Financial product specification and trading via a blockchain

orbifold.io

22 November 2017

Abstract

An outline is given for a financial product specification language that uses blockchain confirmation. Various implementation and commercial challenges are discussed. The proposed system allows for a broad class of capital markets instruments to be defined and traded.

1 Introduction

Recent interest in blockchains has provided for a natural focus on formal financial contract specification (see [JES00], [A06] and [F09] as examples of literature about formal contract specification and [H12] for some regulatory input). The advent of Bitcoin provides a concrete reference point on which to base any discussion of contract specification. A key realization is that the blockchain, alongside being a public ledger evidencing no double spend, can itself be used to construct further derivative tokens. For example if tokens a and b are being exchanged on the blockchain, the 'market rate' between two tokens a and b can be defined as a function of time. Embedding that functional definition into the specification of a further token c, creates a token that can be traded now and whose value is contingent on the future market rates of exchange between a and b. By further extending the definition space of tokens to include basic mathematical constructions (e.g. $x \mapsto max(x, 0)$ etc) tokens with option payoff functions can be defined and in this way crypto-cash settled derivatives replicated.

Settlement can be enforced on the nodes in the network but, of course, default cannot be eliminated. If c is a token that creates an obligation for coin-address[1] α to pay one base token a to coin-address β at a future point in time t, then the implementing network automatically confirms the transaction onto the blockchain at point in time t. If α no longer has any as then the confirmation is rejected. Below it is proposed that the blockchain confirmation process is extended to record the rejections that arise from the failure to meet an

[1] A coin-address should be viewed as representing a node in a network implementing a blockchain based trading system; akin to a counterparty

obligation. Recording such rejections can be used as the basis for credit default contracts; i.e. tokens that pay out when a specified coin-address fails to meet its obligations. However, establishing the relationship between a coin-address and a legal entity that has a commercial incentive to meet the obligations of the address is an area that has an inherent conflict. This is because no legal entity would want a list of all of its coin-addresses publicly known as then the full financial crypto-risk profile of the legal entity would be known. On the other hand it would be hard to see how any coin-address could successfully trade commercially without its relationship to a legal entity being known. Potential mitigants are discussed below.

The purpose of this paper is to provide a broad specification for a blockchain based financial products token trading system and to evaluate at a high level the various challenges that will need to be addressed for any eventual commercial or widespread retail adoption. Whilst implementation is some way off, the language has been designed with the following markets in mind: money market loans and deposits, FX forwards, interest rate swaps, exchange traded futures and options, securitizations and OTC derivatives.

The paper is structured as follows. The first three sections define the proposed language of tradeable tokens and describe the blockchain confirmation process. The specification is reasonably complete, with the exception of signing protocols that are only outlined in the text. The remaining sections discuss design challenges, focusing on the blockchain 'memory' challenge and credit.

2 Tokens

The collection of Tokens is defined iteratively. Controlling the complexity of the iteration is a key design consideration as the blockchain confirmation process must parse and verify the operational soundness of each Token. The Token constructors are made as simple as possible focusing on the core components of any financial contract which are timed functional payouts, options and securities.

Definition of Token

Tokens are are built up iteratively, using the following rules from Base Tokens, Value Functions and Time Lag Functions (jointly, Functions), terms which are all defined later:

1. A Base Token is a Token.

2. If a_i are Tokens, $i = 1, ..., n$, then $\{a_1, ...a_n\}$ is a Token (i.e. a list of Tokens is a Token).

3. For any Value Function F, Time Lag Function Lag, timestamp t and Token a then the 4-tuple (F, Lag, t, a) is a Token. Meaning: the owner of (F, Lag, t, a) receives $F(t)$ units of a at point in time $Lag(t)$. Any Token of this

form is a *Function* Token. The Token is evaluated at time t and paid at time $Lag(t)$

4. If t_1 and t_2 are two timestamps, x_i a list of non-zero real numbers[2] and a_i a list of Tokens (both of length n) then $\lambda^{t_1,t_2}[(x_1,a_1),...,(x_n,a_n)]$ is a Token; i.e. a Token that consists of a pair of time points together with a list of tokens, each with an amount. This Token gives the owner the right but not the obligation to receive x_i units of a_i for every $x_i > 0$ in exchange for paying $-x_i$ units of a_i for every $x_i < 0$ at any point in time between t_1 and t_2 (once only). Any Token that starts with a λ is an *Option* Token.

