SmartFin Financial Smart Contract Client - User Manual

Daniel Dean - DD2415

Contents

1 Getting Started

1.1 Overview

The financial smart contract client is a client which allows the user to define a financial contract in a domain-specific language called SmartFin, and publish it to a blockchain in the form of a smart contract. These financial contracts can be as simple as one party paying the other 1 Wei (the minimal amount of currency payable on an Ethereum blockchain), but they can also be more complex - like a European option. For more information on writing financial contracts in SmartFin, see section 2.

The client can be connected to a blockchain of your choice (see sections 1.2 and 3), and used to publish *financial smart contracts* (smart contracts which represent SmartFin financial contracts). These contracts are made between two parties. One party is the *holder* - the party that can sign, or *acquire*, the contract. The second party is the *counter-party* - the party that authors the contract. It is not currently possible to create a contract with any number of parties other than 2 in this client.

When the holder *acquires* a financial smart contract, the contract will track the balance of Wei between the two parties. For example, a smart contract representing a SmartFin financial contract requiring the counter-party to pay the holder 1 Wei immediately will track the holder's balance increasing by 1 Wei, and the counter-party's balance decreasing by 1 Wei, upon acquisition by the holder.

This manual contains instructions on using the financial smart contract client to compose and evaluate SmartFin financial contracts, and deploy and interact with financial smart contracts.

1.2 Installation

In order to use the financial smart contract client, run the provided `fsc-server.sh` file (requires Python 3 and Google Chrome). This will run a local server, and open a tab in Google Chrome to the client page.

If no tab is opened automatically, open a browser window in Google Chrome and navigate to `http://localhost:8080`.

2 SmartFin - The Financial Contract DSL

2.1 Overview

The financial smart contract client deals with smart contracts that represent financial contracts defined in SmartFin, a combinator language used to describe financial contracts. A combinator language is a functional language in which a program is made up of chained function calls. As such, each contract written in SmartFin can be used as a sub-contract for any combinator. This means that a whole financial contract can be represented by a single combinator, or by some composition of combinators.

Each SmartFin financial contract has a *holder*, and a *counter-party.* Typically, the counter-party will be the party making payments, and the holder will be the party receiving payments. All payments are made in Ether (and all amounts of currency will be in the form of Wei, the smallest denomination of Ether).

A SmartFin financial contract can be *acquired* by the holder at any point in time, but the responsibilities of each party may differ depending on when the contract is acquired. For example, consider a contract C_1 which requires the counter-party to pay the holder 100 Ether on noon of January 1st 2019 and again on noon of January 1st 2020. C_1 requires 2 payments to occur if acquired before 12:00 on 01/01/19, 1 payment to occur if acquired by the 12:00 01/01/20, or none otherwise. The acquisition date of a SmartFin financial contract will therefore affect the value of the contract for each party.

A SmartFin financial contract may also *expire* where no responsibilities outlined in the contract take effect if the contract is acquired after a certain time. For example, the contract C_1 has no effect if acquired after 12:00 on 01/01/20. This date is called the *horizon* of the SmartFin contract. An important thing to note is that a contract's responsibilities could potentially extend past the contract's horizon, but a contract acquired after its horizon will have no effect.

Some SmartFin financial contracts may be dependent on not just sub-contracts, but also parameters. The contract C_1, for instance, defines payments of a specific amount on two specific dates. This contract would need to be defined with a constant representing 100 Ether, and two date/times. A SmartFin financial contract could also be dependent on a variable value, such as the temperature in London in Celsius, or the distance between two people in metres. Such a value is called an *observable*.

2.2 Combinators

The set of combinators defined in SmartFin is described below, along with the type signature of each combinator (using the function signature notation of Haskell). The notation used to describe SmartFin is defined in table 1.

c, d	*Contract*
o	*Observable*
t	*Date/Time*

Table 1: Conventions for SmartFin's Description

```
zero :: Contract
```

This combinator represents a contract with no terms. It can be acquired at any time.

```
one :: Contract
```

This combinator represents a contract which requires the counter-party to immediately pay the holder one Wei upon acquisition. This contract can be acquired at any time.

```
give :: Contract -> Contract
```

`give c` represents `c` with all responsibilities reversed (e.g. if the holder acquires `give one`, they must pay the counter-party 1 Wei immediately).

```
and :: Contract -> Contract -> Contract
```

When `and c d` is acquired, both c and d are acquired immediately. Expired sub-contracts are not acquired.

```
or :: Contract -> Contract -> Contract
```

When `or c d` is acquired, the holder immediately acquires either c or d. If one has expired, the holder cannot acquire it (and must acquire the other if possible).

```
truncate :: Date -> Contract -> Contract
```

When `truncate t c` is acquired, the holder acquires `c`. The horizon of `truncate t c` is the earliest of `t` and the horizon of `c` (thus `truncate t c` does nothing after either horizon has passed).

Dates in SmartFin must be provided in either the format `<DD/MM/YYYY HH:mm:ss>`, the format `<DD/MM/YYYY HH:mm:ss Z>`, or in UNIX Epoch time format. For more information on how to format times, see section 5.1.1.

```
then :: Contract -> Contract -> Contract
```

When acquiring `then c d`, the holder acquires `c` if `c` has not expired, or `d` if `c` has expired and `d` has not.

```
scale :: Observable -> Contract -> Contract
```

`scale o c` represents `c` with all payments multiplied by the value of the observable `o` at the time of acquisition.

An observable is represented by either a number (e.g. `scale 5 one` requires the counter-party to pay 5 Wei to the holder), or by a name and address if the observable has a time-varying value. The name is used to refer to the observable in the financial smart contract client, and the address is the user address of an arbiter for the observable's value that will provide its value at some point. This is written in the form `scale <name> <addr> c`, e.g. `scale tempInLondon 0xA0a4D3524dC3428884c41C05CD344f9BcB5c79f3 one`. Observable names can be in any form as long as they contain at least 1 non-mathematical character, such as a letter.

```
get :: Contract -> Contract
```

Acquiring `get c` acquires `c` at the moment in time when the horizon of `c` is reached. For example, `get truncate t one` will require the counter-party to pay the holder 1 Wei at time `t` (if acquired before it expires).

```
anytime :: Contract -> Contract
```

After `anytime c` is acquired, `c` can be acquired by the holder at any time before it expires, and must be acquired by this point.

2.3 Examples

2.3.1 Zero-Coupon Discount Bond

One example of a simple financial contract is a *zero-coupon discount bond*. This is a contract between a holder and a counter-party that requires the counter-party to pay a specified amount of currency to the holder at a certain date.

A zero-coupon discount bond which requires the counter-party to pay 100 Wei to the holder at 12:00pm on 01/01/2020 is defined in SmartFin as:

```
get truncate <01/01/2020 12:00:00> scale 100 one
```

Once the `get` combinator is acquired, its sub-contract will be acquired at the acquisition date, i.e. 12:00pm on 01/01/01. The `truncate` combinator will not yet have expired, and so its underlying contract will be acquired at this point. The acquisition of the `scale` combinator causes its underlying contract (with values multiplied by 100) to be acquired immediately, thus acquiring the `one` combinator. This results in the counter-party paying 100 Wei to the holder at 12:00pm on

01/01/2020, if acquired before this time.

2.3.2 European Options

A European option is another type of financial contract, which states that the holder can choose whether or not to acquire a contract on a given date.

A European option over the contract `c` at 12:00pm on 01/01/2020 is defined in SmartFin as:

```
get truncate <01/01/2020 12:00:00> or c zero
```

Similarly to the previous contract, acquiring the `get` combinator acquires the sub-contract at its horizon, i.e. 12:00pm on 01/01/2020. This acquires the non-expired `truncate` combinator, and thus the underlying `or` combinator.

