Linear Logic as a blockchain language

Denis Erfurt

ABSTRACT

asd asdas

1 MOTIVATION

- Linear Logic is a **Logic**, which is one of the oldest and best studied mathematical systems.
- proof terms Curry-Howard correspond to the pi-calculus - a process calculus with atomic notion of concurrency, making it an ideal candidate for blockchain application.

1.1 Requirements

(1) R01: Programming obvious applications has to be intuitive.
(2) R02: Practically proving the correctness of programs must be possible.
(3) R03: Execution of compiled programms must be relativelly effecient.
(4) R04: Programs has to be composable with other programs (e.g. they should be able to call and be called by other programs).

2 GENERAL IDEA

The general Idea is to view the blockchain [contract] as a Multiset $\mathcal{B}$ of Ressources.

$$\frac{\dfrac{\mathcal{D}}{\Delta \vdash C} \quad \exists\theta.\Delta\theta \subseteq \mathcal{B}}{\mathcal{B} \longrightarrow \mathcal{B} \setminus \Delta\theta \cup \{C\theta\}}\ \mathcal{B}\text{-transition} \tag{1}$$

2.1 Unification

During a blockchain transition step a Most General Unifier(MGU) θ has to be found. Initially θ contains the assigment of reserved variables, such as $Sender$, $Origin$, $BlockNumber$, etc, which don't change during unification.

3 EXAMPLE PROGRAMMS

Lets considder a simple token.

4 SECURITY

We are interested to proof different properties about our written programs. E.g. that the total value of a system is not changed with different operations, this we want to illustrate on a case study - a coin exchange: Let d denote a dime, n a nickel and q a quarter together with the following persistent rules given:

$$\left.\begin{aligned} r1 &: q \multimap d \otimes d \otimes n \\ r2 &: d \otimes d \otimes n \multimap q \\ r3 &: d \multimap n \otimes n \\ r4 &: n \otimes n \multimap d \end{aligned}\right\} R$$

Also we want to have a persistent notion of value:

$$\left.\begin{aligned} v1 &: q \multimap value(25) \\ v2 &: d \multimap value(10) \\ v3 &: n \multimap value(5) \end{aligned}\right\} V$$

Additionally we can add up values:

$$sum : value(X) \otimes value(Y) \multimap value(X + Y)$$

Now our task is to prove the stability of the system. This means that the total value don't change if a rule in R is applied. We can do this easily by proof expansion of every rule in R. We want to illustrate this for $r1$, the other cases can be shown analogous: Without loss of generality let $\Delta = value(X), q$, we want to proof $\Delta \vdash value(X + 25)$ first by applying only Rules out of V + sum as seen in Proof 1 and expand this proof by applying $r1$ first and then restricting ourselves with only using rules out of V, which can be seen in Proof 2. This proves that the application of rule $r1$ didn't change the total value. With this we have shown that rules in R satisfy our stability criteria. The proves are rather technical and included for the purpose to demonstrate, that those proves can be easily automatized.

TODO: provided that the computation is deterministic and arbitrary values can't be derived. Find a way how to prove those properties automatically.

5 EFFICIENCY

As computation is done via proof-search and unification, executing programms tend to be slow in practice. Therefore we are interested in methods which reduces the computational overhead. One such method could be the recuction of proof search to the execution of a minimal set of evm instructions, which yield the same result. For this the equivalence of the proof search to the evm instructions has to be **proven**.

In order to achieve this, we will reconstruct the evm in linear logic and show for a proposition A, a proposition B which represents an evm program and a substitution θ:

$$\Gamma; \cdot \vdash A(\bar{X})\theta \Leftrightarrow \Gamma; \cdot \vdash B(\bar{X})\theta$$

TODO: Or is it $\Gamma; \cdot \vdash A(\bar{X}) \multimapboth B(\bar{X})$?

Meaning that the evm programm B is behaviourally isomorphic to A.

