

Marlowe - 3 Steps to Working DApp

Steps

- Design
- Test
- Integrate

Project

On-Chain Betting Platform

Steps

- Design
- Test
- Integrate

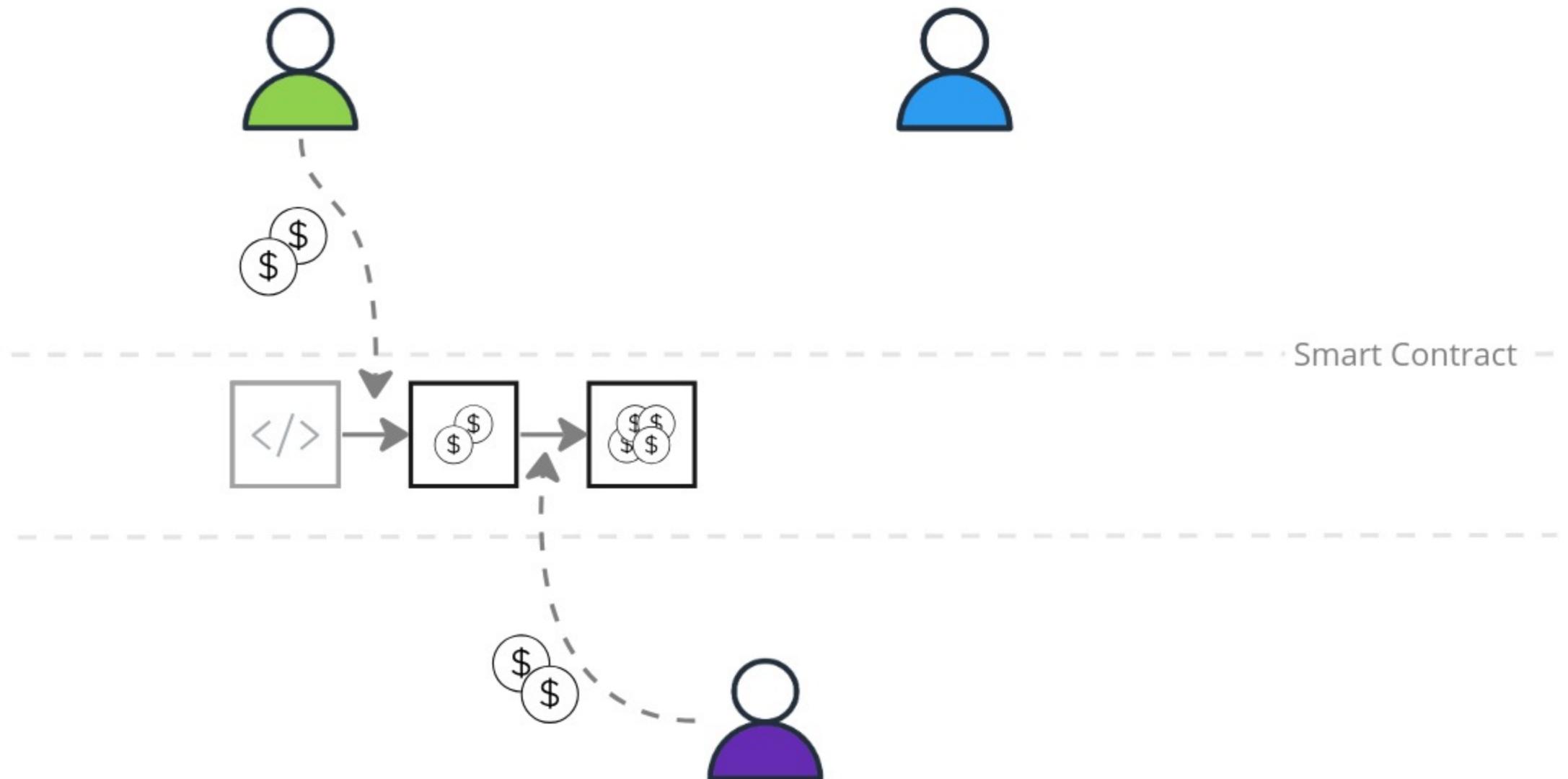
"On-Chain Betting"



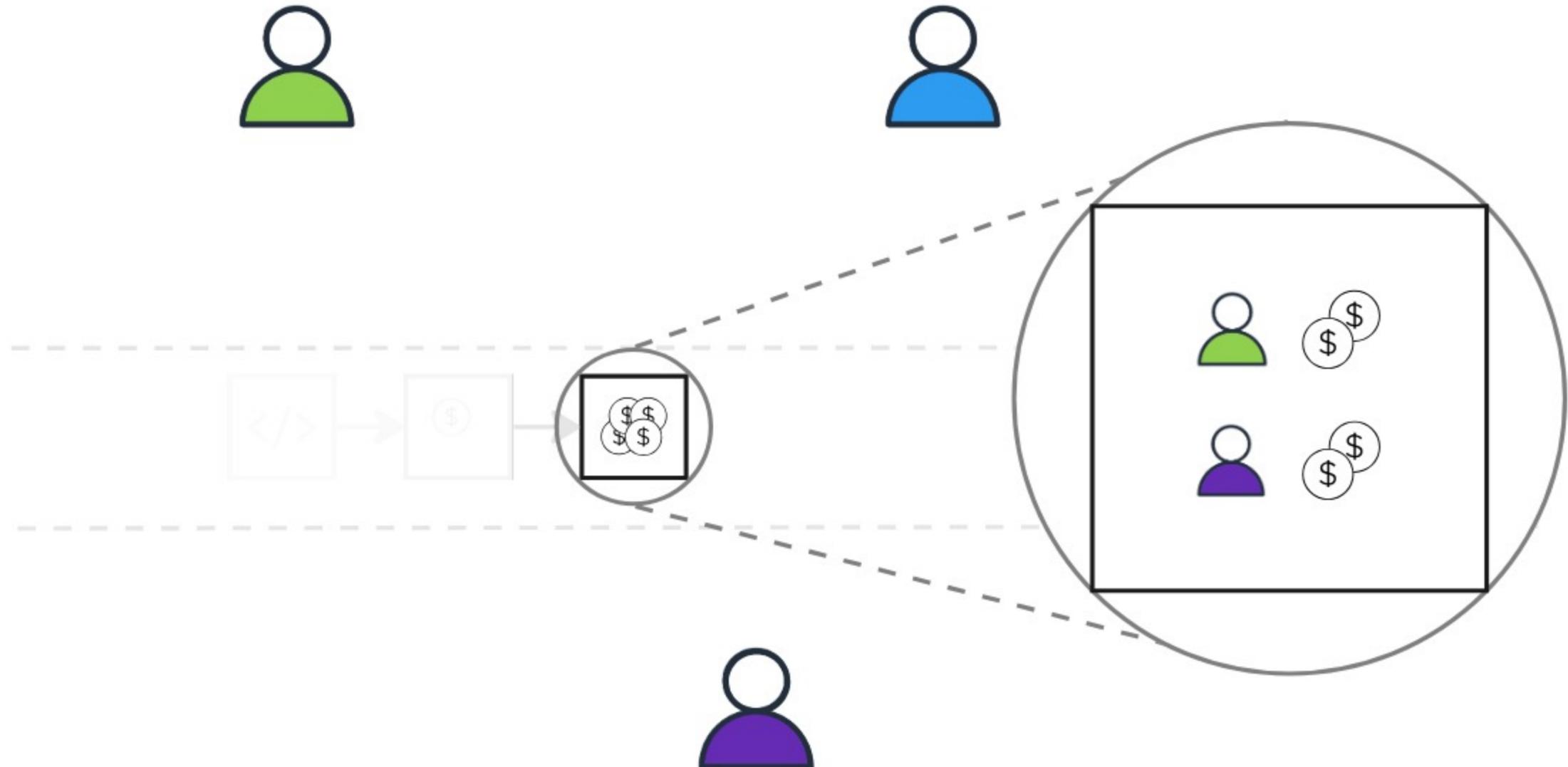
Smart Contract



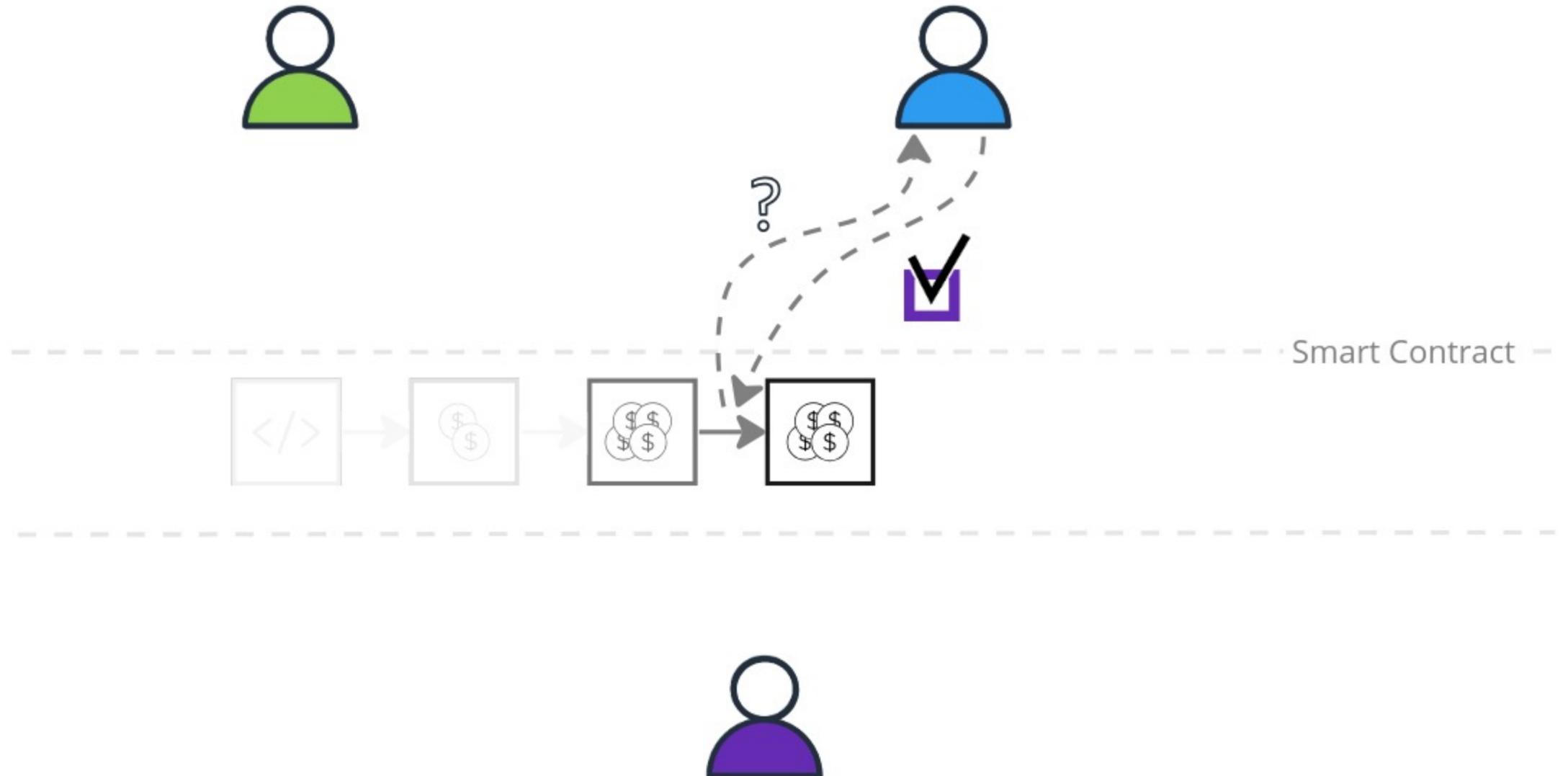
"On-Chain Betting"



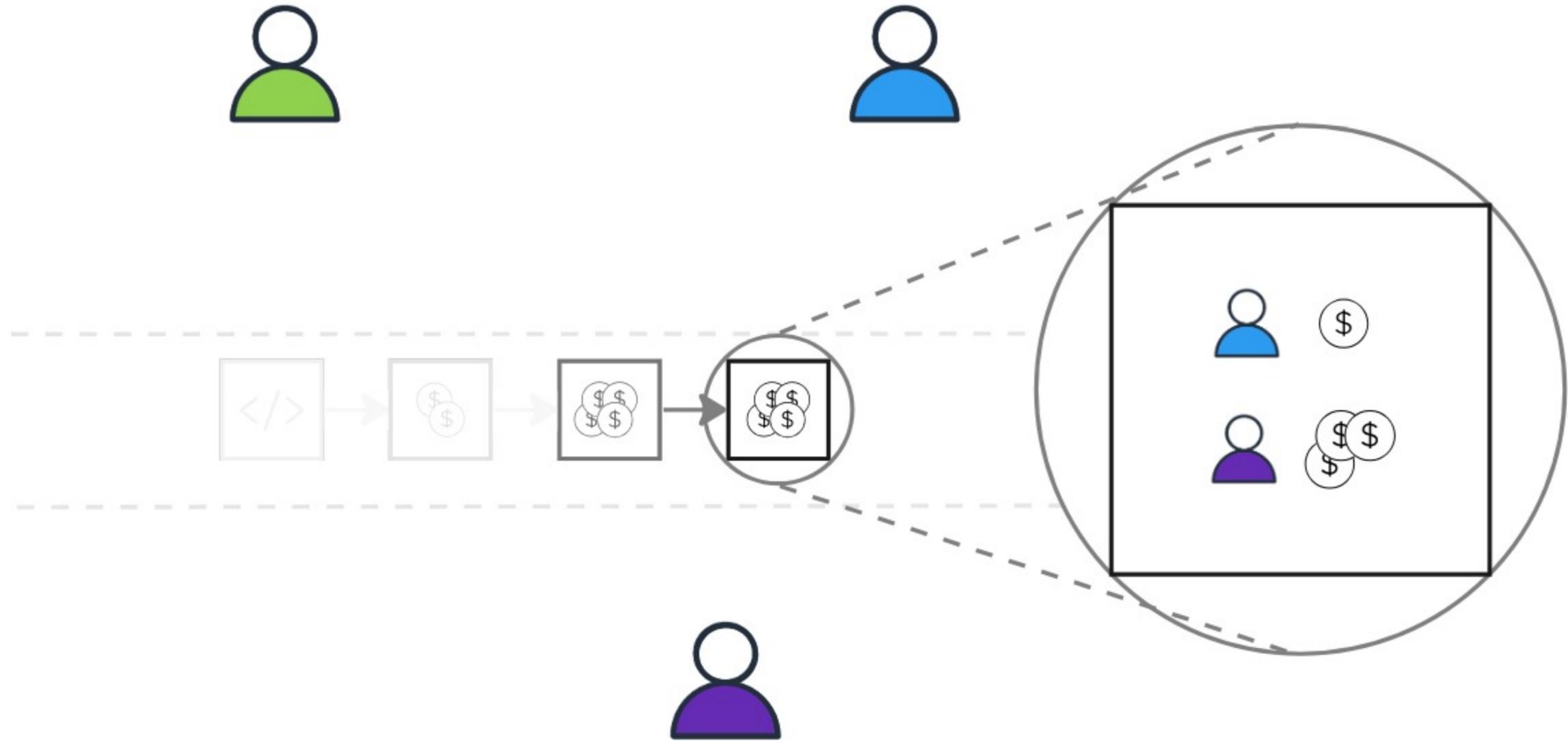
"On-Chain Betting"



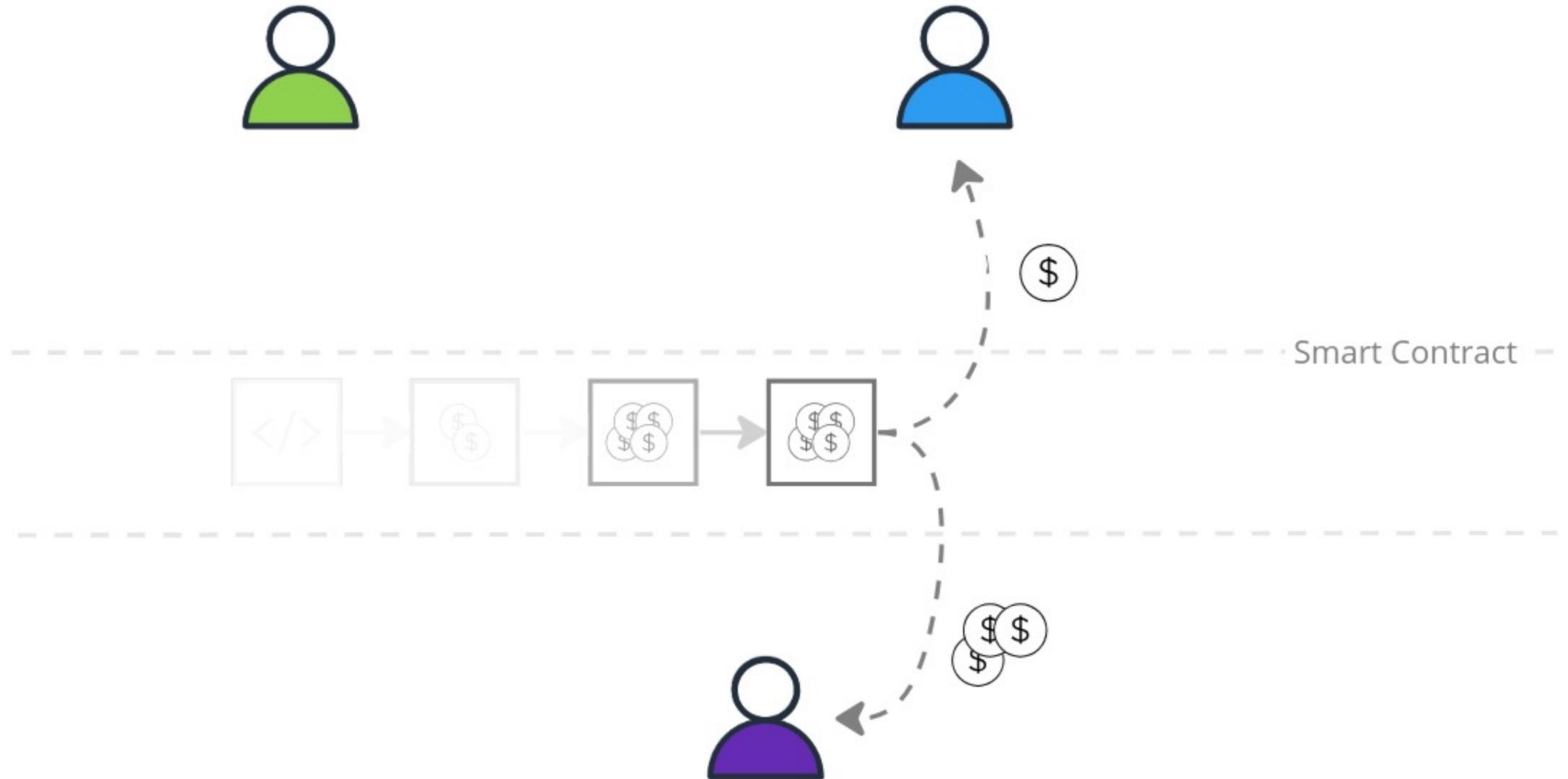
"On-Chain Betting"