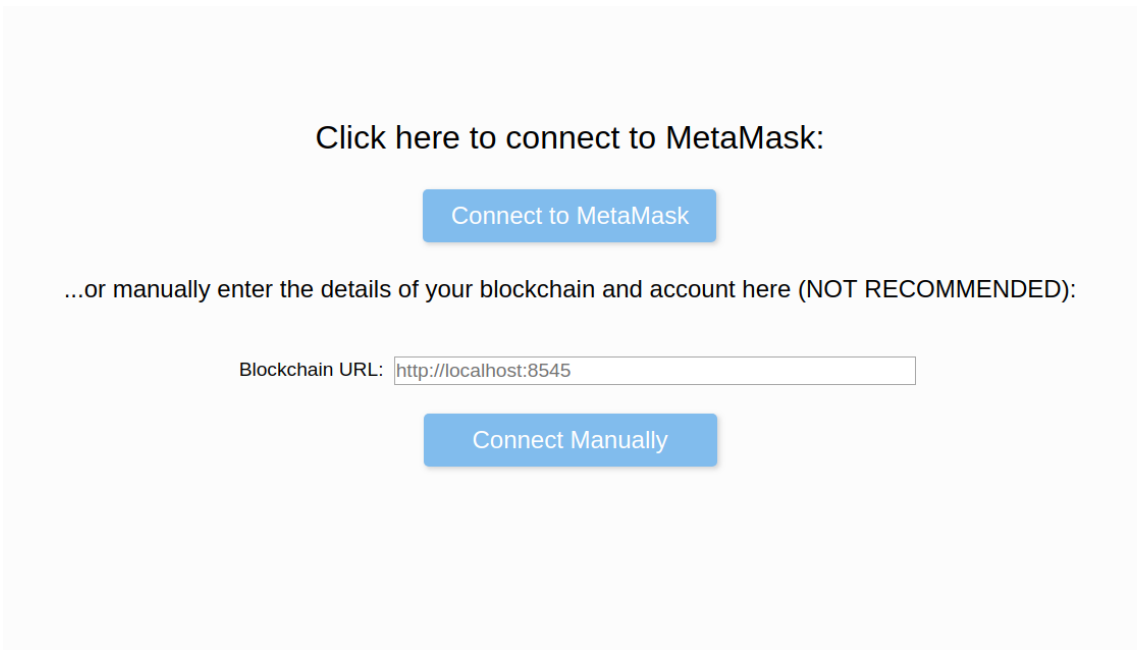
At any point in time, the holder may specify which branch of the `or` combinator they would like to acquire. In the financial smart contract implementation, the contract will not proceed until a choice is made. After a choice is made, the chosen branch is evaluated based on the acquisition time of the `or` combinator (i.e. 01/01/2020 12:00pm), regardless of when the `or` choice is actually supplied.

This means that the user will select between the underlying contract `c` and `zero`, and the result will be paid out at 01/01/2020 12:00pm, or as soon as the `or` choice is provided (whichever is latest).

3 Connecting to a Blockchain

3.1 Overview

Once the server has been started and the client is open in a browser tab, you will be greeted by the blockchain connection screen, shown in figure 1.


Click here to connect to MetaMask:

...or manually enter the details of your blockchain and account here (NOT RECOMMENDED):

Blockchain URL: http://localhost:8545

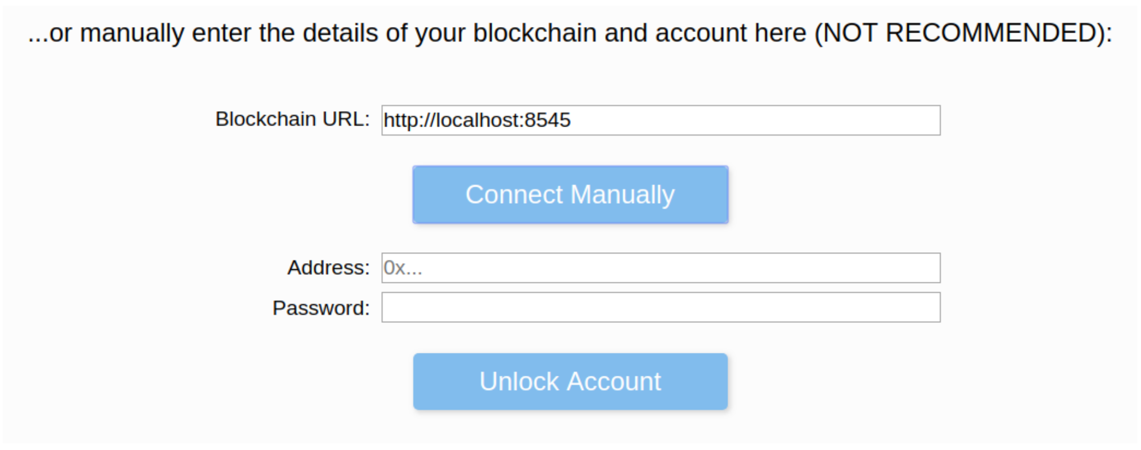
Figure 1: The *Connect* view, through which you can connect the client to a blockchain.

If you are using MetaMask in your browser, you can connect to it using the *Connect to Meta-Mask* button. This will automatically detect your account's address, and allow you to approve any blockchain interaction through MetaMask.

3.2 Manual Connection

If you are running a local test blockchain, it may be easier to enter your blockchain's details manually. Please note, this is **not recommended** unless you are connecting to a local test blockchain, as your credentials will be processed directly through `web3` in plaintext, which is not secure.

Pressing the *Connect Manually* button will prompt the client to communicate with the blockchain at the address in the *Blockchain URL* input. If the blockchain is successfully found, you will be prompted for the account's address and password, as shown in figure 2.


...or manually enter the details of your blockchain and account here (NOT RECOMMENDED):

Blockchain URL: http://localhost:8545

Address: 0x...

Password:

Figure 2: The account detail inputs in the *Connect* view, after a manually-entered blockchain is found.

If the account's details are correct, the account will be unlocked permanently (or until the connected blockchain node is restarted), and the client will proceed to the main menu.

If the connection is not successful, please ensure that the blockchain URL is correct, and that the account credentials are correct.

4 Main Menu

Once you have connected the client to a blockchain, the client will progress to the main menu (shown in figure 3).

From the main menu, there are 2 available options. You may proceed to the *contract composition* view, to create, evaluate, and deploy new SmartFin financial contracts and their corresponding financial smart contracts. The other option is to proceed to the *contract monitoring* view, to monitor and interact with a deployed financial smart contract.

Press one of the two buttons to proceed to the appropriate view. You may return to the main menu from either of these menus.

5 Composing a SmartFin Contract

From the main menu, you may have progressed to the contract composition view (shown in figure 4). If you progressed to the monitoring view instead, please see section 6. To return to the main menu, press the button at the top left of the screen. The contract composition menu allows you to write, evaluate, and deploy a SmartFin financial contract and its financial smart contract counterpart.

5.1 Composing a SmartFin Financial Contract

In the large text input, you may input a SmartFin financial contract. If you need to know how to write a financial contract with SmartFin, please refer to section 2. Alternatively, the *Help* button in the monitoring view will display a similar guide for writing in SmartFin.

