Tensile

Ergoscript batch-0 DeCo 2022

Introduction

What is Tensile?

- decentralized, open source and non-custodial trading platform
- offers first and second order derivative contracts
- trading with leverage
- Market hedging and speculative betting

What is a Future contract?

- An agreement to buy/sell something at a future date
- Price and amount are decided in the present
- perpetual/non-perpetual characteristics

P2P non-perpetual Futures with self-provided liquidity

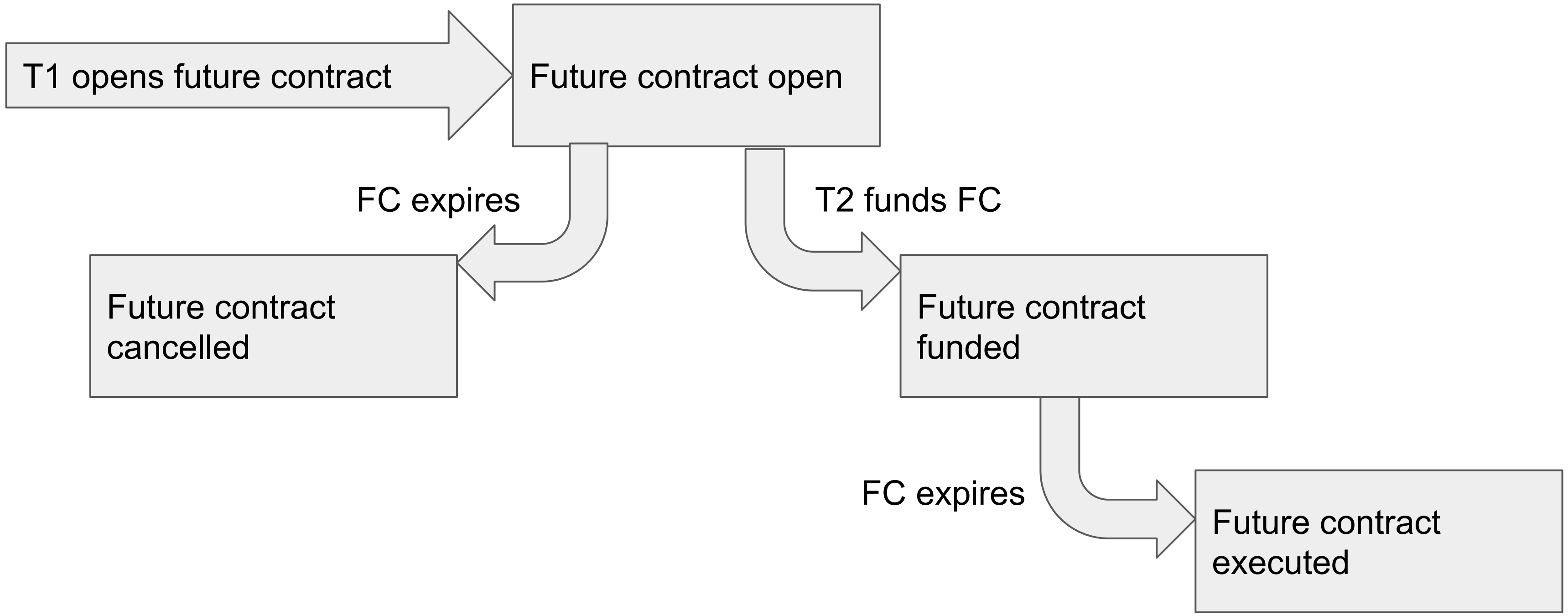
Parties in a long future: Trader 1, Trader 2

1. Trader 1 opens a future contract for offer
    a. T1 sets future expiration date, price and amount of sigUSD to buy
    b. T1 locks in ERG

