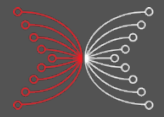
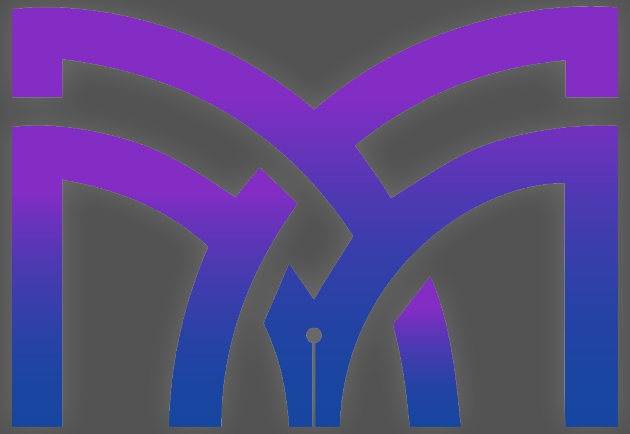


IOHK | Summit 2019
APRIL 17–18 MIAMI

Marlowe

Financial Contracts on Blockchain



Simon Thompson Pablo Lamela Seijas Alex Nemish

Marlowe: financial contracts on blockchain

Marlowe language for financial contracts on blockchain and particularly Cardano.

Meadow for development and simulation.


LEO MARK I

Fortran

Algol

BASIC

C++

Java

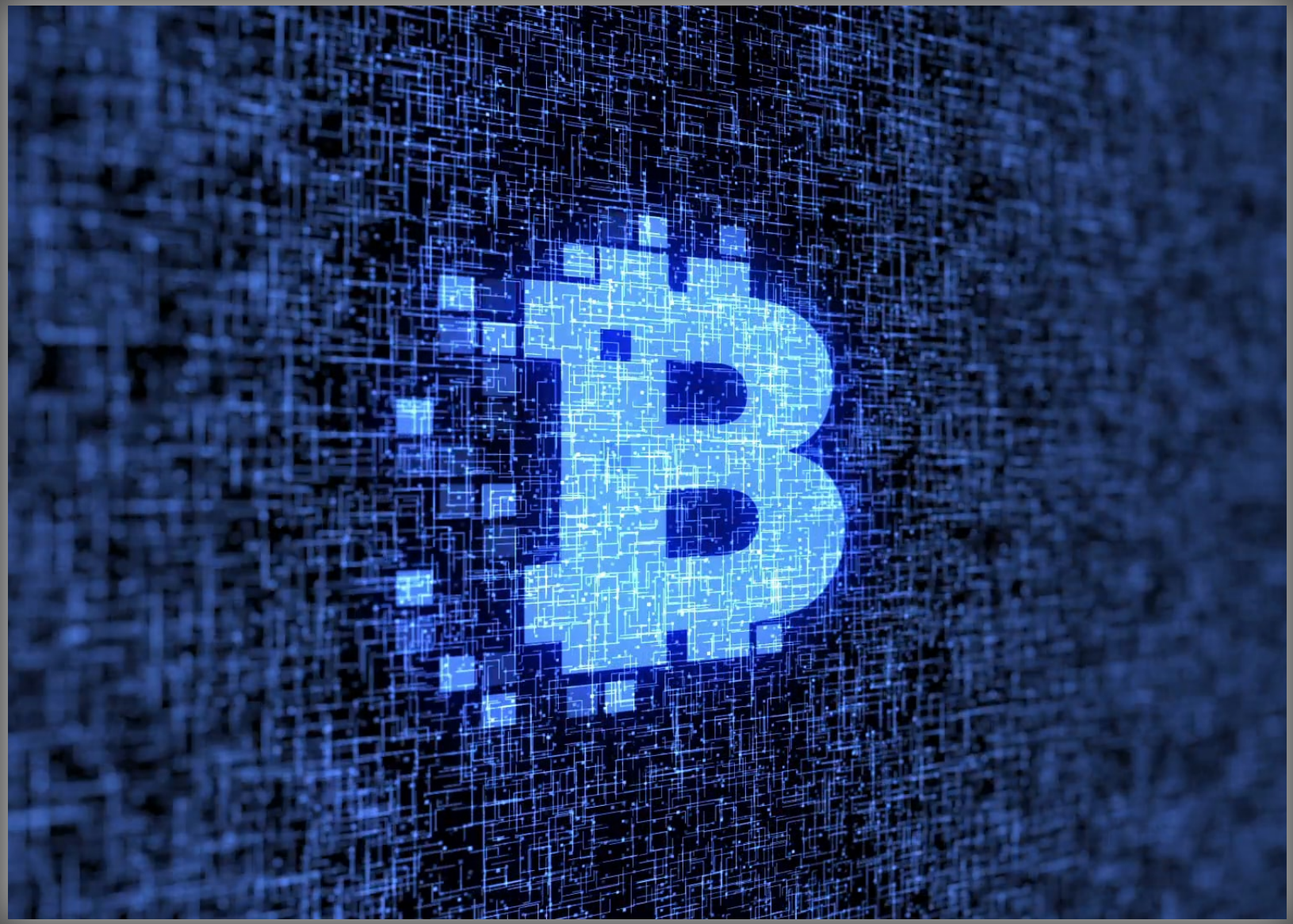
Haskell

Solidity

Bitcoin script

Plutus

Michelson



CSL

DAML

Corda UC

Financial DSLs aren’t new

We have a model …

Composing contracts: an adventure in financial engineering

Functional pearl

Simon Peyton Jones
Microsoft Research, Cambridge
simonpj@microsoft.com

Jean-Marc Eber
LexiFi Technologies, Paris
jeanmarc.eber@lexifi.com

Julian Seward
University of Glasgow
v-sewardj@microsoft.com

23rd August 2000

Abstract

Financial and insurance contracts do not sound like promising territory for functional programming and formal semantics, but in fact we have discovered that insights from programming languages bear directly on the complex subject of describing and valuing a large class of contracts.

