Chain

# Introducing TxVM

Cathie Yun, Software Engineer

JANUARY 25, 2018

# Background

Chain builds **ledger services** for financial institutions.

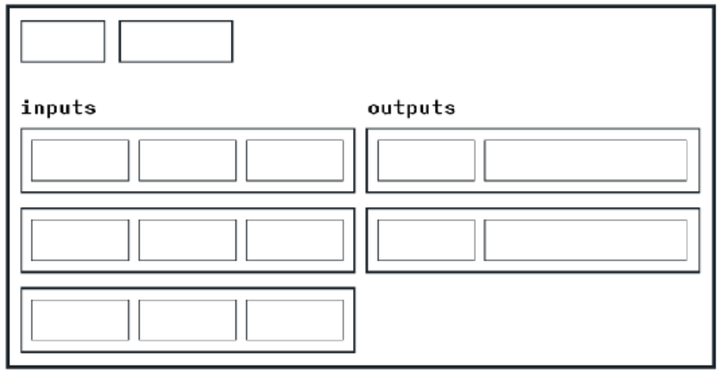
Underlying technology: transactions on a **blockchain**.

Supports **high transaction volumes**, issued assets, and **smart contracts.**

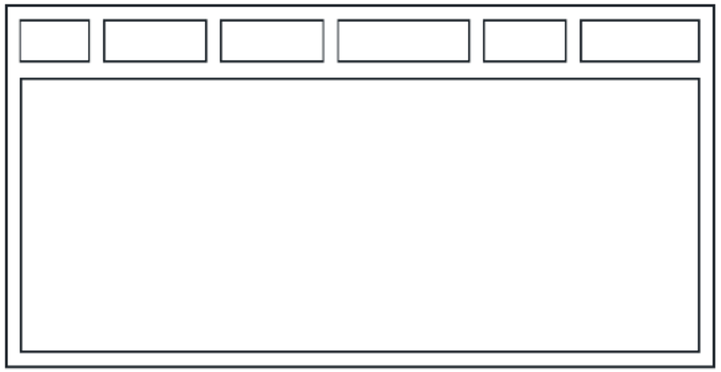
# Existing approaches

Two popular blockchains use different models for transactions and contracts.

