

Keywords

linear logic, blockchain, types, Curry-Howard

ABSTRACT

We present an interpretation of classical linear logic in terms of operations on the blockchain.

Linear types can change the blockchain!

L.G. Meredith
Biosimilarity, LLC

lgreg.meredith@biosimilarity.com

1. BACKGROUND AND MOTIVATION

i've had a lot of time to think this week because i've been waiting on machines to synchronize with the blockchain. As a result, i've been more motivated than ever to find a scalable and reliable architecture for the blockchain. i believe i'm getting closer to a way to explain and test a hypothesis that linear proofs provide the basis for a much more scalable architecture for the blockchain. For background on what i mean by linear proofs, here's a paper that interprets them in terms of games, and here's another that interprets them in terms of traditional computational calculi like the lambda calculus.

A linear proof is a formal structure representing a proof of a formula in linear logic. The Curry-Howard isomorphism tells us that formulae are types (as in data types in a programming language), and that proofs are programs. This is a very broad and deep idea. In the 90's, for example, Abramsky extended it to proofs as processes, which Wadler was only very recently able to realize as a correspondence between linear proofs and λ -calculus processes. In this context it means that linear proofs provide a representation of both data (blocks) and program (executable transactions) that gives several advantages over the current choices made by the blockchain.

The blockchain is a great example of data that is also program; it's a giant ledger spread out over the Internet, that's made of a bunch of distributed, but interacting servers. To become more scalable, and reliable, both ledger and servers will need certain characteristics of data/program that have to do with a

property called compositionality. Scalability is always all about being able to build composite systems from components. For example, if we can prove that sections of the blockchain can be safely isolated from other sections, for example, if all blocks necessary to prove that Alice has sufficient funds to send M btc to Betty, can be isolated from the blocks necessary to prove that Alfred has sufficient funds to send N btc to Bob, then Alice and Betty, and Alfred and Bob can safely work with projections of the blockchain, and thus complete their transactions, not only in isolation of each other, but without the onerous need to sync the entire blockchain.

One analogy is the use of separation logic (a child of linear logic) to prove things about the structure of the heap which can, in turn, be used to guarantee that two threads can operate at the same time safely. The blockchain is like the heap. The Alice - Betty and Alfred - Bob transactions are like the two threads. A proof that the heap is of the form $H1 * H2$ together with a proof that $T1 : H1 \rightarrow H1'$, $T2 : H2 \rightarrow H2'$ constitutes a proof that $T1 * T2$ (thread T1 running concurrently with T2) operate effectively in isolation and thus safely. Likewise, a proof that the blockchain is of the form $B1 * B2$ (none of the transactions in B1 connect to addresses in B2 and vice versa), together with a proof that $\text{AliceBettyTxn} : B1 \rightarrow B1'$ (this txn uses only addresses in B1), and $\text{AlfredBobTxn} : B2 \rightarrow B2'$ (this txn uses only addresses in B2) constitutes a proof that $\text{AliceBettyTxn} * \text{AlfredBobTxn}$ can operate with isolated projections of the blockchain.

If the blockchain is built using the primitives of linear logic, it becomes easier and easier to construct these proofs, but also to construct the blockchain, itself, in terms of smaller blockchains.

2. INTERPRETING LINEAR PROOFS AS OPERATIONS ON THE BLOCKCHAIN

Here's the most basic interpretation.

$$\vdash 1blkchnaddr : A \otimes \dots \otimes A$$

is a statement that there are N A 's available at the address, $1blkchnaddr$. A 's can be any resource, BTC's, AMP's, DogeCoin, etc.

$$\vdash txn : A \otimes \dots \otimes A \multimap B \otimes \dots \otimes B$$

is a statement that txn will generate M B 's if provided N A 's.

Terminologically, we say that $1blkchnaddr$ is a *witness* or a *proof* of $A \otimes \dots \otimes A$, and similarly, that txn is a witness or proof of $A \otimes \dots \otimes A \multimap B \otimes \dots \otimes B$. Given two such proofs we can use the cut rule of linear logic to produce a proof

$$\vdash txn(1blkchnaddr) : B \otimes \dots \otimes B$$

where $txn(1blkchnaddr)$ is a new address in the blockchain constructed from the information in txn together with $1blkchnaddr$. This should look remarkably like function application, because it is.

