FIRMO: Secure Execution of Financial Contracts

FIRMO, 12 April 2018

Abstract

The Firmo protocol executes financial contracts on blockchain technology. Contracts are written using the domain specific language; FirmoLang. FirmoLang provides a secure execution environment for financial contracts on decentralized asset-classes. Similar to software built for airplane coordination, FirmoLang is formally verified, yielding needed security benefits for smart contracts in finance. FirmoLang compiles directly to Ethereum Virtual Machine bytecode. The Firmo Protocol is designed to integrate with and support the advancing decentralized economy, including: Decentralized exchanges, p2p lending platforms, prediction market platforms and more.

The Firmo Protocol is currently in the alpha phase of development. This document:

1. Introduces the Firmo protocol and the objective behind introducing financial contracts to the blockchain.
2. Provides a non-specific overview of the domain specific, domain specific programming language: FirmoLang.
3. Introduces the FirmoLang Ethereum Virtual Machine Compiler.

Please be advised: The Firmo protocol is in constant development, please find the most recent version of this publication on: www.firmo.network. Please contact the authors at omri@firmo.network and johannes@firmo.network with any comments or suggestions

Content:

1.0 Introduction

Bitcoin introduced the concept of a decentralized, digital currency, paving the way for the social acceptance of a decentralized monetary system. [1] Ethereum introduced the idea of extending the decentralized monetary system with a virtual machine, supporting smart contracts on blockchain technology. [2] Firmo attempts to augment the development of a decentralized economy by introducing secure and automated financial contracts without the involvement of trusted third parties.

Firmo is in the process of creating an alternative infrastructure for financial contracts [which is/will be] transparent, fair and trustless. In our view, the application of financial services is a particularly interesting use case for distributed ledgers. Financial assets are abstract products, deriving their value from the ongoing price-discovery mechanism on the markets. Indeed, financial operations are essentially a series of transfers of value between counterparties, at selected time intervals, governed by certain conditions.

Arguably, the majority of financial processes in today's markets lend themselves to disintermediation and automation in smart contracts or representation of value through tokenization. Expressive Turing complete[1] programming languages offer a variety of benefits for formulating complex transfers of value in smart contracts. Such is the case for public infrastructure with native scripting languages, where users are building sophisticated business models using distributed ledger technology.

However, as emphasized in the Rice theorem, [3] expressiveness comes at a price. Smart contracts written in advanced programming languages frequently suffer from bugs, even when the code has been fully audited. In fact, [4] have found that 45 percent of the smart contracts examined from a set of over 19,000 contracts deployed on the Ethereum mainnet contained a subset of known exploits or vulnerabilities.

When mistakes happen in the conventional financial markets, we trust our service providers to resolve the issue. When mistakes occur in the construction of smart contracts, a technology that is supported by a vast network of nodes in consensus, resolving issues is a complicated governance process. [5] FirmoLangis designed to be simple and secure. FirmoLang is formally verified, allowing tools for proof assistance at compile-time. This, in our view, is the most efficient way to produce secure code. FirmoLang is built on the insight that all financial instruments are reducible to the same triad of events: A series of transfers of value between counterparts, at selected time intervals, governed by a set of conditions like the price on the market or a counterparty calling an option.

[1] A programming language is considered Turing complete, when it is able to simulate a *Turing Machine,* that is,

The Firmo Protocol enables users to create what we call smart derivatives, a secure type of smart financial contract, offering products including but not limited to, futures, loans, options and swaps. The Firmo Protocol is designed to operate with a diverse selection of external liquidity providers and exchanges. Smart derivatives are non-fungible and can be traded on any regulated exchange which supports the Firmo Protocol. With the introduction of the Firmo Protocol, Firmo is paving the way for the logical evolution of the emerging crypto markets.

2.0 The Firmo Protocol Stack

The Firmo Protocol enables any business or individual user to create secure and automated smart derivatives using the domain specific programming language; FirmoLang. Any smart derivative or alternative financial contract that uses FirmoLang can be compiled to the Ethereum Virtual Machine (EVM). The Firmo Protocol Stack consists of 4 general groups of processes which are set out in the table below:

Fig. 2.a The Firmo Protocol Stack

Any person or entity that is tokenizing an asset class using an ERC20 format, can utilize the Firmo Protocol to create secure smart derivatives or other financial instruments in the proprietary programming language; FirmoLang. Firmolang is a versatile programming language with a broad scope of applications. This makes the Firmo Protocol an obvious choice for cryptocurrency exchanges, financial institutions or individual users looking to create secure and automated financial contracts on distributed infrastructure.