We introduce a combinator library that allows us to describe such contracts precisely, and a compositional denotational semantics that says what such contracts are worth.

At this point, any red-blooded functional programmer should start to foam at the mouth, yelling "build a combinator library". And indeed, that turns out to be not only possible, but tremendously beneficial.

The finance industry has an enormous vocabulary of jargon for typical combinations of financial contracts (swaps, futures, caps, floors, swaptions, spreads, straddles, captions, European options, American options, ...the list goes on). Treating each of these individually is like having a large catalogue of prefabricated components. The trouble is that

Example: escrow contract

Alice wants to buy a cat from Bob,
but neither of them trusts the other.

They both trust Carol, though.

Example: escrow contract

Alice wants to buy a cat from Bob,

but neither of them trusts the other.

They both trust Carol, though.

Example: escrow contract

Alice wants to buy a cat from Bob,

but neither of them trusts the other.

They both trust Carol, though.

Example: escrow contract

Alice wants to buy a cat from Bob,

but neither of them trusts the other.

They both trust Carol, though.

Example: escrow contract

Alice wants to buy a cat from Bob,

but neither of them trusts the other.

They both trust Carol, though.

Example: escrow contract

Alice wants to buy a cat from Bob,
but neither of them trusts the other.

They both trust Carol, though.

Example: escrow contract

Alice wants to buy a cat from Bob,

but neither of them trusts the other.

They both trust Carol, though.

Example: escrow contract

Alice wants to buy a cat from Bob,

but neither of them trusts the other.

They both trust Carol, though.

Example: escrow contract

Alice wants to buy a cat from Bob,
but neither of them trusts the other.

They both trust Carol, though.

Example

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Example

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Example

When this condition becomes true, do this …

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Example

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

A choice of two options, depending on this value

Onto blockchain

Onto blockchain

Enforcement

The legal system ensures financial contracts …

… but should a contract on blockchain enforce itself?

Onto blockchain

Enforcement

The legal system ensures financial contracts …

… but should a contract on blockchain enforce itself?

Double spend

Blockchain designed to prevent spending the same money twice …

… but that's precisely how credit works.

Operation

Modality: “pull” not “push”

The contract does not make actions happen…

… but instead allows them to take place.

Operation

Modality: “pull” not “push”

The contract does not make actions happen…

… but instead allows them to take place.

Correctness

Not just a matter of avoiding bad behaviour …

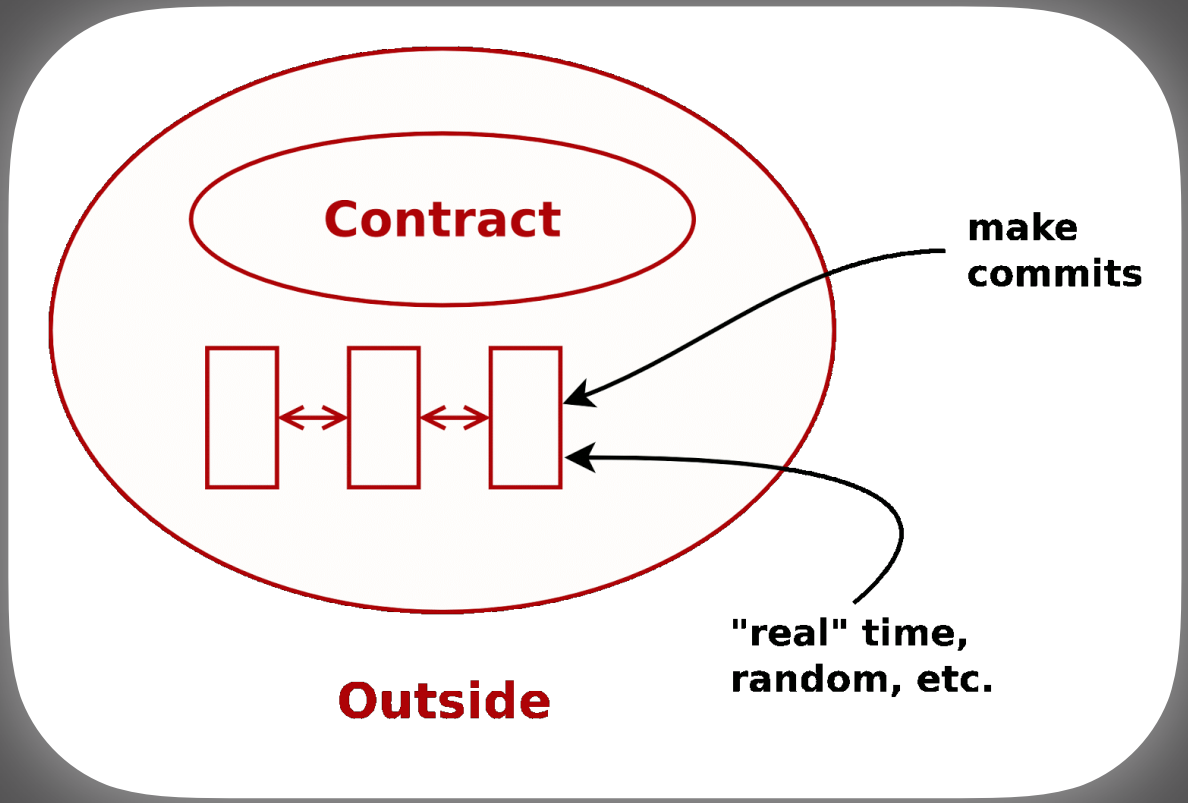
… but also make sure that good things happen.

Interactions with the outside world

“The spot price of oil in Oslo at 12:00, 31-12-17”.

Random values.