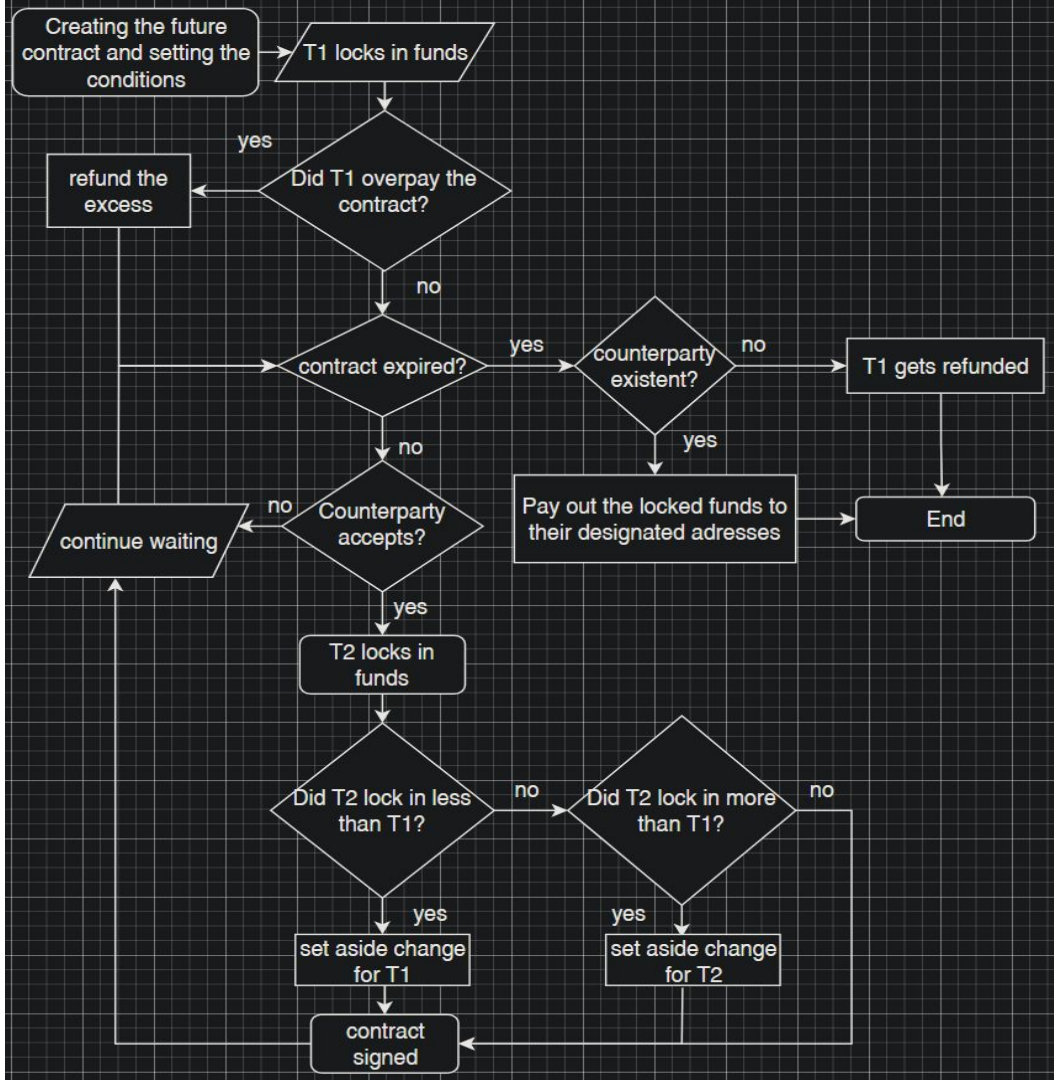
2. Trader 2 accepts future contract before expiration
    a. T2 accepts the contract
    b. T2 locks in sigUSD depending on the future exchange rate provided in the contract

3. At expiration
    a. T1 receives sigUSD and any ERG that have not been sold
    b. T2 receives ERG for the amount of provided sigUSD/rate