## Bitcoin

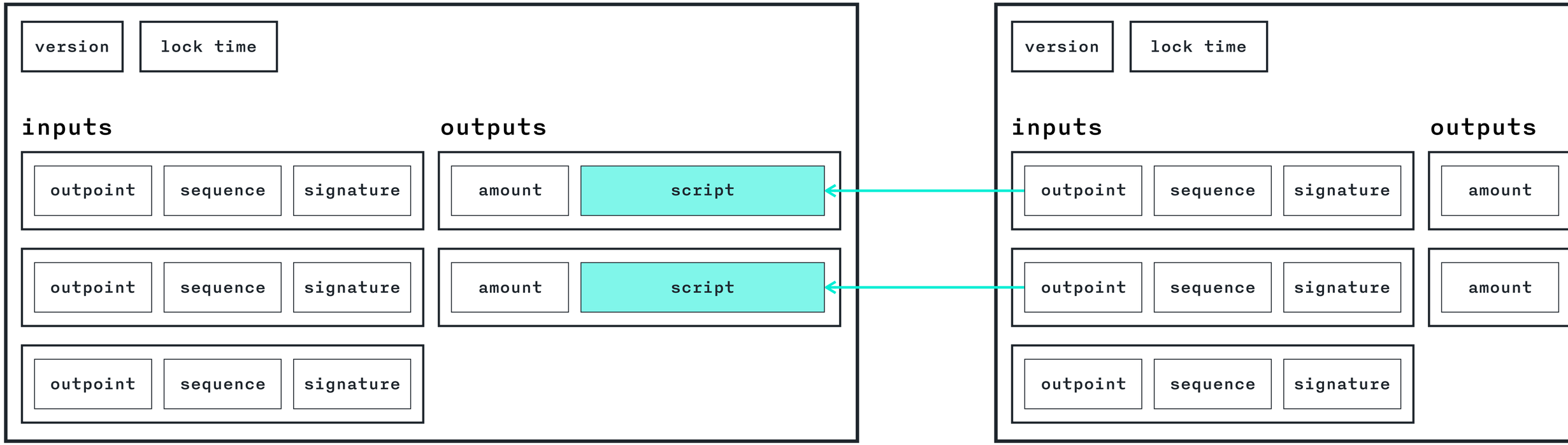



## Ethereum



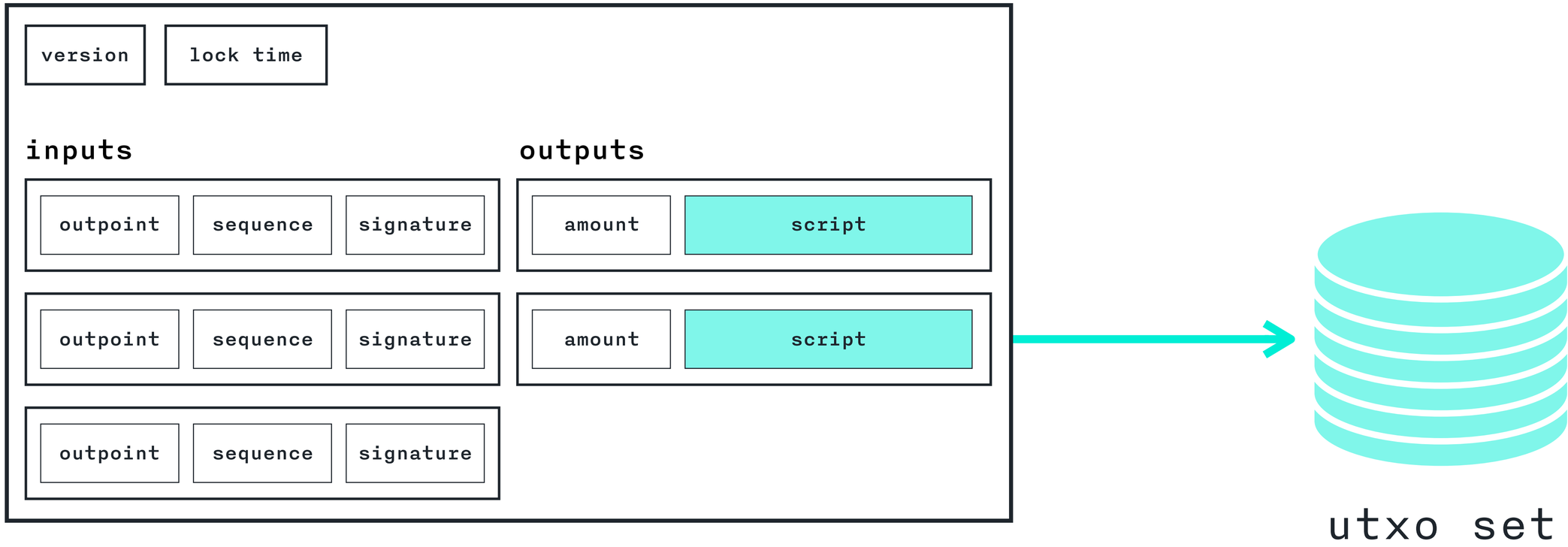
# Bitcoin

Transactions and contracts are **declarative**.