Notice that now we can see, recursively, what a proof of a statement like

$$\vdash 1blkchnaddr : A \otimes \dots \otimes A$$

looks like. In most cases it will be a proof made from a previous application of the cut rule. This tree of cuts will trace all the way back to some genesis block – which is the only other way to have a proof of a statement like

$$\vdash 1blkchnaddr : A \otimes \dots \otimes A$$

Now, where does the return address associated with $txn(1blkchnaddr)$ come from?

To see this we have to look into the mechanics of \multimap (called linear implication or, more affectionately, lol-lipop).

$$A \multimap B = A^\perp \wp B$$

Linear implication is decomposed much like classical implication in terms of a negation and a disjunctive connective. It is literally an expression capturing the sentiment we need A to get B , or B comes with a cost of A . The use of a proof rule for these kinds of links looks like

$$\frac{\vdash G, 1blkchnaddr : A^\perp, 2blkchnaddr : B}{\vdash G, txn(1blkchnaddr, 2blkchnaddr) : A^\perp \wp B}$$

where we snuck in the rest of the blockchain as G . As we saw above, $A^\perp \wp B = A \multimap B$; so, we can write

$$\frac{\vdash G, 1blkchnaddr : A^\perp, 2blkchnaddr : B}{\vdash G, txn(1blkchnaddr, 2blkchnaddr) : A \multimap B}$$

Now we see that forming a txn comes with the requirement to provide an address where A 's will be sent and an address where B 's will be received. Expanding on these intuitions, we can see that the rules of classical linear logic correspond exactly to a specification of operations on the blockchain.

2.1 Linear Sequents

In more detail, a proof rule in linear logic is usually written in terms of a transformation,

$$\frac{S_1, S_2, \dots, S_N}{S}$$

taking *sequents* S_1, \dots, S_N to a *sequent* S , where a sequent is of the form

$$\vdash G, t_1 : A_1, \dots, t_N : A_N$$

A sequent is really just a statement about what is distributed where in an instance of the blockchain.

- t_1, \dots, t_N are either addresses or programs that take addresses as input; they constitute the "focus" of the proof rule, or where the action is going to happen.
- A_1, \dots, A_N are (types built from) the different types of coin
- G is the rest of the blockchain – it is necessary to establish the distribution of resources we see at t_1, \dots, t_N , but it's not the focus of the operation of the proof rule.

Putting it all together, a proof rule of the form

$$\frac{S_1}{S_2}$$

is then a statement about how the blockchain in state S_1 goes to a blockchain in state S_2 . If you think about it, that's just what we need to reason about transactions. In a transaction where Alice sends Betty N coin, we can think of the transaction as a rule that takes a blockchain in a state where Alice has N btc to a blockchain in a state where Betty has N btc.

2.2 The Multiplicatives

Linear logic, however, allows to build bigger blockchains from smaller ones, and manages the dependency and information flow so that everything remains consistent. Here's an example. The proof rule for the tensor $A \otimes B$ looks like this

$$\frac{\vdash G, t : A, \vdash H, u : B}{\vdash G, H, t \otimes u : A \otimes B}$$

It says that if you have one blockchain, $\vdash G, t : A$, and another completely independent blockchain, with a totally separate address space, $\vdash H, u : B$, then you can make a new one

$$|- G, H, t \otimes u : A \otimes B$$

in which you just combine all the data of assignments of addresses to resources in G and H in one big blockchain, G, H , and you can make a kind of composite address (or program), $t \otimes u$, at which can be found the combined $A \otimes B$ resource.

Now, comparison of the par ($A \wp B$) rule, which establishes transaction links, is even more illuminating.

$$\frac{\vdash G, t : A, u : B}{\vdash G, t \wp u : A \wp B}$$

This rule insists that the transaction link, $t \wp u$, is made in the same piece of the blockchain, G .

The fragment of linear logic that includes, A^\perp , $A \otimes B$, $A \wp B$, $A \multimap B$, is called the multiplicative fragment of linear logic, or MLL. It talks about the basics of transactions, loading up addresses with resources and

establishing dependencies between addresses, essentially recording transaction history. However, it does so in a way that keeps track of how the blockchain itself is segmented. This allows us to determine things like how much of the blockchain do i have to see in order to safely conduct this transaction, or can i conduct this transaction without needing visibility into that region of the blockchain.

