I am analyzing a compiler which was called Bamboo Compiler.

For me, the design was something unreadable and it was an incomplete model as an escrow language.

Now, I clean up and aliasing the former work and I put it on my github as pen

compiler project.

github.com

ghasshee/pen

A compiler into EVM code. Contribute to ghasshee/pen development by creating an account on GitHub.

Loops and Program Graph

Bamboo does not have loops in the language design, claiming that we do not need loops because we have gas limit in the execution.

I think this is not the word of those who create a User Friendly Language.

The dijkstra's Guarded Command Language(GCL) is a way of defining formal method and solving this kind of problems. GCL is translated into Program Graphs easily.

It is compiler that transform loops into a safe finite sequences of opcodes.

This might allow us to use loops

in the language design.

Functional OOP

Unlike Bamboo, I would like the approach of Functional OOP language design which is developped by B.C. Pierce and so on around 90's, which is the original model of Objective Caml.

I heard that Bamboo cannot fill the gap between escrow language which @pirapira

pursued and the EVM design which is quite similar to OOP design.

I would like to see and borrow ideas from a lot of projects now going , e.g. michelson in Tezos.

I found Bamboo compiler lacks some of EVM opcodes like CALLCODE

which I believe caused the parity bug. I do not know now, how to avoid this operation called from some sender, and in the design of Functional OOP. The first thing is to do the whole design of FOOP on EVM roughly.

Stack Verification

Recently, a team on tezos language developped a formal way to anlayze the language in order to grasp if the language terminates as intended.

This is done by applying Refinement Type to the Stack. Refinement Type can have a "predicate" on every Type. We can have every subset of every type as a set with applying the predicate.

Thus, type checking on the type of stack contents is exactly formal vericafition of the program.

(However, this type checking sometimes does not terminate if it has second-order logic.)

The implementation depends on SMT solver for the computation of the logical part.

I would like to try this on EVM.

TODO

- make Guarded Command Langauge for EVM
- define Functional OOP
- add stack verification

I would like to know much more about language designs.

If anyone know about them or interested in them, feel free to let me know !