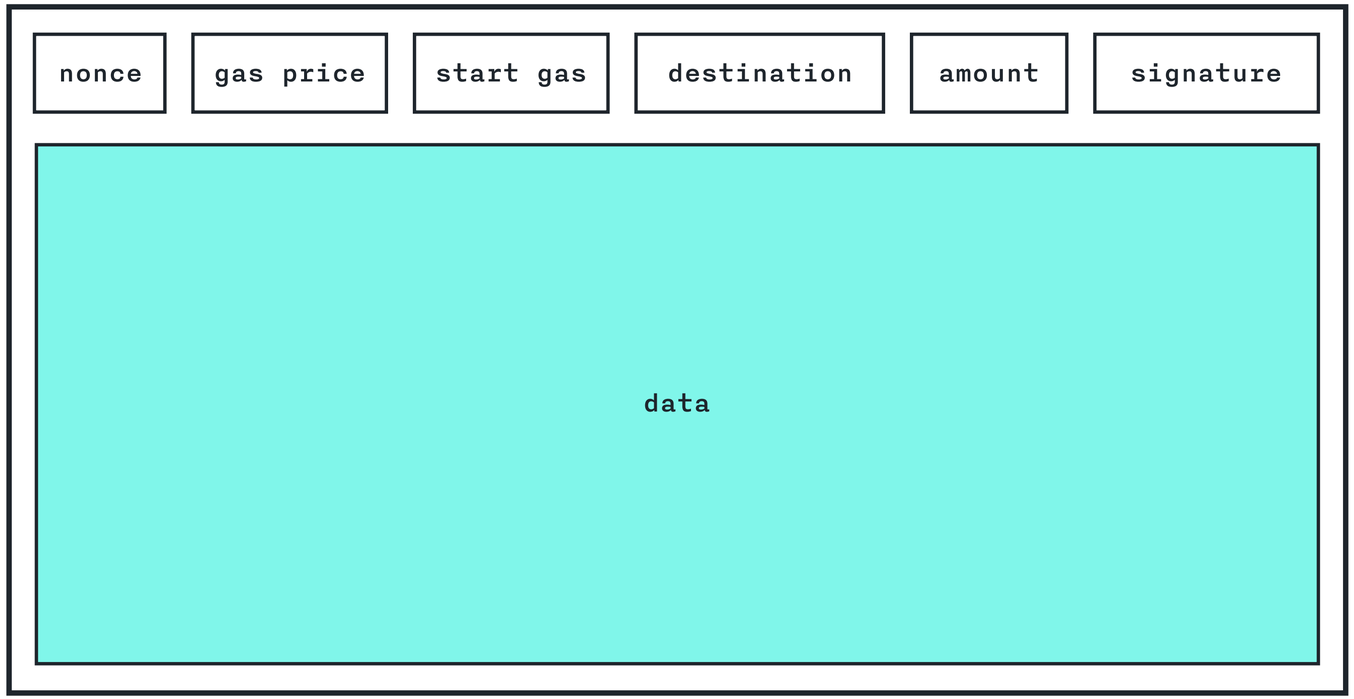
# Bitcoin

Programs are simple predicates. Resulting state is known before publication.