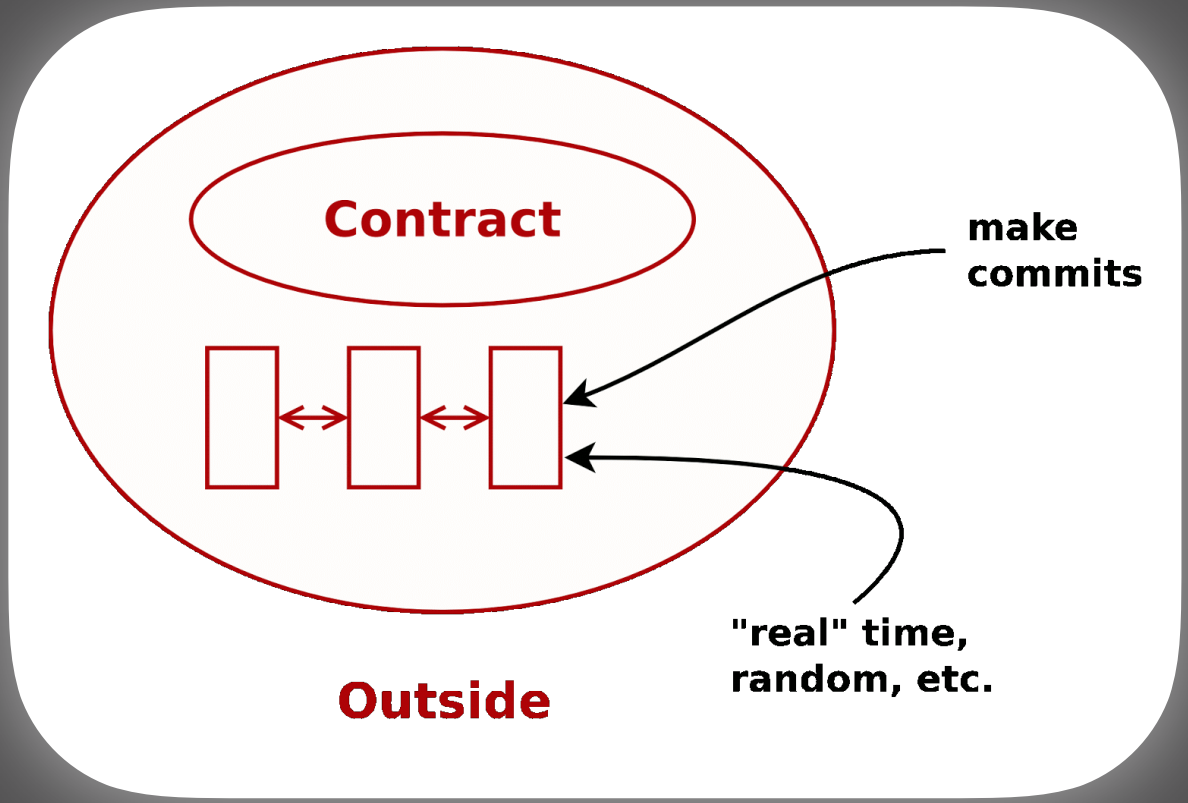
Participants sign-off, …



Commitments

Commit a certain amount of cash for a finite time.

Need to avoid “walk away” …



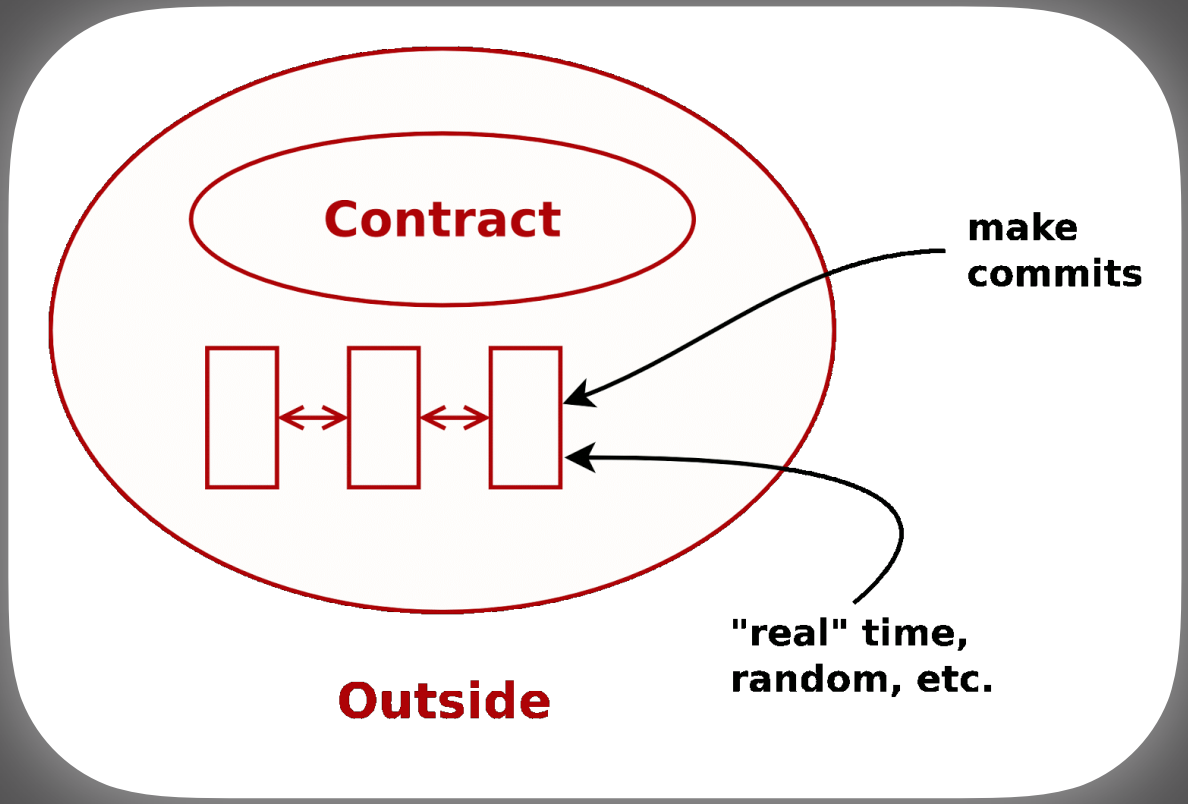
Commitments and timeouts

Commit a certain amount of cash for a finite time.