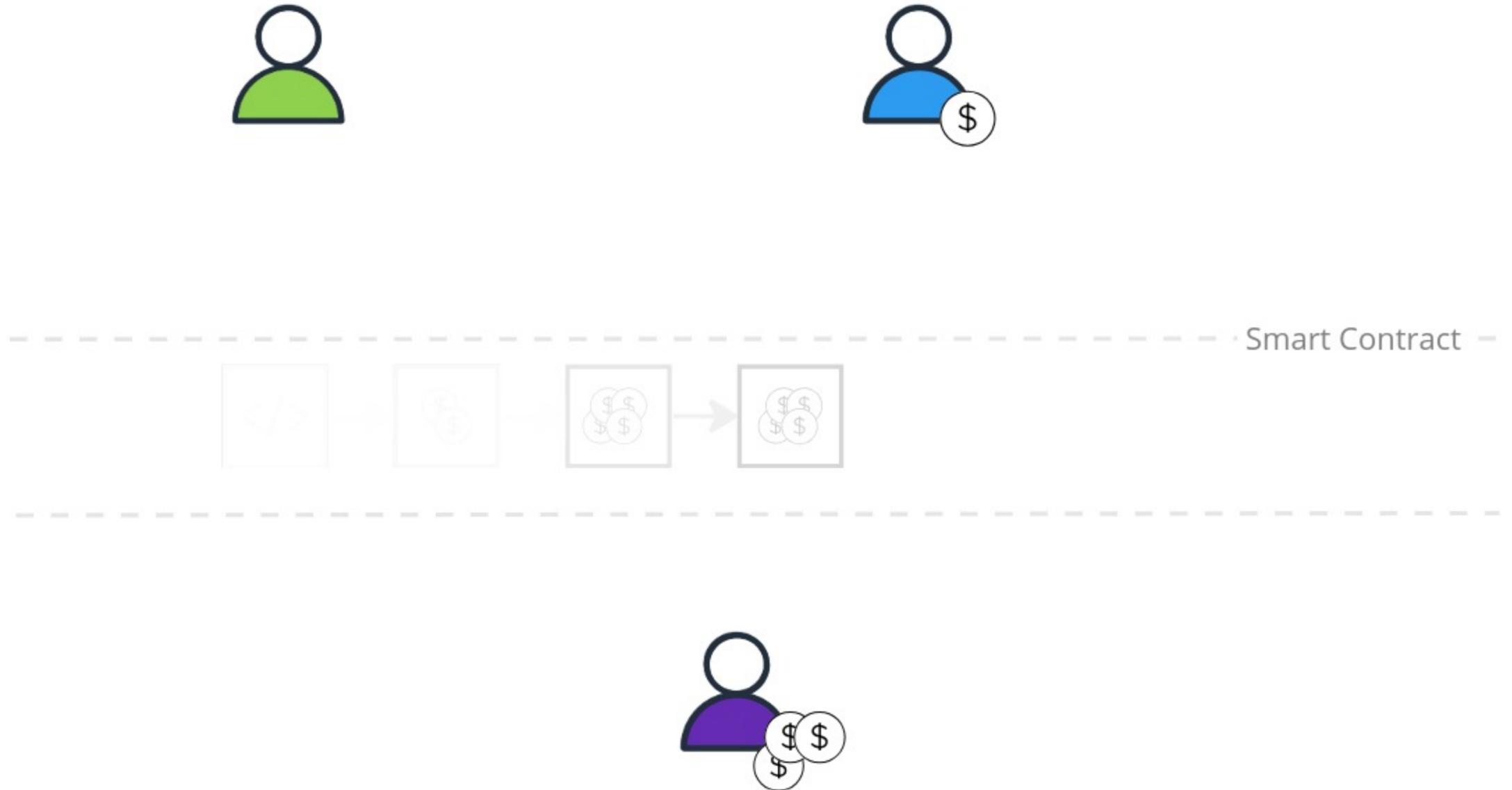
"On-Chain Betting"



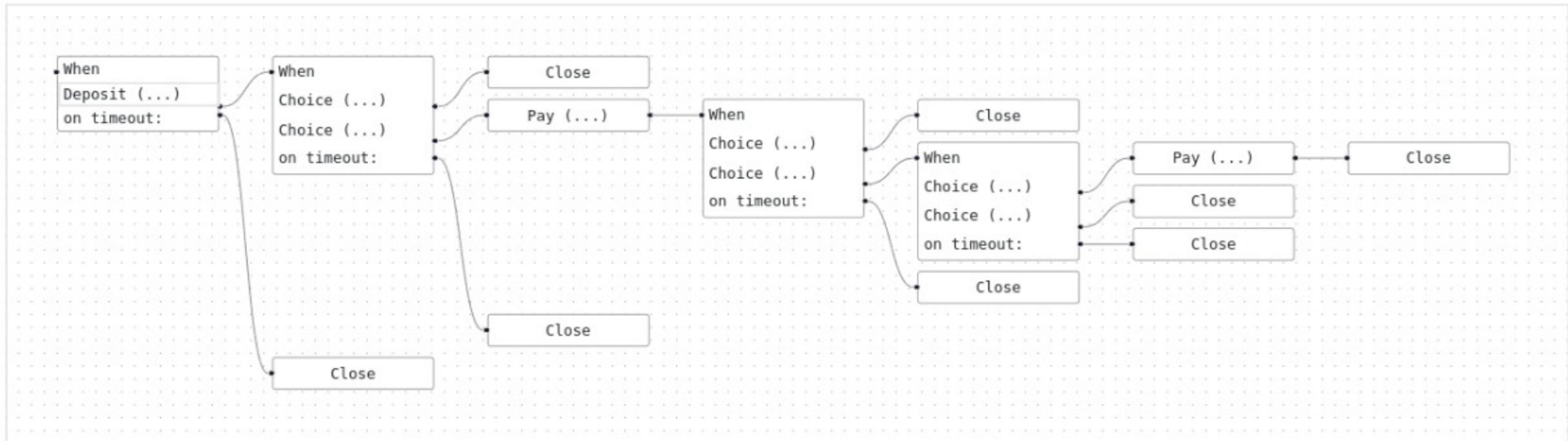
"On-Chain Betting"



"On-Chain Betting"

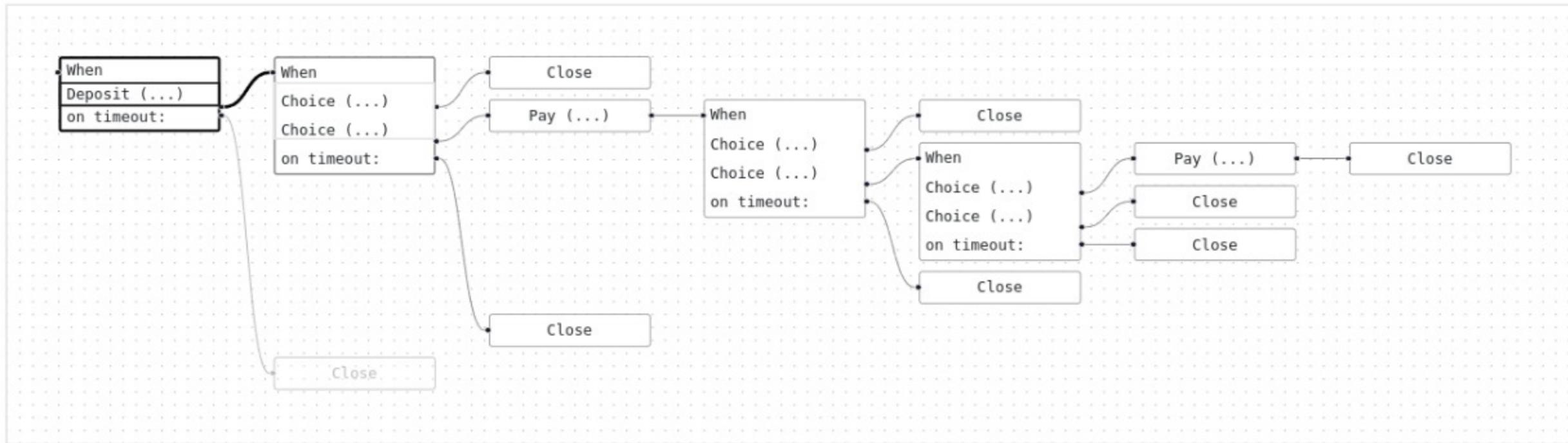


How Marlowe Operates



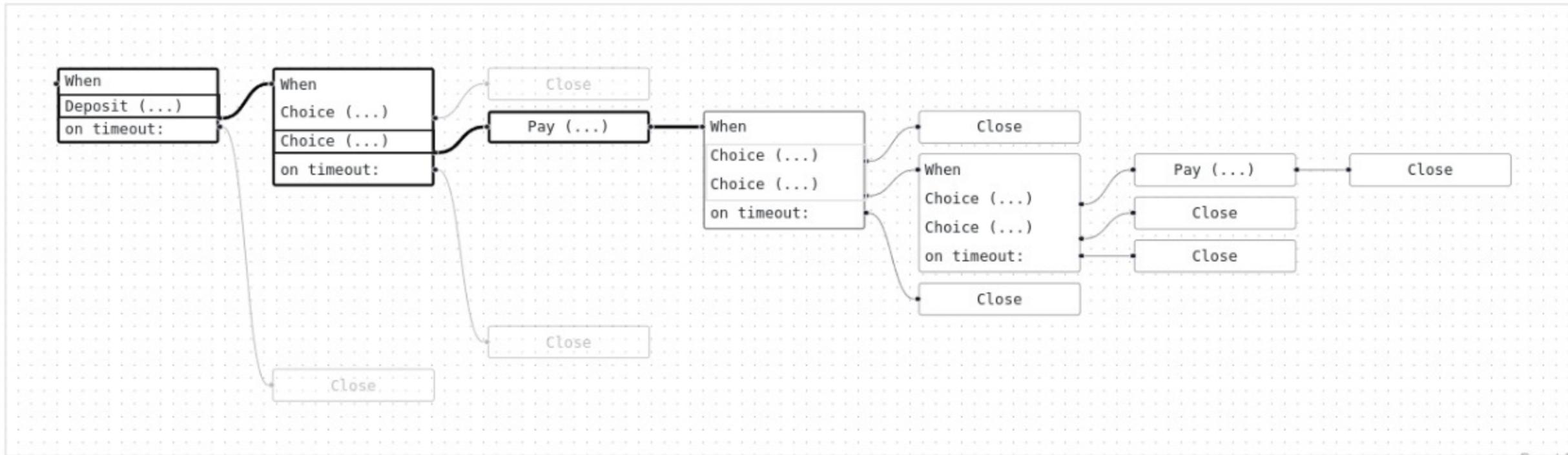
Contract structure is known upfront.

How Marlowe Operates



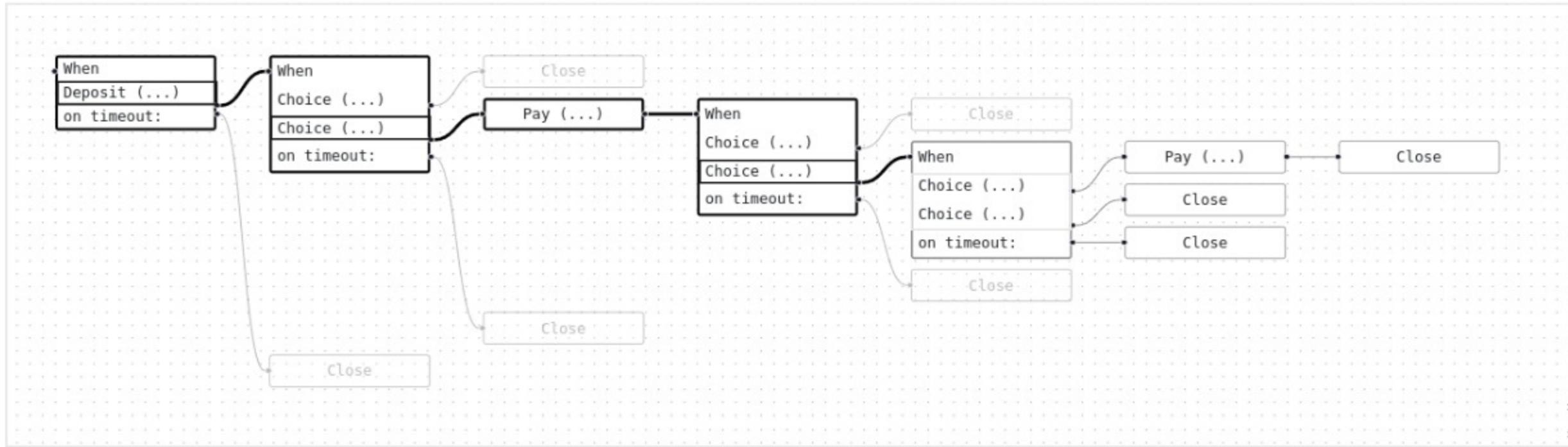
User inputs drives the execution.

How Marlowe Operates



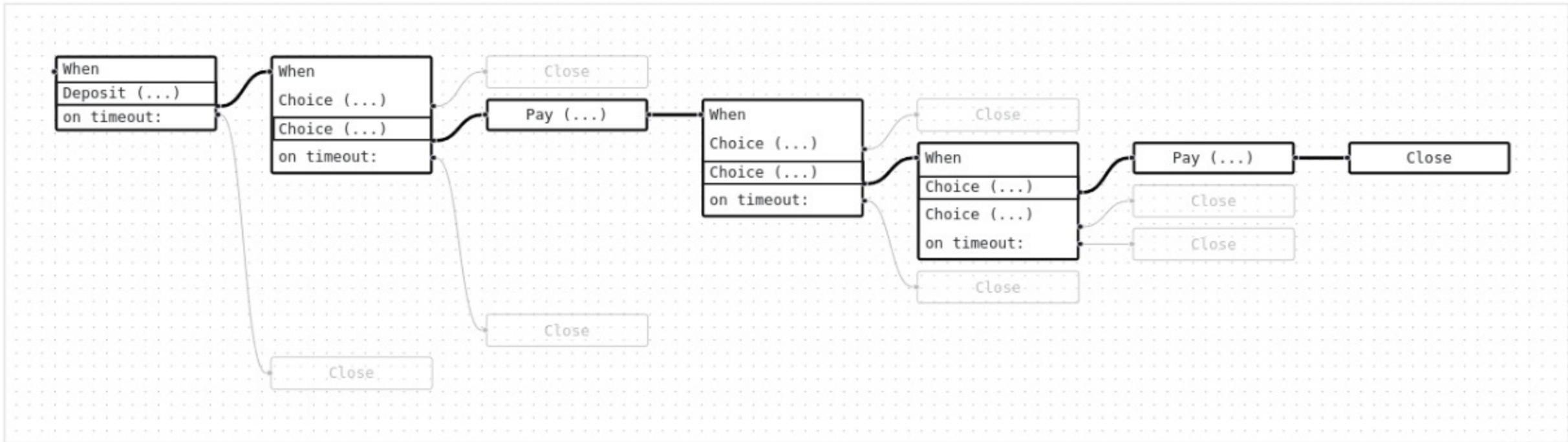
User inputs drives the execution.

How Marlowe Operates



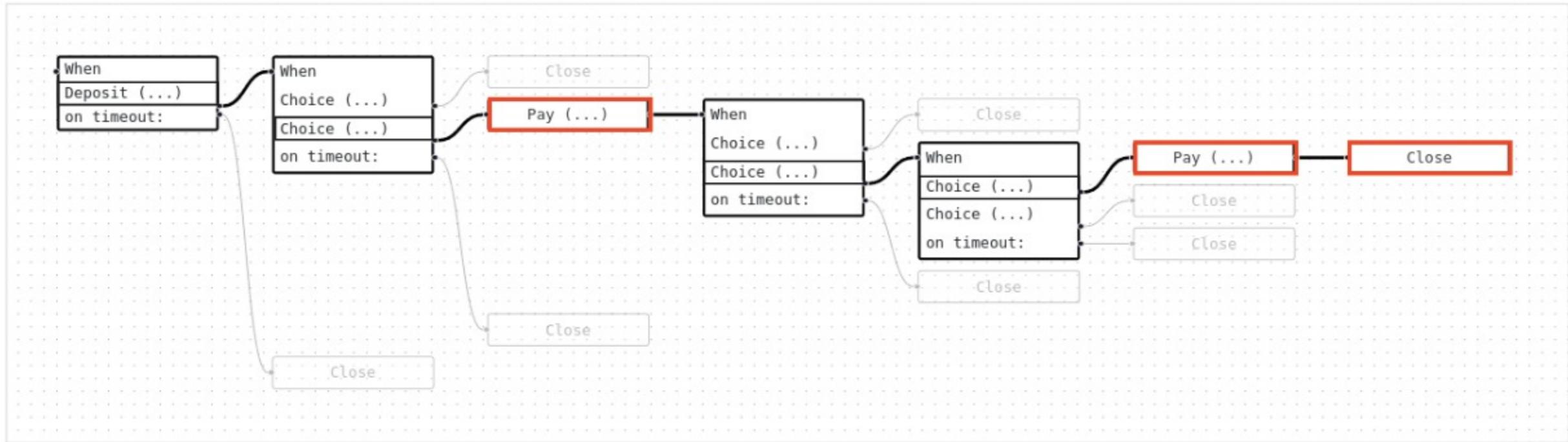
Circumstances drives the execution.

How Marlowe Operates



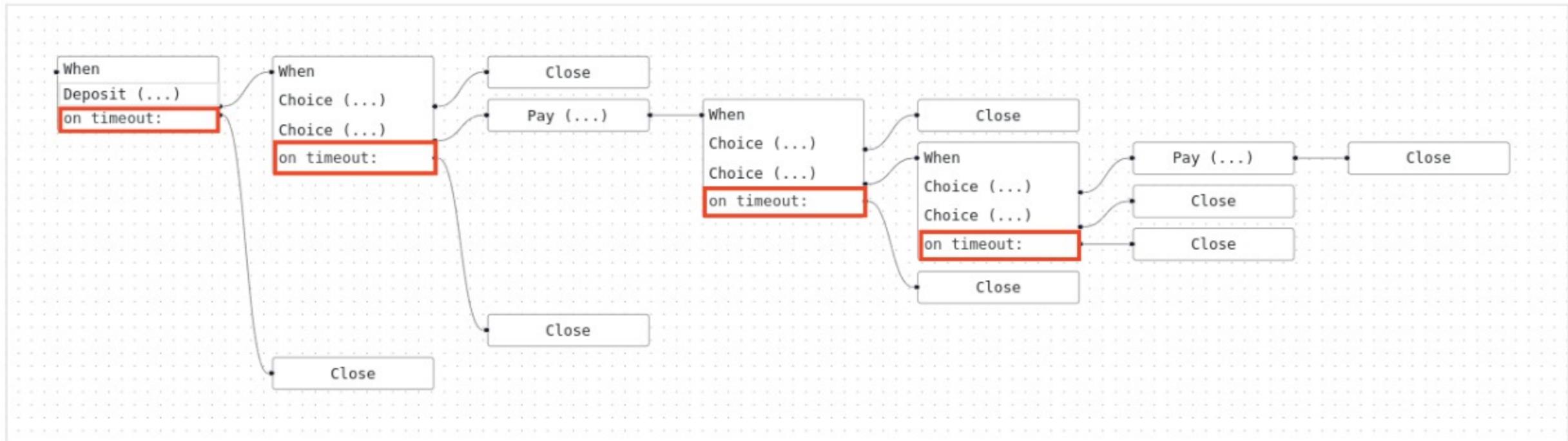
Path and values drive the cash flows.