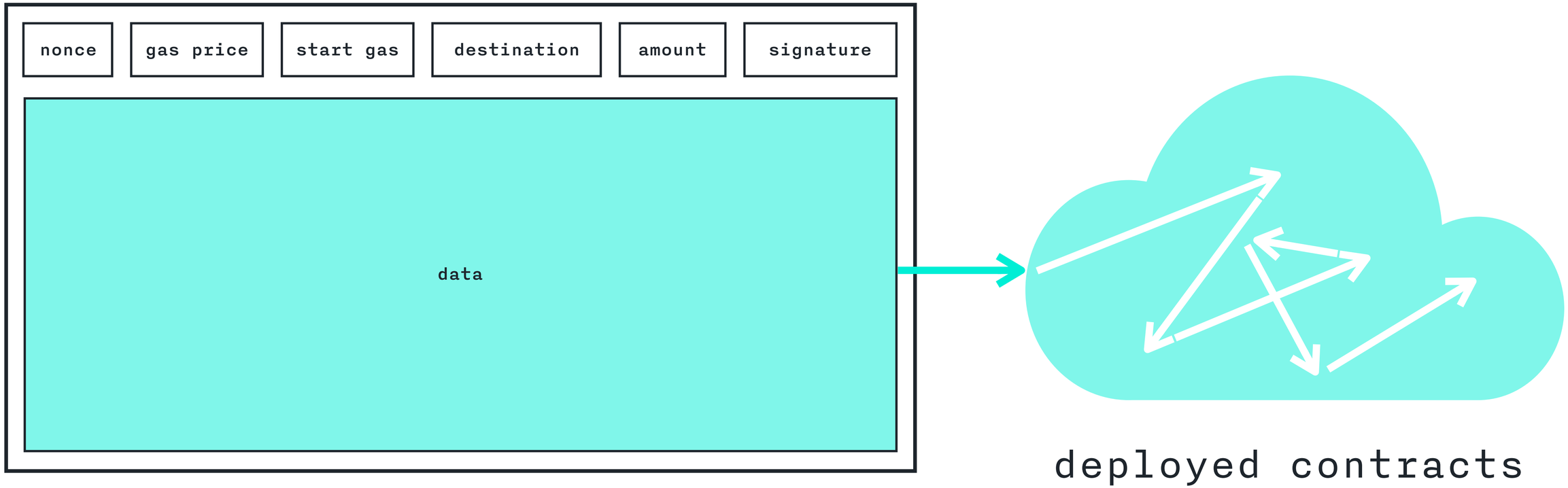
# Ethereum

Transactions and contracts are **imperative**.

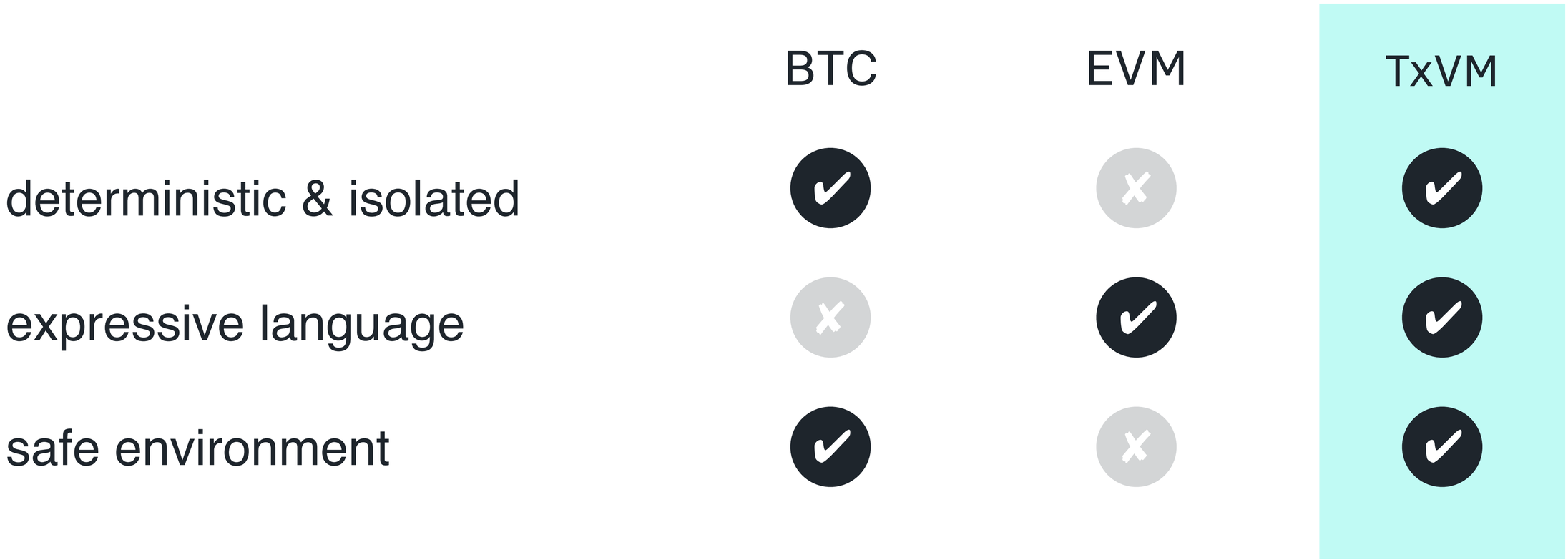


# Ethereum

Contracts can call other contracts. Resulting state is unknown until tx is published.