Need to avoid “walk away” …

We don’t *require* a commitment: can only *ask for* one …

… and only wait a bounded time for the commitment to be made.



Escrow – timeouts

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Escrow – timeouts

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      90
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Escrow – timeouts

Timeout: waiting ends at this block height.

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      90
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Escrow – timeouts

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      90
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original)
      redeem_original)
```

Escrow – timeouts

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      90
      (Choice (majority_chose p
              (Pay alice bob Av
              redeem_original))
      redeem_original)
```

Do this at timeout, if trigger hasn't happened.

Escrow – commitments

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      90
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original)
      redeem_original)
```

Escrow – commitments

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      90
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original)
      redeem_original)
```

Escrow – commitments

```
(CommitCash id1 alice 15000 10 100
                (When (Or (majority_chose refund)
                          (majority_chose pay))
                      90
                      (Choice (majority_chose pay)
                              (Pay alice bob AvailableMoney)
                              redeem_original)
                      redeem_original)

                Null)
```

Escrow – commitments

Waits for Alice to make a commitment of 15k ADA

```
(CommitCash id1 alice 15000 10 100
                (When (Or (majority_chose refund)
                          (majority_chose pay))
                      90
                      (Choice (majority_chose pay)
                              (Pay alice bob AvailableMoney)
                              redeem_original)
                      redeem_original)

                Null)
```

Escrow – commitments

Waits for Alice to make a commitment of 15k ADA

Commitment until slot height 100

```
(CommitCash id1 alice 15000 10 100
                (When (Or (majority_chose refund)
                          (majority_chose pay))
                      90
                      (Choice (majority_chose pay)
                              (Pay alice bob AvailableMoney)
                              redeem_original)
                      redeem_original)

                Null)
```

Escrow – commitments

Waits for Alice to make a commitment of 15k ADA

Commitment until slot height 100

Can be redeemed after that

```
(CommitCash id1 alice 15000 10 100
             (When (Or (majority_chose refund)
                       (majority_chose pay))
                   90
                   (Choice (majority_chose pay)
                           (Pay alice bob AvailableMoney)
                           redeem_original)
                   redeem_original)

             Null)
```

Escrow – commitments

Waits for Alice to make a commitment of 15k ADA

Commitment until slot height 100

Can be redeemed after that

Only wait until slot height 10…

```
(CommitCash id1 alice 15000 10 100
                (When (Or (majority_chose refund)
                          (majority_chose pay))
                      90
                      (Choice (majority_chose pay)
                              (Pay alice bob AvailableMoney)
                              redeem_original)
                      redeem_original)

                Null)
```

Escrow – commitments

```
(CommitCash id1 alice 15000 10 100
             (When (Or (majority_chose refund)
                       (majority_chose pay))
                   90
                   (Choice (majority_chose pay)
                           (Pay alice bob AvailableMoney)
                           redeem_original)
                   redeem_original)

             Null)
```

Waits for Alice to make a commitment of 15k ADA

Commitment until slot height 100

Can be redeemed after that

Only wait until slot height 10…

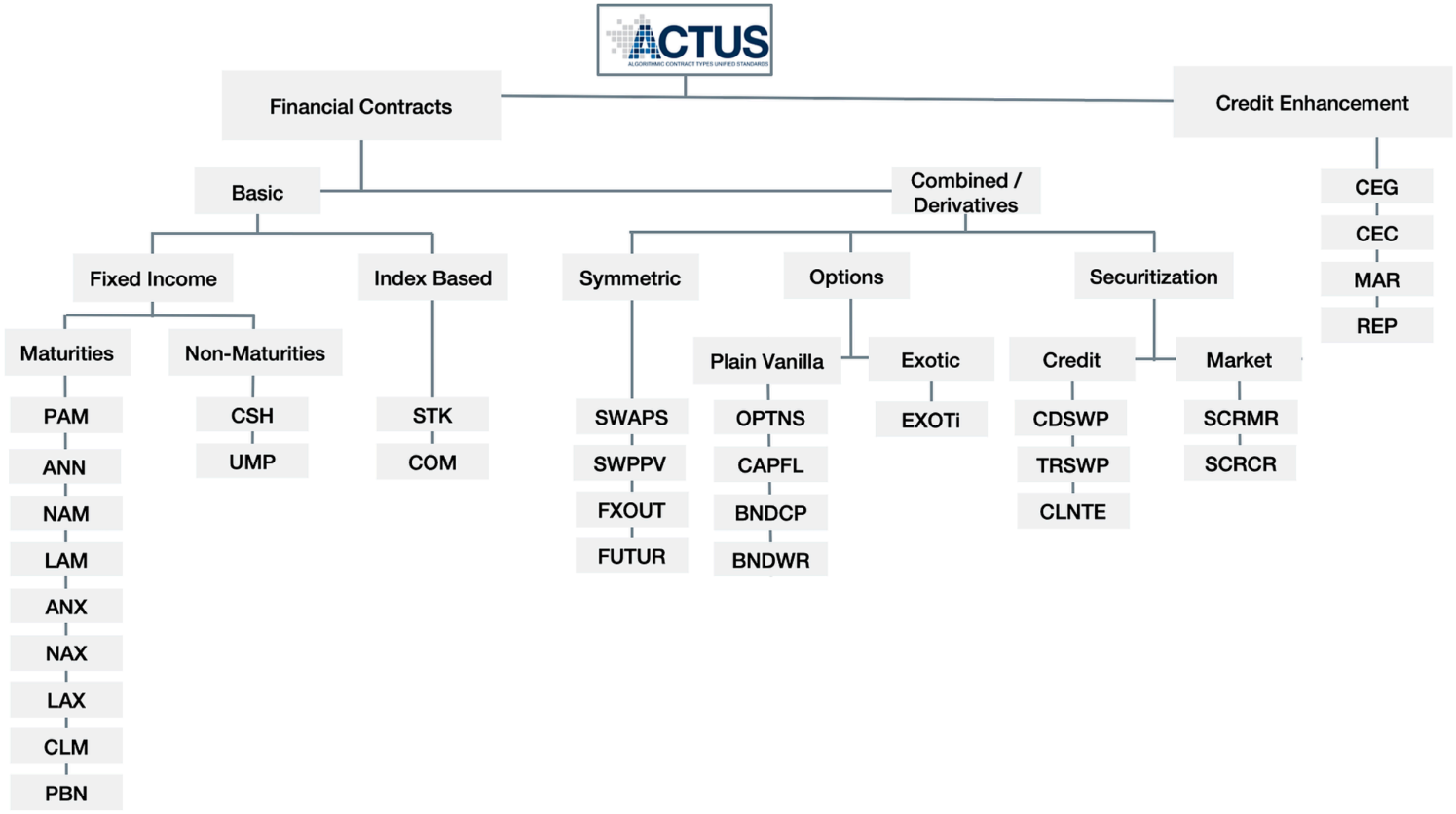
… and if no commitment do this.