How Marlowe Operates



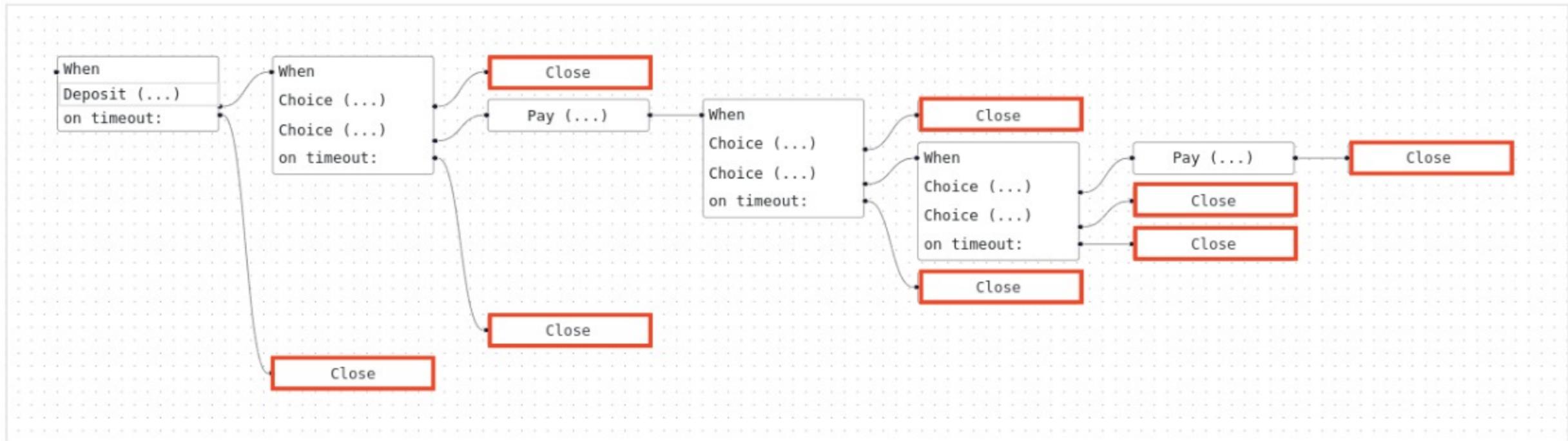
Path and values drive the cash flows.

How Marlowe Operates



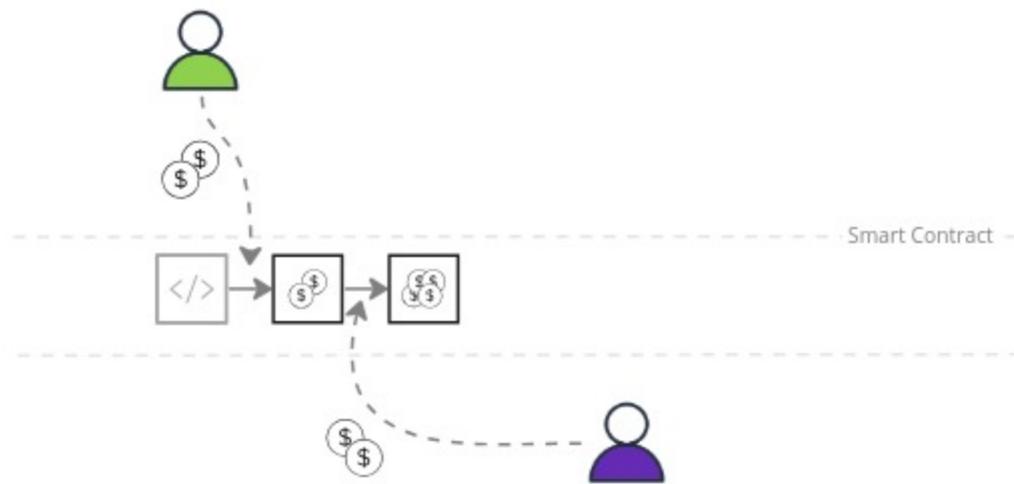
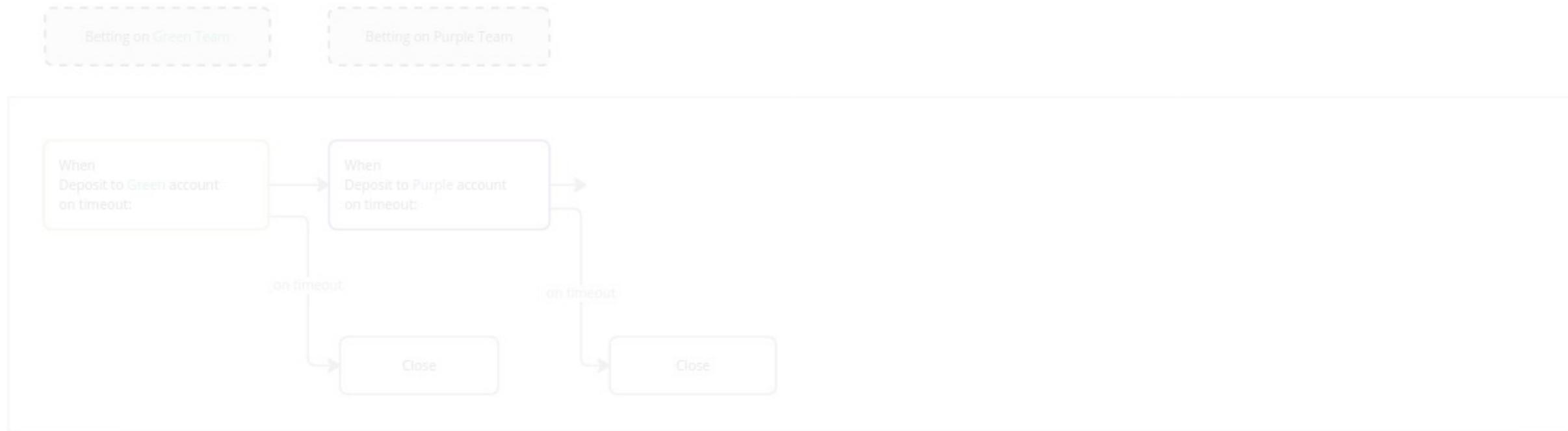
Time can influence the execution.

How Marlowe Operates

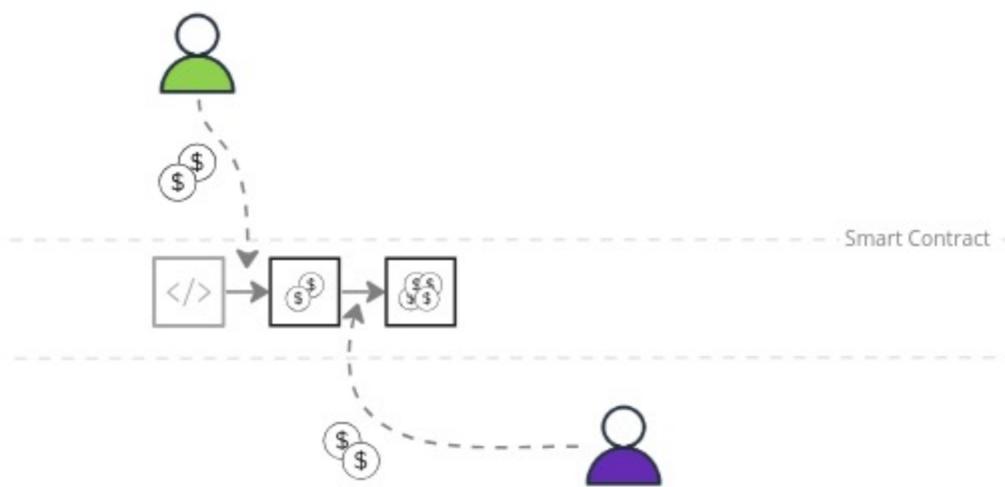
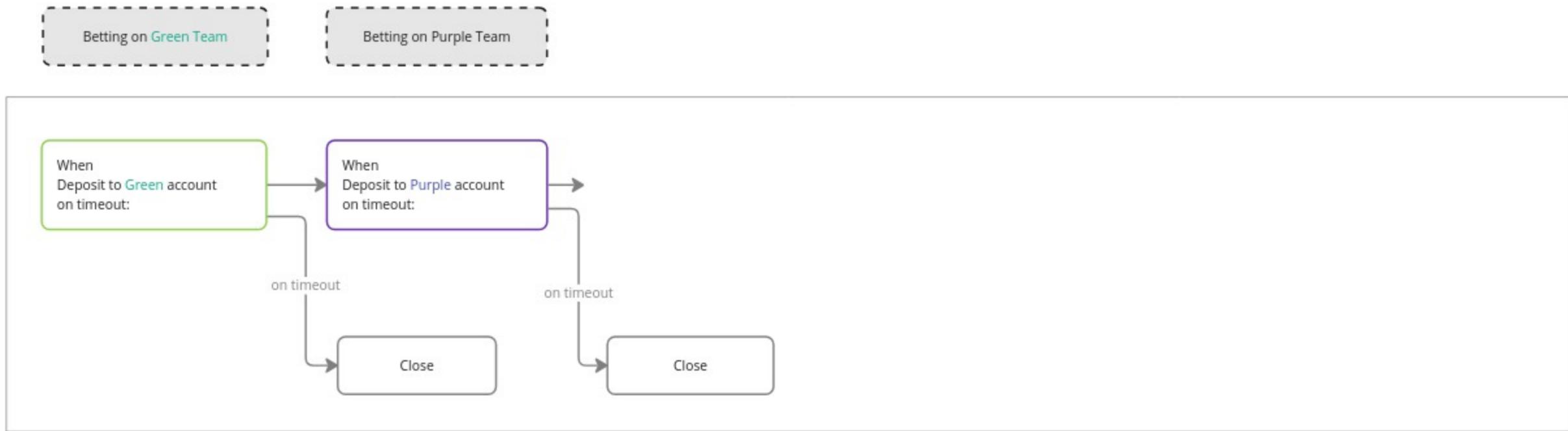


Every path is closed at the end.

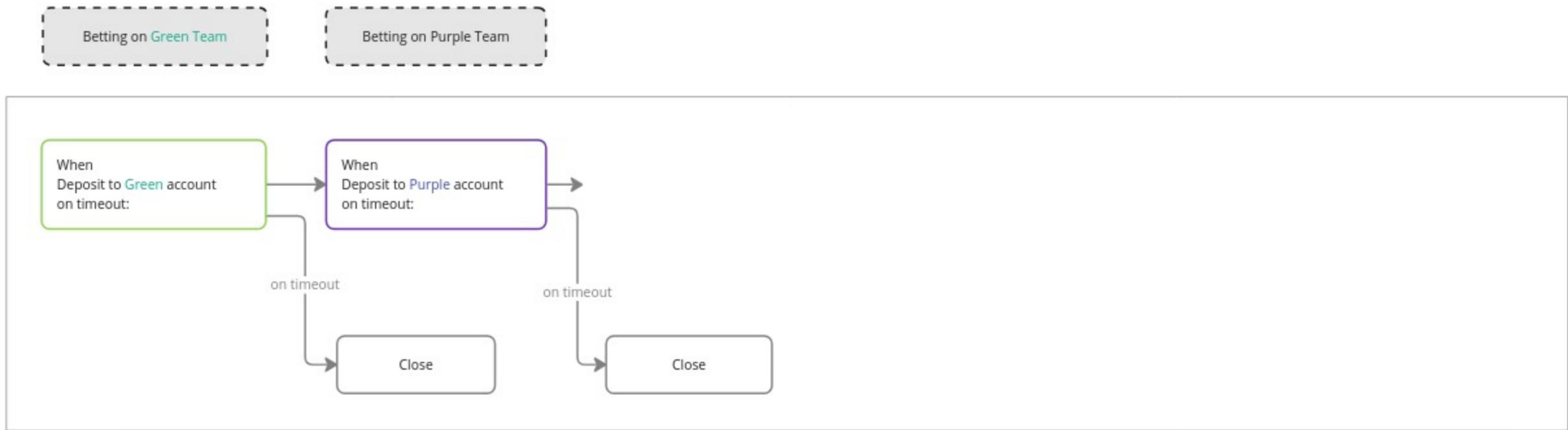
Example: "On-Chain Betting"



Example: "On-Chain Betting"

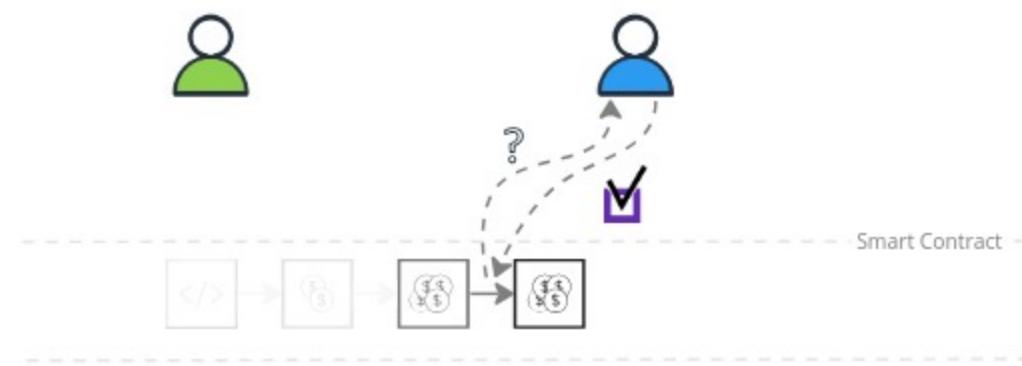
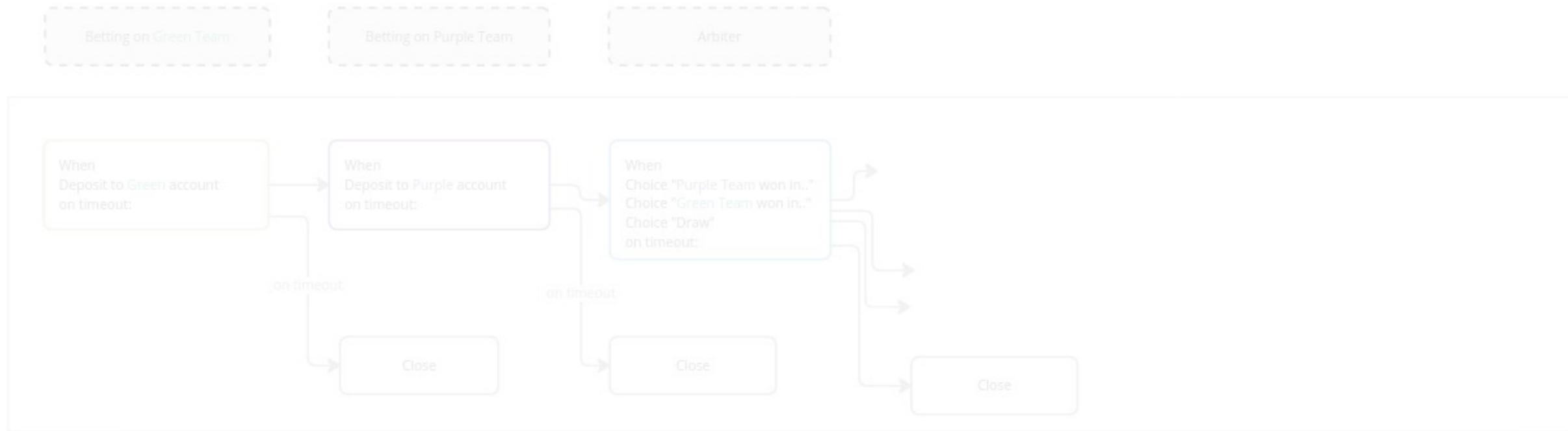


Example: "On-Chain Betting"

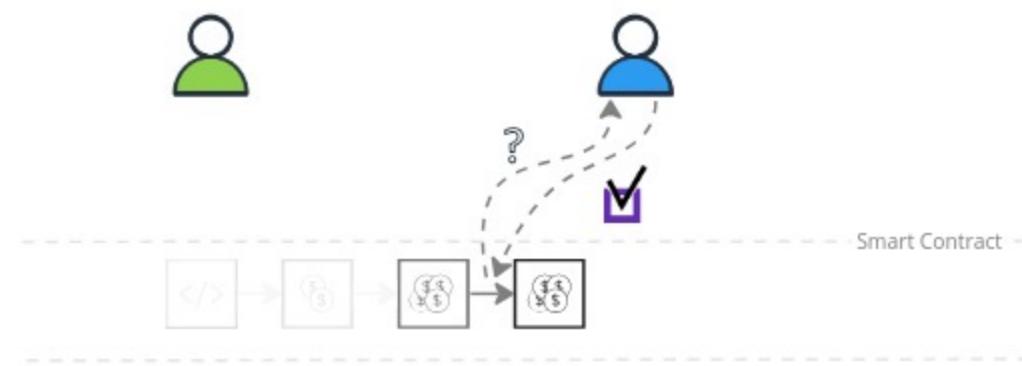
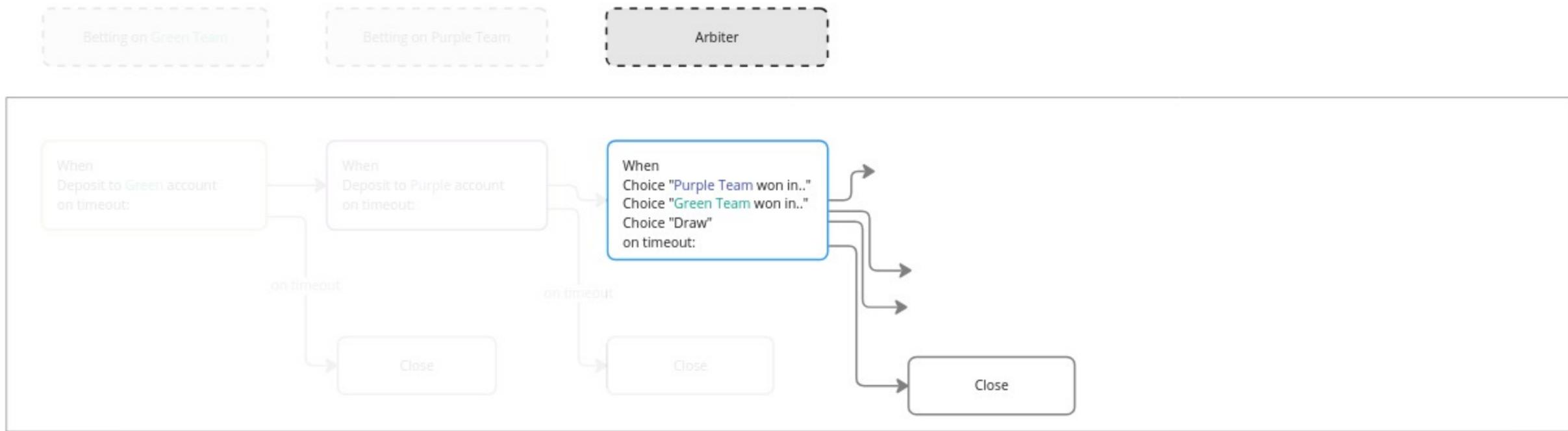


play.marlowe.iohk.io

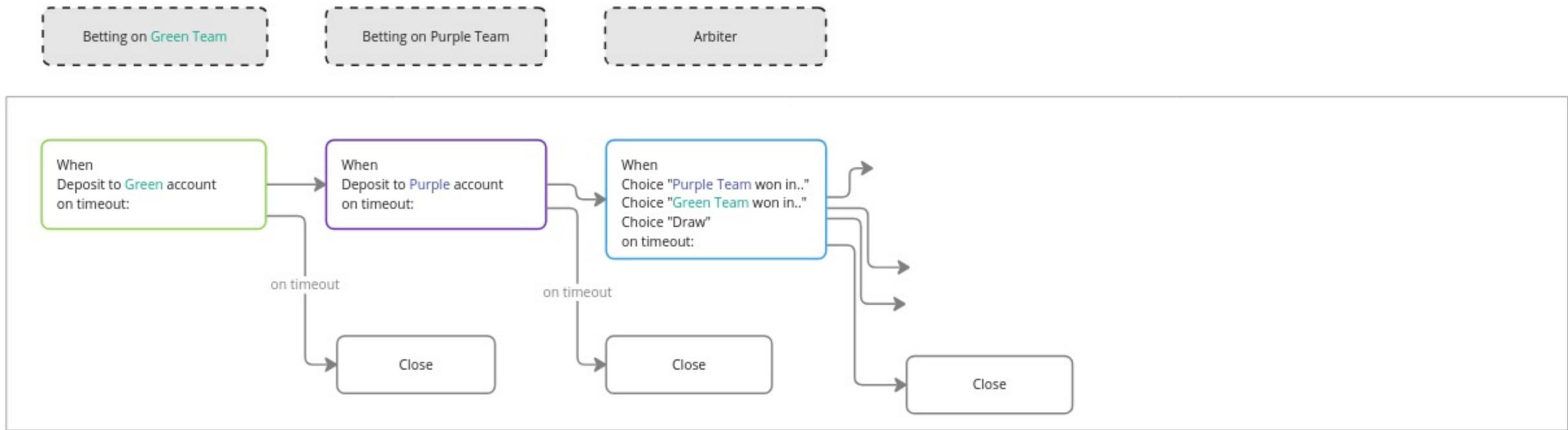
Example: "On-Chain Betting"