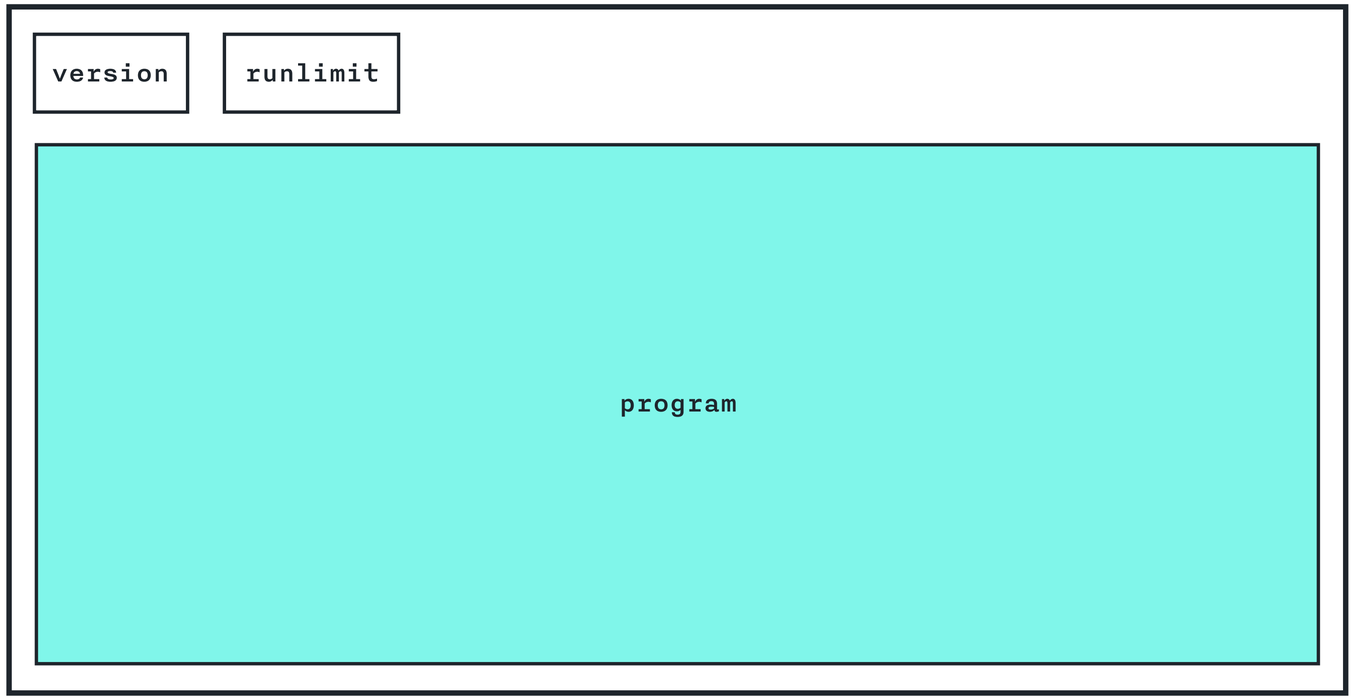


# Introducing TxVM



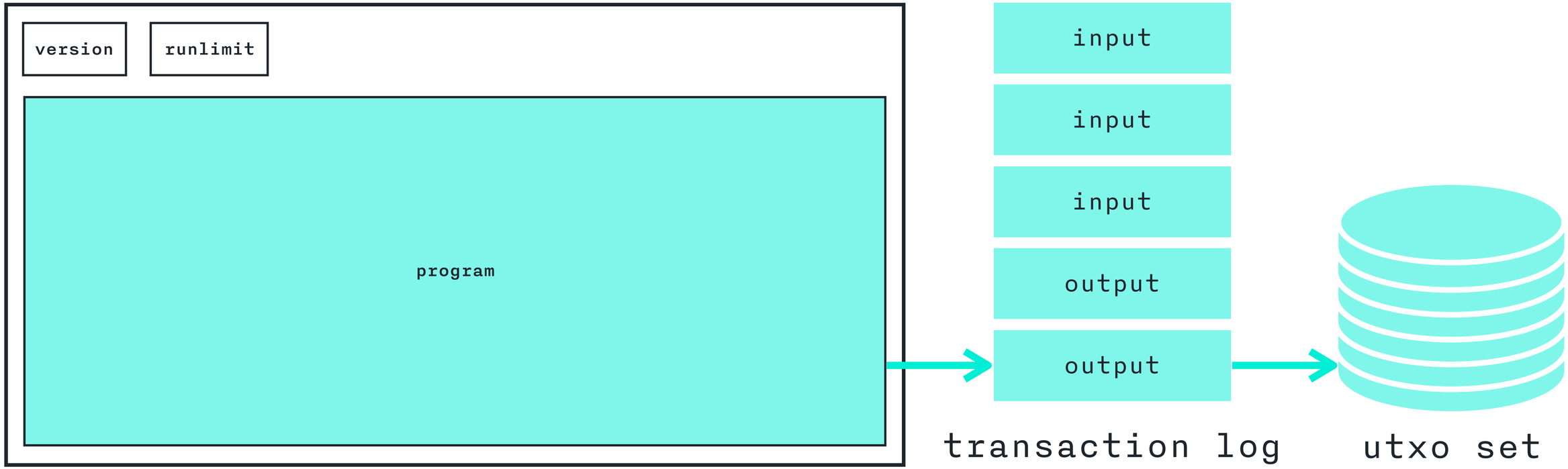
# TxVM

Transaction **is** the program: executes contracts, controls value flow and provides signatures.



# TxVM

Executing the program produces a **deterministic** transaction log.



# TxVM

First class **values** and **contracts** as a part of the VM type system.

1. Values and contracts have **constrained operations** that preserve a "law of conservation”
2. Values and contracts **must be cleared** from the VM by the end of execution.