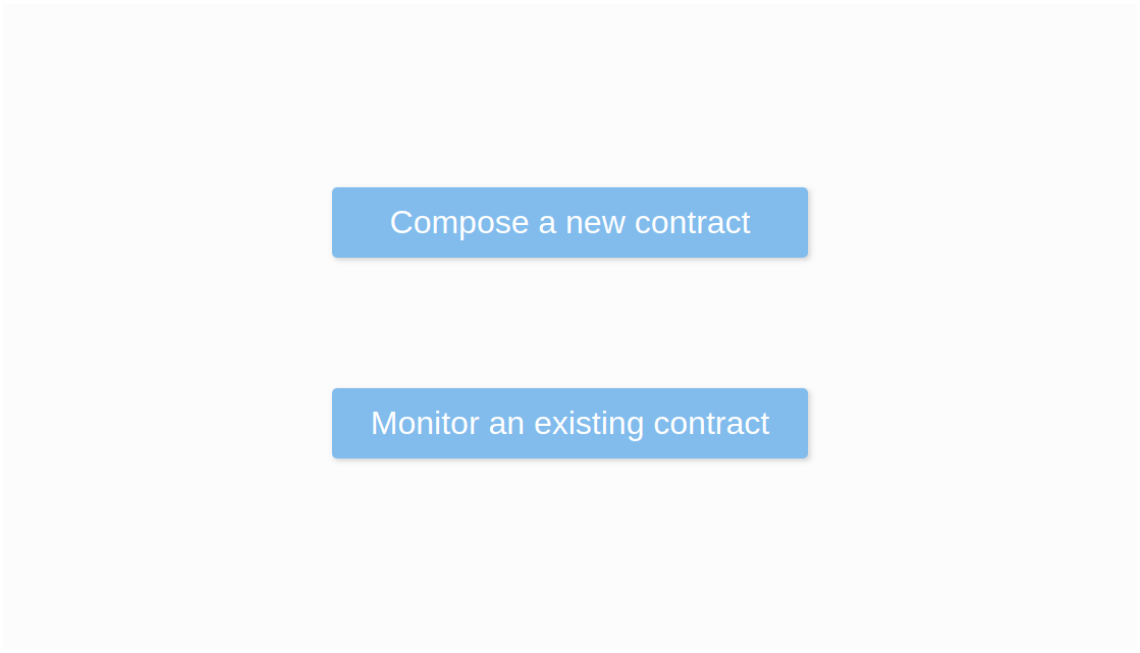


Figure 3: The *Main Menu* view.

5.1.1 Inputting Time

Some SmartFin contracts require a time value to be provided, which will be a specific time on a specific date. For example, the `truncate` combinator causes a contract to expire at a given time (if not earlier). These time values can be provided in a few ways.

One way to provide times to a SmartFin contract is in the format `<DD/MM/YYYY HH:mm:ss>`. For example, the contract `truncate <01/02/2020 13:45:01> one` is worth 1 Wei up until 13:45:01 on the 1st of February 2020.

By default, the time zone for times input in this manner is set to the time zone of your client's locale. In order to change this, the time zone can be provided by inputting a time in the format `<DD/MM/YYYY HH:mm:ss Z>`, where `Z` represents a 2 or 4 digit time zone offset. For example, the time `"<01/02/2020 12:34:00 +1234>"` is equivalent to the time `"<01/02/2020 00:00:00 +0000>"`. Times returned from the client are usually displayed in UTC, with the time zone visible to prevent ambiguity.

Times can also be provided in UNIX Epoch time format. To input a UNIX Epoch time more conveniently, pressing the *Input UNIX Time* button will open a view to input the time using a calendar and time selector. Upon pressing the *Input UNIX Time* button in this menu, the UNIX Epoch time corresponding to the input date/time (in UTC) will be inserted into the SmartFin contract definition at the text cursor's current location.

5.1.2 Verification

Once you have input a SmartFin financial contract definition, you may verify it by pressing the *Evaluate Contract* button, or the *Deploy Contract* button (pressing `enter` is equivalent to pressing *Deploy Contract*).

Upon pressing either of these buttons (or the `enter` key), the SmartFin contract will be verified. If it contains any errors, then they will be displayed below the contract input (as shown in figure 5). The error message can be expanded by clicking on it, which will show a more detailed trace of the error.

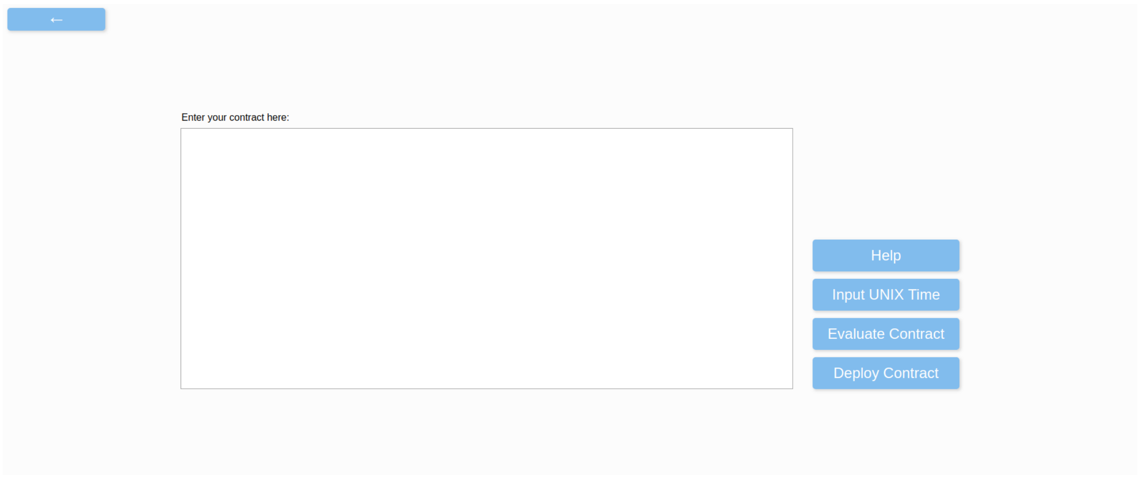
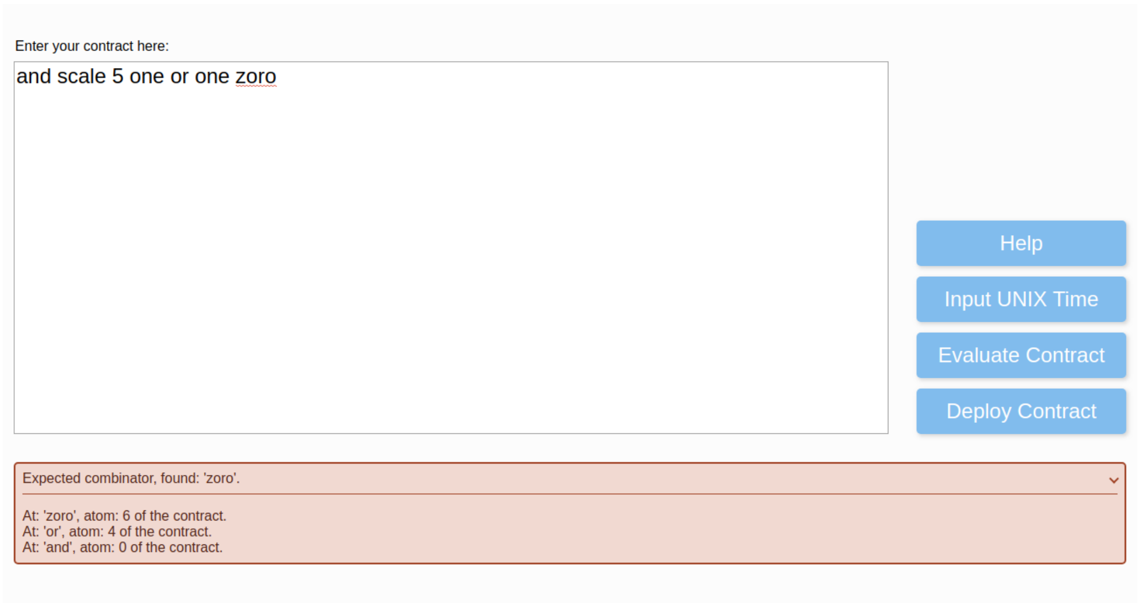


Figure 4: The *Composition* view, consisting of a large text input on the left for a SmartFin financial contract definition, and several options to the right. The top-left button will return you to the main menu.



```
and scale 5 one or one zoro
```

```
Expected combinator, found: 'zoro'.

At: 'zoro', atom: 6 of the contract.
At: 'or', atom: 4 of the contract.
At: 'and', atom: 0 of the contract.
```

Figure 5: The *Composition* view, with an error message and expanded error trace.