Example: "On-Chain Betting"

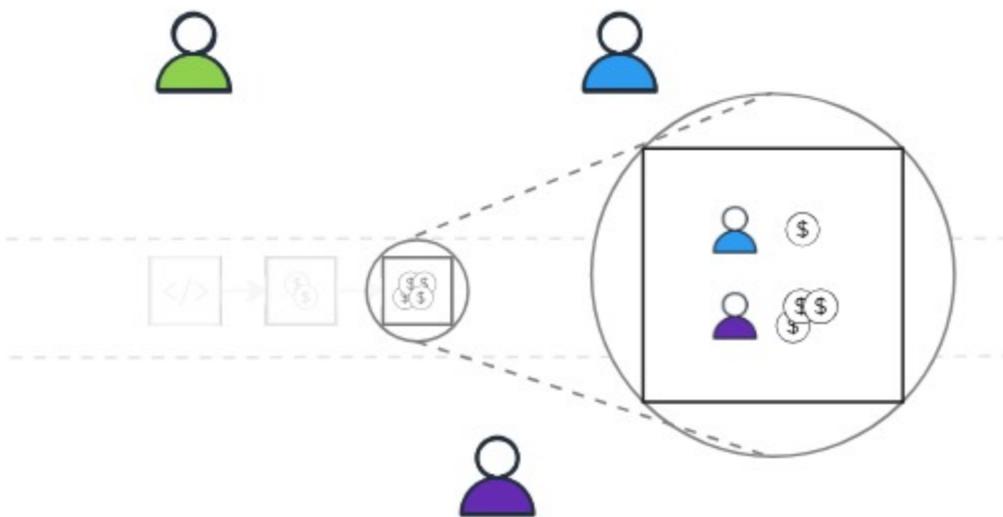
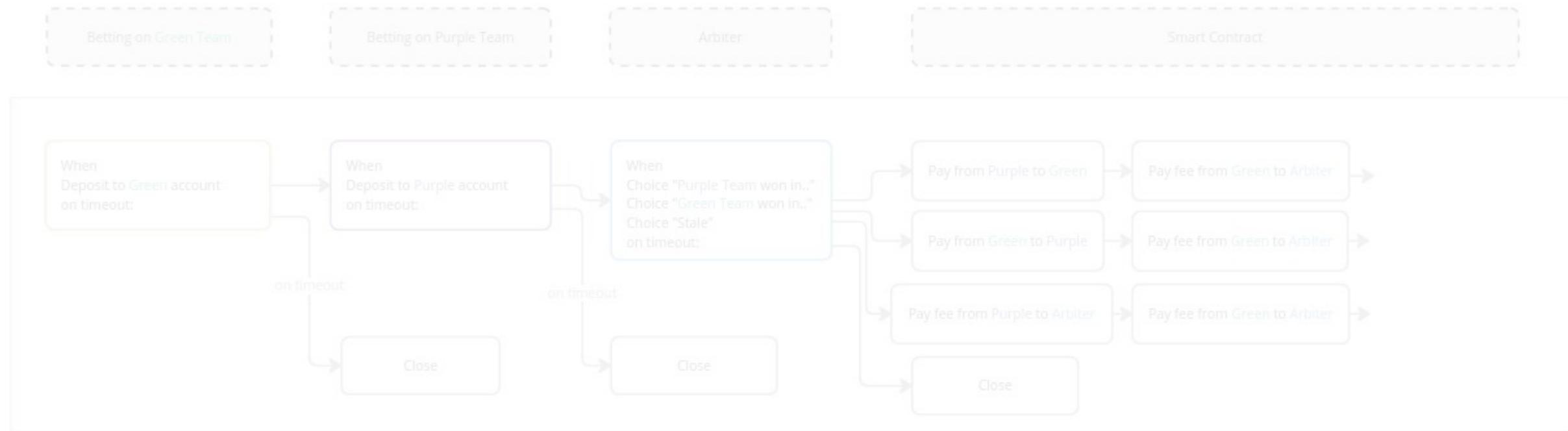


Example: "On-Chain Betting"

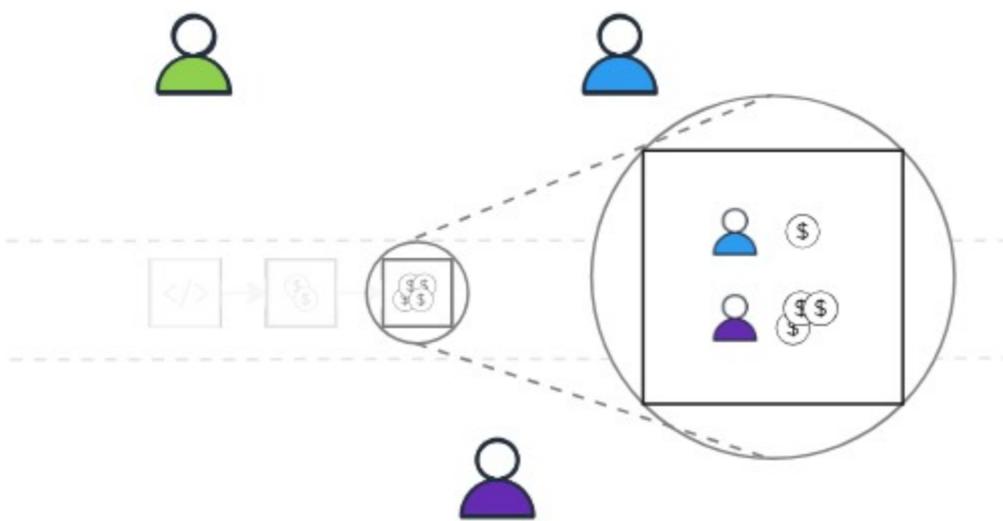
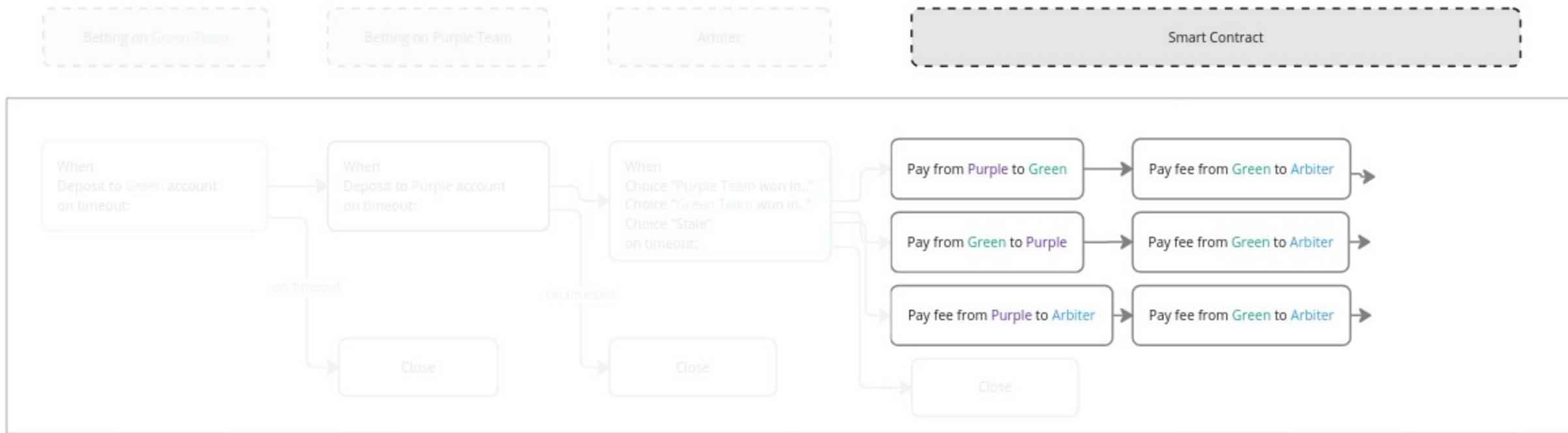


play.marlowe.iohk.io

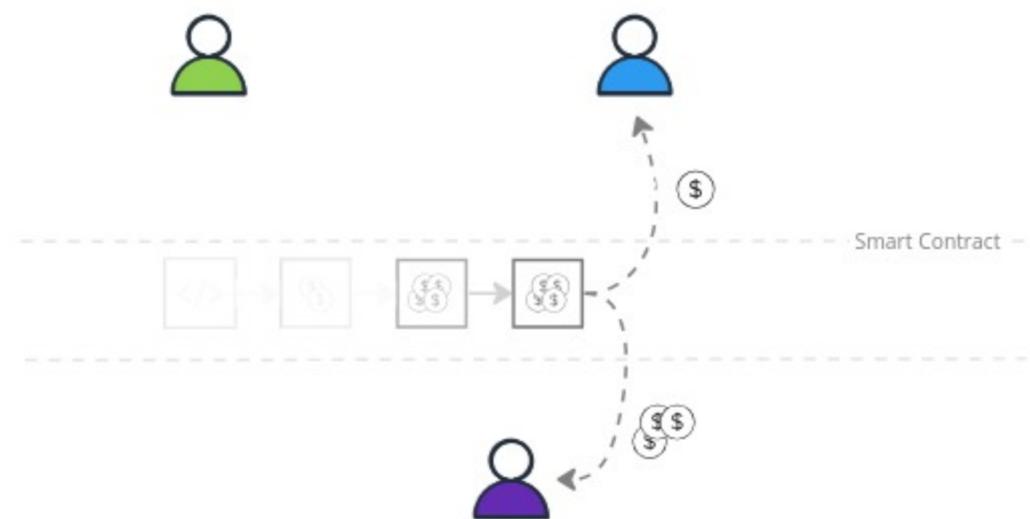
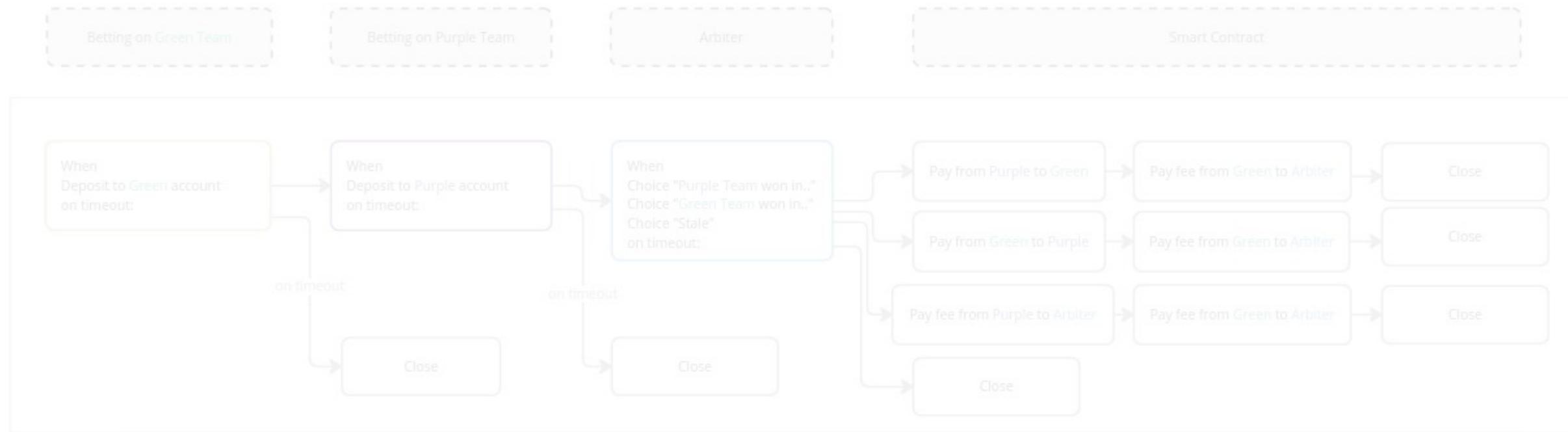
Example: "On-Chain Betting"



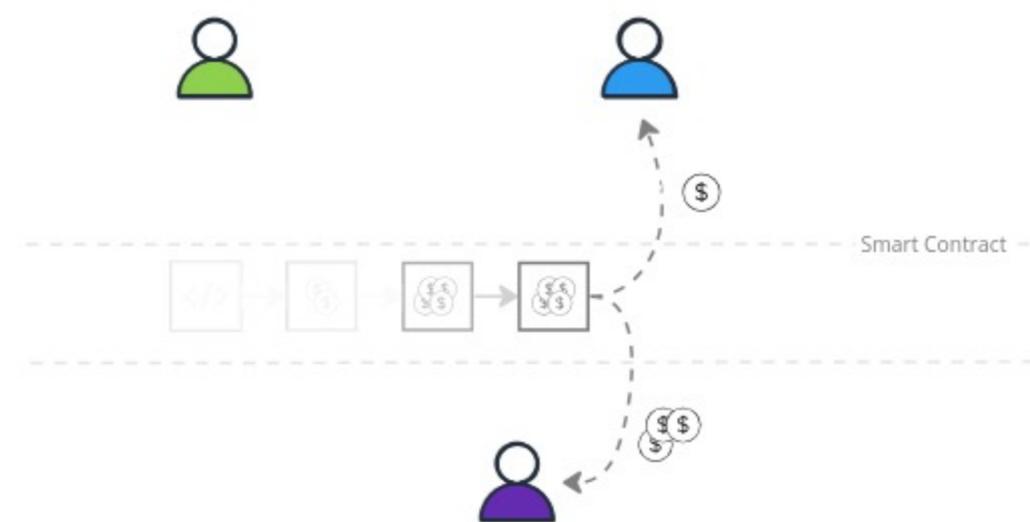
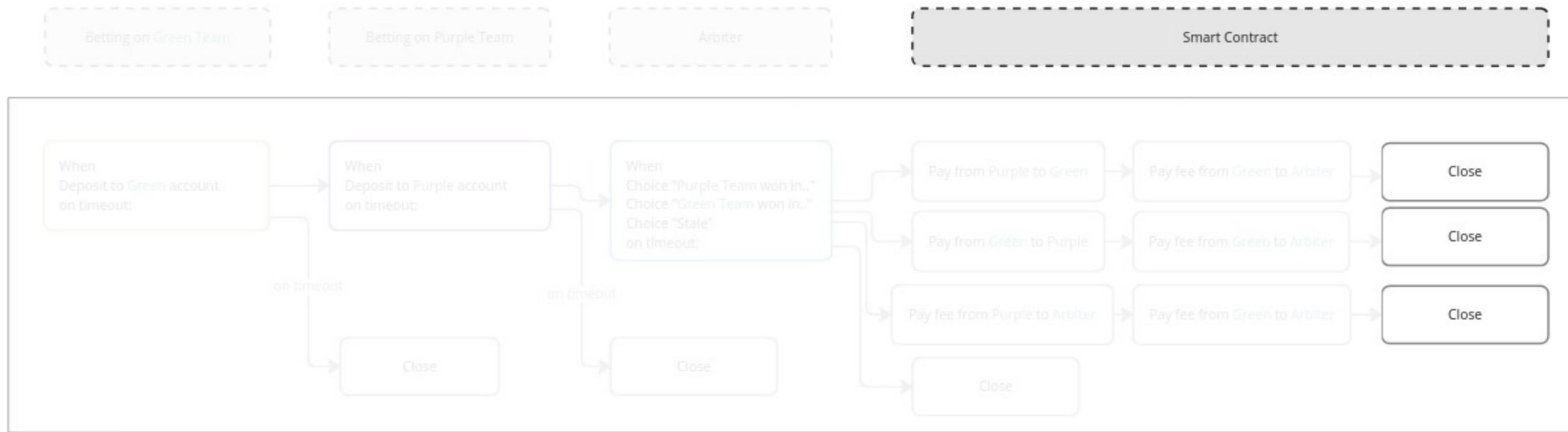
Example: "On-Chain Betting"



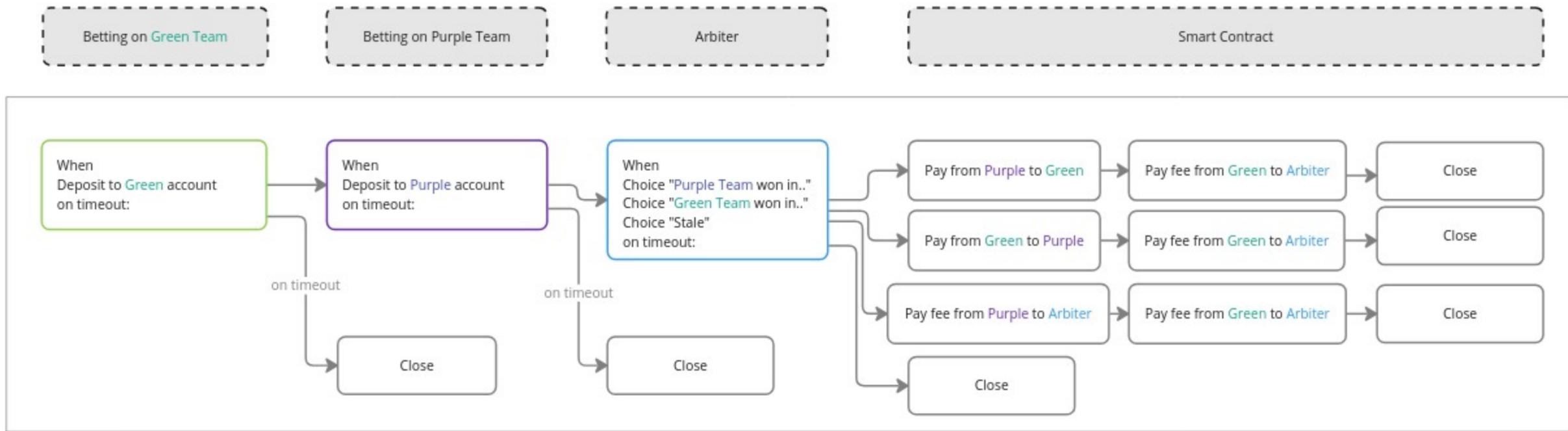
Example: "On-Chain Betting"



Example: "On-Chain Betting"



Example: "On-Chain Betting"



play.marlowe.iohk.io

Steps

- Design
- Test
- Integrate

Steps

- Design
- **Test**
- Integrate

Test

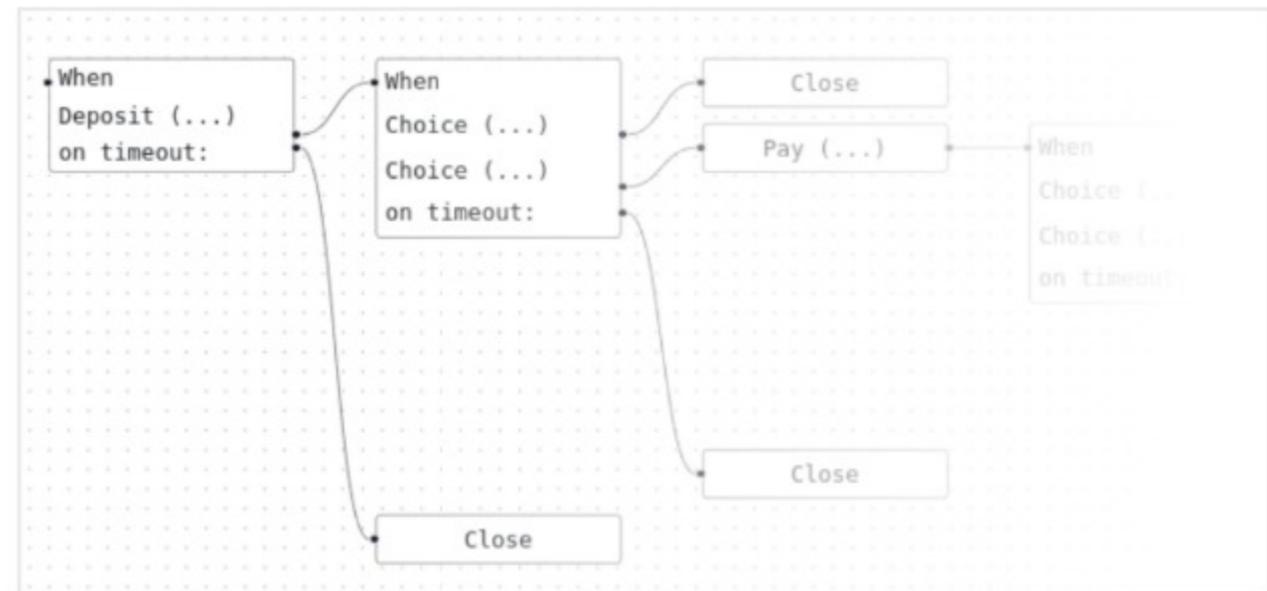
- Safety by construction

- Static analysis

- Runner

- CLI tools

- Testing framework



Test

- Safety by construction

- Static analysis

- Runner

- CLI tools

- Testing framework

The screenshot displays a blockchain testing environment with the following components:

- Code Editor:** Shows Solidity code for a contract named "Myer". The code includes logic for depositing tokens (A1) to accounts and paying them out. It uses choices based on "Green Team won" or "Purple Team won" to determine the amount paid.
- Current State:** A table showing the state of accounts. It lists two accounts: "addr_test...m4uryt" and "addr_test...ceter". Both accounts have a balance of "A 1".
- Transactions:** A log of events with their times:
 - Contract started at 10:34
 - addr_test...ceter deposited A 1 from his/her wallet into addr_test...ceter account at 10:34
 - addr_test...m4uryt deposited A 1 from his/her wallet into addr_test...m4uryt account at 10:34
- ACTIONS:** A sidebar with buttons for choosing a team and other actions like moving current time.