# First-class Values

Example ride-sharing transaction

1. Operations preserve "law of conservation”
2. Objects must be cleared from the VM by the end

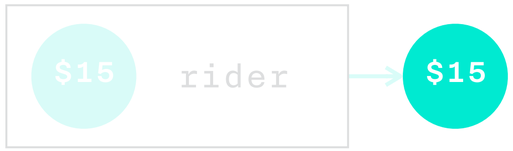
# First-class Values

Example ride-sharing transaction


1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

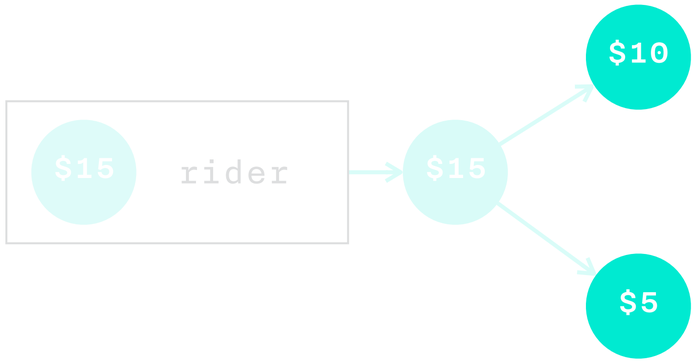
Example ride-sharing transaction




1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

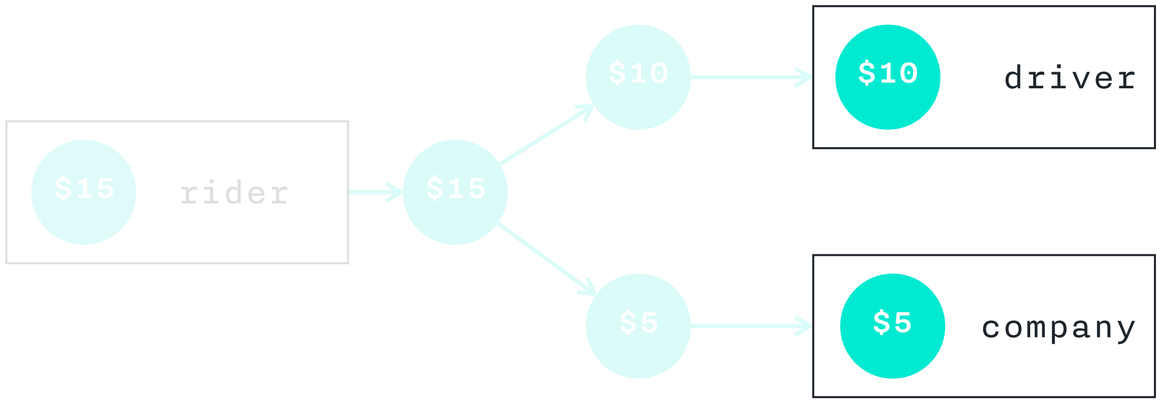
Example ride-sharing transaction




1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

Example ride-sharing transaction




1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

Example ride-sharing transaction




✔ 1. Operations preserve "law of conservation”