ACTUS
ALGORITHMIC CONTRACT TYPES UNIFIED STANDARDS

www.actusfrf.org

ACTUS TAXONOMY

The figure below shows the taxonomy of financial contracts as suggested by ACTUS. The ACTUS Taxonomy orients on building classes of financial contracts that are homogenous in terms of the cash flow (or business events more generally) pattern they implement.



How should we write real contracts?

Embedded DSL

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Embedded DSL

```
(When (Or (majority_chose refund)
          (majority_chose pay))
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Embedded DSL

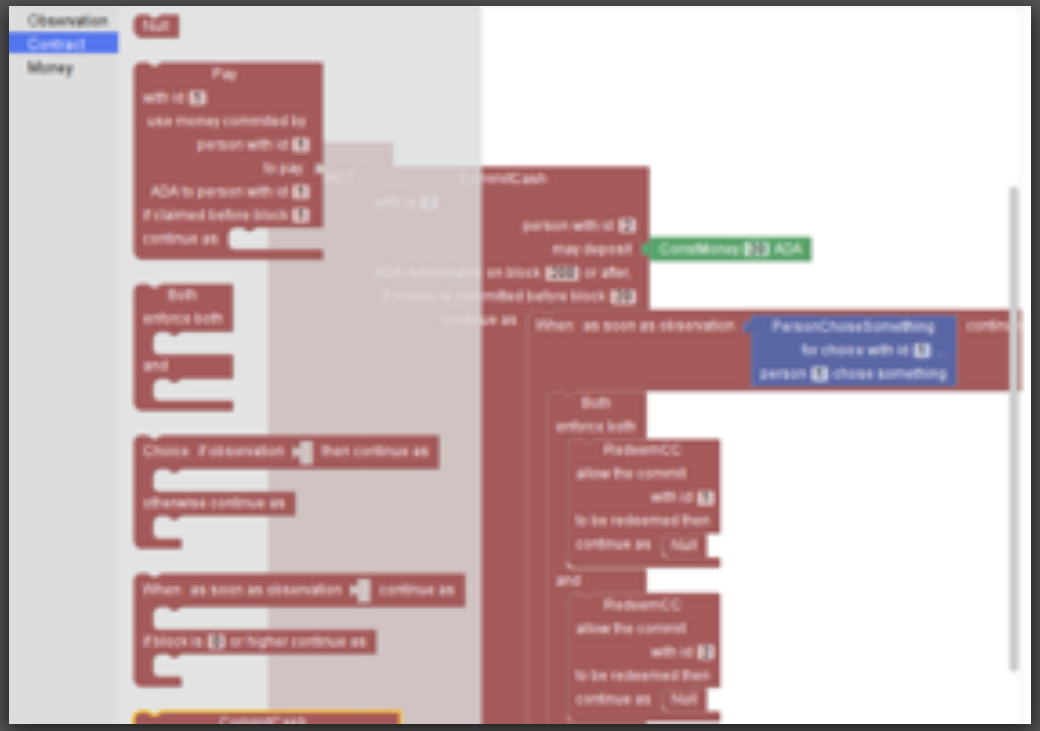
```
(When (Or (majority_chose refund)
          (majority_chose pay))
      (Choice (majority_chose pay)
              (Pay alice bob AvailableMoney)
              redeem_original))
```

Embedded DSL