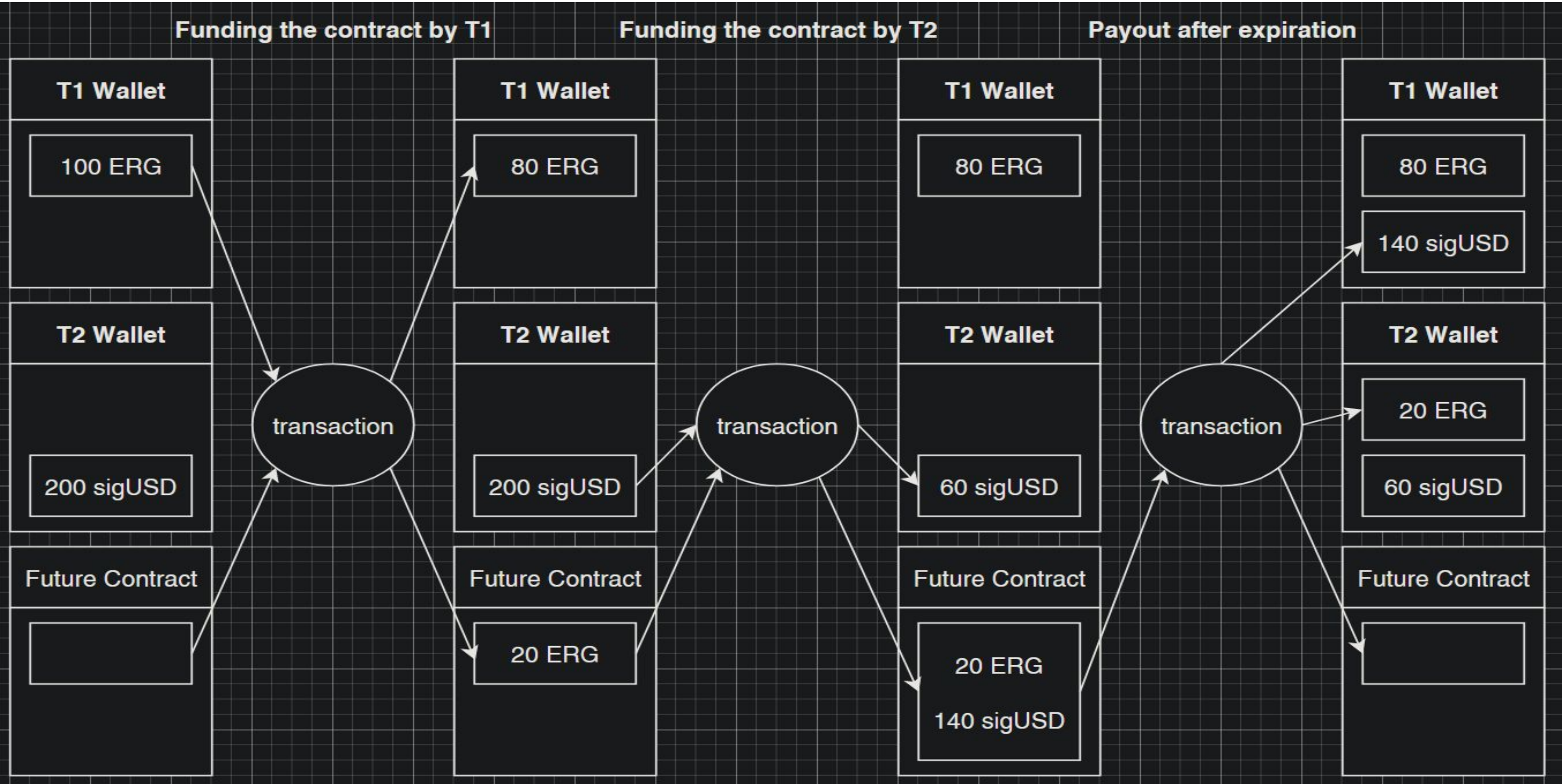
Simple process flow chart



Flow Chart


Creating the future contract and setting the conditions
T1 locks in funds
Did T1 overpay the contract?
yes
refund the excess
no
contract expired?
yes
counterparty existent?
no
T1 gets refunded
yes
Pay out the locked funds to their designated adresses
End
no
Counterparty accepts?
no
continue waiting
yes
T2 locks in funds
Did T2 lock in less than T1?
no
Did T2 lock in more than T1?
no
yes
set aside change for T1
yes
set aside change for T2
contract signed

UTXO transactions



Future Contract Box Guard Script - Registers

1. **Value**: ERG amount to swap
2. **Tokens**: amount to be swapped (can be less than requested)
3. **Registers**:

R4 Job info	jobID	Status (empty, opened, funded)				
Coll(	long,	Coll[Byte]	)			
R5 Opening info	expiry date	exchange rate	amount provided	amount needed	tokenID1	openerpk
Coll(	int,	long,	long,	long,	Coll[Byte])	SigmaProp,
R6 Funding info	funded	funderpk				
Coll(	Boolean,	SigmaProp	)			

Future Contract Box Guard Script - Actions

4.1 Opening a future trade						
INPUTS:	FCB	T1(opener)	**OUTPUS**:	FCB	T1(opener)	Fee
Trigger:	Trader 1(Opener) sends ERG funds to contract and sets Opening info, jobID					
Conditions:	Job Info and Opening info are updated Others stay the same					
4.2 Funding existing contract						
INPUTS:	FCB	T2(funder)	**OUTPUS**:	FCB	T2(funder)	Fee
Trigger:	Trader 2 sends tokens with tokenID to an already opened contract.					
Conditions:	Job Info Status and Funding Info updated. Others stay the same					

Future Contract Box Guard Script - Actions

4.3 Expiration of Contract						
INPUTS:	FCB	**OUTPUS**:	FCB	T1(opener)	T2(funder)	Fee
Trigger:	expiry date reached					
Conditions:	If funded: ERG calculated and send to funder, tokens send to opener, any remaining ERG send to opener, status updated If not funded: ERG sent back to opener, status updated					

ErgoScript code - Intro

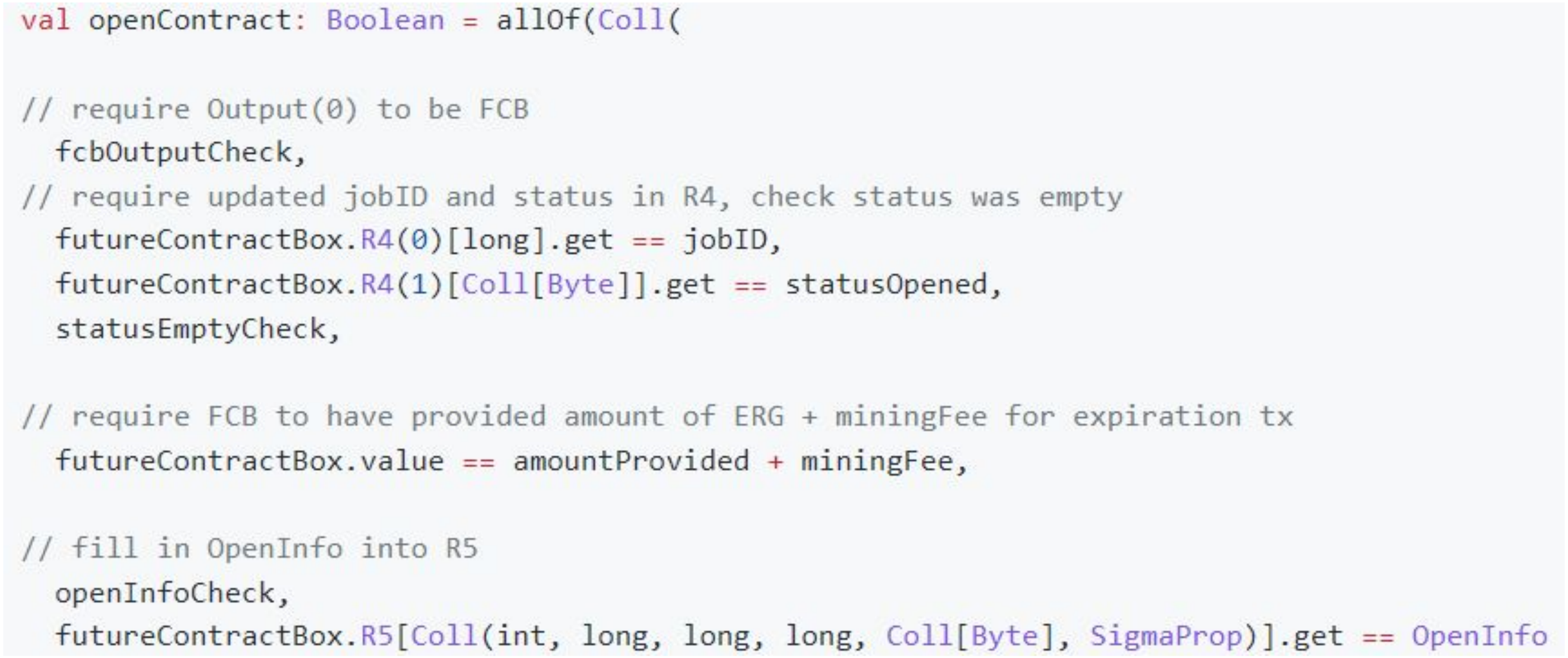
- Draft - not tested
- 3 main spending paths OR-ed at the bottom with:

```
{
[…]

sigmaProp(anyOf(Coll(
  openContract,
  fundContract,
  executeContract
   )))
}
```