✔ 2. All objects must be cleared from the VM

# First-class Values

Impossible application violating rule #1

1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

Impossible application violating rule #1


1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

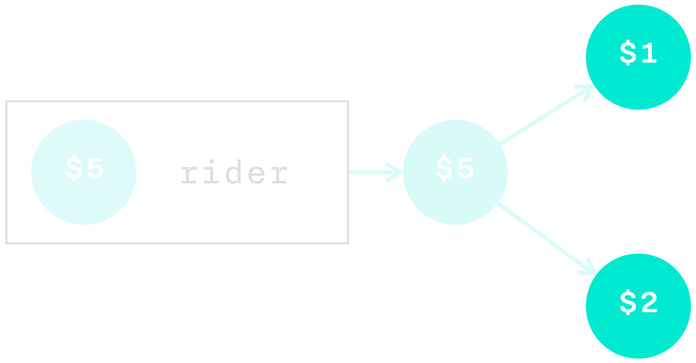
# First-class Values

Impossible application violating rule #1


1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

Impossible application violating rule #1




✗

1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

Invalid transaction violating rule #2

1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

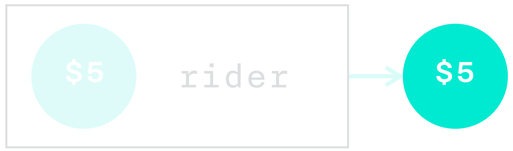