A. **Individual Users:** Can enter into smart derivative contracts and trade their positions to third parties on secondary markets..

B. **Cryptocurrency Exchanges:** Can offer their customers secure derivative products. Cryptocurrency exchanges may also act as the clearing institution or counterparty in a given smart derivative contract.

C. **Financial Institutions:** Traditional financial institutions can utilize the Firmo Protocol to achieve significant reductions in costs associated with the clearing and settlement of derivatives. By eliminating intermediaries and automating clearing processes, financial institutions can further expand their offering with exotic variations of commonly standardized instruments.

2.1 Composition of a smart derivative

All smart derivatives are formulated in the formally verified language, FirmoLang. Initially, Firmo will offer two solutions for users to engage with the Protocol:

A. **Deployment of Template Contracts:** In collaboration with partnering exchanges or other platforms, Firmo will offer a set of template contracts. The template contract will be written in FirmoLang and customized to the service offered by the partner institution. In some cases this might be future contracts, in other it might be forward contracts for peer-to-peer loans or even prediction markets.

B. **Deployment of Customized Contracts:** The Firmo Protocol allows users to formulate customized smart derivatives in FirmoLang by navigating to the console in the user interface. Here, technically adept users can inspect the code or create customized contracts whenever needed. This makes

2.2 The Firmo EVM Compiler

FirmoLang compiles directly to EVM, through the FirmoLang Compiler. The FirmoLang Compiler outputs bytecode format through intermediate representation as described in section 9.0.

2.3 The Ethereum Blockchain

Smart derivatives are executed on the Ethereum Blockchain in the EVM, through the use of the ERC20 compliant methods described in section 9.0.

3.0 The Forward Contract

This section introduces the initial use case for the Firmo Protocol, available in the beta version: The forward contract. The Forward Contract gives the buyer the *obligation* to buy an underlying asset at a specified time in the future, with *physical settlement* of the underlying asset. The Forward Contract is traditionally used for hedging against perceived risk in volatile commodity markets. Forward contracts are often used both to secure supply of essential raw material for the production of consumer goods or to secure a certain currency rate at the market for international trade in the Forex markets. Today, the closely related futures contract (where only the price difference, not the underlying asset is settled) are issued on publicly listed exchanges as NASDAQ, the CME and the CBOE.

Fig. 3.a A forward contract as a smart derivative

3.1 FirmoLang Definition

The Forward contract is defined in FirmoLang as the following:

```
translate(
    days(30),
    both(
          scale(
              50,
              50,
              transfer(BNT address, alice, bob)
          ),
          scale(
              400,
              400,
              transfer(GNT address, bob, alice)
          )
      )
)
```

4.0 Community Use Cases

FirmoLang is a versatile language with an intuitive syntax. This section describes two potential use cases for the Firmo Protocol for how exchanges or other service providers can use the Firmo Protocol to enhance their offering to their user base.

4.1 The European Vanilla Option

The European vanilla option gives the buyer the option, not the obligation, to buy an asset at a set price at a predetermined date. Options are frequently used to hedge or speculate on volatile assets in various industries. When counterparties engage in a call option defined as a smart derivative, the buyer pays the seller a premium. Depending on the perceived valuation in t2, the buyer can trade the option on an exchange, for an additional premium. Should the price of the underlying asset not appreciate above the buyer's initial price, including the premium, at the time of maturity, the smart derivative can automatically close the position, unless other instructions have been made.

Fig. 4.a An Option contract as a smart derivative

4.1.1 FirmoLang Definition

An example of an `option` contract defined in FirmoLang could be the following:

```
translate(
    months(3),
    scale(
        100,
        max(obs(int, datafeed address, BNT) - 3, 0),
        transfer(BNT contract address, alice, bob)
    )
)
```

4.2 Vanilla Interest rate swap

An interest rate swap is a financial instrument, in which borrower and lender exchanges the respectable interest rates paid on a nominal amount. The differential value between the two interest rates can be swapped in multiple ways, with variable maturity periods. The exchange of value in a plain vanilla interest rate swap can be expressed as:

Let D be the differential, where F denotes the notional loan amount or value of a bond, where C is the rate associated with period t. R defines the external data feed queried from a user defined exchange or the LIBOR rate for conventional financial contracts. The value of the swap can be reversed according to the position. When defined in FirmoLang, recurring differential payments on the two positions will be automated according to the user defined period t. As all payments are automated and executed on chain, multi party contracts can be automated to execute at minimal intervals without compromising safety or financial integrity of the clearing and settlement processes.