2.3 The Additives

Linear logic also enjoys another fragment, called the additives. This aspect of the logic is all about conditionals and contingencies, this or that, but not both. The linear logic connective called 'with', and denoted $A \& B$, collects options together into a menu for subsequent selection by interaction with choices indicated by the linear connective 'plus', $A + B$. In symbols,

$$\frac{\vdash G, t : A, u : B}{\vdash G, t \& u : A \& B}$$

while

$$\frac{\vdash G, t : A}{\text{inl}(t) : A + B}$$

and

$$\frac{G, u : B}{\text{inr}(u) : A + B}$$

If during a more complex transaction $t \& u$ gets tied to $\text{inl}(t')$, via $\text{txn}(t \& u, \text{inl}(t'))$, then this will reduce to a transaction of the form $\text{txn}(t, t')$. On the other hand, $\text{txn}(t \& u, \text{inr}(u'))$ will reduce to a transaction of the form $\text{txn}(u, u')$.

2.4 The Exponentials

The fragment of linear logic that includes the multiplicative and additive connectives is called MALL. The remaining connectives are called the exponentials, $?A$, and $!A$. They denote copyable, non-conserved resources. When we write $\vdash G, t : !A$, we are saying that you can get as many A 's from the address (or program) t as you want. Thus, unlike currency, that address is linked to a copyable resource like a document, or a jpeg, or audio file, or ... that can be shared widely. When we write $\vdash G, t : ?A$, we are saying that you can put as many A 's into the address (or program) t as you want. You can think of it as a place to store A 's, or discard them.

What’s critically important about the use of the exponentials is that they mark resources that ought not to stay on the blockchain. They indicate content and content types that can be better served by a different kind of content delivery network. This is another important function in helping with a scalable blockchain – use blockchain technology where it makes sense and use other means where it doesn’t.

Taken all together, we have an interpretation of full classical linear logic in terms of operations on the blockchain.

3. CONCLUSIONS AND FUTURE WORK

We have developed a view of full classical linear logic in terms of operations against the blockchain. The view we have been developing not only extends to provide a meaningful interpretation of full classical linear logic to natural and intuitive operations on the blockchain, it also extends and expands how we think about the blockchain and what transactions on it are. Additionally, it provides guarantees, mathematical certainties about the correctness of transactions structured and executed this way. In particular, notice that we focused mostly on the connectives governing A ’s and B ’s (the resources to be found at addresses or programs). We didn’t really talk about the structure of t ’s and u ’s. These provide us with a simple and intuitive syntax for transactions. Of equal importance, these transactions are *typed* programs. When we write $\vdash G, t : A$, we are not only saying something about the resources produced or manipulated by t , we are saying something about how t can be used, and in what blockchain context we can expect t to perform correctly.

Understood this way, the blockchain interpretation gives new meaning and perspective on some theorems from the linear logic literature. In particular, it is well established that there is a natural notion of execution of t ’s. That is, when thought of as programs, we know how to run them. When they are well typed, that is, if we have established $\vdash t : A$, then t is *terminating*. That’s a theorem from Abramsky’s computational interpretations of linear logic. What this means for the blockchain is that proof terms and their linear connectives provide a scripting language for transactions that, on the one hand, provides termination for all well typed scripts, and on the other is highly expressive. Further, if it turns out that this scripting language is not expressive enough, then there is a natural extension of proof terms via a correspondence between linear proof terms and π -calculus processes that we mentioned at the top of these notes.

proof term	blockchain meaning
address	address
$t * u$	isolated concurrent transactions
$t \wp u$	interacting or linked concurrent transactions
$t \& u$	menu of transaction options
$inl(t), inr(u)$	transaction option selection
$!t$	copyable resource server
$?t$	copyable resource storage
$txn(t, u)$	joined transactions

This correspondence is not just useful for extending a scripting language for blockchain transactions. It turns out the π -calculus the premier formalism for specifying, reasoning about, and executing protocols in distributed systems. Since one of the real values of the blockchain is the fact that it is a distributed means to conduct transactions, the need to tie this formalism to one for specifying protocols in distributed systems is plain.

3.1 Proof-of-work

The glaring lacunae in this discussion is, of course, the relationship to proof-of-work. Consider the following example. Suppose C_1 and C_2 are blockchains both of height N .

$$C_1 = B_{1N} \leftarrow B_{1N-1} \leftarrow \dots \leftarrow B_{10}$$

$$C_2 = B_{2N} \leftarrow B_{2N-1} \leftarrow \dots \leftarrow B_{20}$$

We can define

$$C_1 \otimes C_2 = (B_{1N} \otimes B_{2N}) \leftarrow (B_{1N-1} * B_{2N-1}) \leftarrow \dots \leftarrow (B_{10} * B_{20})$$