- Arbitrary set values that would come from off-chain code

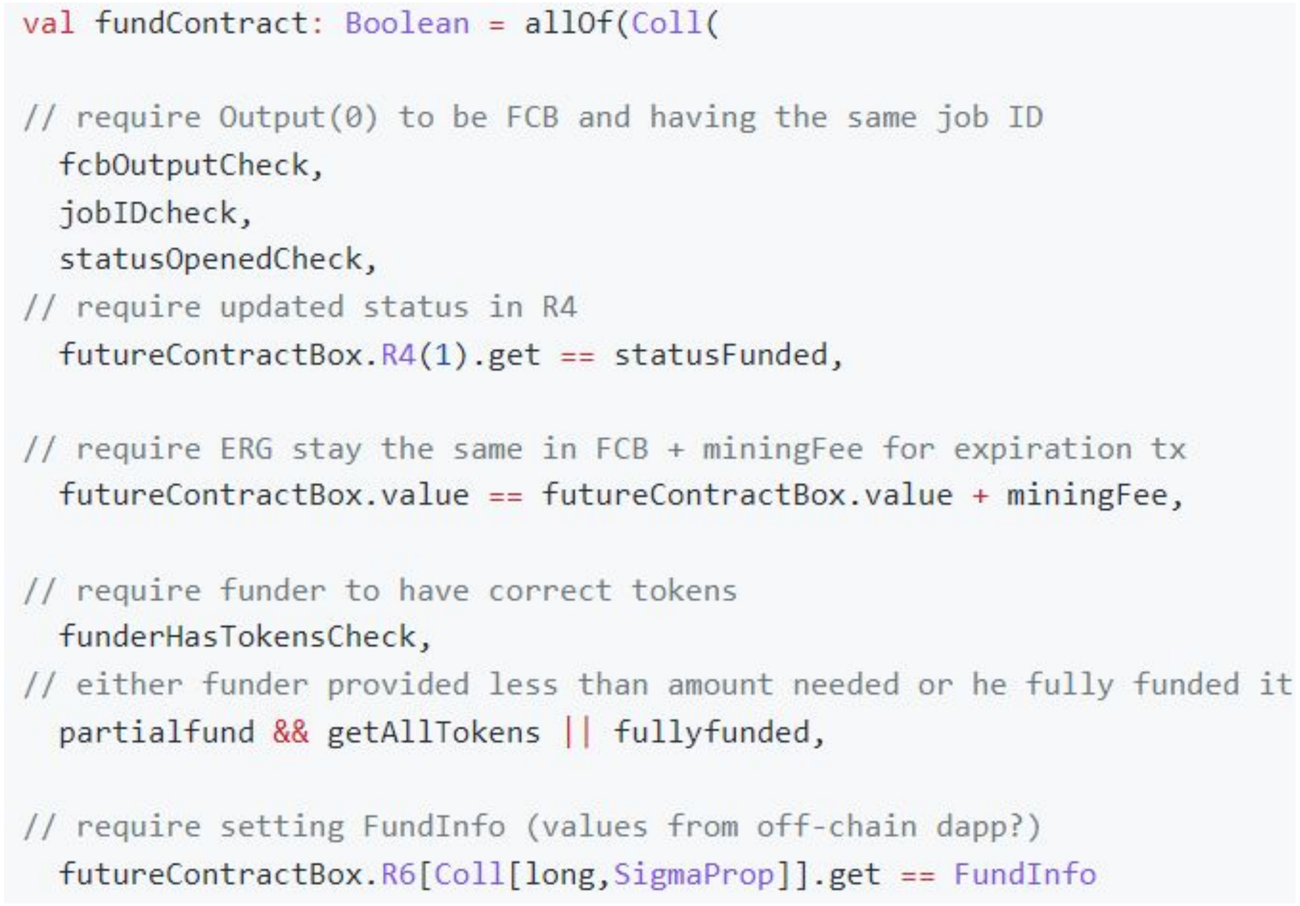
```
{
[...]
val jobID: long = 1234567L
val expirydate: int = 1000000
val exchangeRate: long = 5
val amountProvided: long = 20 * ErgInNanoErg
val amountNeeded: long = exchangeRate *
amountProvided / ErgInNanoErg
val tokenID1: Coll[Byte] =
fromBase58("11111111111111111111111111111111
1")
[…]
}
```