Fig. 4.b An IR swap as a smart derivative

4.2.2 FirmoLang Definition

An interest rate swap defined in FirmoLang can potentially be formulated as the following:

```
translate(
      days(30),
      if((fixed rate * principal amount) > (obs(int, df address,
position) * principal amount)) within now
            then
                  scale(
                        upper limit,
                        ((fixed rate * principal amount) - (obs(int, df
address, position) * principal amount)),
                        transfer(currency, alice, bob)
                  )
            else
                  scale(
                        2nd upper limit,
                        ((obs(int, df address, position) * principal
amount) - (fixed rate * principal amount)),
                        transfer(currency, bob, alice)
                  )
)
```

5.0 Collateral Accounts and Margins

In recent post-trade legislative acts, including MiFIR and MiFID II [16] regulators emphasize the necessity of transparent clearing and settlement processes. While the legislative agenda on blockchain based smart financial contracts still remains formally unaddressed, the legitimacy of distributed financial infrastructure is entirely contingent on the industry's ability to provide retroactively compliant solutions. The Firmo Protocol is an intermediate service layer, providing 3rd party applications with secure infrastructure to create and deploy smart financial contracts on any blockchain supporting scripting languages. This enables users of the Firmo infrastructure to contract with each other. Blockchain technology has a unique potential for disintermediating the traditional clearing, settlement and OTC matching processes in the traditional financial markets. Margins in the Firmo Protocol are case specific. Different clients will have different needs for margins, escrow or collateral applied in the trade and execution of smart derivatives. Firmo will not offer any margin or collateral accounts as the products launches.

5.1 Full Coverage

When users choose to execute smart derivatives with full coverage, the underlying assets will be stored in the smart derivative until maturity of the contract, upon which the transaction will take place. This function resembles the role of escrow or collateral in standardized contracts. The Firmo Protocol will launch with this feature.

5.2 Variable Collateral Coverage

The Firmo Protocol will allow users to lock margins, substituting the value of the underlying asset with the equivalent value in another asset class. Variable margin accounts will issue margin calls if the aggregate value of the collateral does not exceed the value of the underlying with a predefined percentage. This function is the equivalent of the margin call in the traditional brokerage roles.
This feature will be made available after the launch of the Firmo Protocol on the testnet.
See Fig. 5.a.

6.3 Zero Coverage

The Firmo Protocol allows users to execute smart derivatives with little or no collateral. Zero coverage can be used in several circumstances: When the smart derivatives are executed on permissioned infrastructure, or when the sufficient trust exists in between the counterparties. This feature will be built by the Firmo Community.

6.0 Data Feeds and Oracle Integrations

Smart derivatives can be triggered by external data feeds, such as the price of an asset on an exchange or a verifiable event, such the cancellation of an airplane or the LIBOR rate at a given date. A standard smart derivative can utilize input from a single data feed determining the price of an asset on a selected exchange, whereas a more exotic smart derivative may utilize inputs from several sources or exotic sources to form prediction markets with futures or similar instruments. Such external data feeds will be settled with external oracle partnerships. [7]

6.1 Proof-of-Authenticity with Oracles

Any external data feed from non-trusted entities utilized by the Firmo protocol will be required to deliver a proof-of-authenticity. The current discussion on the legitimacy of centralized data sources powering distributed platforms is a general concern in the industry, and must be dealt with appropriately by any protocol or platform provider. When introducing off-chain empirical events as a trigger function in the deterministic sequence of events portrayed on the blockchain, we are faced with the problem of translating a non-sequentially recorded series of events into a deterministic series of events. To solve this issue, users will be required to provide retroactive proof that any decisive data is authentic. Several industry grade solutions have been proposed to solve this issue by various third-party providers. The industry standard for handling data feeds with Oracles, is currently the proof-of-authenticity model. A proof can be constructed either by hashing the content of the query and delivering the data by signing the transaction with hash, or by referencing and hashing a trusted source for confirmation.