Note that it is insufficient merely to guarantee for $B \otimes B'$ that all the transactions in B are isolated from the transactions in B' . The counterexample is

$$C_1 = \text{Block}\{1\text{AliceAddr} \xrightarrow{5\text{btc}} 1\text{AllanAddr}\} \leftarrow \text{Block}\{1\text{BobAddr} \xrightarrow{7\text{btc}} \dots\}$$

$$C_2 = \text{Block}\{1\text{BobAddr} \xrightarrow{7\text{btc}} 1\text{BettyAddr}\} \leftarrow \text{Block}\{1\text{AliceAddr} \xrightarrow{5\text{btc}} \dots\}$$

Clearly B_{11} is isolated from B_{21} , and B_{10} is isolated from B_{20} ; but, B_{20} is not isolated from B_{11} , and B_{10} is not isolated from B_{21} . As a result, the spends in the earlier blocks could impact the spends in the later blocks.

Instead, the entire address space of C_1 must be isolated from C_2 . In this case the network of servers, N_1 , that maintain C_1 can be safely combined with the network of servers, N_2 , that maintain C_2 , and we can safely define the composite chain as above. The proof-of-work protocol organizing N_1 is completely separate

from that in N_2 . They do not interact. Yet, it is safe to combine the chains using a glorified zip function. In this example,

$$C_1 = \text{Block}\{1\text{AliceAddr} \xrightarrow{5\text{btc}} 1\text{AllanAddr}\} \leftarrow \text{Block}\{2\text{BobAddr} \xrightarrow{7\text{btc}} 2\text{BettyAddr}\} C_2 = \text{Block}\{1\text{BobAddr} \xrightarrow{7\text{btc}} 1\text{BettyAddr}\} \leftarrow$$

The address spaces of these chains are completely isolated (often written $\text{addresses}(C_1) \# \text{addresses}(C_2)$). We are free to calculate

$$\begin{aligned} & C_1 \otimes C_2 \\ &= \\ & \text{Block}\{1\text{AliceAddr} \xrightarrow{5\text{btc}} 1\text{AllanAddr}\} \otimes \text{Block}\{1\text{BobAddr} \xrightarrow{7\text{btc}} 1\text{BettyAddr}\} \\ & \leftarrow \text{Block}\{2\text{BobAddr} \xrightarrow{7\text{btc}} 2\text{BettyAddr}\} \otimes \text{Block}\{2\text{AliceAddr} \xrightarrow{5\text{btc}} 2\text{AllanAddr}\} \\ & \text{Block}\{1\text{AliceAddr} \xrightarrow{5\text{btc}} 1\text{AllanAddr}; 1\text{BobAddr} \xrightarrow{7\text{btc}} 1\text{BettyAddr}\} \\ & \leftarrow \text{Block}\{2\text{BobAddr} \xrightarrow{7\text{btc}} 2\text{BettyAddr}; 2\text{AliceAddr} \xrightarrow{5\text{btc}} 2\text{AllanAddr}\} \end{aligned}$$

The ordering of transactions provided by the two independently executing proof-of-work protocols is combined in a completely safe.

Note that there are at least two possible interpretations of $C_1 * C_2$. One is that the requirement is to verify that $\text{addresses}(C_1) \# \text{addresses}(C_2)$. Another is to ensure this is the case by rewiring the transactions. Under this latter interpretation even the counterexample becomes safe

$$\begin{aligned} & C_1 \otimes C_2 \\ &= \\ & \text{Block}\{1\text{AliceAddr} \xrightarrow{5\text{btc}} 1\text{AllanAddr}\} \otimes \text{Block}\{1\text{BobAddr} \xrightarrow{7\text{btc}} 1\text{BettyAddr}\} \\ & < - \text{Block}\{1\text{BobAddr} \xrightarrow{7\text{btc}} 01\text{BettyAddr}\} * \text{Block}\{1\text{AliceAddr} \xrightarrow{5\text{btc}} 01\text{AllanAddr}\} \\ &= \\ & \text{Block}\{01\text{AliceAddr} \xrightarrow{5\text{btc}} 01\text{AllanAddr}; 11\text{BobAddr} \xrightarrow{7\text{btc}} 11\text{BettyAddr}\} \\ & \leftarrow \text{Block}\{01\text{BobAddr} \xrightarrow{7\text{btc}} 01\text{BettyAddr}; 11\text{AliceAddr} \xrightarrow{5\text{btc}} 11\text{AllanAddr}\} \end{aligned}$$

There is much more to be said, but that must be left to a later revision!

Acknowledgments. We would like to acknowledge Vlad Zamfir for asking some thoughtful and stimulating conversation about the blockchain protocol.