5. If a is a Token and γ is a coin-address then $\mathcal{S}_i^\gamma a$ is a Token where $i = 0, 1$ or 2; i.e. a Token that consists of Token and coin-address pair together with a flag i. The meaning of $\mathcal{S}_i^\gamma a$ is that γ pays and receives any crypto-cashflow of a evaluated at time t to/from whoever owns $\mathcal{S}_i^\gamma a$ at point in time t. This is a *Security* Token. The integer i can be $0, 1$ or 2. Setting $i = 0$ ($i = 1$) means that the security balance can never go negative (positive). Setting $i = 2$ means no restriction.

2.1 Base Tokens

A Base Token represents a crypto-currency, tied to a particular coin-address. The coin-address can redeem Base Tokens by agreeing a transaction that returns the issuance to them. A legal entity with commercial credibility will have an incentive to offer crypto-USD, crypto-GBP etc on the basis of one-to-one redemption with the real (fiat) currency. Having the relevant central bank take the step would further eliminate risk, but it is probably more realistic to envisage that a commercial entity could issue, for example, crypto-USD. A commercial entity may issue a Base Token at will. However, if it issues without the backing of a client deposit in fiat currency then it would be creating an unbacked liability; its own accounting standards and regulatory constraints provide a level of assurance that crypto-currencies would be appropriately backed. So, in summary, it is not unreasonable to assess as possible the idea that key mainstream banking entities' crypto-currency issuances could be valued on a par with real currencies such as USD, EUR or GBP.

One can further envisage Base Tokens being backed by equity or bond securities, allowing for crypto-derivatives on these further asset classes. In these cases having a clearing house issue the crypto-equity or bond may provide clearer legal ownership line of sight through existing mechanisms of account segregation[3]. Onward trading of Base Tokens on a blockchain allows for ownership at any point in time to be recorded, providing for the crypto-settlement of any

[2] e.g. implemented as DOUBLE

[3] Determining the necessary legal framework would need to be thought through in both the cash and the security case; this remains as further work and has not been explored as part of the analysis here.

dividends or coupons. The data of the Base Token can contain the hash of the issuing prospectus so that the legal obligations of the issuer can be confirmed.

The underlying data of a Base Token consists of the coin-address of the issuer, an identifier, and some text (e.g. 'PAY BEARER ONE USD'; or a hash of more complex legal text, as required).

2.2 Blockchain

The definition of Functions, to follow, will build on the blockchain and so we now clarify what the blockchain will look like. Ignoring the blockchain proof of work steps, the underlying data of the blockchain is a list of timed input and output Token exchanges between coin-addresses. So, omitting any tree structure or nonces, the blockchain records a list of transactions each one of the following form:

$$...|t,(x_1,a_1),...,(x_n,a_n),[\alpha,\beta],Fail|...$$

Here t is the timestamp of the confirmation, the x_is are non-zero real numbers $(i=1,...,n)$, the a_is are Tokens and α and β are coin addresses. The meaning of the transaction is that at time t the blockchain is confirming that for every i coin-address $\alpha(\beta)$ is receiving (paying) x_i units of token a_i from (to) coin-address β if x_i is positive (negative). In other words, a transaction is a list of amounts of Tokens exchanged between two coin addresses. The final item, $Fail$ is a list containing either α or β or both or empty. It records transactions where α (or β or both) overspend a Security or Base Token.

A list of transactions is only considered settled by the blockchain going forward if $Fail$ is empty. Duplicate a_is are either netted or not allowed in the string in the first place.

Tokens that are lists of Tokens are confirmed in the blockchain as lists of transactions with the obvious mapping; i.e. $(x,\{a,b\}) \mapsto (x,a),(x,b)$, so that agreeing to pay x units of the list a, b is settled by paying x units of a and x units of b. This is done iteratively, so that lists of lists are reduced to lists of tokens none of which are themselves lists. Similarly for Securities whose underlying asset is a list: $(x,\mathcal{S}_i^\gamma\{a,b\}) \mapsto (x,\mathcal{S}_i^\gamma a),(x,\mathcal{S}_i^\gamma b)$ etc.