Test

- Safety by construction

- Static analysis

- Runner

- CLI tools

- Testing framework

The screenshot shows a web-based application for managing Marlowe contracts. The interface includes a header with the Marlowe logo and a user ID (addr_test@pl2qjntf9..). A 'Create a contract' button is located in the top right. Below the header is a search bar labeled 'Filter contracts...'. The main area is a table listing six contracts, each with columns for Created, Updated, Contract Id, Tags, and Actions.

Created	Updated	Contract Id	Tags	Actions
11/8/2023 10:46:11 AM	11/8/2023 10:46:11 AM	36a82775e39286936bfff0d94c...	bet-between-lace-and-nami-typhon	
11/4/2023 7:03:42 AM	11/4/2023 7:07:01 AM	6b27639e9b791cd219f07b888f...	Complete	
11/4/2023 6:52:26 AM	11/4/2023 6:57:50 AM	e7948f77f0cb407979c212af9...	Complete	
11/4/2023 6:51:39 AM	11/4/2023 6:58:17 AM	12b395bf547660f7cf122a537b...	Complete	
11/3/2023 9:24:08 PM	11/3/2023 9:24:08 PM	de3ff1ad2eabc92fb6d630e2ac...	Advance	
11/3/2023 9:22:20 PM	11/3/2023 9:22:20 PM	fate3e4a0a36554eca44533a4f...	Advance	

Test

- Safety by construction

- Static analysis

- Runner

- CLI tools**

- Testing framework

```
1  $ marlowe-cli run analyze --help
2 Usage: marlowe-cli run analyze [--mainnet | --testnet-magic INTEGER]
3                                         --socket-path SOCKET_FILE
4                                         --marlowe-file MARLOWE_FILE [--preconditions]
5                                         [--roles] [--tokens] [--maximum-value]
6                                         [--minimum-utxo] [--execution-cost]
7                                         [--transaction-size] [--best] [--verbose]
8
9 [EXPERIMENTAL] Analyze a Marlowe contract.
10 Available options:
11   --mainnet                                Execute on mainnet.
12   --testnet-magic INTEGER                   Network magic. Defaults to the CARDANO_TESTNET_MAGIC
13                                         environment variable's value.
14   --socket-path SOCKET_FILE                 Location of the cardano-node socket file. Defaults to
15                                         the CARDANO_NODE_SOCKET_PATH environment variable's
16                                         value.
17   --marlowe-file MARLOWE_FILE               JSON file with the state and contract.
18   --preconditions                          Whether to check preconditions for valid Marlowe
19                                         state.
20   --roles                                  Whether to check lengths of role names.
21   --tokens                                 Whether to check lengths of token names.
22   --maxUnum-value                         Whether to check the 'maxValueSize' protocol limit.
23   --minimum-utxo                           Whether to check the 'utxoCostPerWord' protocol
24                                         limit.
25   --execution-cost                        Whether to check the 'maxTxExecutionUnits' protocol
26                                         limit.
27   --transaction-size                     Whether to check the 'maxTxSize' protocol limit.
28   --best                                    Whether to compute tight estimates of worst-case
29                                         bounds, instead of generous estimates of those
30                                         bounds.
31
32
```

Test

- Safety by construction
- Static analysis
- Runner
- CLI tools
- Testing framework

```
17 - CreateWallet: Other provider
18
19 - Fund:
20   wallet: Other provider
21   utxo: 40000000
22
23 - Mint:
24   issuer: Ada provider
25   nickname: RoleTokenCurrency
26   minLovelace: 2000000
27   tokenDistribution:
28     - [Other provider, Other provider, 1]
29     - [Ada provider, Ada provider, 1]
30
31 - CheckBalance:
32   wallet: Other provider
33   balance:
34     - [ADA, 40000000]
35     - [RoleTokenCurrency, Other provider, 1]
36
37 - Initialize:
38   minLovelace: 2000000
39   roleCurrency: RoleTokenCurrency
40   submitter: Ada provider
41   source:
42     when:
43       - case:
44         party:
45           role_token: Ada provider
46           deposits:
47             multiply: 1000000
48             times: 20
49             of_token:
50               currency_symbol: ''
51               token_name: ''
52             into_account:
53               role_token: Ada provider
54             then:
55               when:
56                 - case:
57                   party:
58                     role_token: Other provider
59                     deposits:
60                       multiply: 1000000
61                       times: 25
```

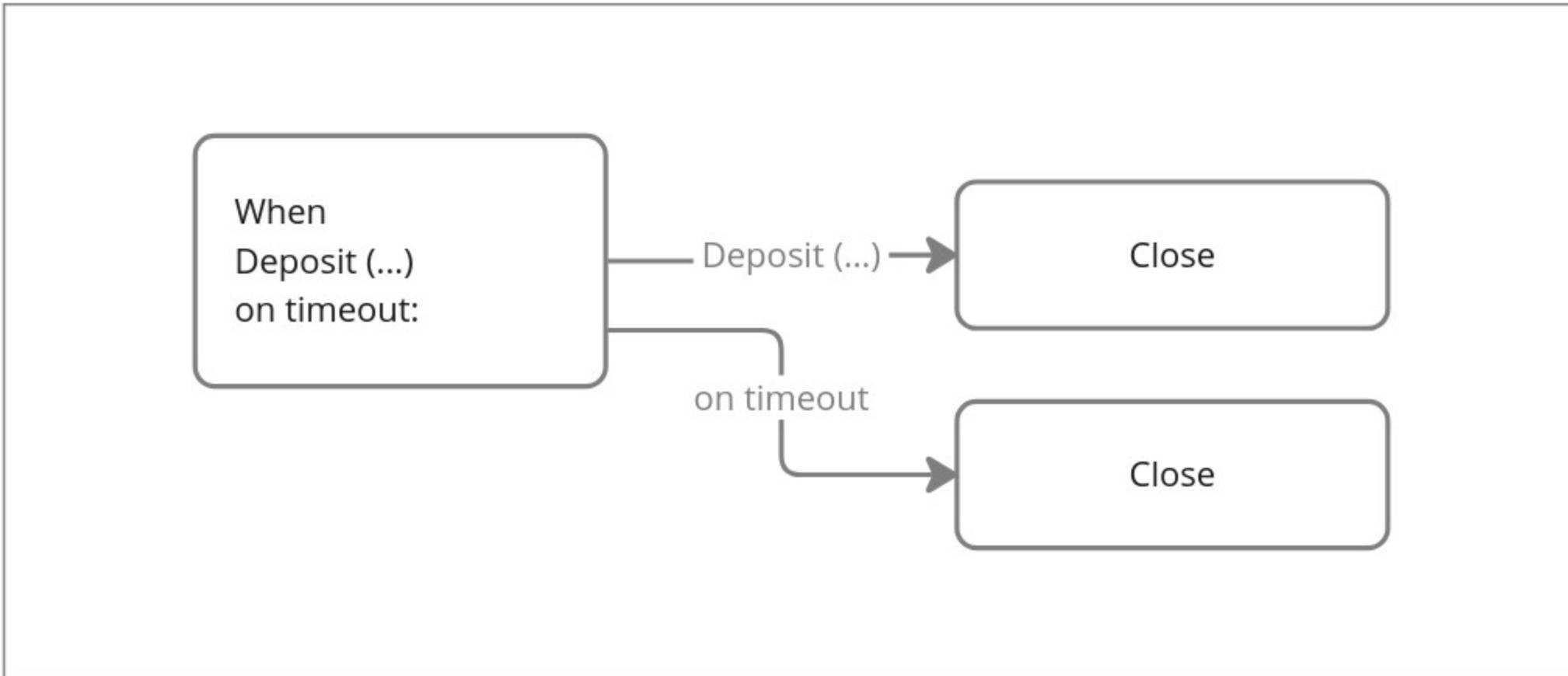
Steps

- Design
- **Test**
- Integrate

Steps

- Design
- Test
- **Integrate**

Dynamic Contract Generation



play.marlowe.iohk.io

Dynamic Contract Generation

```
1  {
2      "when": [
3          {
4              "case": {
5                  "party": {
6                      "address": "addr_test1qq0acgkfkgeueezdy2fn2y5mxhn9zcvrjesxxen4k2d2t2zrhp2e
7                          tmnsef6wnpvrsu5n80kxceafnxpv5te923agndxs5c4ter"
8                  },
9                  "deposits": 1000000,
10                 "of_token":
11                     ...
12                 }
13             },
14         ],
15         "timeout": 1701252900000
16         "timeout_continuation": "close",
17     }
```

Dynamic Contract Generation

```
1  export const mkContract = (donor: Address, acceptor: Address): Contract => {
2    const inTwentyFourHours = datetoTimeout(new Date(Date.now()) + ...
3    return {
4      "when": [
5        {
6          "then": "close",
7          "case": {
8            "party": donor,
9            "deposits": 100n,
10           ...
11           "timeout_continuation": "close",
12           "timeout": inTwentyFourHours,
13         }
14       };

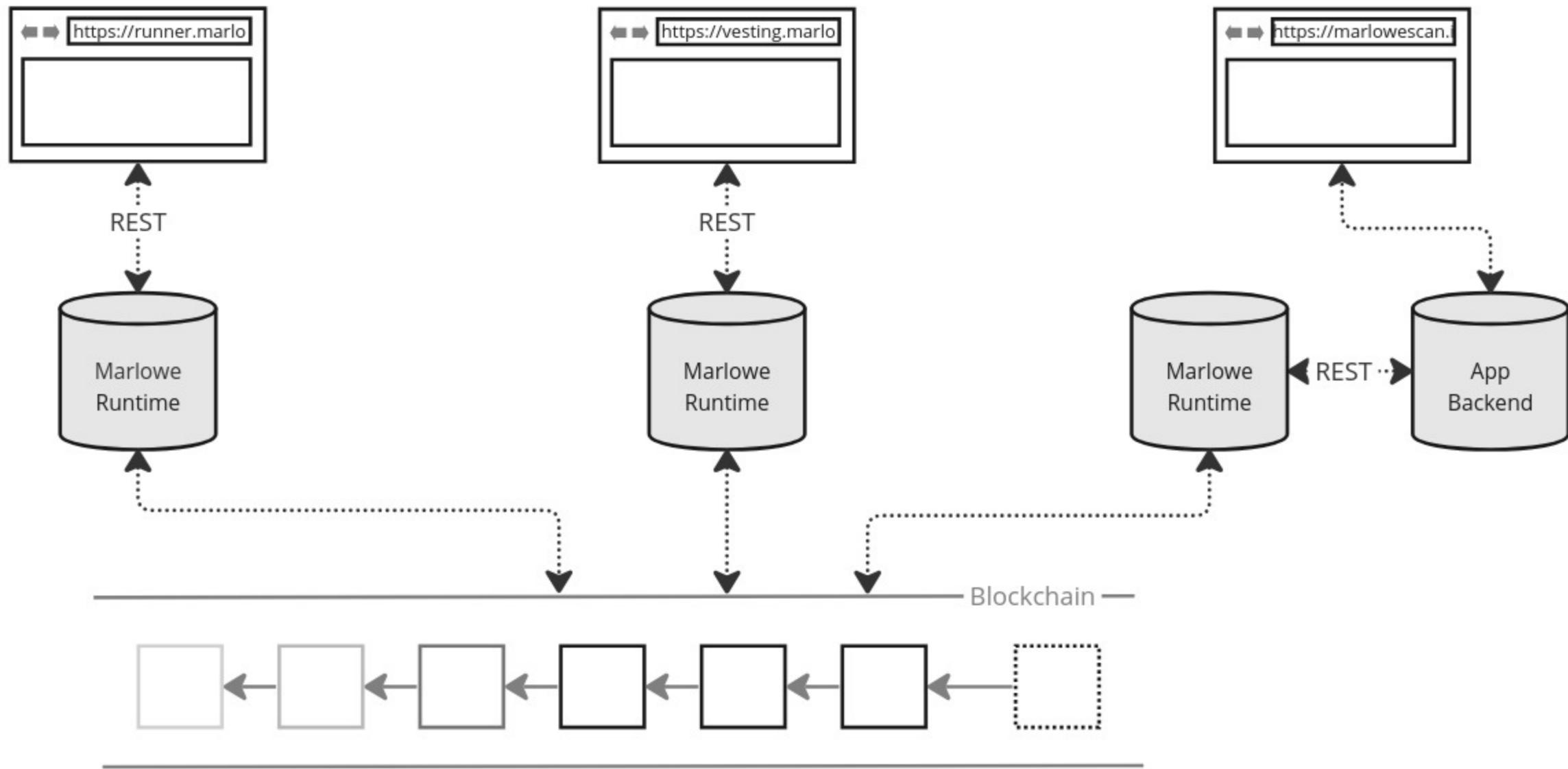
```

- [Full exercise](#)
- [marlowe-ts-sdk](#)

Dynamic Contract Generation

- TS/JS: github.com/input-output-hk/marlowe-ts-sdk
- Rust: github.com/OlofBlomqvist/marlowe-rs
- Go: github.com/menabrealabs/marlowe
- Python: github.com/OlofBlomqvist/marlowe-py
- C# and more: projectcatalyst.io/funds/10/f10-developer-ecosystem-the-evolution/marlowe-runtime-sdks#about

Marlowe Runtime



Blockchain interaction

- Creation

```
1 await runtimeLifecycle.contracts.createContract({ contract, tags });
```

- Execution

```
1 const depositRequest = {inputs: [deposit]};  
2 await runtimeLifecycle.contracts.applyInputs(contractId, depositRequest)
```

Marlowe and Your Dev Team

- Marlowe itself is small, safe and easy to learn
- Marlowe can be used from any programming language (Go, Rust, C#, Python, TypeScript)
- Marlowe blockchain interaction is provided through simple REST API

Marlowe and Your Business

- Low entry / prototyping cost
- Quick delivery - when problem fits Marlowe
- Easy to develop - UTx0 agnostic, REST, online tools.
- Easy to maintain - no deep dev expertise required

DApps Prototypes

- Vesting app: vesting-preprod.scdev.aws.iohkdev.io
- Marlowe Runner: preprod.runner.marlowe.iohk.io
- Marlowe Scanner: marlowescan.com
- Workshop DApp: github.com/paluh/marlowe-workshop-dapp

Thank you

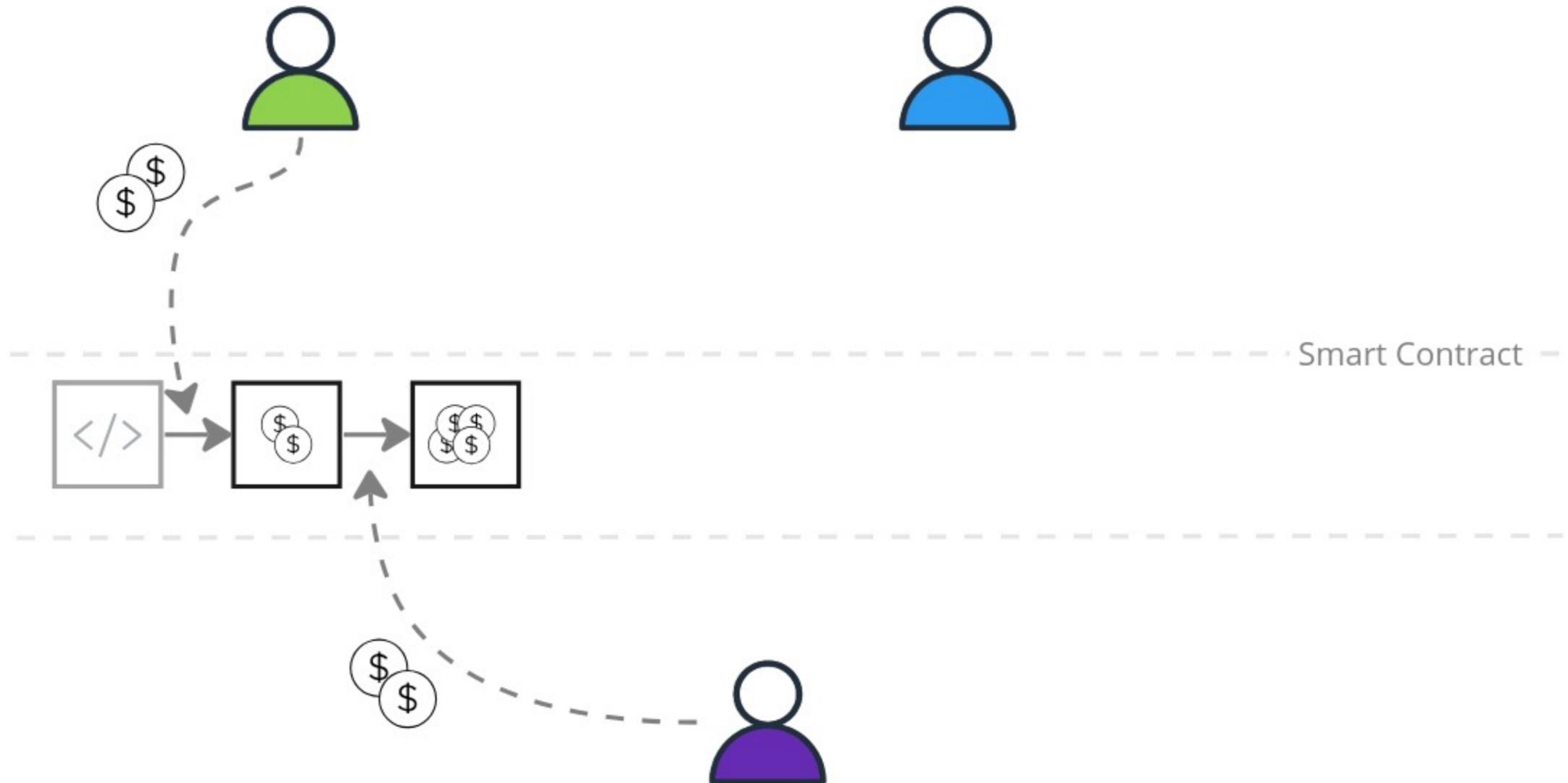
What are Smart Contracts?