5.1.3 Evaluating a SmartFin Financial Contract

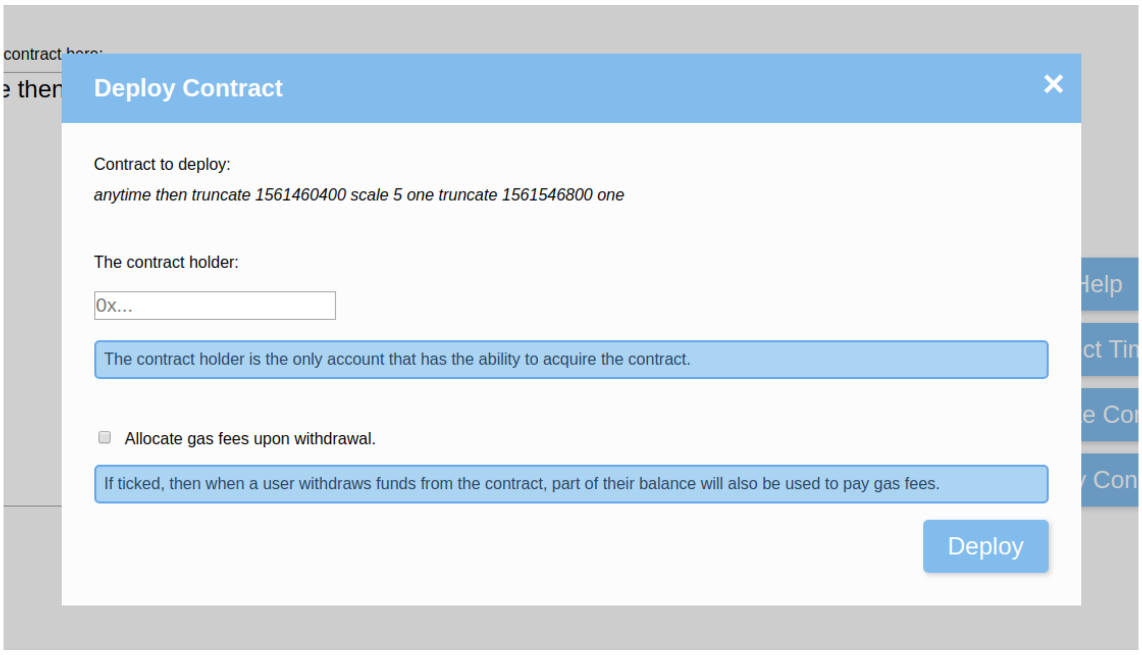
In order to evaluate the SmartFin contract before deploying it, press the *Evaluate Contract* button. This will display the *Evaluate Contract* menu. For more information about this menu, please see section 7.

5.2 Deploying a Financial Smart Contract

Once the SmartFin financial contract definition is fully-written and error-free, its financial smart contract representation can be deployed to the connected blockchain. The smart contract is compiled to the `wasm` format, so please ensure that the connected blockchain is compatible with `wasm` smart contracts before proceeding.

To open the deployment menu, press the *Deploy Contract* button on the composition view (or press `enter` in the SmartFin contract text input). This will display a short menu for setting some options on the contract (as shown in figure 6).

A deployed financial smart contract has a counter-party (the account used to deploy the con-


Deploy Contract

Contract to deploy:

anytime then truncate 1561460400 scale 5 one truncate 1561546800 one

The contract holder:

0x...

The contract holder is the only account that has the ability to acquire the contract.

Allocate gas fees upon withdrawal.

If ticked, then when a user withdraws funds from the contract, part of their balance will also be used to pay gas fees.

Figure 6: The *Deploy Contract* menu, which displays the final SmartFin contract, as well as several options.

tract), and a prospective holder. The prospective holder of the financial smart contract is fixed, to ensure that only the intended party may acquire the contract. The holder's address must be provided to the contract during deployment, and can be entered in the *Contract Holder* input.

The financial smart contract also has an option for whether or not it should use gas upon withdrawal. If only externally-owned accounts (i.e. users, not other smart contracts) will interact with the contract, or the blockchain the contract is running on does not require gas fees, then this box does not need to be ticked. If one of the parties who will withdraw funds from the contract is a smart contract, then gas fees will need to be paid upon withdrawing in case the smart contract receiving the funds needs to execute code. In this case, the gas fees (2300 Wei) will be taken out of the withdrawing party's balance upon withdrawal. If there are not enough funds to pay for gas fees, then withdrawal will fail.

Once you have provided all options to the deployment menu, the *Deploy* button will deploy the financial smart contract to the connected blockchain from the connected account. If this fails for any reason, the error message will be displayed at the bottom of this menu. This message can be expanded by clicking to see the blockchain's error message.

6 Monitoring a Financial Smart Contract

From the main menu, you may have progressed to the monitoring view. If you progressed to the composition view instead, please see section 5. The monitoring view enables you to interact with already deployed financial smart contracts in various ways.

If you have previously deployed a financial smart contract in the composition view during the current session on the client, then the monitoring view will automatically display the details of the deployed contract. If not, you will be prompted for the address of the financial smart contract you would like to monitor (as shown in figure 7). If the given address corresponds to a financial smart contract then its details will be loaded, otherwise an error message will be shown.