```
(When (Or (majority_chose refund)
          pay_chosen)
      (Choice pay_chosen
              (Pay alice bob AvailableMoney)
              redeem_original))
 where
 pay_chosen          = majority_chose pay
```

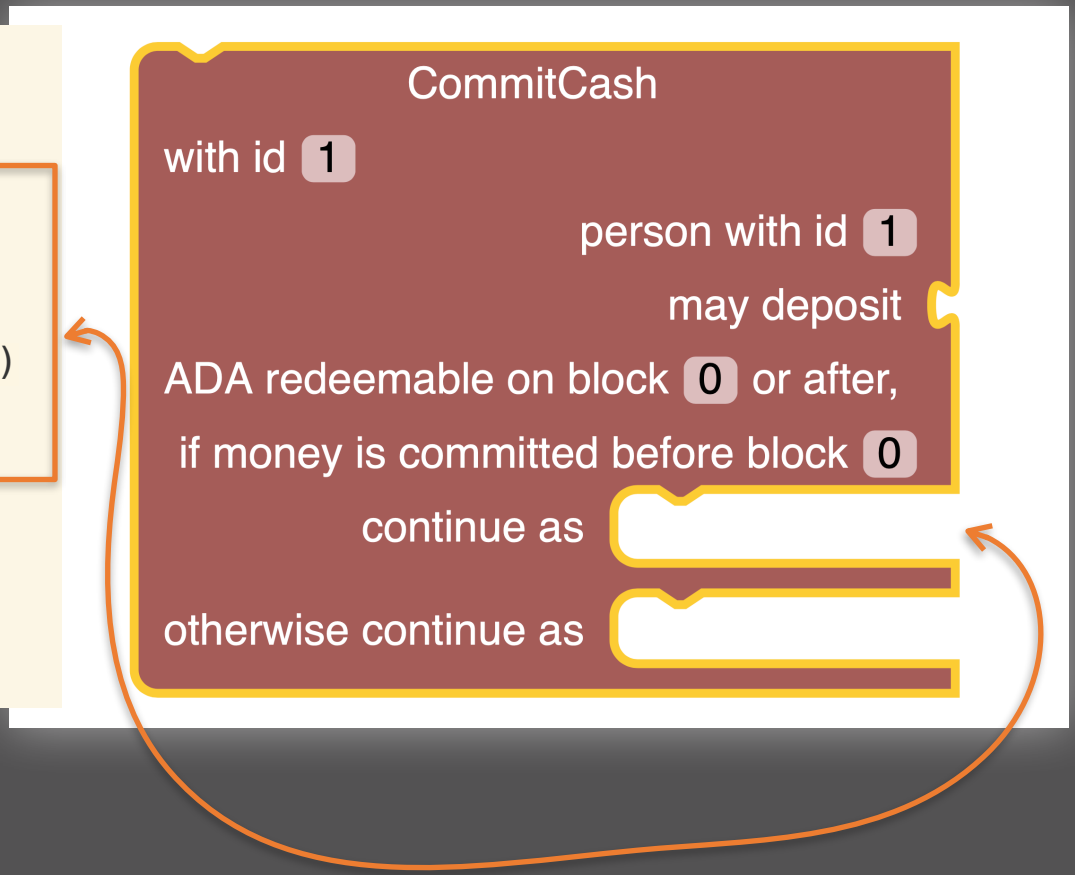
Text or blocks?

```
CommitCash com1 alice ada100 10 200
           (CommitCash com2 bob ada20 20 200
               (When (PersonChoseSomething choice1 alice) 100
                     (Both (RedeemCC com1 Null)
                           (RedeemCC com2 Null))
                     (Pay pay1 bob alice ada20 200
                         (Both (RedeemCC com1 Null)
                               (RedeemCC com2 Null))))
                (RedeemCC com1 Null))
           Null
```



Text or blocks?