openContract spending conditions



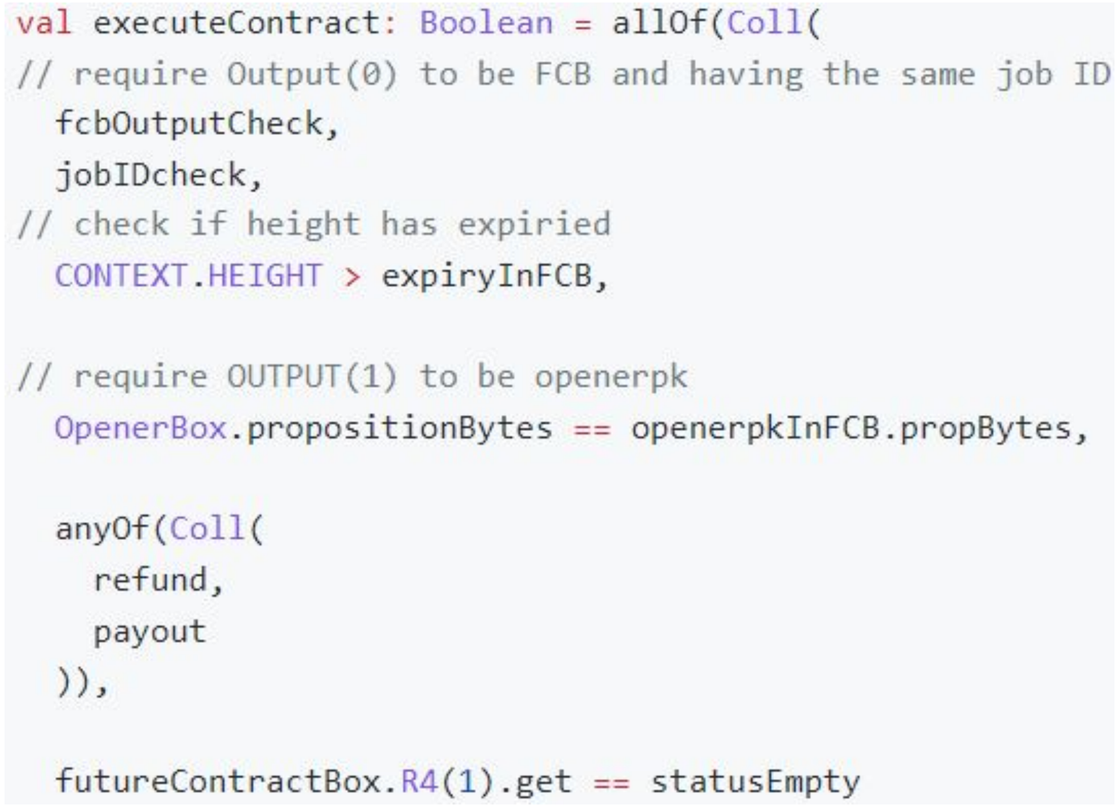
```
val openContract: Boolean = allOf(Coll(

// require Output(0) to be FCB
  fcbOutputCheck,
// require updated jobID and status in R4, check status was empty
  futureContractBox.R4(0)[long].get == jobID,
  futureContractBox.R4(1)[Coll[Byte]].get == statusOpened,
  statusEmptyCheck,

// require FCB to have provided amount of ERG + miningFee for expiration tx
  futureContractBox.value == amountProvided + miningFee,

// fill in OpenInfo into R5
  openInfoCheck,
  futureContractBox.R5[Coll(int, long, long, long, Coll[Byte], SigmaProp)].get == OpenInfo
```