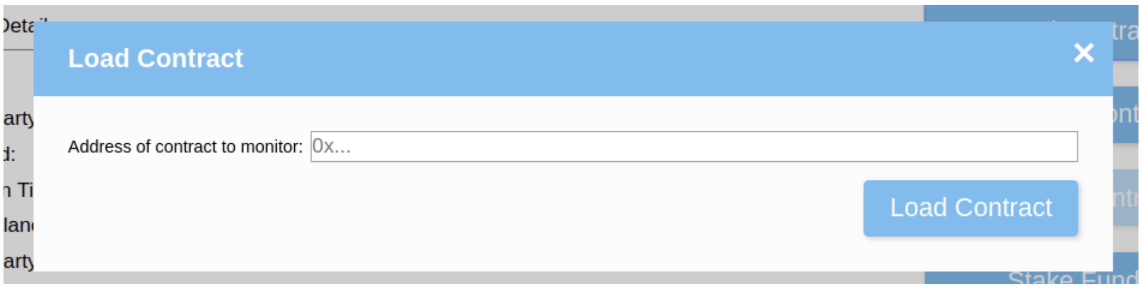
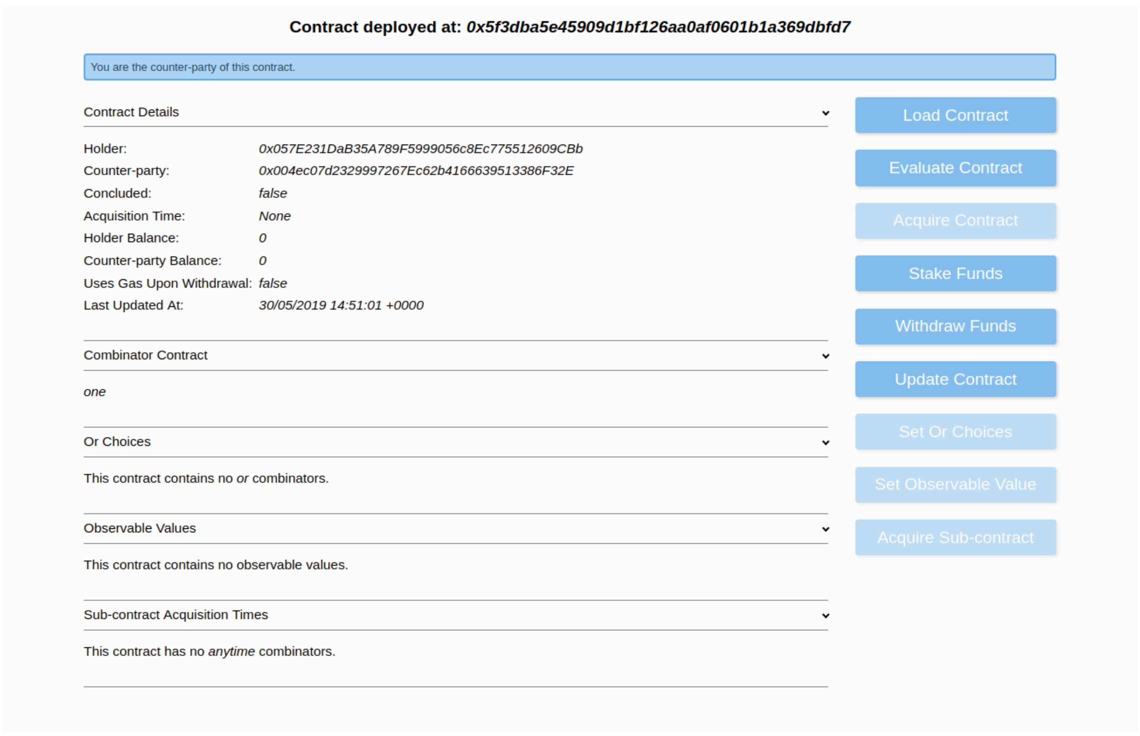


Figure 7: The *Load Contract* menu, which allows you to enter the address of a financial smart contract you would like to monitor.

6.1 Financial Smart Contract Details

The monitoring view allows the user to view several details from the financial smart contract, as shown in figure 8. These details are contained with drop-down boxes, which can be opened or closed by clicking on the titles. The *Contract Details* box contains information regarding the financial smart contract's current state, including the contract holder and counter-party's addresses, whether or not the contract has concluded, the top-level acquisition time, the holder and counter-party's balances, whether or not the contract uses gas, and the last time the contract was updated. The SmartFin financial contract definition for the financial smart contract is also displayed. All of the information in the monitoring view is refreshed every few seconds. The functions on the smart contract which return this information are *pure*, and so no gas is used to obtain these details.



Contract deployed at: *0x5f3dba5e45909d1bf126aa0af0601b1a369dbfd7*

You are the counter-party of this contract.

Contract Details

Holder:	*0x057E231DaB35A789F5999056c8Ec775512609CBb*
Counter-party:	*0x004ec07d2329997267Ec62b4166639513386F32E*
Concluded:	*false*
Acquisition Time:	*None*
Holder Balance:	*0*
Counter-party Balance:	*0*
Uses Gas Upon Withdrawal:	*false*
Last Updated At:	*30/05/2019 14:51:01 +0000*

Combinator Contract

one

Or Choices

This contract contains no *or* combinators.

Observable Values

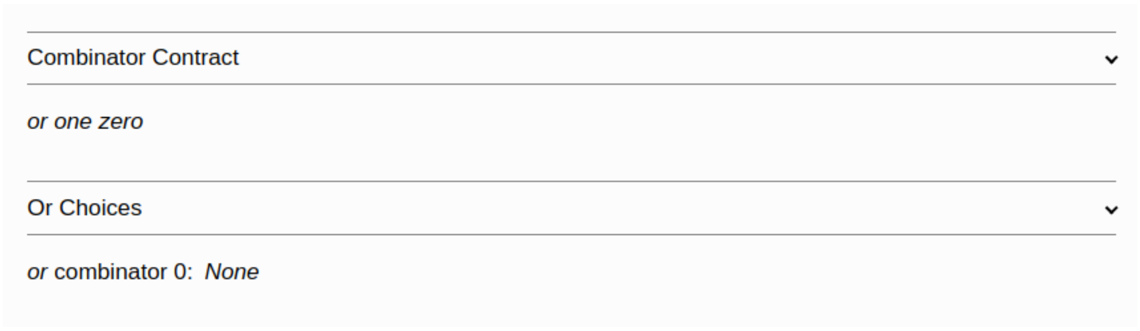
This contract contains no observable values.

Sub-contract Acquisition Times

This contract has no *anytime* combinators.

Figure 8: The *Monitoring* menu, which displays the details of a financial smart contract in several drop-down boxes, and several options.

Besides the basic details, the contract also shows a list of available or-choices, observable values, and anytime acquisition times (and the state/value of each). This can be seen in figures 9, 10, and 11.

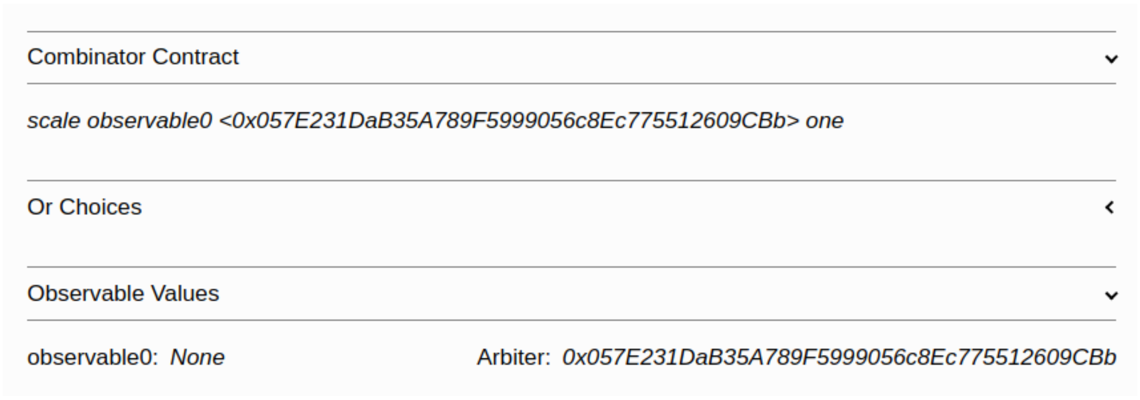

Combinator Contract

or one zero

Or Choices

or combinator 0: *None*

Figure 9: An or-choice (before choosing) in the *Monitoring* menu.


Combinator Contract

scale observable0 <0x057E231DaB35A789F5999056c8Ec775512609CBb> one

Or Choices

Observable Values

observable0: *None* Arbiter: *0x057E231DaB35A789F5999056c8Ec775512609CBb*

Figure 10: An observable value named *observable0* (before setting) in the *Monitoring* menu.

6.2 Interacting with a Financial Smart Contract

Besides viewing the details of the financial smart contract, the monitoring view also allows the user to interact with it. The user can step through the contract to evaluate it based on certain parameters (see section 7 for more details), acquire the contract (if the user is the contract holder), stake funds in the contract, withdraw funds from the contract, update the contract, or set values in the contract.