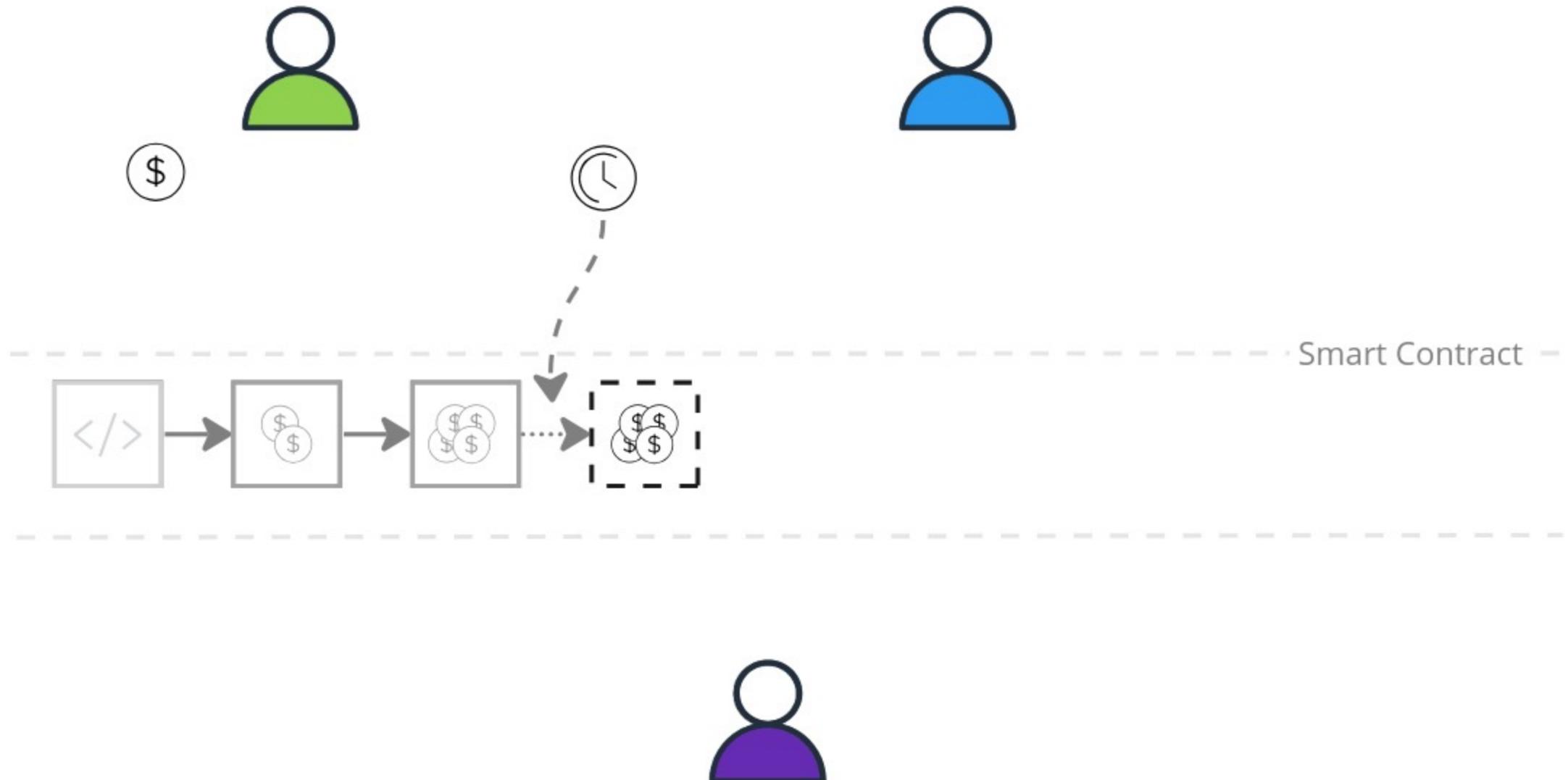
— Smart Contract —



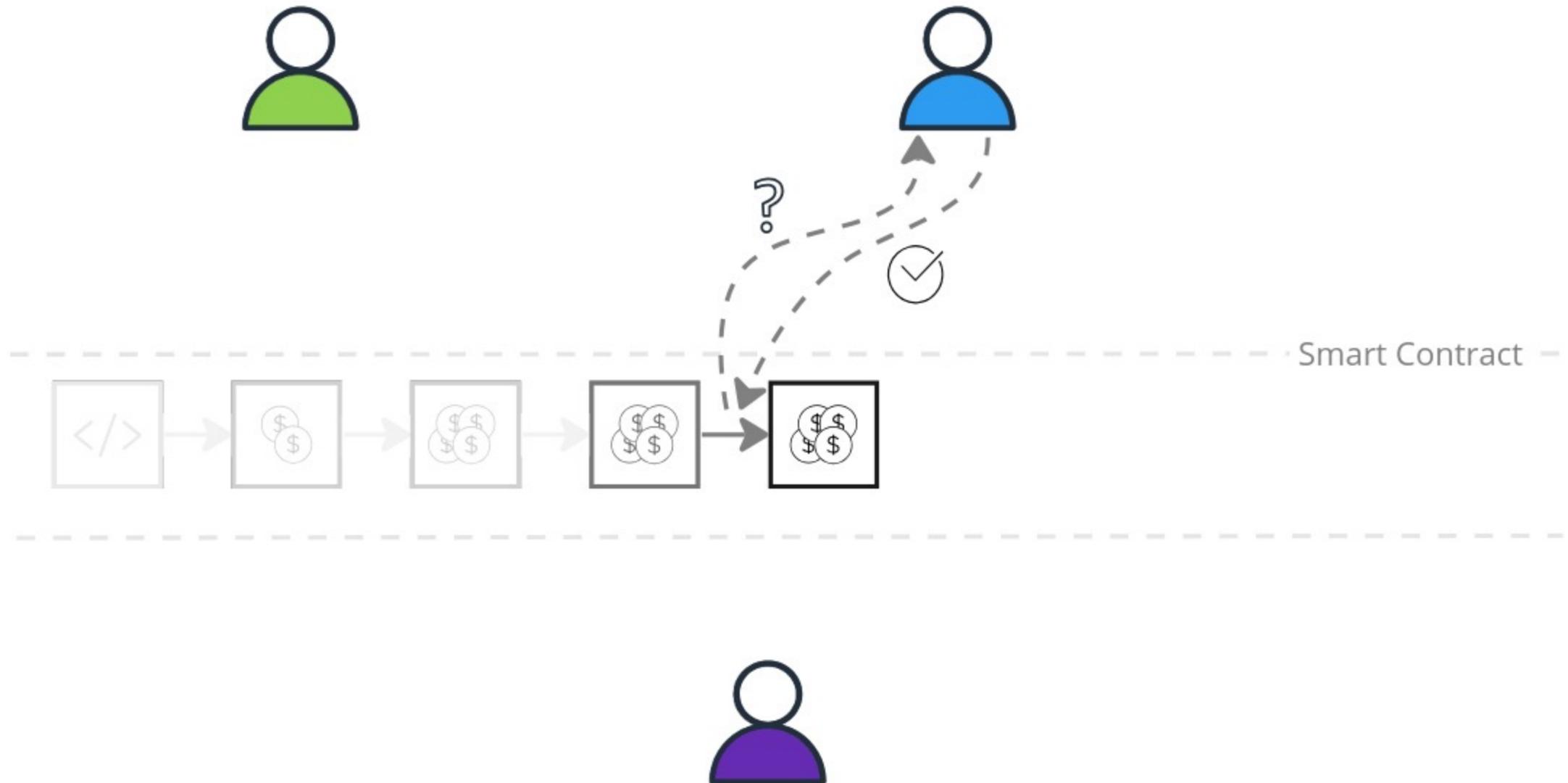
What are Smart Contracts?



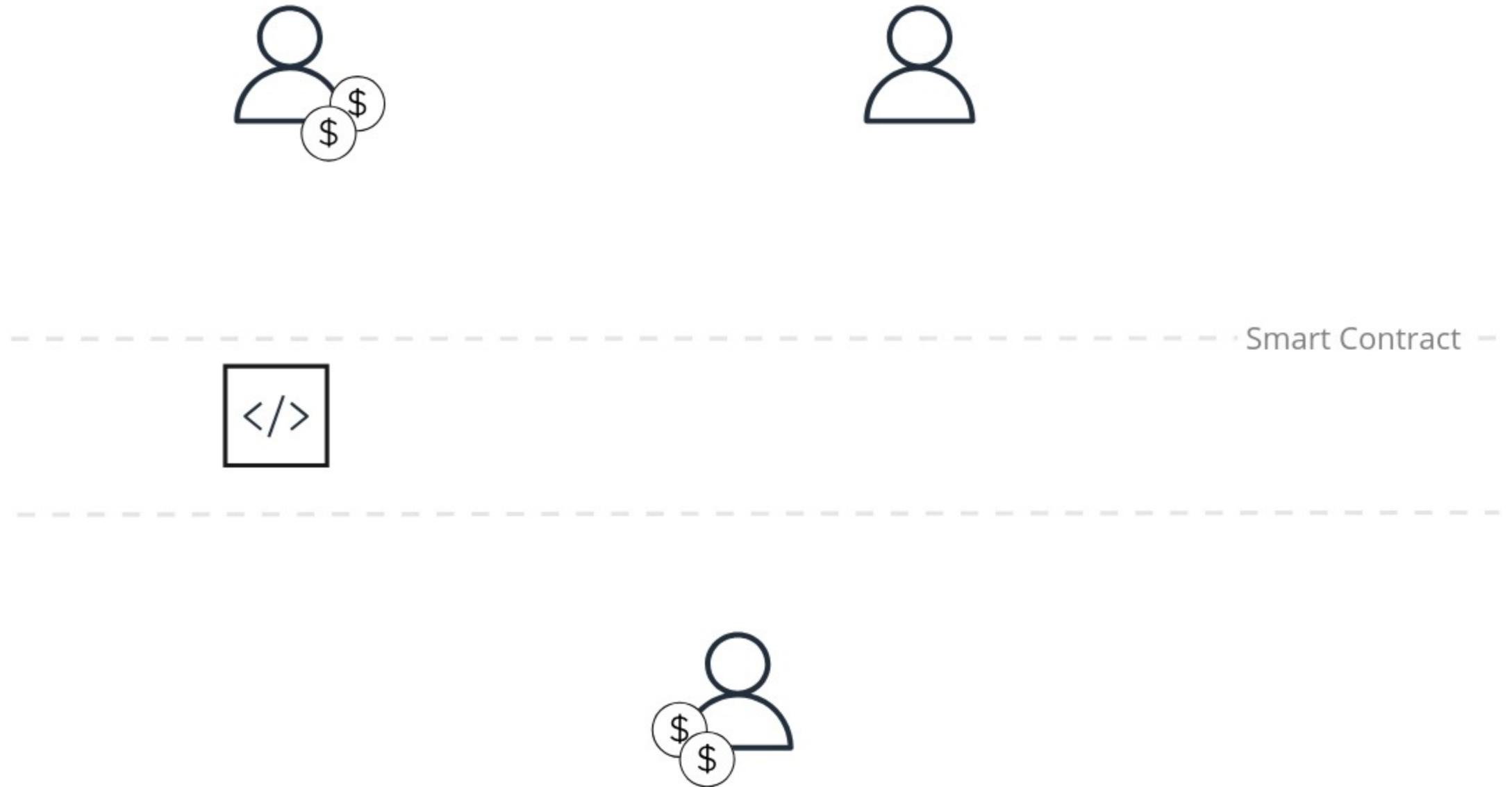
What are Smart Contracts?



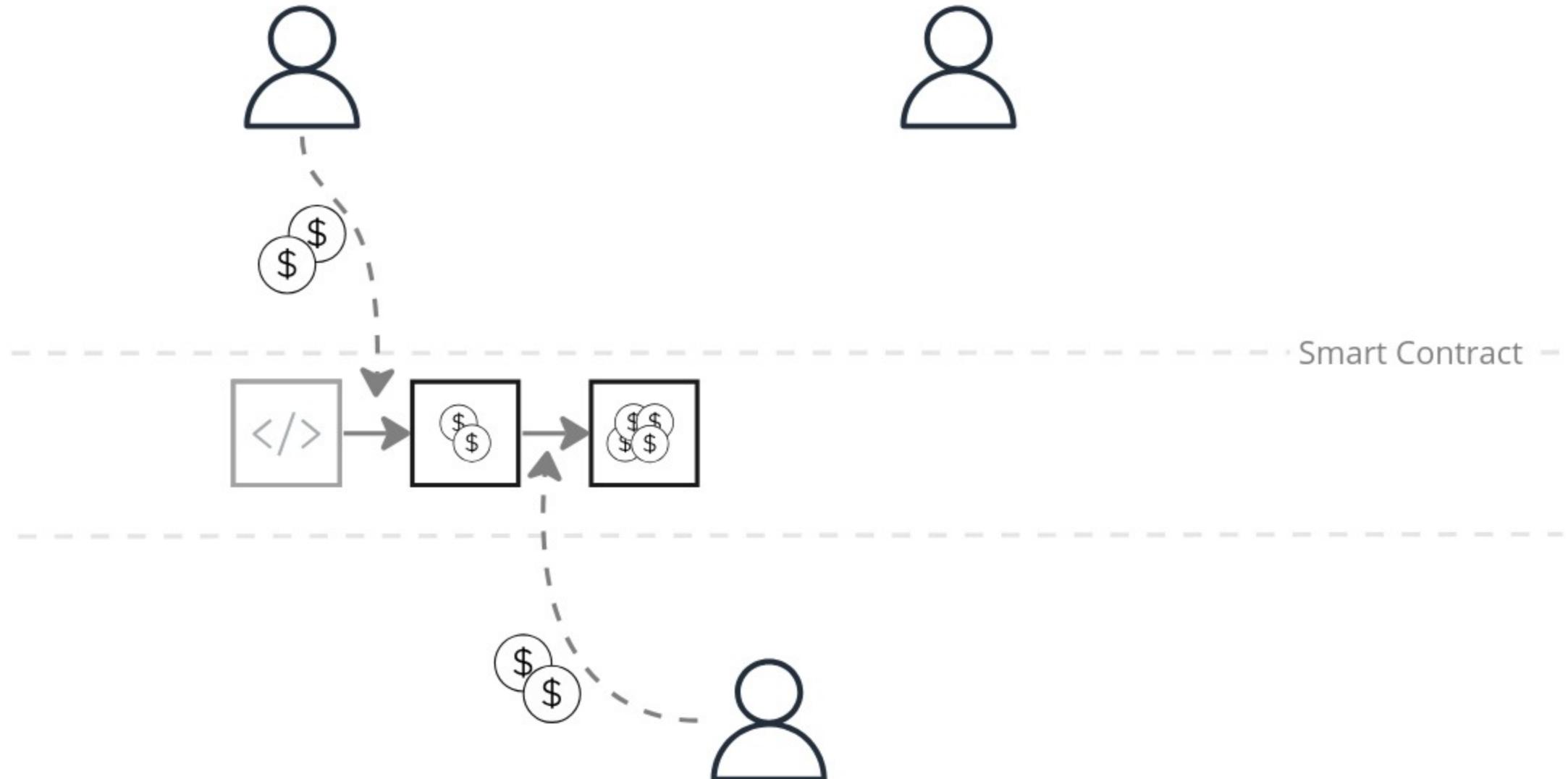
What are Smart Contracts?



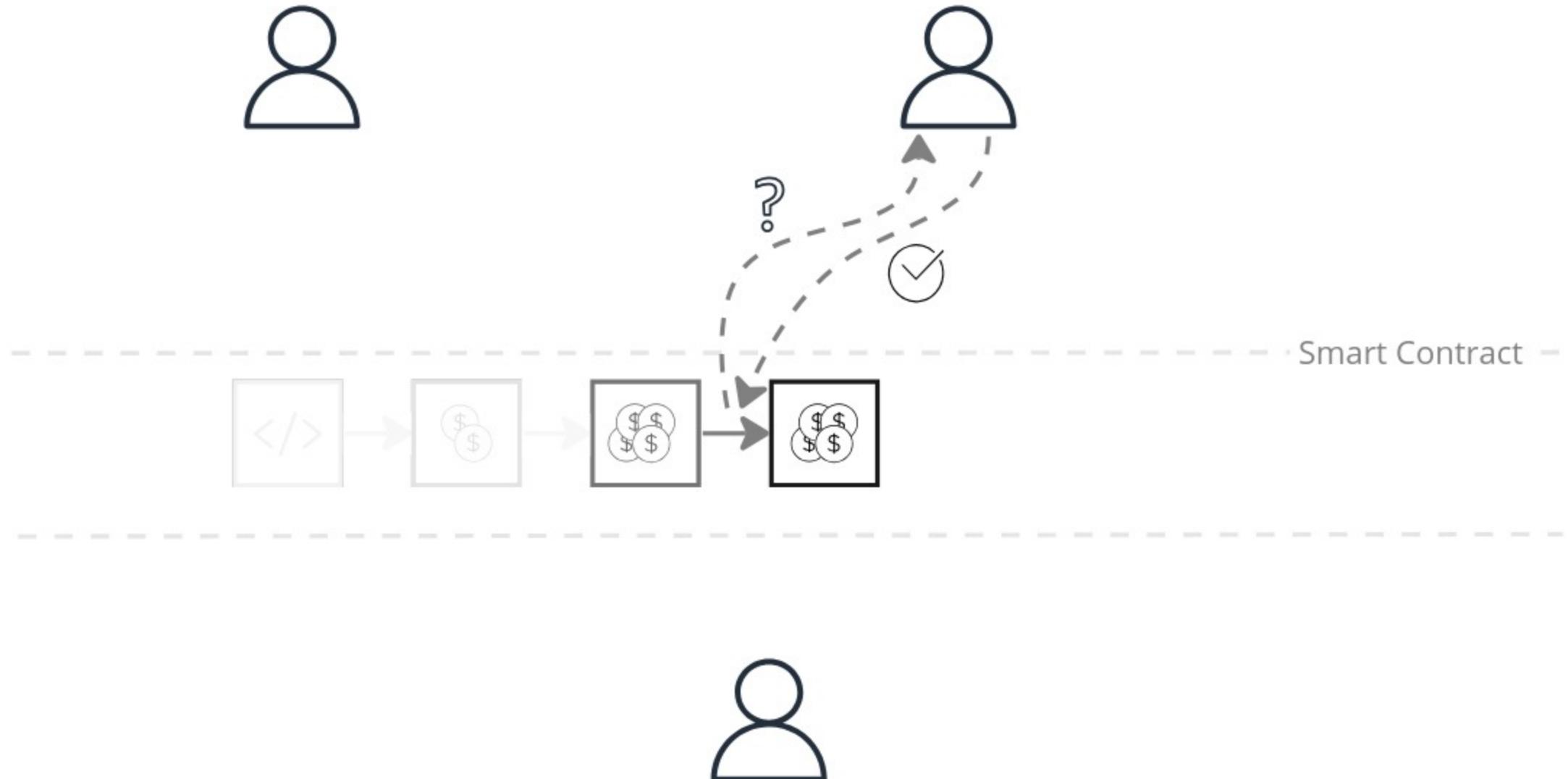
Example: "On-Chain Betting"