```
(CommitCash id1 alice 15000 10 100
            (When (Or (majority_chose refund)
                      (majority_chose pay))
                  90
                  (Choice (majority_chose pay)
                          (Pay alice bob AvailableMoney)
                          redeem_original)
                  redeem_original)
            Null)
```

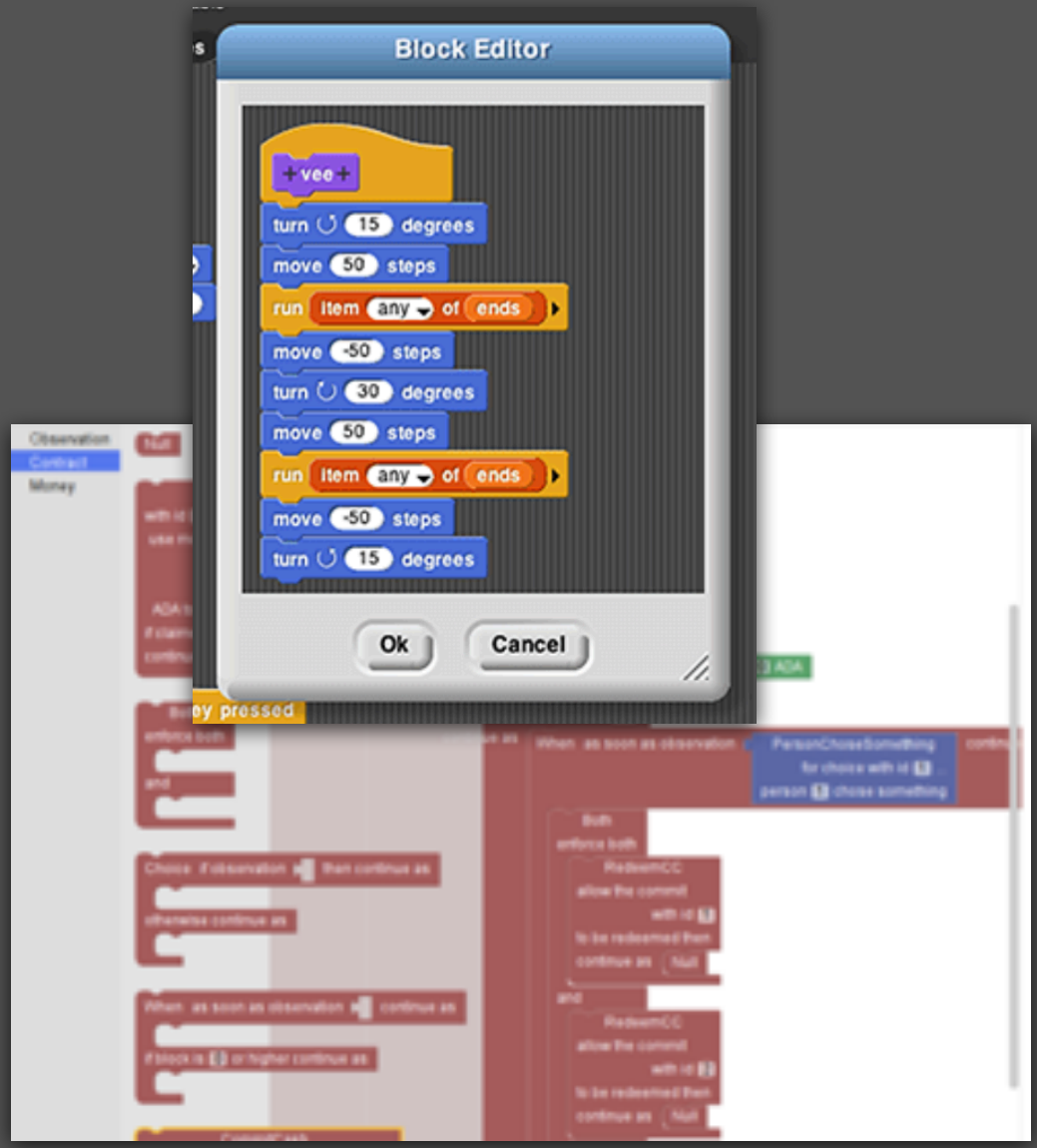


Text or blocks?

Defining your own blocks …

… gives scalability.

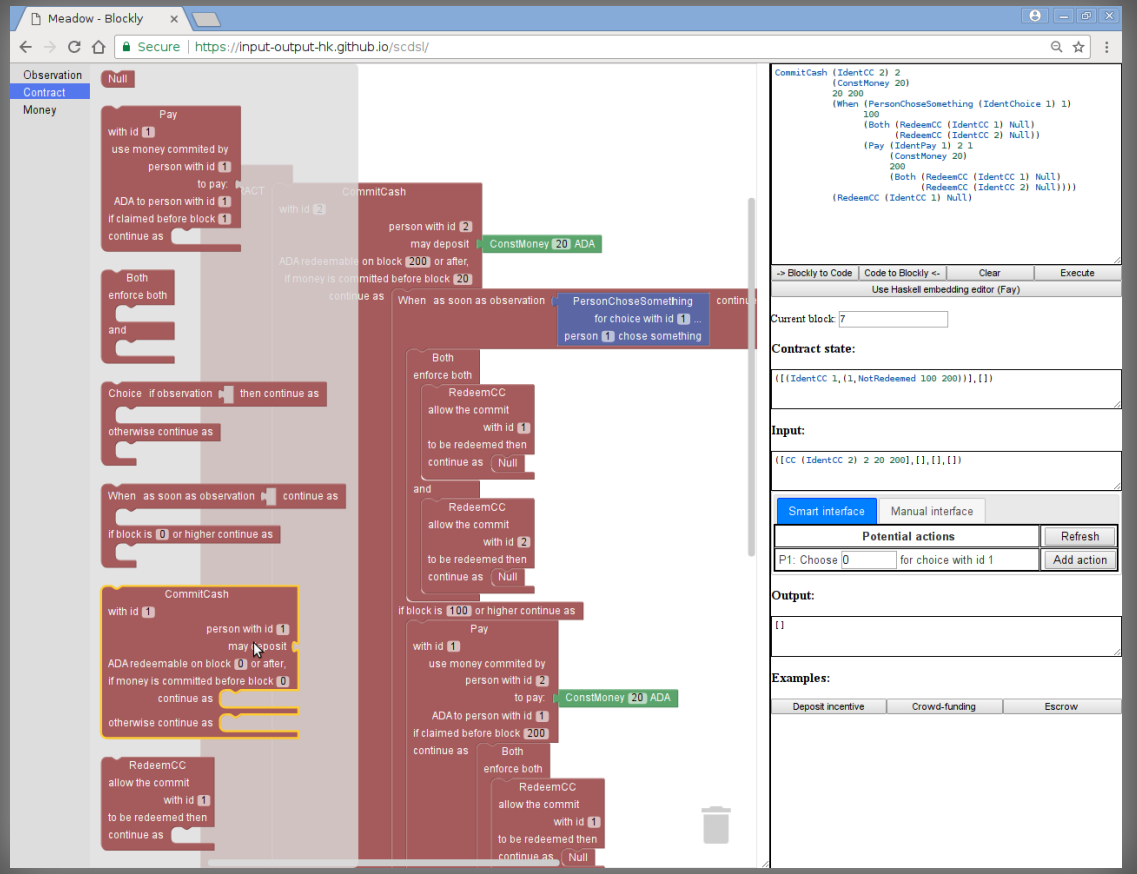
```
CommitCash com1 alice ada100 10 200
           (CommitCash com2 bob ada20 20 200
               (When (PersonChoseSomething choice1 alice) 100
                     (Both (RedeemCC com1 Null)
                           (RedeemCC com2 Null))
                     (Pay pay1 bob alice ada20 200
                         (Both (RedeemCC com1 Null)
                               (RedeemCC com2 Null))))
                (RedeemCC com1 Null))
          Null
```



Meadow: browser-based tool

End-to-end development

- Develop embedded contracts
- Convert to pure Marlowe
- Simulate with smart inputs



Meadow

HASKELL EDITOR | SIMULATION

Demos: CouponBondGuaranteed | Escrow | ZeroCouponBond

```haskell
module Escrow where

import           Marlowe

{-# ANN module "HLint: ignore" #-}

main :: IO ()
main = putStrLn $ prettyPrint contract

------------------------------------
-- Write your code below this line --
------------------------------------

-- Escrow example using embedding

contract :: Contract
contract = Commit 1 iCC1 alice
                  (Constant 450)
                  10 100
                  (When (OrObs (majority_chose refund)
                               (majority_chose pay))
                        90
                        (Choice (majority_chose pay)
                                (Pay 2 iCC1 bob
                                     (Committed iCC1)
                                     100
                                     Null
                                     Null)
                                (redeem_original 3))
                        (redeem_original 4))
                  Null
    where majority_chose = two_chose alice bob carol

-- Participants

alice, bob, carol :: Person
alice = 1
bob = 2
carol = 3

-- Possible votes

refund, pay :: Choice
```

Compile

```
Commit 1 1 1
```

Send to Simulator

```
import           Marlowe

{-# ANN module "HLint: ignore" #-}

main :: IO ()
main = putStrLn $ prettyPrint contract

-------------------------------------
-- Write your code below this line --
-------------------------------------

-- Escrow example using embedding

contract :: Contract
contract = Commit 1 iCC1 alice
                  (Constant 450)
                  10 100
                  (When (OrObs (majority_chose refund)
                               (majority_chose pay))
                        90
                        (Choice (majority_chose pay)
                                (Pay 2 iCC1 bob
                                     (Committed iCC1)
                                     100
                                     Null
                                     Null)
                                (redeem_original 3))
                        (redeem_original 4))
                  Null
    where majority_chose = two_chose alice bob carol