EVM implementation

For demonstration purposes we will just implement a basic stack machine here, which should be sufficient to illustrate the approach.

The EVM has a few distinct components, such as:

- code
- pc
- stack
- …

We can model the Stack Machine with the following predicates:

- code(X,Y) - OpCode Y is on position X in the code.
- pc(X) - Program Counter is at position X
- stack(X,Y) - word Y is on position X on the stack
- …

Additionally we create a few extra predicates which will help during the execution:

- stackLength(X) - X is the current stack length

We now want to model the predicate $inc(X, Y) := X + 1 = Y$ in the evm with the following program: *PUSH X INC*
The result should then be found at stack height 0. We can retrieve it with:

$$stack(0, X) \otimes pc(3) \multimap result(X)$$

Also initially we have the following predicates given: $pc(0) \otimes stackLength(0)$

PUSH

$$pc(X) \otimes code(X, PUSH) \otimes code(sX, V) \otimes stackLength(L)$$

$$\multimap$$

$$stack(L, V) \otimes pc(ssX) \otimes code(X, PUSH) \otimes code(sX, V) \otimes stackLength(sL)$$

This pushes the value V to the stack, increments the stack length and preserves the code.

INC

Assume we have the following axiomatic rule for inc given:

- inc(X, sX)

Thus:

$$inc(X, Y) \Leftrightarrow X + 1 = Y$$

So the following transition rule models our INC opCode:

$$pc(X) \otimes code(X, INC) \otimes stack(L, V) \otimes stackLength(sL) \otimes inc(V, W)$$

$$\multimap$$

$$pc(sX) \otimes code(X, INC) \otimes stackLength(sL) \otimes stack(L, W)$$

proof

Together we have to prove:

$$inc(X, Y) \multimapboth (C \otimes \mathcal{R} \otimes pc(0) \otimes stackLength(0) \multimap result(Y))$$

with the code:

$$C := code(0, PUSH) \otimes code(1, X) \otimes code(2, INC)$$

and the termination condition:

$$\mathcal{R} := pc(3) \otimes stackLength(1) \otimes stack(0, Z) \multimap result(Z)$$

TODO: Make the actual proof, but the conjecture is that this is proovable.

TODO: How does this work with non-primitives, e.g. loops, math-function e.g. log, exp

6 APPENDIX

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathit{value}(X) \otimes \mathit{value}(25) \vdash \mathit{value}(Y) \otimes \mathit{value}(Y')}\,\mathit{id}(\sigma) \quad \dfrac{}{\mathit{value}(Y + Y') \vdash \mathit{value}(X + 25)}\,\mathit{id}(\sigma)}{\mathit{value}(X) \otimes \mathit{value}(25), \mathit{value}(Y) \otimes \mathit{value}(Y') \multimap \mathit{value}(Y + Y') \vdash \mathit{value}(X + 25)}\,\multimap\mathrm{L}}{\mathit{value}(X) \otimes \mathit{value}(25) \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{sum}}{\mathit{value}(X), \mathit{value}(25) \vdash \mathit{value}(X + 25)}\,\otimes\mathrm{L} \quad \dfrac{}{q \vdash q}\,\mathrm{id}}{\mathit{value}(X), q, q \multimap \mathit{value}(25) \vdash \mathit{value}(X + 25)}\,\multimap\mathrm{L}}{\mathit{value}(X), q \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{v1}$$

