Psamathe: A DSL for Safe Blockchain Assets

Reed Oei [1], Michael Coblenz [2], Jonathan Aldrich [2]

[1]University of Illinois [2]Carnegie Mellon University

Introduction

Blockchains are becoming more commonly used as platforms for programs called ***smart contracts***, which usually manage ***digital assets***. For example, there are smart contracts for managing lotteries, auctions, games, and the most common kind of digital asset on the Ethereum blockchain, a ***token***. Tokens typically are used as a kind of digital currency, the most common kind of tokens being defined by the ERC-20 or ERC-721 standards, which account for approximately 73% of high-activity contracts [1] as well as many other less-used contracts.

Hacks of smart contracts regularly occur, often causing the loss of large quantities of money, such as the DAO attack in 2016, which led to the loss of over 40 million dollars worth of ether [2].

Psamathe is a new programming language we are designing for writing smart contracts that provides a way to mark values as ***assets***, with various ***modifiers*** to control their use, as well as a new abstraction, called a ***flow***, representing an atomic transfer operation, which is widely applicable to smart contracts managing digital assets.

Contributions

- **Flow abstraction**: Psamathe uses the new flow abstraction to encode semantic information about the flow of assets into the code.
- **Safety guarantees**: Psamathe ensures that assets are properly managed, eliminating reuse, asset-loss, and duplication bugs through the flow abstraction.
- **Conciseness**: Psamathe makes writing typical smart contract programs more concise by handling common patterns and pitfalls.

Example

```
contract ERC20 {
  type Token is fungible asset uint256
  balances : map address => Token
  transaction transfer(dst : address, amt : uint256):
    balances[msg.sender] ——[ amt ]—> balances[dst]
}
```

Figure 1. A Psamathe contract implementing the ERC-20 function transfer, which transfers amount tokens from the sender's account to the destination account. It is implemented with a single flow, which checks all the preconditions to ensure the transfer is valid.

```
contract ERC20 {
  mapping (address => uint256) balances;
  function transfer(address dst, uint256 amt)
    public returns (bool) {
    require(amt <= balances[msg.sender]);
    balances[msg.sender] = balances[msg.sender].sub(amt);
    balances[dst] = balances[dst].add(amt);
    return true;
  }
}
```

Figure 2. An implementation of ERC-20's transfer function in Solidity, the most commonly used smart contract language on Ethereum. All preconditions are checked manually. Note that we must include the SafeMath library (not shown), which checks for underflow/overflow, to use the add and sub functions.

In this example, we see how flows can be used to clearly express the **intent** behind the transfer function, which is obscured by the Solidity code. Furthermore, using Psamathe will provide additional safety, e.g., by ensuring that no tokens are lost or duplicated; this is particularly helpful for more complicated transactions which include more flows or more complicated flows (e.g., transfer fees).

Formalization

$$\frac{\begin{array}{c}\Gamma \vdash A\ \mathtt{provides}_{\mathcal{Q}}\ \tau \\ \Gamma \vdash s\ \mathtt{selects}_{\mathcal{R}}\ \tau \qquad \mathtt{validSelect}(s, \mathcal{R}, \mathcal{Q}) \\ \Delta = \mathtt{update}(\Gamma, A, \Gamma(A) \ominus \mathcal{R}) \qquad \Delta \vdash B\ \mathtt{accepts}\ \tau\end{array}}{\Gamma \vdash (A \xrightarrow{s} B)\ \mathsf{ok} \dashv \mathtt{update}(\Delta, B, \Delta(B) \oplus \min(\mathcal{Q}, \mathcal{R}))}$$

Figure 3. The rule checking the well-formedness of a single flow.

We are also developing a ***formal semantics*** for Psamathe, i.e., mathematical definitions and proofs of

- the precise behavior of a program when it is executed,
- which programs are well-typed,
- and the guarantees that being well-typed gives, such as no accidental asset loss.

Conclusion and Future Work

Psamathe is a new programming language for making smart contracts managing assets safer. Flows can be used to encode transfers of assets in a safe, and concise, manner. In the future, we hope to fully implement the Psamathe language and evaluate it via longer case studies and user studies.

References

[1] Gustavo Oliva, Ahmed E. Hassan, and Zhen Jiang.
An exploratory study of smart contracts in the ethereum blockchain platform.
Empirical Software Engineering, 11 2019.

[2] Emin Gün Sirer.
Thoughts on the DAO hack, 2016.