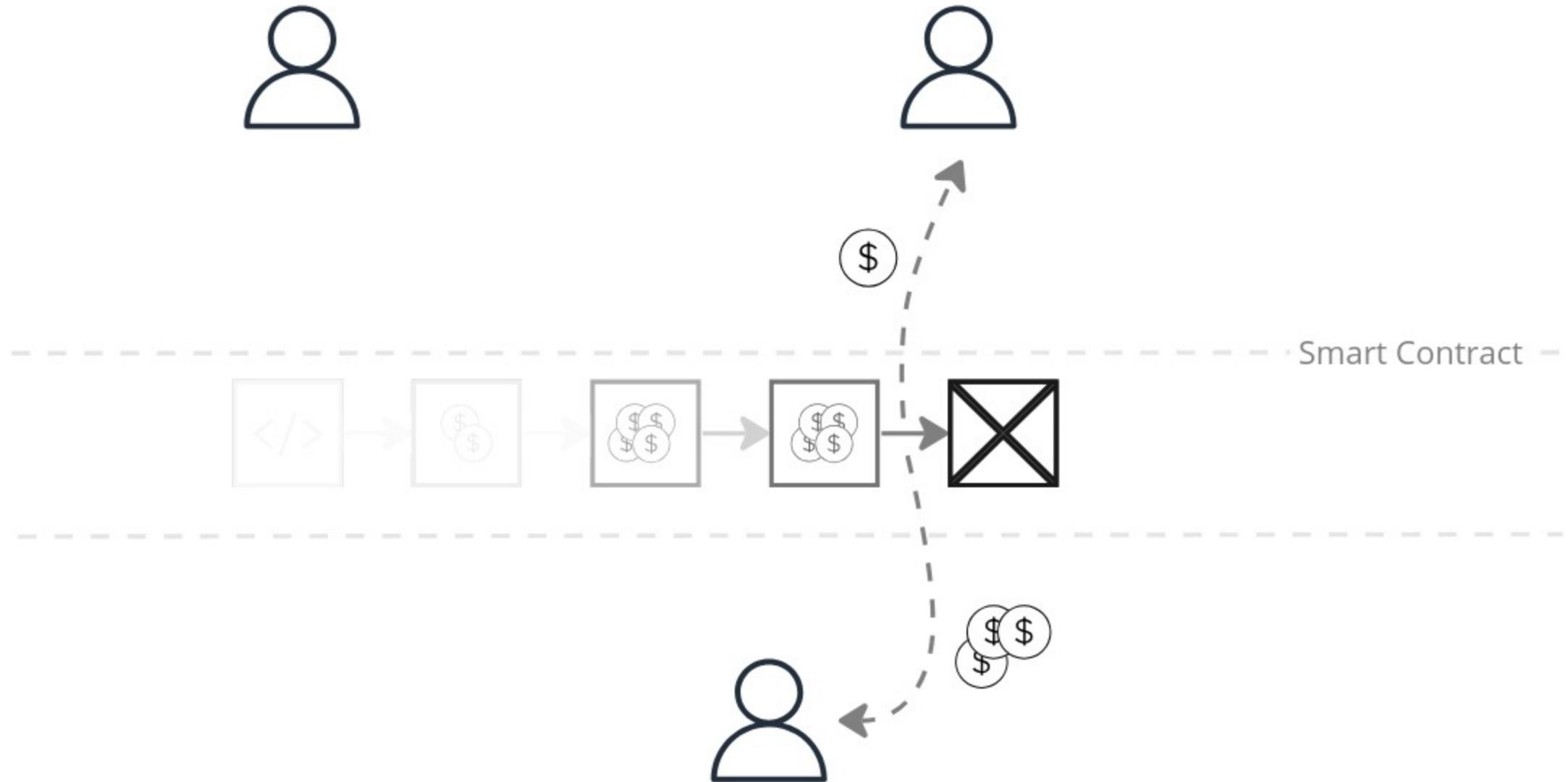
Example: "On-Chain Betting"



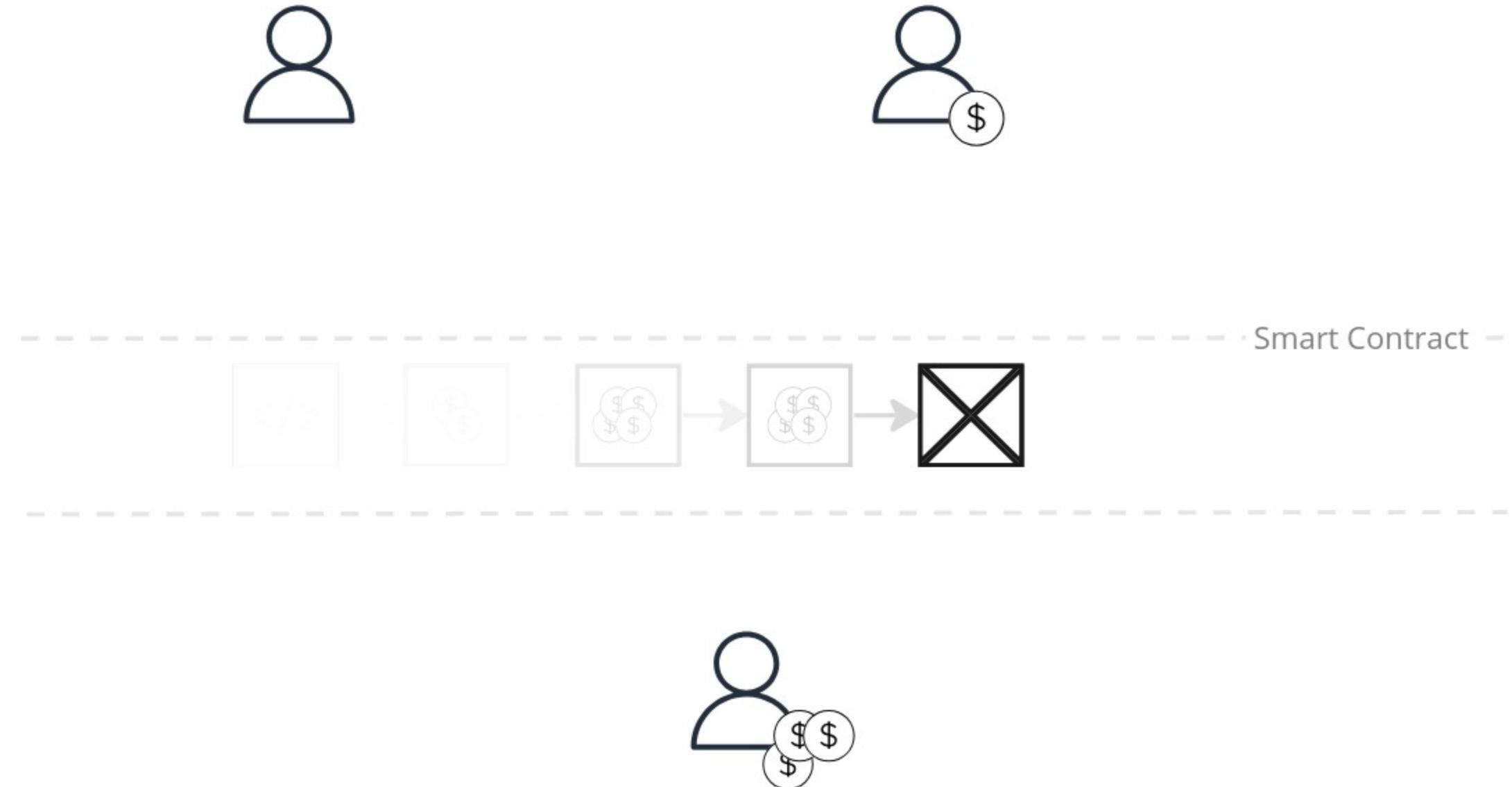
Example: "On-Chain Betting"



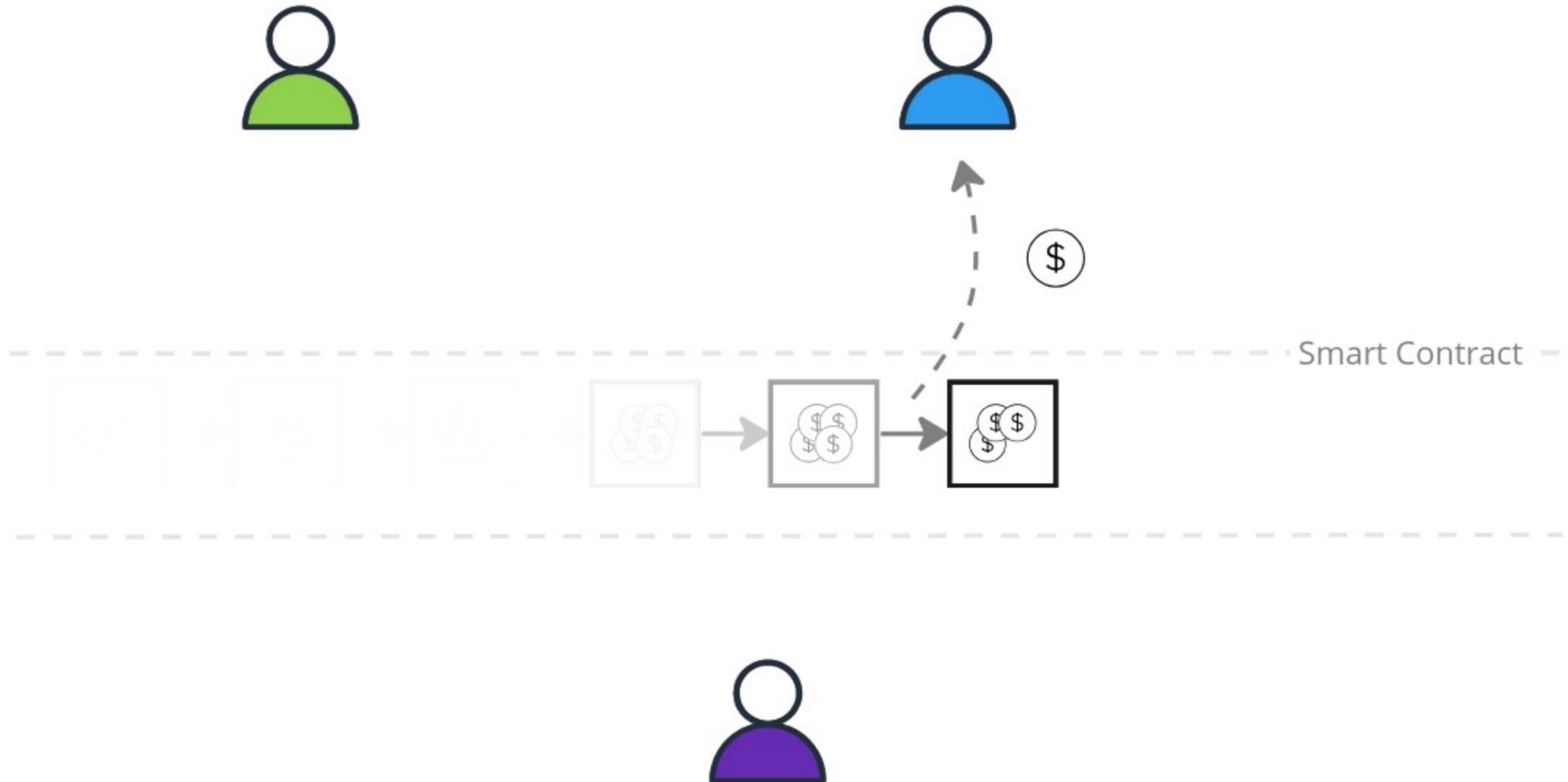
Example: "On-Chain Betting"



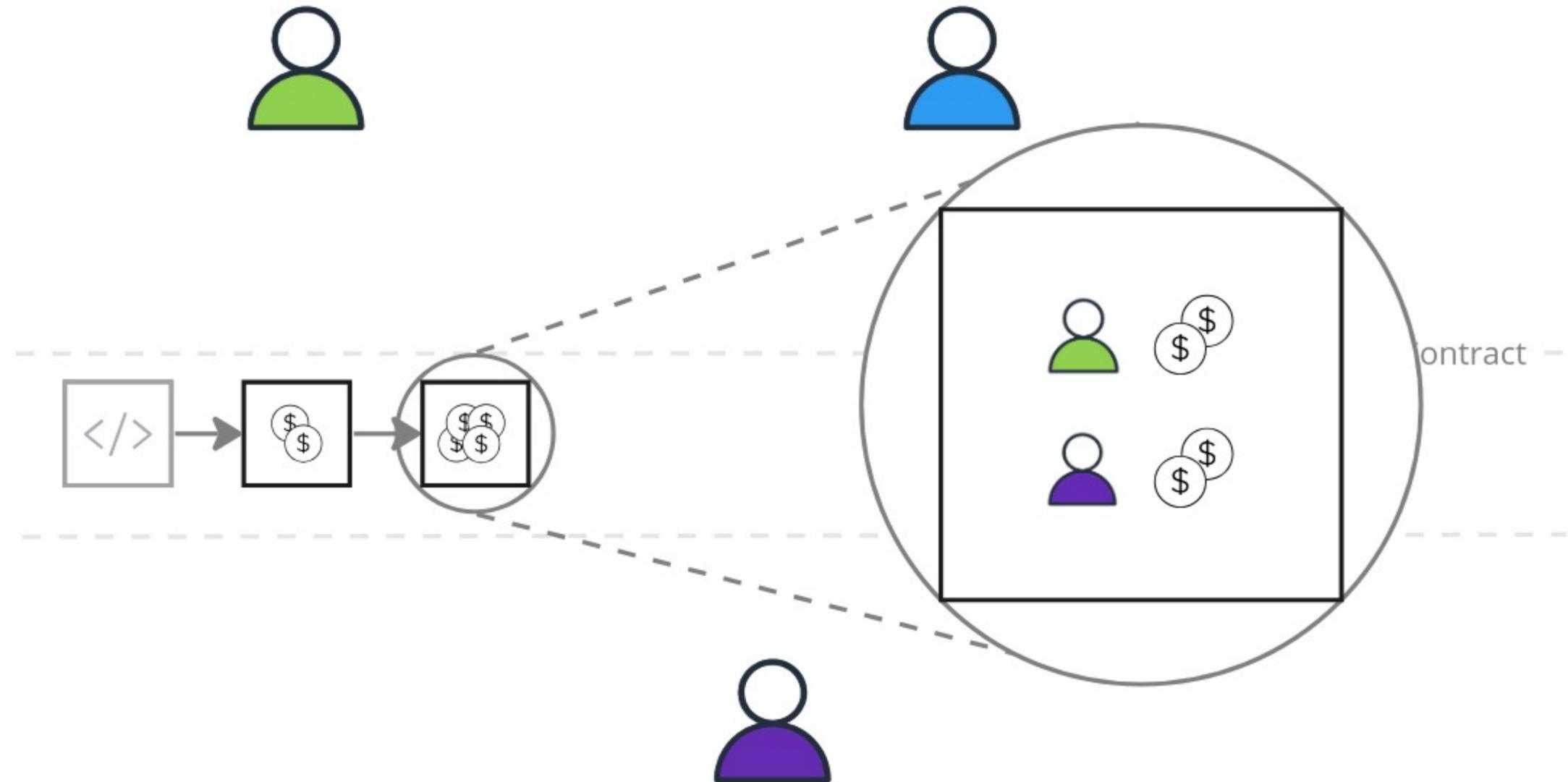
Example: "On-Chain Betting"



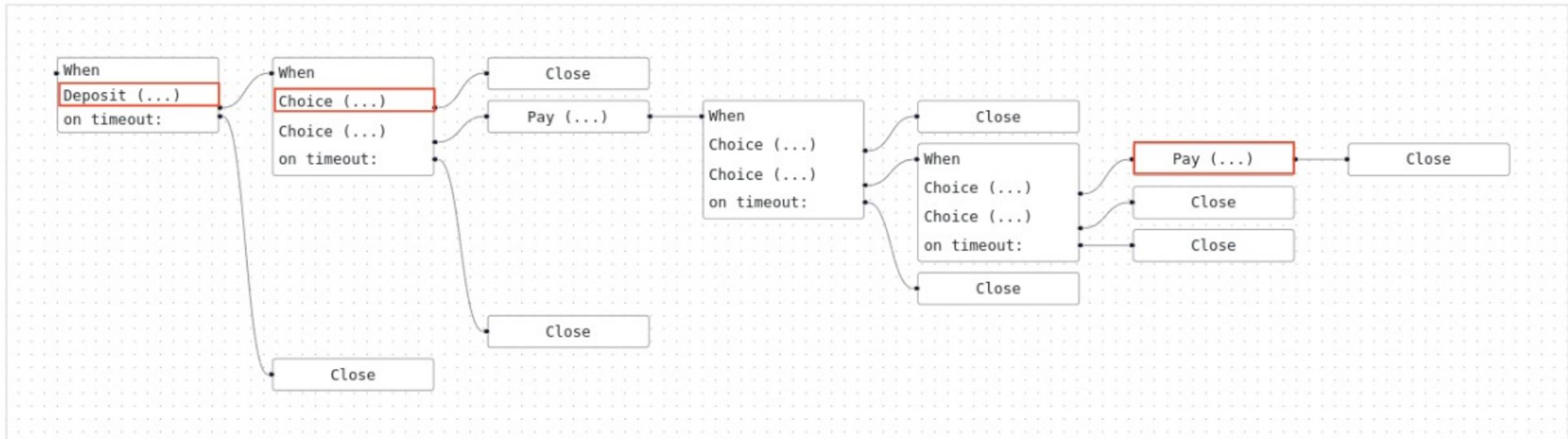
What are Smart Contracts?



What are Smart Contracts?