2.3 Ledger

The blockchain uniquely determines a ledger at any point in time. For every Token and coin-address pair (a,α), at any point of time t, there is a balance $Bal(a,\alpha,t)$, which is the amount of a that α has settled on the blockchain up to any confirmation timestamped t. Any network node implementing our approach will reject any confirmation that takes a coin address' Base Tokens below zero, with the exception of a coin address' own Base Token. Security Tokens also cannot go below (above) zero if the flag i is set to 0(1).

2.4 Function

We now specify the rules for constructing Functions. These rules effectively define the collection of methods that can be applied to derive the payouts of Function Tokens (both the payout amount and the timing of the payout). The collection of methods that results is not Turing complete as Functions are defined iteratively and without loops.

To provide context we assume the following base types/classes and constructors on types/classes are available:

Base types/classes: $BOOL$, R (reals; e.g. double, float etc), T (timestamp), Tok (tokens) and CA. We use CA for the class 'coin-address'; this class is effectively a $TEXT$ type, with some checksum or similar test validating text as a coin-address.

Basic constructors. For any two classes A and B, $A \times B$ is a class (pairs). A LIST structure can operate on any class.

Definition of Function: Base Functions split into three types: 1. mathematical and logical, 2. blockchain lookup and 3. external URI lookup. By definition something is a Function if it can be built up iteratively from Base Functions and the constructors accepting that all Functions may look up the blockchain, or external URIs.

Value Functions are $T \longrightarrow R$; i.e. one free variable of type T, with output a real. Time Lag Functions are $T \longrightarrow T$.

As an implementation point we observe that our expectation is that Functions will be recorded within Tokens as the hash of the code recording the function. A node will deem a Token invalid if it does not have in its memory function code that hashes to each function hash in a token. Of course, Function code will be deemed invalid if it is not well typed.

Basic Mathematical and Logical

E.g.

1. $+ : R \times R \longrightarrow R$, $* : R \times R \longrightarrow R$, $\div : R \times R \longrightarrow R_{\perp}$, $max : R \times R \longrightarrow R$, $min : R \times R \longrightarrow R$, $ABS : R \longrightarrow R^{+}$, $exp : R \longrightarrow R^{+}$ and $ln : R \longrightarrow R$.

2. $\geq, \leq : T \times T \longrightarrow BOOL$ and $\geq, \leq : R \times R \longrightarrow BOOL$.

3. $\neq, = : A \times A \longrightarrow BOOL$ and $\neq, = : A \times A \longrightarrow BOOL$ for any type A.

4. $BOOL$s; i.e. $\neg$, AND, OR etc.

Blockchain Lookup

This function extracts transactions (i.e. outputs a list of blocks) given a pair of timestamps. The output is a list of confirmed transactions on the blockchain which have been confirmed between the timestamps.

By projecting down components of the output, and applying mathematical and logical Functions, it should be clear how to write out (non-uniquely) a Function $Market_{a,b} : T \longrightarrow R$ for any two tokens a and b. There are a great many design choices that could go into the definition of a market function; the period of time before or after t that is considered, the method of averaging, size constraints, sidedness (i.e. bid, offer or mid) and counterparty constraints (i.e. constraints on which coin-addresses transactions should be considered in any average).

Given that a great many different choices can be made for a market function, there is a risk that as markets evolve different conventions come and go. This means that a longer dated crypto-derivative may settle to a market that is no longer active. Aside from the 'cancel' feature, introduced below, we have not been able think of any design feature that substantially reduces this risk. Allowing for some sort of equivalence relationship to be established between the various functional forms that $Market_{a,b}$ could take and embedding that definition into Tok would greatly increase the complexity of the language. In practice existing derivatives markets do navigate these issues, usually with some sort of basis between new and old conventions being liquid for a period whilst contracts are moved to new market conventions; a similar process could work in our context.