fundContract spending conditions



```
val fundContract: Boolean = allOf(Coll(

// require Output(0) to be FCB and having the same job ID
  fcbOutputCheck,
  jobIDcheck,
  statusOpenedCheck,
// require updated status in R4
  futureContractBox.R4(1).get == statusFunded,

// require ERG stay the same in FCB + miningFee for expiration tx
  futureContractBox.value == futureContractBox.value + miningFee,

// require funder to have correct tokens
  funderHasTokensCheck,
// either funder provided less than amount needed or he fully funded it
  partialfund && getAllTokens || fullyfunded,

// require setting FundInfo (values from off-chain dapp?)
  futureContractBox.R6[Coll[long,SigmaProp]].get == FundInfo
```

executeContract spending conditions



```
val executeContract: Boolean = allOf(Coll(
// require Output(0) to be FCB and having the same job ID
  fcbOutputCheck,
  jobIDcheck,
// check if height has expiried
  CONTEXT.HEIGHT > expiryInFCB,

// require OUTPUT(1) to be openerpk
  OpenerBox.propositionBytes == openerpkInFCB.propBytes,

  anyOf(Coll(
    refund,
    payout
  )),

  futureContractBox.R4(1).get == statusEmpty
```

Discussion summary

- Collection must have elements of same data type
- -> Rearrange Registers
-
- If-scope has to end with sigmaProp, values only available inside that scope
-
- Can create a Box with off-chain code and set register values
- -> no need for empty FCB box
-
- Check number of input boxes - instead if a box exists
-
- Setting futureContractBox.value == futureContractBox.value + miningFee
- will always result to a false

Thanks for your attention!

Notes

- Coll has to be same data type
- If scope has to end with sigmaProp
- Can create a Box with off-chain code
- Check number of input boxes
- Can increase Register size with box re-creation

Future Contract Box Guard Script - Actions

- **4.3 Expiration of Contract**
    - **INPUTS: FCB**
    - **OUTPUTS: FCB, T1 box, T2 box and Mining Fee**
    - **Trigger:**
        - expiry date reached
    - **Contract Conditions:**
        - If funded: ERG send to T2, tokens send to T1, any remaining ERG send to T1, Status updated
        - If not funded: ERG sent back to T1

Future Contract Box Guard Script - Actions

- **4.1 Opening a future trade**
  - **INPUTS: FCB and T1 box -> OUTPUTS: FCB, T1 box (for change) and Mining Fee**
  - **Trigger**: Trader 1 sends ERG funds to contract and sets Opening info, jobID
  - **Contract Conditions:**
    - Job Info Status and Opening info are updated
    - Others stay the same

- **4.2 Funding existing contract**
  - **INPUTS: FCB and T2 box -> OUTPUTS: FCB, T2 box (for change) and Mining Fee**
  - **Trigger:** Trader 2 sends tokens with tokenID to the contract
  - **Contract Conditions:**
    - Job Info Status and Funding Info updated
    - Others stay the same

Future Contract Box Guard Script - Registers

1. **Value**: ERG amount to swap

2. **Tokens**: amount to be swapped (can be less than requested)

3. **Registers**:
    3.1. R4 -> Job info
        3.1.1. Type: Coll(long, Coll[Byte])
        3.1.2. List: *JobID*, Status: opened, funded, expired
    3.2. R5 -> Opening info
        3.2.1. Type: Coll(int, long, long, long, Coll[Byte], SigmaProp)
        3.2.2. List: expirydate, exchangerate, amountProvided, amountNeeded, tokenID1, openerpk
    3.3. R6 -> Funding info
        3.3.1. Type: Coll(Boolean, SigmaProp)
        3.3.2. List: funded, funderpk