Anatomy of Marlowe



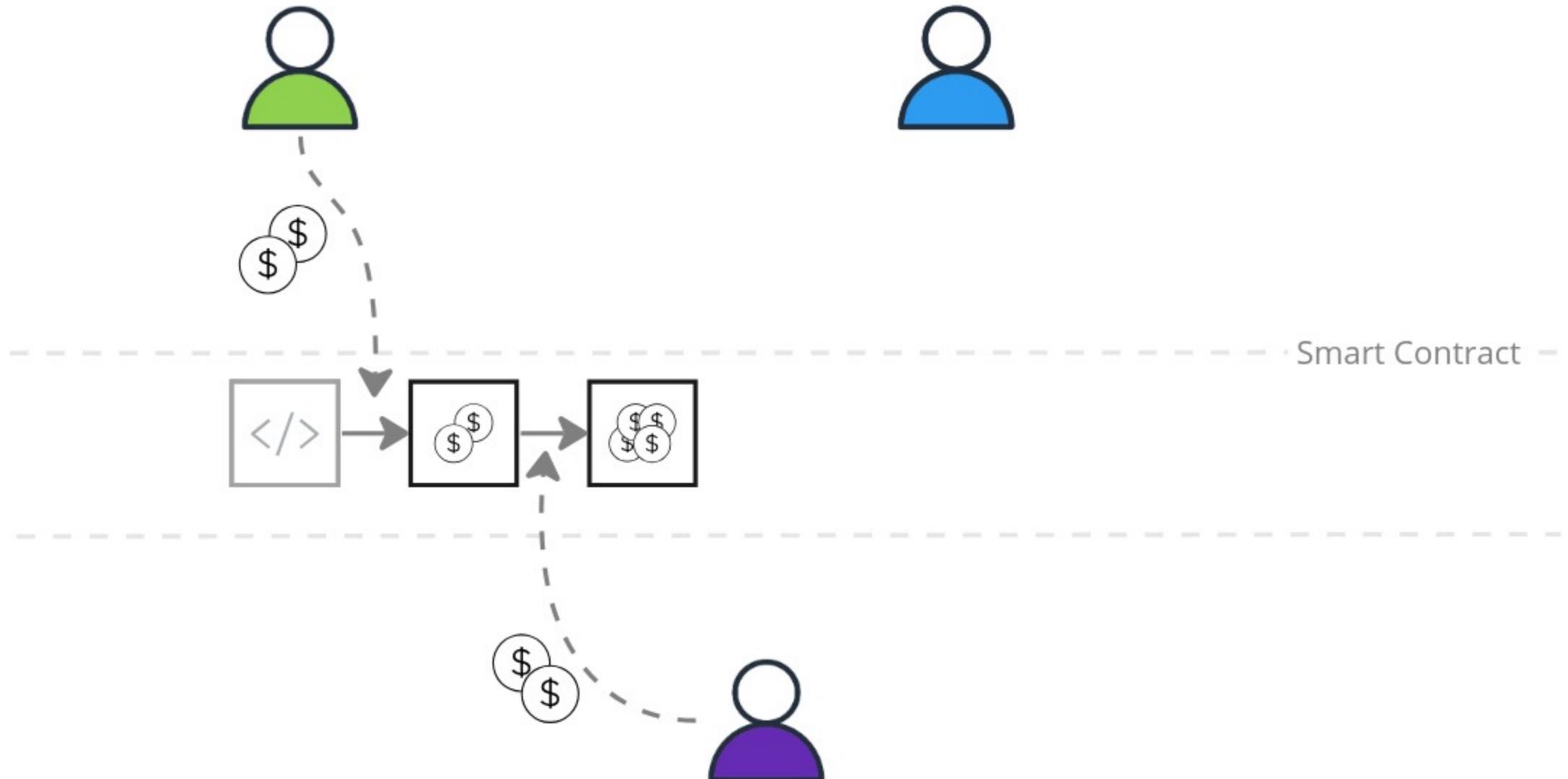
"Deposit" Locks The Assets



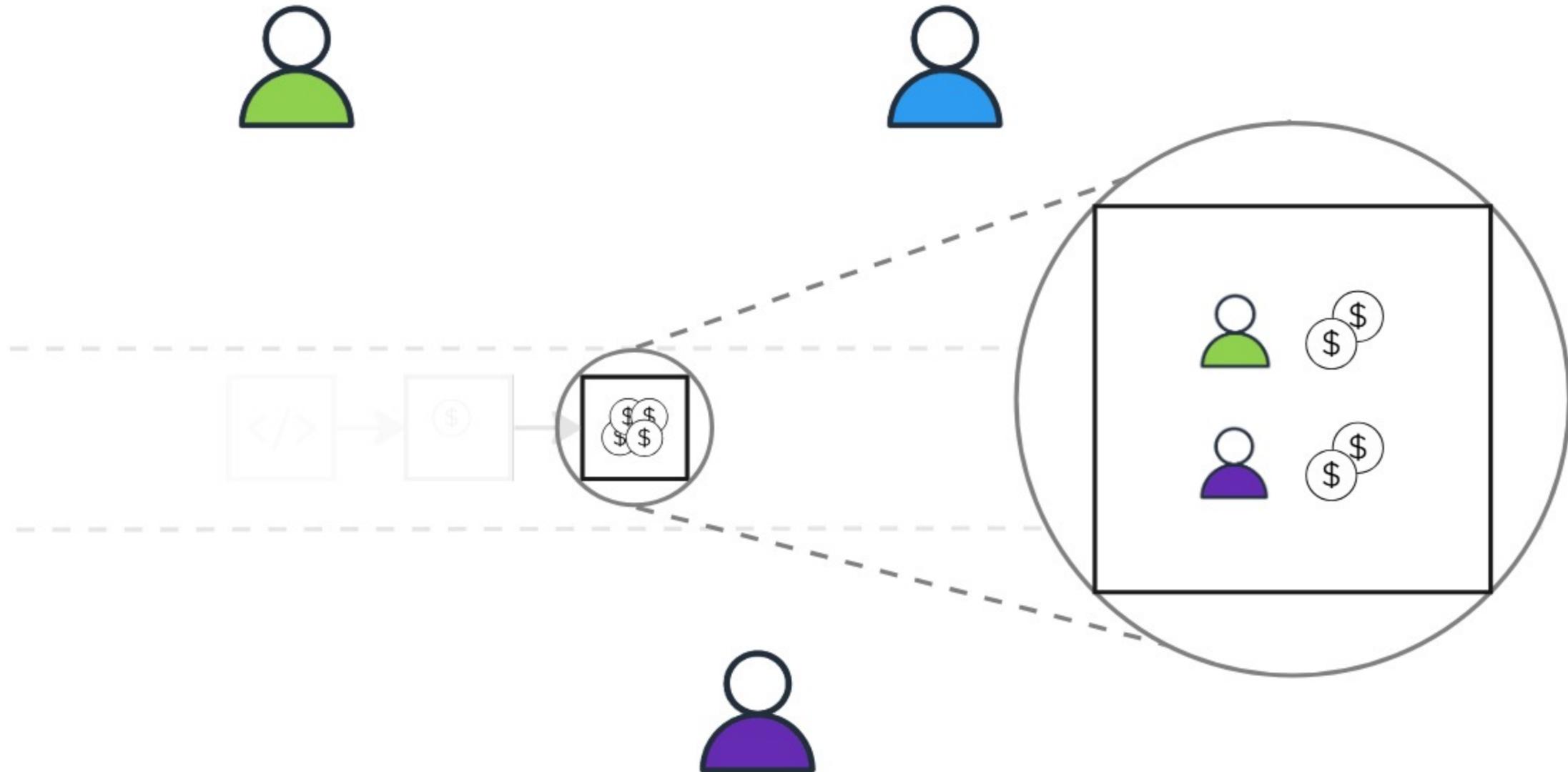
----- · Smart Contract · -----



"Deposit" Locks The Assets



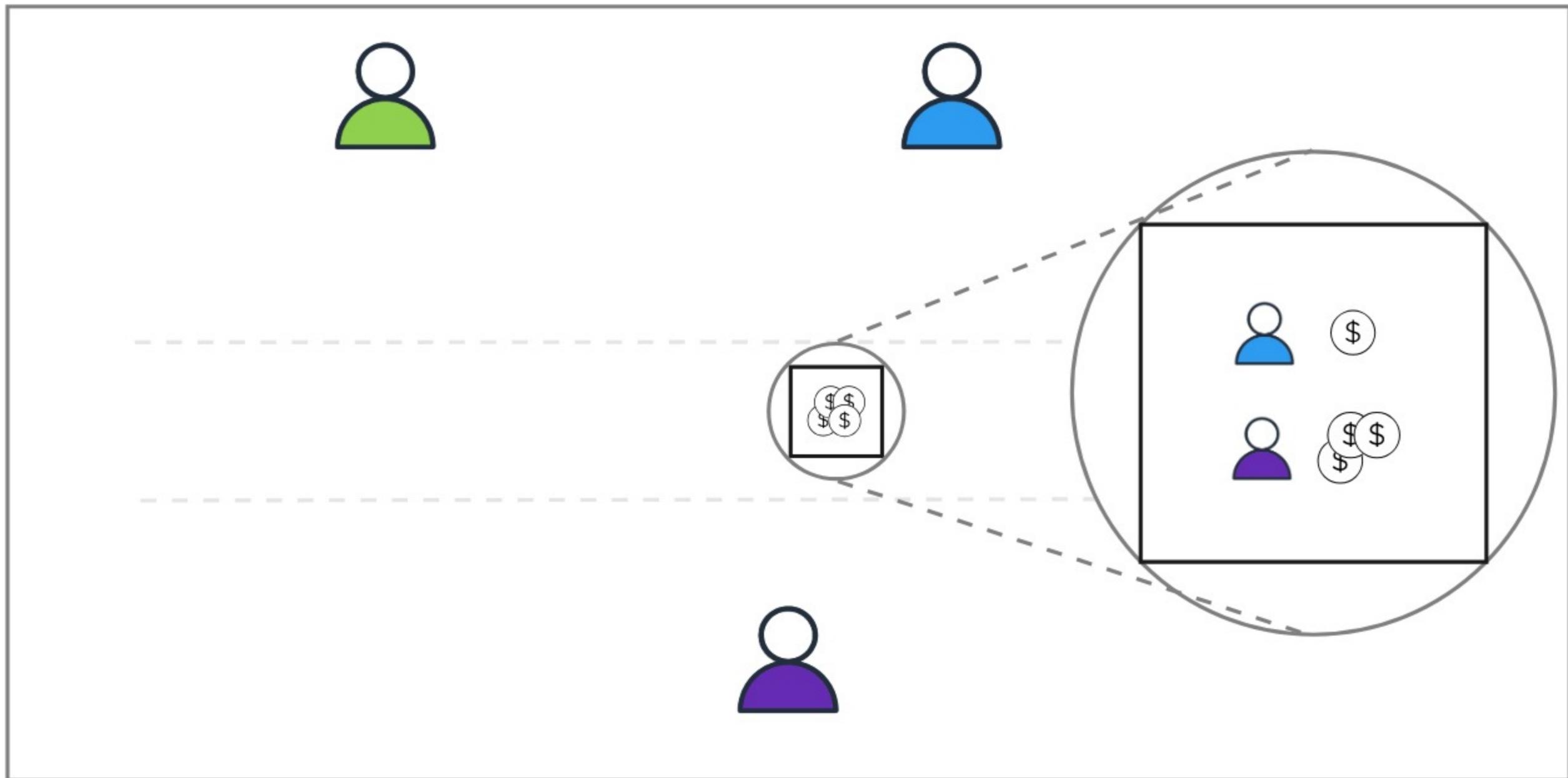
"Deposit" Locks The Assets



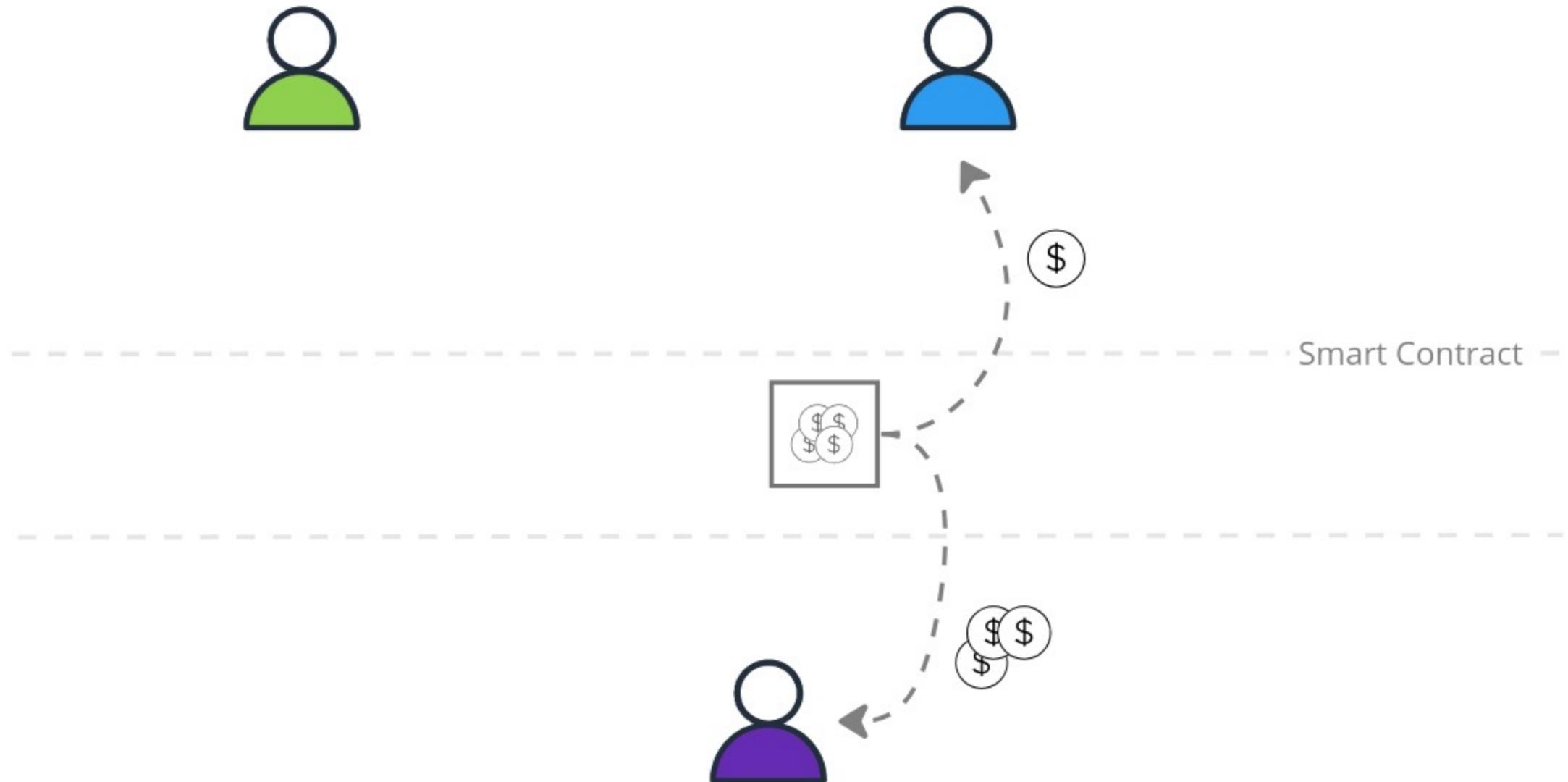
"Close" Releases The Assets

- When we reach "Close" contract releases remaining assets
- The assets are paid out according to account state

"Close" Releases The Assets



"Close" Releases The Assets



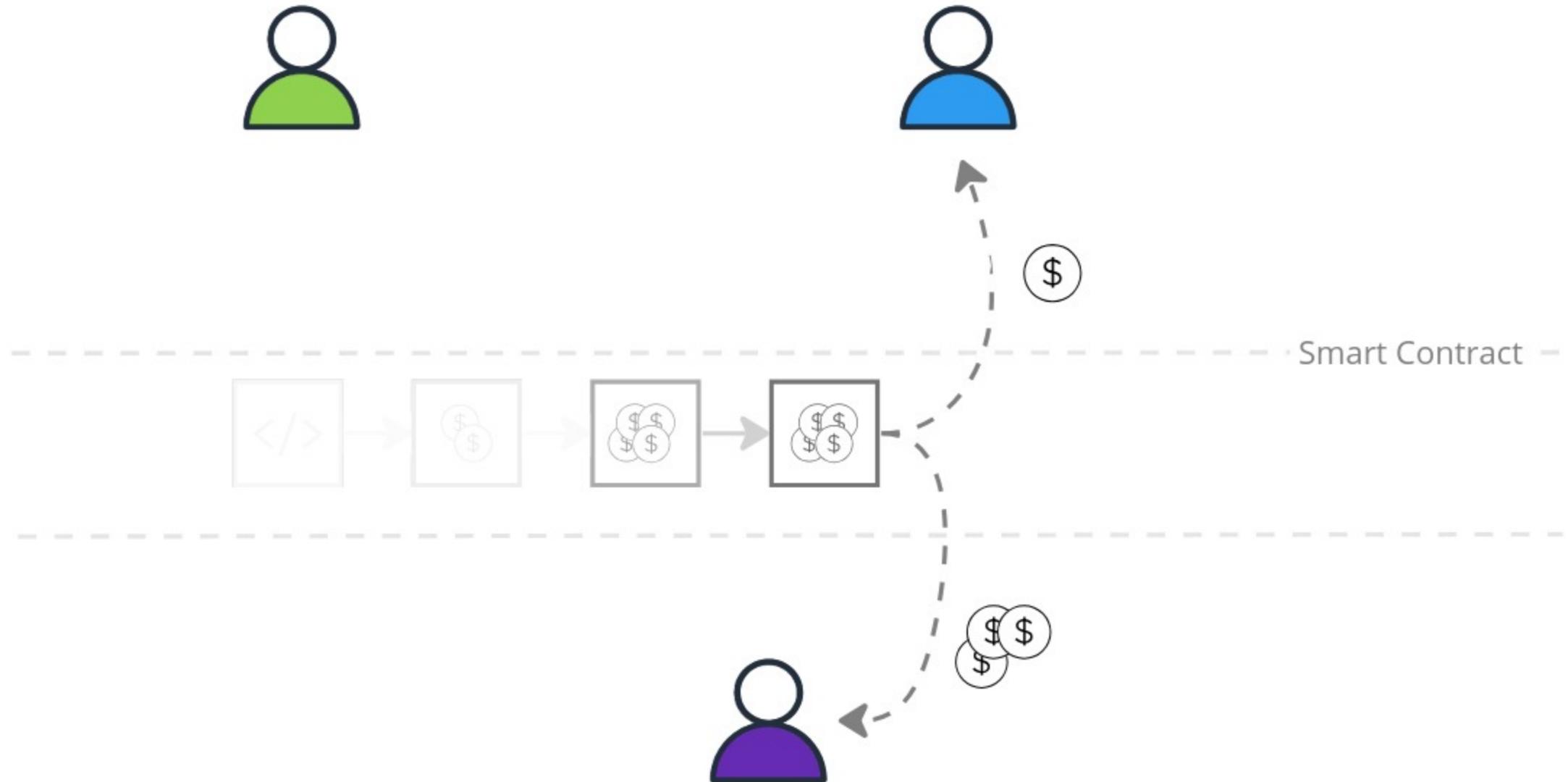
"Close" Releases The Assets



----- · Smart Contract · -----



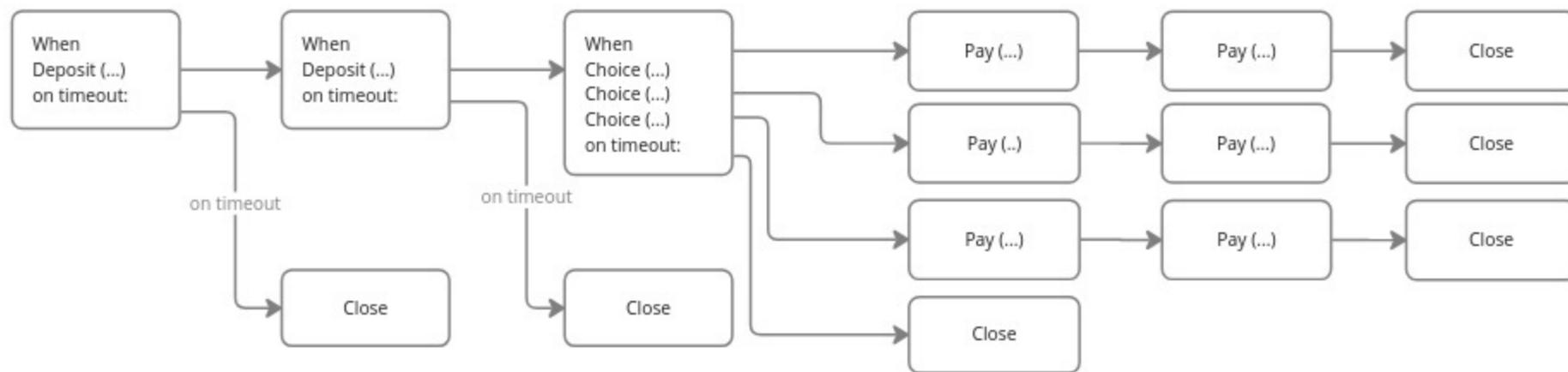
Example: "On-Chain Betting"



On-Chain Betting Benefits

- No up front deposits / account charging etc.
- Predictable cash flow
- Global reaching service
- Everything is transparent
- Arbiter (oracle) is the only trusted party in this setup
- Arbiter has reputation (whole business) at stake

Example: "On-Chain Betting"



play.marlowe.iohk.io