7.0 FirmoLang: A Formally Verified Domain Specific Language

This section provides a description of FirmoLang, the domain specific language for executing derivative contracts on Firmo. FirmoLang is a non-Turing complete domain specific language, designed with the explicit intention of diminishing the margin of error in self-executing contracts on decentralized platforms.

7.1 Language Objectives

In our view, object or contract oriented native languages such as Solidity perform well in large and complex environments with multiple agents and vectors. However, as emphasized by [4] and recent events [12], smart contracts or code written in general purpose Turing-complete languages frequently contain vulnerabilities. Such vulnerabilities have been exploited on several occasions, fuelling criticisms of decentralized infrastructure and raising concerns amongst investors. [13] While no system can be completely fault tolerant, financial contracts must execute in outstandingly secure environments, if the system is to be considered trustworthy. FirmoLang is designed for zero redundancies, expressing only the absolutely necessary functions for financial contracts. FirmoLang is formally verified with the Coq proof assistant. [14] Formal verification means that the Firmo protocol can run a static causal check of a smart derivative, exposing any vulnerabilities or faults , before a contract is submitted to the blockchain.

7.2 Simplicity is security

The FirmoLang language is formally constructed to produce an unambiguous and rigorous representation of contracts, in order to enable their automatic validation, execution, and analysis activities that are collectively referred to as contract lifecycle management. Domain specific languages hold several advantages over broad purpose, Turing-complete programming languages. While the narrow scope of the syntax described above may not suit complex expressions, FirmoLang enables a stable and secure execution environment for versatile and exotic financial contracts. With the upcoming open-source release, developers, regardless of their experience, will be able to formulate exotic smart derivatives out-of-the-box.

7.3 The FirmoLang Syntax

All derivative contracts can essentially by reduced to a series of transfers at a predetermined time, conditional on predetermined and verifiable events. This makes the task of computing derivative contracts inherently simple. FirmoLang has been adapted for from previous versions of a financial contract languages [15] and inspired by easily recognizable XML languages like the current

industry standard, FpML. [16] Similar approaches to formal contract languages has been proposed by [17] and [18] and more recently in the strictly Typed Pluto's Core Language [19]

FirmoLang is built for a compositional approach to derivative contracts. This facilitates a versatile range of expressions, with a minimum of well defined constructs. A smart derivative is composed by any number of smaller, independent components, called contract units. A contract unit utilizes the construct functions to execute a set of commands, delaying the contract unit execution or linking the contract to a data feed or subsequent contract unit or smart derivative. A simple smart derivative can consist of a single contract unit, where more advanced smart derivatives will typically contain several contract units with multiple expressions.

Fig. 7.a The FirmoLang Syntax

7.4 Constructs

The basic constructs are: zero denoting the end of the smart derivative and transfer denoting the transfer of one unit of the specified asset class between two parties. The translate construct offsets the contract unit execution to a specified time, and the scale construct multiplies transfers to a factor of (e, c1). The both construct denotes the execution of multiple contract units by the same expression, facilitating the compositional approach. The constructs if e within *t1* denotes the condition in which a predefined expression triggers a Boolean (true/false) value within a given timeframe. The constructs then *c1* else *c2* denotes the selected outcomes of the expression. FirmoLang allows multiple sources for parameterization of data feeds. This is detailed in the expressions for handling observables with n data feeds for multiple assets: obs (f,a1,...,a*n*,t). Selected data feeds from oracles, can trigger any predefined construct function. This feature makes FirmoLang an especially useful language for automation of complex tasks calling for an unmatched level of security.

7.5 Denotational Semantics

The Firmo Protocol gradually reduces a smart derivative as it executes. This is computed via intermediate expression of the contract terms to the construct zero, representing the completion of the smart derivative. The reduction semantics are represented by a small number of computer interpretable rules that dictate how a smart derivative is evaluated.

The reduction semantics pictured below, shows the intermediate reduction of a basic smart derivative from the Forward contract use case above. The smart derivative is submitted at t0. Here, the first reduction is applied when the translate construct postpones the execution by t+90 days. The next reduction is executed as the both constructs executes the two transfer constructs, completing the pre-defined cash flows. This sequence of events reduces the smart derivative to the zero construct.