The blockchain lookup function, as written above, may be too greedy for implementation. Practically a Token list and coin-address list would be required as input with the output only consisting of transactions that contain the Tokens that are on the coin-address list. Since any confirmed Token with a blockchain lookup function will force all implementing nodes to record the results of any lookup, getting this right is a key challenge; the paragraph 'Key implementation challenge' below discusses this further.

External Lookup

This function takes as input a URI, e.g. a webpage, and returns a real. For example it could look up the publication of a LIBOR rate or the close of an exchange traded futures from a reputable website. This part of the definition of Function allows the blockchain to settle transactions based on external data. The reliability of any Token based on such a lookup is of course dependent on the reliability of the URI.

3 Implementing Tokens on a blockchain

In summary, blockchain implementation proceeds as usual but with additional rules that automate the settlement of future obligations as they fall due.

If two coin-addresses, α and β, both sign $(x_1, a_1), ..., (x_n, a_n)[\alpha, \beta]$ then the network will add it to the blockchain for validation through proof of work. This can be done following a similar method as the existing Bitcoin mechanism. The difference here is that (i) there are a number of potentially different Tokens

being exchanged rather than having the same crypto-token (i.e."BTC") within the transaction; and, (ii) there are two counterparties that need to sign.

The additional rules are as follows:

1. Any network node, which reaches a time stamp $t' \geq t$ adds a new block (or blocks) for confirmation based on the following schema:

$$\begin{aligned} ...(x,(F,Lag,t,a))...[\alpha,\beta] &\mapsto (xF(t),a)[\alpha,\beta] \\ ...(x,\mathcal{S}_i^{\gamma}(F,Lag,t,a))...[\alpha,\beta] &\mapsto (xF(t),a)[\alpha,\gamma] \text{ if } x>0 \\ ...(x,\mathcal{S}_i^{\gamma}(F,Lag,t,a))...[\alpha,\beta] &\mapsto (xF(t),a)[\gamma,\beta] \text{ if } x<0 \end{aligned}$$

The real number $F(t)$ is calculated by evaluating the blockchain, up to and including any blocks timestamped t. The meaning of the schema is that if the left hand side has been confirmed in the blockchain and not spent then the right hand is pushed for confirmation by each node once a point in time t_0 with $t_0 \geq Lag(t)$ is reached. The Lag here is similar to the Locktime feature that exists in Bitcoin.

2. (i) If at time t, with $t_1 \leq t \leq t_2$, a node sees that

(a) coin-address α has signed $(xx_1,a_1),...,(xx_n,a_n)[\alpha,\beta]$,

(b) $...(x,\lambda^{t_1,t_2}[(x_1,a_1),...,(x_n,a_n)])...[\alpha,\beta]$

(respectively $...(x,\mathcal{S}_i^{\gamma}\lambda^{t_1,t_2}[(x_1,a_1),...,(x_n,a_n)])...[\alpha,\beta]$)

have already been confirmed on the blockchain and not spent; and,

(c) $x>0$,

then the node confirms $(xx_1,a_1),...,(xx_n,a_n)[\alpha,\beta]$ (respectively $(xx_1,a_1),...,(xx_n,a_n)[\alpha,\gamma]$) without requiring β (respectively γ) to sign. The Option or Security Token are then marked as spent.

2. (ii) Same as 2.(i) but with β switched with α and $x<0$ in place of $x>0$.

In other words option exercise is achieved by one counterparty (the option owner) signing the transaction that it has the right to, and each implementing node accepting this transaction for confirmation without requiring the option seller to sign the transaction.

3. For 1,2(i) and 2(ii) the flag $Fail$ in the confirmation block records any overspend by either address. If $Fail$ is non-empty each node waits a system wide determined constant (24 hours suggested) before trying to re-confirm the transaction. This is repeated up to a system determined constant (30 suggested).

4 Key implementation challenge

How much information can we put on the blockchain? Removing Turing completeness helps - everything is a sentence, or a list of sentences - there are no loops. We are further taking as read that spent transactions are removed from a node's memory (any transaction that changes a balance of a Token sends the change to its owner; the old transaction can be discarded).