$$\sigma = \{\tfrac{X}{Y}, \tfrac{25}{Y'}\}$$

Abbildung 1: Proof using only rules from V

$$\dfrac{\dfrac{}{q \vdash q}\,\mathrm{id} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{d \vdash d}\,\mathrm{id} \quad \dfrac{\dfrac{\mathcal{D}}{\mathit{value}(Y + Y') \otimes \mathit{value}(10), n \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{sum}}{\mathit{value}(Y + Y'), \mathit{value}(10), n \vdash \mathit{value}(X + 25)}\,\otimes\mathrm{L}}{\mathit{value}(Y + Y'), d, d \multimap \mathit{value}(10), n \vdash \mathit{value}(X + 25)}\,\multimap\mathrm{L}}{\mathit{value}(Y + Y'), d, n \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{v2} \quad \dfrac{}{\mathit{value}(X) \otimes \mathit{value}(10) \vdash \mathit{value}(Y) \otimes \mathit{value}(Y')}\,\mathit{id}(\sigma)}{\mathit{value}(X) \otimes \mathit{value}(10), \mathit{value}(Y) \otimes \mathit{value}(Y') \multimap \mathit{value}(Y + Y'), d, n \vdash \mathit{value}(X + 25)}\,\multimap\mathrm{L}}{\mathit{value}(X) \otimes \mathit{value}(10), d, n \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{sum}}{\mathit{value}(X), d, n, \mathit{value}(10) \vdash \mathit{value}(X + 25)}\,\otimes\mathrm{L} \quad \dfrac{}{d \vdash d}\,\mathrm{id}}{\mathit{value}(X), d, d, n, d \multimap \mathit{value}(10) \vdash \mathit{value}(X + 25)}\,\multimap\mathrm{L}}{\mathit{value}(X), d, d, n \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{v2}}{\mathit{value}(X), d, d \otimes n \vdash \mathit{value}(X + 25)}\,\otimes\mathrm{L}}{\mathit{value}(X), d \otimes d \otimes n \vdash \mathit{value}(X + 25)}\,\otimes\mathrm{L}}{\dfrac{\mathit{value}(X), q, q \multimap d \otimes d \otimes n \vdash \mathit{value}(X + 25)}{\mathit{value}(X), q \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{r1}}\,\multimap\mathrm{L}$$

With $\mathcal{D}$:=

$$\dfrac{\dfrac{}{\mathit{value}(Y + Y') \otimes \mathit{value}(10) \vdash \mathit{value}(Z) \otimes \mathit{value}(Z')}\,\mathit{id}(\sigma) \quad \dfrac{\dfrac{\dfrac{\mathcal{E}}{\mathit{value}(Z + Z'), \mathit{value}(5) \vdash \mathit{value}(X + 25)}\,\otimes\mathrm{L} \quad \dfrac{}{n \vdash n}\,\mathrm{id}}{\mathit{value}(Z + Z'), n, n \multimap \mathit{value}(5) \vdash \mathit{value}(X + 25)}\,\multimap\mathrm{L}}{\mathit{value}(Z + Z'), n \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{v3}}{\mathit{value}(Y + Y') \otimes \mathit{value}(10), \mathit{value}(Z) \otimes \mathit{value}(Z') \multimap \mathit{value}(Z + Z'), n \vdash \mathit{value}(X + 25)}\,\multimap\mathrm{L}$$

With $\mathcal{E}$:=

$$\dfrac{\dfrac{\dfrac{}{\mathit{value}(Z + Z') \otimes \mathit{value}(5) \vdash \mathit{value}(Q) \otimes \mathit{value}(Q')}\,\mathit{id}(\sigma) \quad \dfrac{}{\mathit{value}(Q + Q') \vdash \mathit{value}(X + 25)}\,\mathit{id}(\sigma)}{\mathit{value}(Z + Z') \otimes \mathit{value}(5), \mathit{value}(Q) \otimes \mathit{value}(Q') \multimap \mathit{value}(Q + Q') \vdash \mathit{value}(X + 25)}\,\multimap\mathrm{L}}{\mathit{value}(Z + Z') \otimes \mathit{value}(5) \vdash \mathit{value}(X + 25)}\,\mathit{copy}_{sum}$$

With $\sigma := \{\tfrac{X}{Y}, \tfrac{10}{Y'}, \tfrac{X+10}{Z}, \tfrac{10}{Z'}, \tfrac{X+10+10}{Q}, \tfrac{5}{Q'}\}$

Abbildung 2: Proof using rule $r1$ and only rules from V