-- Participants

alice, bob, carol :: Person
alice = 1
bob = 2
carol = 3

-- Possible votes
```

Compile

```
Commit 1 1 1
  (Constant 450) 10 100
  (When
     (OrObs
        (OrObs
           (AndObs
              (ChoseThis (1, 1) 0)
              (OrObs
                 (ChoseThis (1, 2) 0)
                 (ChoseThis (1, 3) 0)))
           (AndObs
```

Send to Simulator

Meadow

Tutorial Privacy

HASKELL EDITOR | SIMULATION

Input Composer

Person 1

+ Action 1 : Commit 450 ADA with id 1 to expire by 100

+ Choice 1 : Choose value 0

Person 2

+ Choice 1 : Choose value 0

Person 3

+ Choice 1 : Choose value 0

Transaction Composer

Input list

No inputs in the transaction

Signatures

Person 1 (+0 ADA) Person 2 (+0 ADA) Person 3 (+0 ADA)

Apply Transaction | Next Block | Reset

State

Current Block: 0 Money in contract: 0 ADA

Money owed

No participant is owed money

Commits

There are no commits in the state

Choices

No choices have been recorded

Oracle values

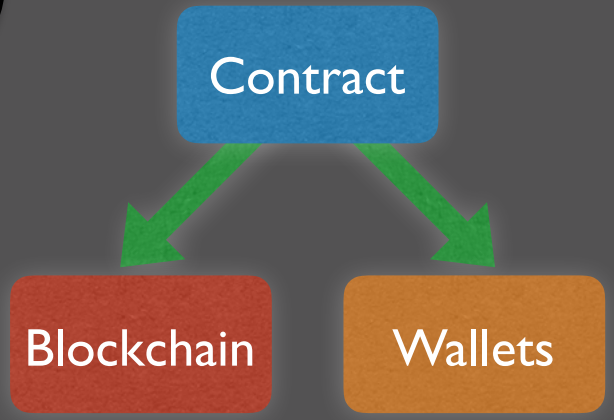
No oracle values have been recorded

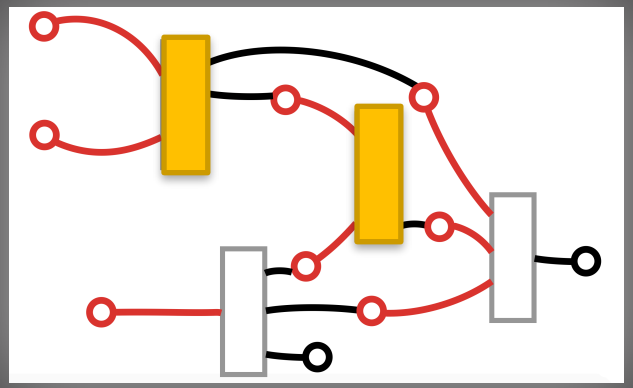
Marlowe Contract

Demos: Crowd Funding | Deposit Incentive | Escrow

```
1  Commit 1 1 1
2    (Constant 450) 10 100
3    (When
4       (OrObs
5          (OrObs
6             (AndObs
7                (ChoseThis (1, 1) 0)
8                (OrObs
9                   (ChoseThis (1, 2) 0)
10                  (ChoseThis (1, 3) 0)))
11            (AndObs
12               (ChoseThis (1, 2) 0)
```

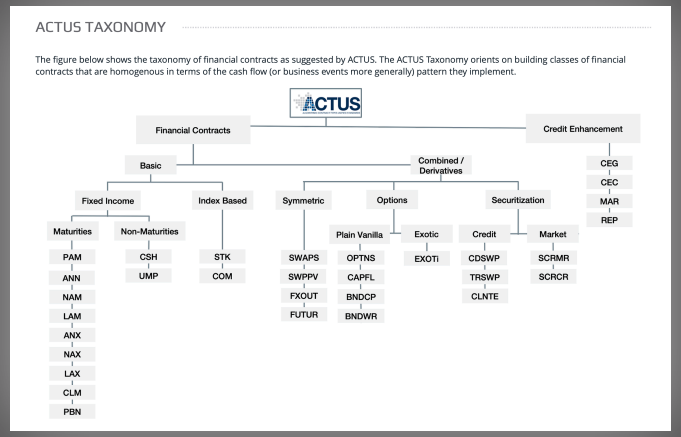
Is it possible for an action to `Fail`?



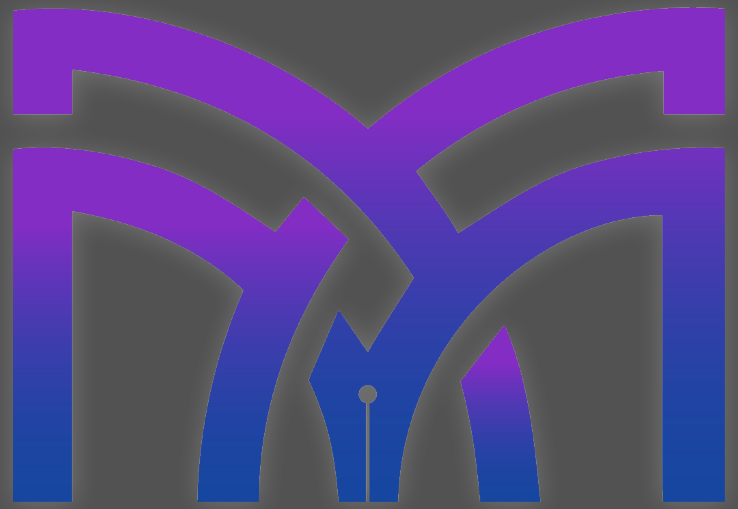


Marlowe

Cardano



It is impossible for this contract to `Fail`.



github.com/input-output-hk/marlowe