Our design is an attempt to make the sentences reasonably straight forward for commercial application. The Token space can expand, but whilst there may be very large numbers of distinct Tokens constructed it seems reasonable to assert that there will not be large numbers of transactions in large numbers of Tokens. Liquidity will only go to a few key Tokens.

The main implementation challenge is controlling blockchain lookup functionality. Firstly, it is probably unfeasible for the blockchain to keep a record of all transactions and so allow a new blockchain lookup to go into the past. But, once a block has been confirmed which requires lookup of blocks between times t_1 and t_2 say, then this data must somehow be recorded by each implementing node. So, as time passes, the nodes must evaluate and store the answer for any Function component within a confirmed Token as it falls due for calculation. If the amount of data captured in a blockchain lookup is excessive and the length of time between t_1 and t_2 is long, then memory requirements could rise. For example, consider an Asian option whose payout is the average of a market rate taken every second for a ten year period. The solution may be dynamic functional evaluation; the average is calculated as you go along and so the data storage requirement drops. However working out whether this is sufficient to control the memory requirement of any implementing node is probably the most pressing technical challenge for any implementation.

5 Controlling onward transfers

A Security Token could be an asset or a liability. If it is a liability, controlling onward sale is important to ensure that it is not transferred to a coin-address with low credit. As a further enhancement the definition of a Security Token can be expanded from $\mathcal{S}_i^\gamma a$ to $\mathcal{S}_{i,\mathcal{A}}^\gamma a$ with two proposals for $\mathcal{A}$: (i) it is a static list of acceptable coin-addresses, or (ii) it is a list of coin-addresses controlled by γ. The confirmation rules of the blockchain are enhanced to only settle a Security Token if it is from/to an address in $\mathcal{A}$. This could also allow for the implementation of rehypothecation clauses in collateralisation contracts, such as credit support annexes.

6 Credit Events

Informally, a credit event occurs when a coin-address does not have enough Tokens to meet an obligation. To encode this event into a further Token, extract from the blockchain (for a period t_1 to t_2 say) the obligations of a coin-address to deliver Tokens as they fall due. For each obligation, wait for a period (e.g. 30 days) and verify that the obligation has been settled on the blockchain. The flag F determines overspend which can be used to excluded 'de-minimus' failures to pay.

7 Credit

A coin-address can only deliver a token if it has the token. How can any participant know whether a coin-adress will deliver? Unfortunately establishing the relationship between a coin-address and a legal entity that has a commercial incentive to meet the obligations of the address is an area that has an inherent conflict. There is a tension between a counterparty wanting to have its credit known to get deals done but not wanting its risk profile to be known.

There are two potential mitigants: 1. collateralisation and 2. partial disclosure of coin-addresses by legal entities.

1. *Collateralisation*

Alongside the agreement to exchange a token a for b at some future point in time t, two coin-addresses could also agree to exchange collateral, based on, say, the daily market exchange rate (for the exchange of a for b at time t). Embedding a market rate of exchange into a token's definition could enforce continual (say daily) collateralisation. However there is the challenge of defining the market function used to settle any collateral obligation. Counterparties would want to tie collateral obligations into the life of any contract and so would have to agree upfront mark-to-market methodologies for the collateral calls. Whilst theoretically the language stipulated here would allow for this for all Tokens, it seems much more reasonable to assess that collateralisation would be initially done on a Token by Token basis rather than on products whose valuation is based on multiple underlyings. In other words, encoding collateral as is done for an exchange traded futures contract should be straightforward, but doing the same for a vanilla OTC swap is more challenging as it would require upfront agreement on market source for each of the multiple pricing inputs that go into a swaps curve as well as the pricing methodology applied.

2. *Partial disclosure of coin-addresses*

A legal entity could disclose some of its nodes and so its risk profile would be less transparent. It could then move risk to coin addresses that are not publicly associated with the legal entity. Consideration could also be given for regulators to deem certain coin addresses as 'regulated' without disclosing the legal entity involved.