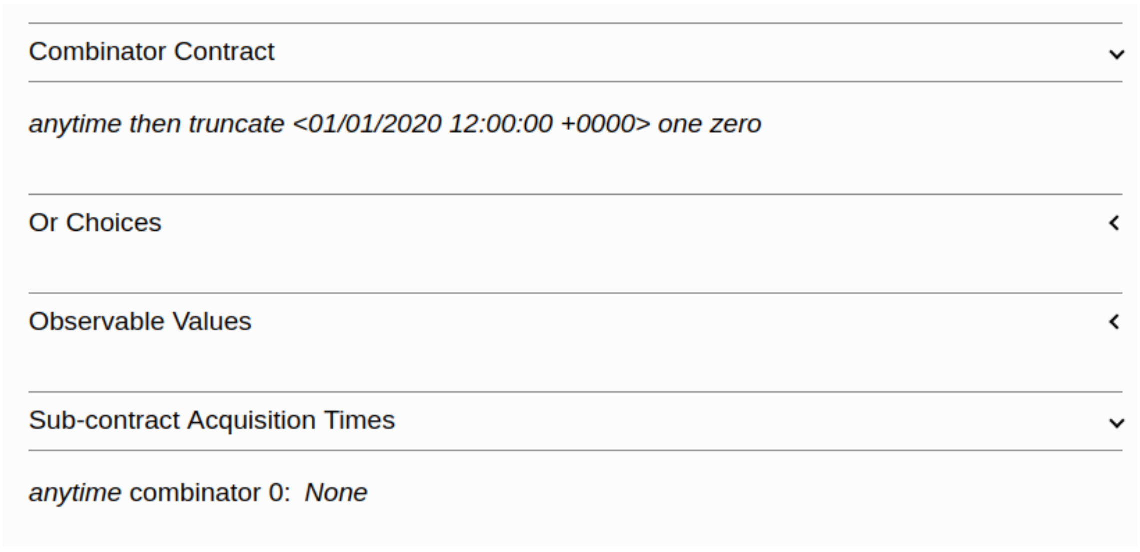
6.2.1 Acquiring and Updating

Acquiring the contract (i.e. *signing*) will set its acquisition time to the current time, and updates the contract based on its combinators. For example, if a user acquires the financial smart contract representing the SmartFin contract `one`, then the holder balance will increase by 1, and the counter-party balance will decrease by 1.

Some balance changes will only occur over time, depending on the combinators . For example, a `get` combinator will only acquire the sub-contract to update the contract balance upon its horizon, and before this its value is 0. The financial smart contract cannot automatically update over time, as this would be non-deterministic, and would require someone to pay gas fees. To bring the contract's value up-to-date, press the *Update Contract* button. *This requires paying gas, as updating requires state-altering execution on the smart contract.*

6.2.2 Staking and Withdrawing Funds

A financial smart contract cannot pay funds to either party until some funds are paid in. To put funds into the contract, press the *Stake Contract* button and enter a value (in Wei) to send. To withdraw funds, press the *Withdraw Funds* button and enter a value (in Wei) to withdraw. You cannot withdraw more funds than your balance, more funds than the total contract balance, or when your balance or the financial smart contract's total balance cannot afford the gas fees (when gas fees upon withdrawal are enabled).


Combinator Contract

anytime then truncate <01/01/2020 12:00:00 +0000> one zero

Or Choices

Observable Values

Sub-contract Acquisition Times

anytime combinator 0: *None*

Figure 11: An anytime acquisition time (before acquisition) in the *Monitoring* menu.

6.2.3 Choosing `or` Branches

You may also set several values on the financial smart contract. A contract with `or` combinators allows the user to choose between the left and right sub-contracts. To do this, you may press the *Set Or Choice* button and choose an `or` combinator and its chosen sub-contract (as shown in figure 12). The `or` combinator is selected by its `or-index`. Each `or` combinator has an `or-index`, starting from zero and increasing sequentially by order of occurrence in the financial smart contract (from left to right). For example, in the contract `or one or one zero`, the `or` at atom 0 has `or-index` 0, and the `or` at atom 2 has `or-index` 1.

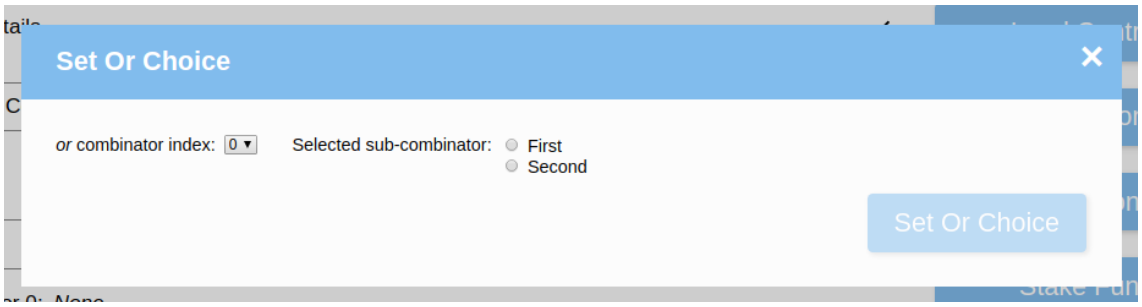


Figure 12: The menu to choose a branch for an `or` combinator.

6.2.4 Setting Observable Values

From the monitoring view, you may also set the value of observables (as shown in figure 13). The `scale` combinator may define an observable and an arbiter. If you are logged in to the client with the address of an arbiter of an observable, then you can set the value of that observable. To do so, press the *Set Observable Value* button, choose the observable to set, and enter the value.

6.2.5 Acquiring `anytime` Sub-contracts

In a financial smart contract, an `anytime` combinator can be acquired at any point before the horizon is passed. If it is not acquired by this time, it will be acquired on the horizon. For more details, see section 2. In order to acquire an `anytime` sub-contract at the current time, press the *Acquire Anytime Sub-contract* button to open the `anytime` acquisition menu (shown in figure 14). Choose the combinator by its `anytime-index` (defined similarly as the `or-index` in section 6.2.3, for `anytime` instead of `or`), and press the *Acquire* button. *If you have not acquired the parent of the `anytime` combinator, then this will fail and display an error message.*

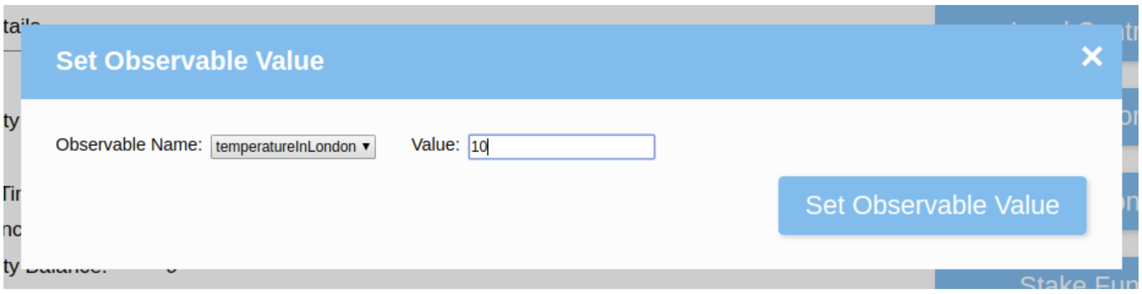


Figure 13: The menu to set the value of an observable (named *temperatureInLondon*).