Fig. 7.b The FirmoLang denotational semantics

Unlike dynamically typed languages, declarative languages like FirmoLang are strictly typed. This means that the Firmo protocol can infer the type the expressions at Compile Time. Because of the type system the Firmo protocol enables parties to verify that a smart derivative is well-defined and causal, meaning that it does not contain time absurdities. The type system makes it impossible for counterparties to exploit the Firmo protocol by creating smart derivatives where constructs such as the transfer function is used incorrectly, or a sequence of events could force parties to make untimely transfers of value.

8.6 An agnostic framework

FirmoLang is blockchain agnostic. This means that, given the right compiler, a smart derivative can potentially compile to any preconfigured byte code format, making the Firmo inter-operational with multiple decentralized, distributed or centralized platforms. The Firmo Protocol has been designed to accommodate a broad ecosystem with multiple languages and protocols. A dominant objective in designing the Protocol has been to support the efficient allocation of risk exposure across various platforms.

8.0 The Ethereum Virtual Machine Bytecode Compiler

FirmoLang initially compiles to the Ethereum Virtual Machine (EVM). The current version of the EVM compiler is ERC20 compliant and supports any asset-class defined as a certified token in solidity.

8.1 Specifications

In simple terms, an ERC20 token contract can be defined as a key/value pair, where the key is an Ethereum address and the value is the balance of the address, in the token. In order to transfer assets between parties and to read observables, FirmoLang relies on token contracts and data feed contracts, written in a more expressive language. The functionality of these contracts with which the compiled code interacts, cannot be expressed in the FirmoLang syntax. For this reason, the Firmo Protocol relies on an interface to perform this communication with other contracts. The communication from FirmoLang to token-contracts on ethereum can be divided into two operations:

1. Evaluating an observable by reading information through the ’get’ function defined in a data feed contract.
2. Executing a transfer of tokens from one party to another by calling the ’TransferFrom’ function defined in a token contract.

Fig. 8.a The FirmoLang Compiler flow on two ERC20 contracts

These simple instructions define how FirmoLang interacts with the EVM. Only the transfer function method call changes the state of the blockchain by transferring value from one party to another. The amount that is to be transferred and which transfers are to be executed will often depend on the value returned by the get method which only reads a state on the blockchain without alterations. This otherwise complex operation is thus reduced to two primary values:

1. A Boolean value to determine if or not a contract executes
2. An integer value defining the amounts transferred from accounts

The simplification of the specified terms, to the bare essentials in the larger contract or object oriented languages, removes any redundancies typically associated with today's smart contracts, creating a secure execution environment for derivative contracts.

9.0 Concluding Remarks: Future Work

This short introduction paper has presented the Firmo Protocol. This version has presented the basic principles for creating smart derivatives with the Firmo Protocol.

The following points will be addressed in detail in the upcoming version the Firmo Whitepaper.

1. A full chapter on the FRM token model.

2. Additional use cases built and supported by Firmo and the Firmo Community.

3. Technical specifications on the current exchange implementations and SDK/API resources.

4. Additional specifications for the use of FRM in Collateral margin accounts.

5. Targeting smart derivatives at the secondary markets by implementing an address changing function to FirmoLang and the compilers internal representation.

Acknowledgements

Firmo would like to thank: Mark Smargon, Amos Meiri, Eyal Hertzog, Guy Benartzi, Yudi Levi, Professor Michel Avital Professor Martin Elsman, Eran Segal, Omri Stein and Martin Koppelmann for their useful suggestions, critical evaluations and continued support on building the Firmo Protocol.

References

[1] Nakamoto, S., 2008. Bitcoin: A Peer-to-Peer Electronic Cash System.
[2] Gavin, W., Ethereum: a secure decentralised generalised transaction ledger.
[3] Hans, H., 2007. Rices Theorem
[4] Luu, L. et al., 2016. Making Smart Contracts Smarter.
[5] https://www.coindesk.com/understanding-dao-hack-journalists/
[7] http://www.oraclize.it/
[8] https://augur.net — https://gnosis.pm
[9] Raskin, Max, The Law and Legality of Smart Contracts (September 22, 2016)
[10] Gnosis.pm
[10] https://www.technologyreview.com/s/609320/can-china-contain-bitcoin/
[11] https://www.coindesk.com/understanding-dao-hack-journalists/
[12] https://coq.inria.fr/
[13] https://www.smartcontract.com/link
[16] https://ec.europa.eu/info/law/markets-financial-instruments-mifid-ii-directive-2014-65-eu_en