchase Deposit</td>
<td>437000</td>
<td>425000</td>
<td>2.74599542334096</td>
</tr>
<tr>
<td>Safe Remote Purchase Constructor</td>
<td>477996</td>
<td>465813</td>
<td>2.54876609846107</td>
</tr>
<tr>
<td>Safe Remote Purchase Confirm Purchase</td>
<td>42184</td>
<td>42036</td>
<td>0.3508439218661 1</td>
</tr>
<tr>
<td>Voting Deposit</td>
<td>446400</td>
<td>540600</td>
<td>-21.1021505376344</td>
</tr>
<tr>
<td>Voting Constructor</td>
<td>487492</td>
<td>709092</td>
<td>-45.4571562199995</td>
</tr>
<tr>
<td>Voting Vote</td>
<td>102527</td>
<td>184327</td>
<td>-79.7838618120105</td>
</tr>
<tr>
<td>Voting Winning Proposal</td>
<td>418</td>
<td>394</td>
<td>5.74162679425837</td>
</tr>
</tbody>
</table>

VOTING CONTRACT EXAMPLE

Student: Sean Tan Jun Yu
Supervisor: Associate Professor Sourav S Bhowmick