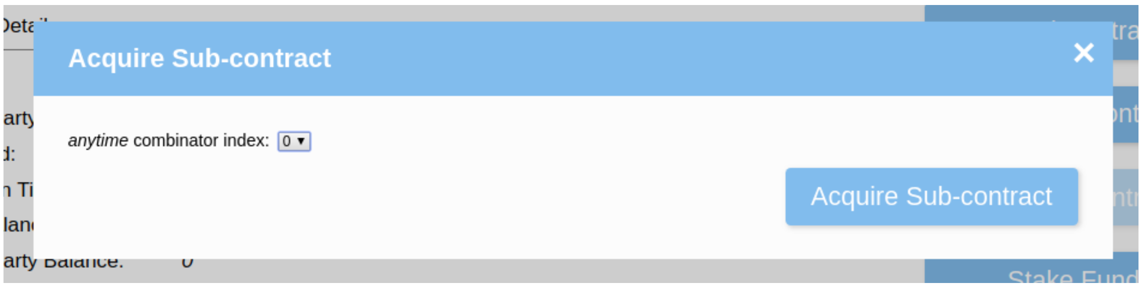


Figure 14: The menu to acquire an `anytime` combinator.

7 Evaluating a SmartFin Financial Contract

The evaluation menu can be viewed from either the *composition* view or the *monitoring* view. The SmartFin financial contract which has been composed or loaded from a monitored financial smart contract can be evaluated in this view, by selecting acquisition times and or-choices for the contract. The evaluation is handled in a step-by-step manner, and at each step you are required to choose one of several options. These options can be deleted at any time, in case you change your mind. The first step of evaluation will be choosing the top-level contract acquisition time, i.e. the time the contract itself is acquired.

7.1 Top-level Acquisition Time

When choosing the top-level acquisition time, several options will be prevented to you (as shown in figure 15). These options are each a distinct range of time, within which the contract's value will not change. For example, the contract `truncate <1st March 2020 12:00> one` will have the value 1 Wei until 12:00 on the 1st March 2020 is passed, and the value 0 Wei afterwards. This makes 2 distinct ranges of time with fixed values.

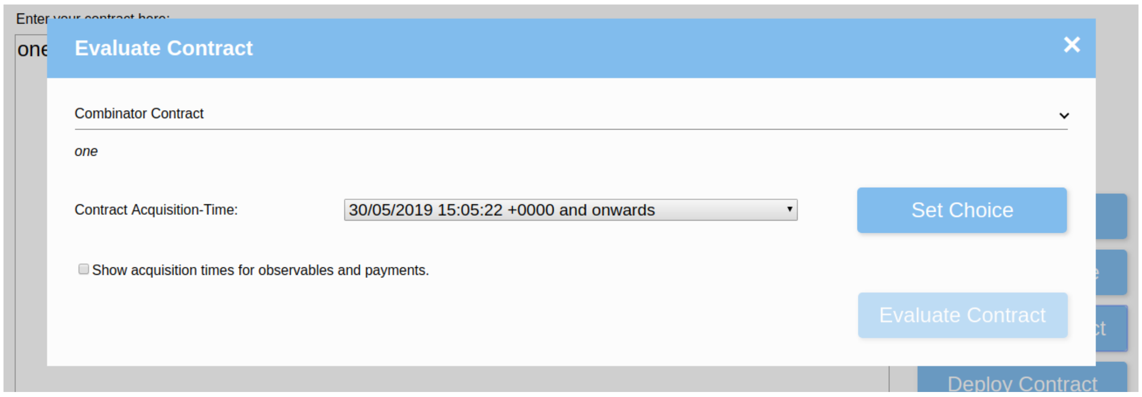

Evaluate Contract
Combinator Contract
one
Contract Acquisition-Time: 30/05/2019 15:05:22 +0000 and onwards
Set Choice
Show acquisition times for observables and payments.
Evaluate Contract
Deploy Contract

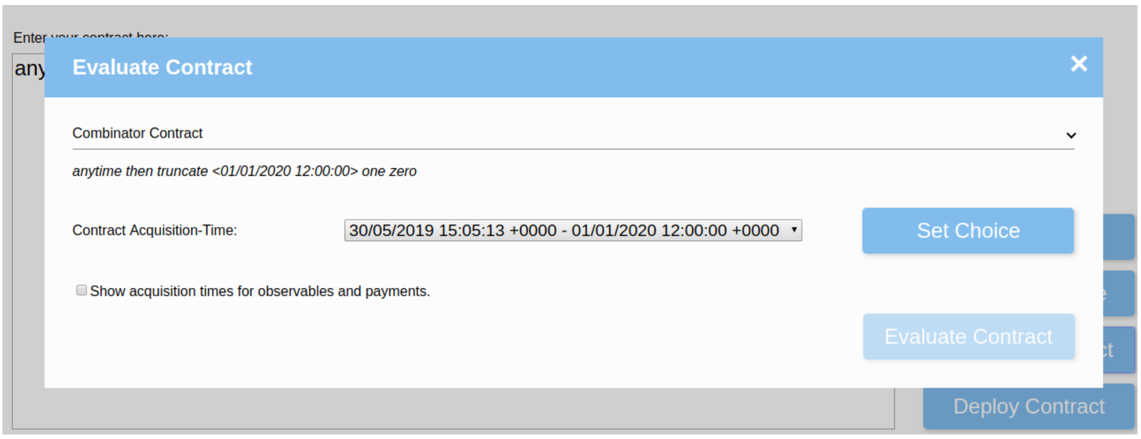
Figure 15: The *Evaluation* menu, before choosing the contract's acquisition time.

These time ranges do not account for the varying of observables; for example, `scale obs`

`<addr> one` will be treated as a single range of time with equal value, even though the value of its observable may change over time.

7.2 Anytime Acquisition Time

Similarly to the top-level acquisition time, you may also need to provide acquisition times for anytime combinators within a contract, as shown in figure 16. The options for this will also be represented as distinct time ranges, in the same way that the top-level acquisition time options are shown. For example, for the contract `anytime then truncate <1st March 2020 12:00:00> one zero` (with a top-level acquisition time of 1st March 2020 12:00:00 or earlier), `anytime 0` will have options consisting of the time before 12:00 on 1st March 2020, and the following time period.



Combinator Contract

anytime then truncate <01/01/2020 12:00:00> one zero

Contract Acquisition-Time: 30/05/2019 15:05:13 +0000 - 01/01/2020 12:00:00 +0000

☐ Show acquisition times for observables and payments.

Figure 16: The *Evaluation* menu, setting an `anytime` acquisition time.

If an `anytime` combinator only has one available option for its acquisition time, then that option will be chosen automatically. For example, in the same contract defined above, if the top-level acquisition time is after 12:00:00 on 1st March 2020, then `anytime 0` can only be acquired in the time range after this time, and so this is automatically set. As such, the user only needs to select `anytime` acquisition times where there is more than one time-range to choose from.

The `anytime` combinator is denoted by its `anytime-index`, which is defined similarly to the `or-index` in section 6.2.3.