8 Pending, booked and cancelled

An additional flag can be added to each transaction and recorded in the confirmation process:

$$...|t, (x_1, a_1), ..., (x_n, a_n), [\alpha, \beta], F, S|...$$

where S is text: either 'P' (pending), 'B' (booked), or; 'C' (cancelled). Each node proceeds as usual ignores any 'P's and only acts on a given 'B' provided there is no instance of the same transaction (but with 'C' in place of

'B') further along the blockchain. The cancellation only impacts future crypto-cashflows/options as they fall due; it does not reverse exchanges that have already occurred. In other words it is only steps 1, 2(i) and 2(ii) of Section 3 that are adapted to check for 'C'. The benefit is that this implementation is more in keeping with current commercial practice and allows for counterparties to withdraw from settlement if both counterparties agree that the blockchain implementation of the settlement amounts due does not coincide with their joint commercial expectation. In other words, having the ability to cancel provides a get out, provided both counterparties agree.

9 Quotes

In implementation the communication protocol between nodes could allow for any coin-address to sign a transaction without a counterparty coin-address or with a list of 'acceptable' coin-address counterparties. The protocols could then allow any coin-address (or coin-address on the list) to sign the transaction. This would allow any coin-address to offer Tokens (or options to exchange Tokens etc) to the market without a specific counterparty and so would allow participants to distribute executable quotes.

10 Market abuse

We have not been able to imagine any operational mechanism that could rule out market abuse without introducing some third-party rule keeper. If two coin-addresses conspire to trade as for bs at an off-market rate and a $Market_{a,b}$ Function used to settle a crypto-derivative is not sufficiently robust to eliminate the trade from its average, then the off-market trading could contribute to the settlement of a crypto-derivative in a way that is not commercially reasonably. Potential additional controls are: (i) monitoring of coin-addresses for off-market transactions on the blockchain, (ii) include in each new transaction (the hash of) legal text asserting that both counterparties believe that the transaction is at market rates; and, (iii) public identification of legal entities with coin-addresses allowing their commercial credibility to be associated with transactions.

11 Implementing real numbers

From a commercial perspective getting the final answers to calculations right to more than 5 or 6 decimal places is probably not a key issue provided the dollar value of any error is demonstrably small. But since our proposal has no absolute dollar value concept, rounding to a predetermined dollar amount is not possible. On the other hand, even with very high implementing standards it is known that small differences can creep in to calculations on the reals. This means that implementing nodes will not agree on amounts to be settled if the level of accuracy is set too high and this would mean that the blockchain

consensus mechanism could fail. It is expected that the only practical way round this is to accept something like 6 decimal place accuracy on any Function output and caution users of any very high value tokens.

12 Conclusion

It is not claimed that the ideas presented here are all new. There is a lot of material on the internet showing that something along these lines is being considered by a number of developers; relevant Google search terms include 'smart contract', 'ethereum', 'clearmatics' and 'hedgy'.

In this paper we have identify areas where further work is required before a liquid crypto-financial derivative market emerges. However what is clear is that establishing a sensible prototype is well within reach. The core option and security constructors needed to create a language of trade-able tokens can be readily defined and in this way a broad class of financial instruments specified. By allowing tokens to look up the blockchain as it evolves cash settled derivatives can be naturally defined and this is an innovation that could have widespread implications should such a system be commercially adopted. We have described how a structured token language in conjunction with a simple extension of the blockchain confirmation protocol should allow for the creation of a capital markets trading system - freely available to any implementing node.

References

[A06] Andersen, Jesper, et al."Compositional specification of commercial contracts." International Journal on Software Tools for Technology Transfer 8.6 (2006): 485-516.

[F09] Frankau, Simon, et al. "Commercial uses: Going functional on exotic trades." Journal of Functional Programming 19.01 (2009): 27-45.

[H12] Haldane, Andrew. "Towards a common financial language." Speech to the Securities Industry and Financial Markets Association Symposium on Building a Global Legal Entity Identifier Framework, New York. Vol. 14. 2012.

[JES00] Jones, Simon Peyton, Jean-Marc Eber, and Julian Seward. "Composing contracts: an adventure in financial engineering(functional pearl)." ACM SIGPLAN NOTICES 35.9 (2000): 280-292. APA