7.3 Or Choices

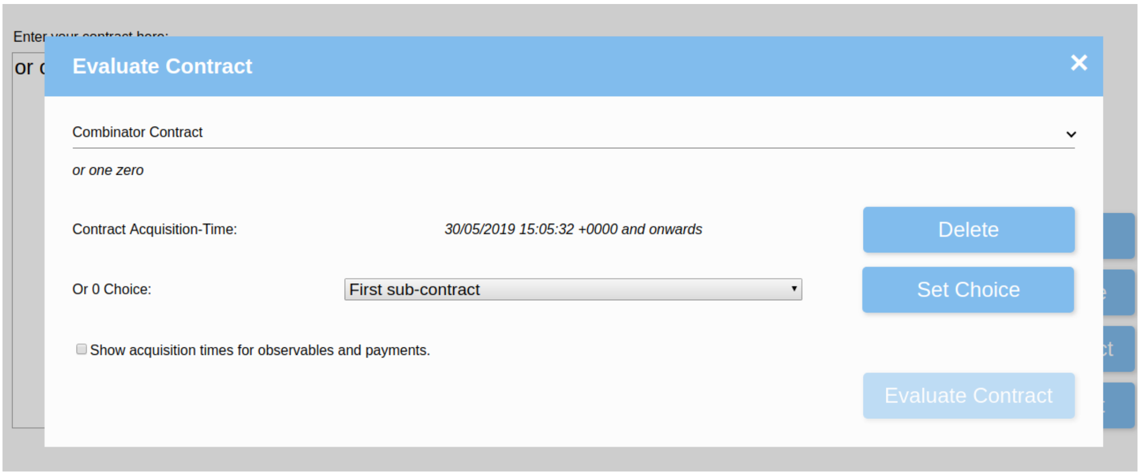
Besides acquisition times, the other combinator which requires user input is the `or` combinator. As such, during step-by-step evaluation the user must input any `or-choices` as they occur, as shown in figure 17. You may choose between the first and second branch of the `or` combinator, ordered by occurrence when reading the SmartFin financial contract definition from left-to-right. For example, in the contract `or one zero`, choosing the first branch will net a value of 1 Wei, and the second branch will net a value of 0 Wei.

If either branch is expired at the time that the `or` combinator is encountered during evaluation (i.e. the time-range chosen for its parent combinators), then that branch cannot be acquired (for more information, see section 2). For example, in the contract `or truncate <1st March 2020 12:00:00> one zero`, if the top-level contract is acquired after 12:00:00 on the 1st March 2020, then no `or` choice will be presented and the value will be 0 Wei. If both branches have expired, then similarly no choice is presented and the `or` combinator will evaluate to 0 Wei.

The `or` combinator is denoted by its `or-index`, which is defined in section 6.2.3.

7.4 Value

The final result of the evaluation process will be a value, displayed at the bottom of the menu (as shown in figure 18). This value consists of a sum of products of observables and values.



Combinator Contract

or one zero

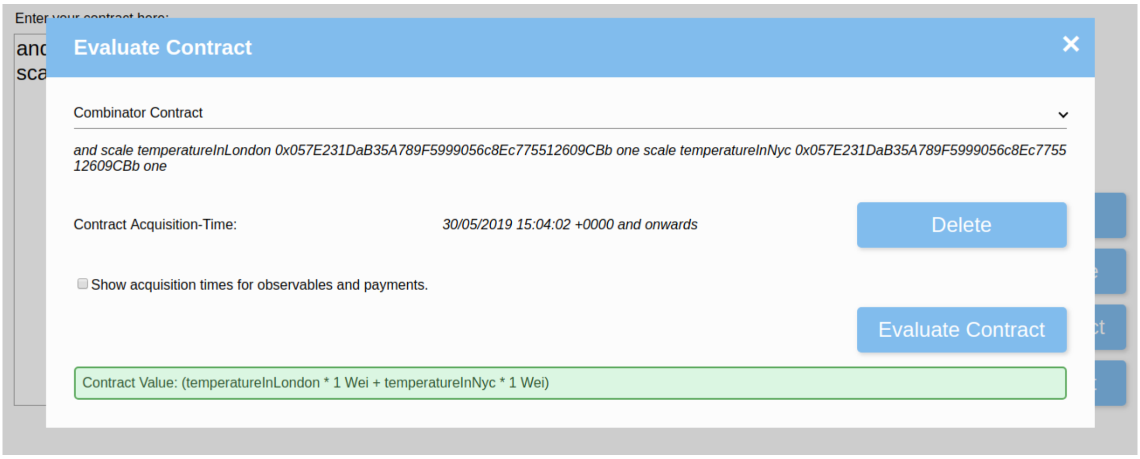
Contract Acquisition-Time: *30/05/2019 15:05:32 +0000 and onwards*

Or 0 Choice: First sub-contract

☐ Show acquisition times for observables and payments.

Figure 17: The *Evaluation* menu, setting an `or` choice.

For example, the contract `and scale obs0 <addr> one scale obs1 <addr> one` will evaluate to *obs0 * 1 + obs1 * 1* Wei. The value of the observable cannot be estimated by the evaluation process, and so it is treated as an unknown variable.



Combinator Contract

and scale temperatureInLondon 0x057E231DaB35A789F5999056c8Ec775512609CBb one scale temperatureInNyc 0x057E231DaB35A789F5999056c8Ec7755 12609CBb one

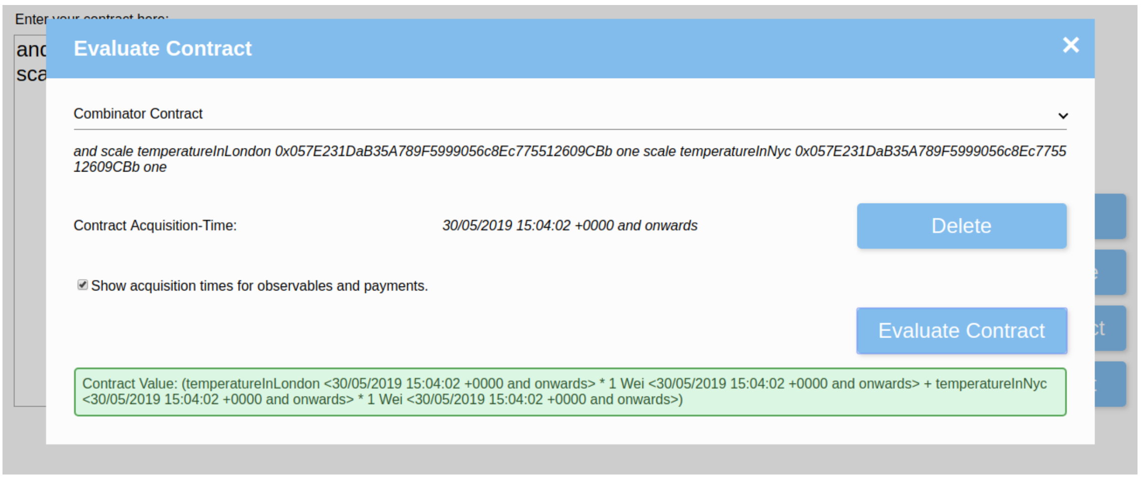
Contract Acquisition-Time: *30/05/2019 15:04:02 +0000 and onwards*

☐ Show acquisition times for observables and payments.

Contract Value: (temperatureInLondon * 1 Wei + temperatureInNyc * 1 Wei)

Figure 18: The *Evaluation* menu, after evaluating a SmartFin financial contract.

By default, the evaluation value represents the total overall value of the contract once it has concluded, and does not take time into account. Some observables may have widely varying values over time, in which case it can be useful to know the specific times in which observables/payments are acquired. In order to show this, check the box labelled *"show acquisition times for observables and payments."* This will display time periods alongside observables and payment values, which represent the times at which they are each queried or obtained respectively (as shown in figure 19).



and scale temperatureInLondon 0x057E231DaB35A789F5999056c8Ec775512609CBb one scale temperatureInNyc 0x057E231DaB35A789F5999056c8Ec7755 12609CBb one

Contract Acquisition-Time: *30/05/2019 15:04:02 +0000 and onwards*

[✓] Show acquisition times for observables and payments.

```
Contract Value: (temperatureInLondon <30/05/2019 15:04:02 +0000 and onwards> * 1 Wei <30/05/2019 15:04:02 +0000 and onwards> + temperatureInNyc
<30/05/2019 15:04:02 +0000 and onwards> * 1 Wei <30/05/2019 15:04:02 +0000 and onwards>)
```

Figure 19: The *Evaluation* menu, after evaluating a SmartFin financial contract with acquisition times displayed.