# First-class Values

Invalid transaction violating rule #2


1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

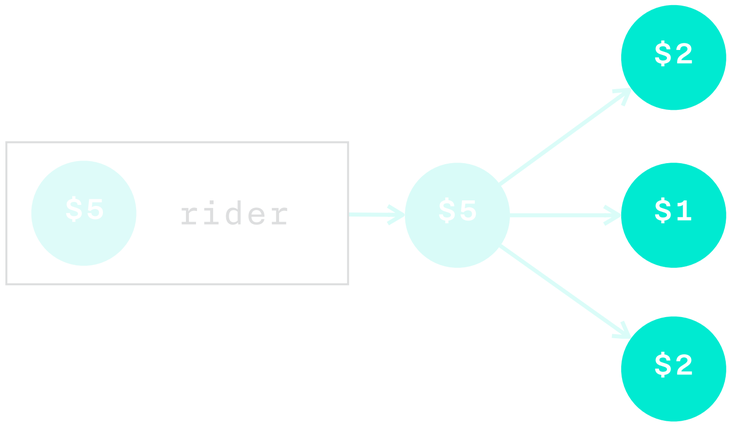
Invalid transaction violating rule #2




1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

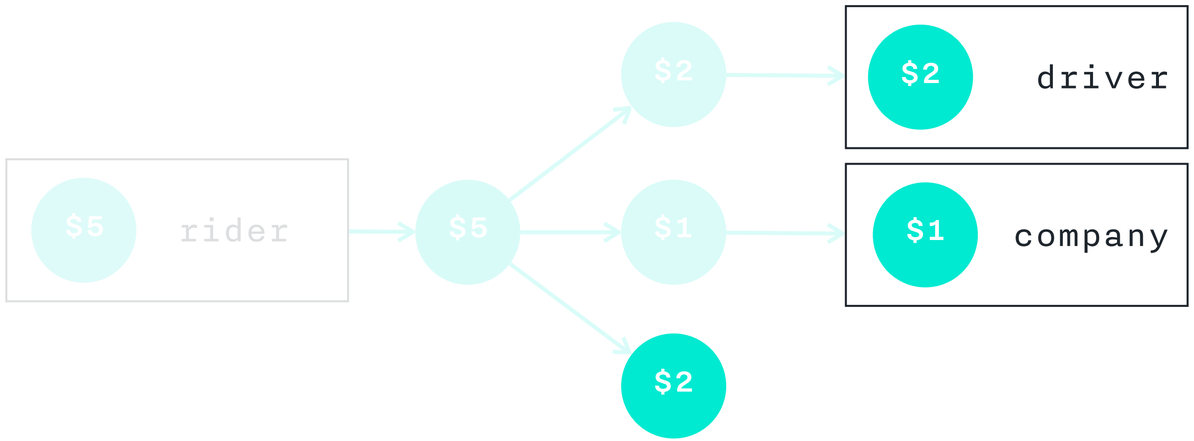
Invalid transaction violating rule #2




1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

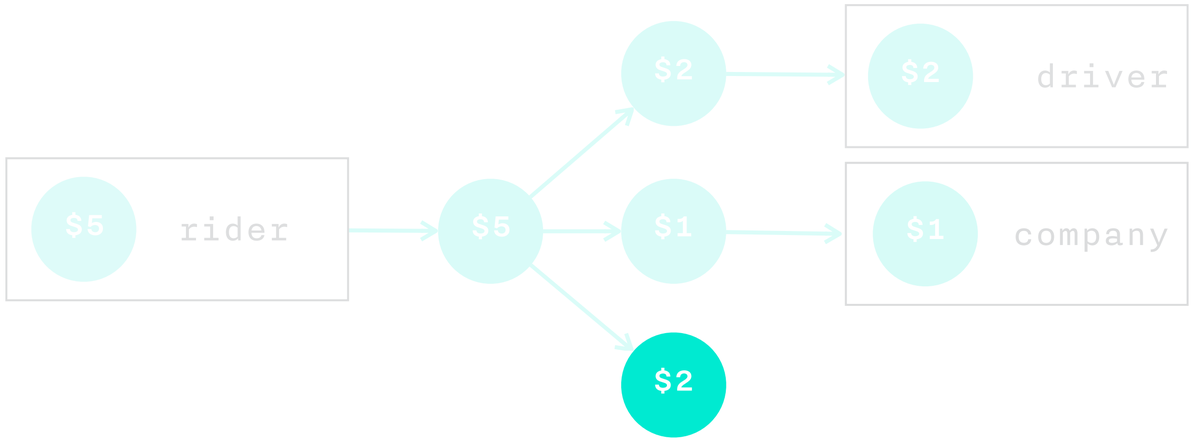
Invalid transaction violating rule #2




1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

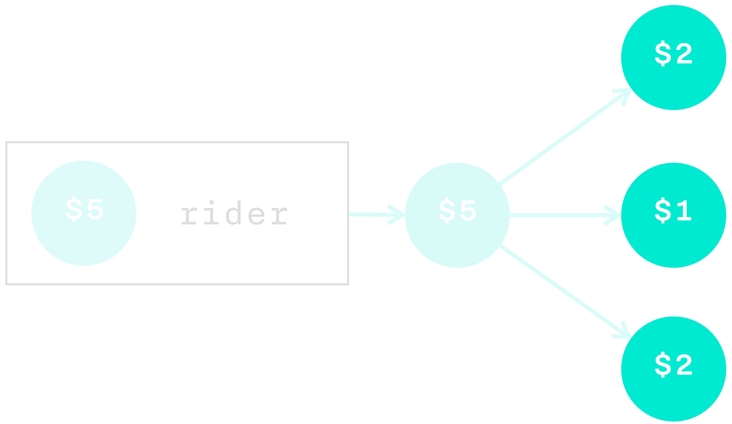
Invalid transaction violating rule #2




1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

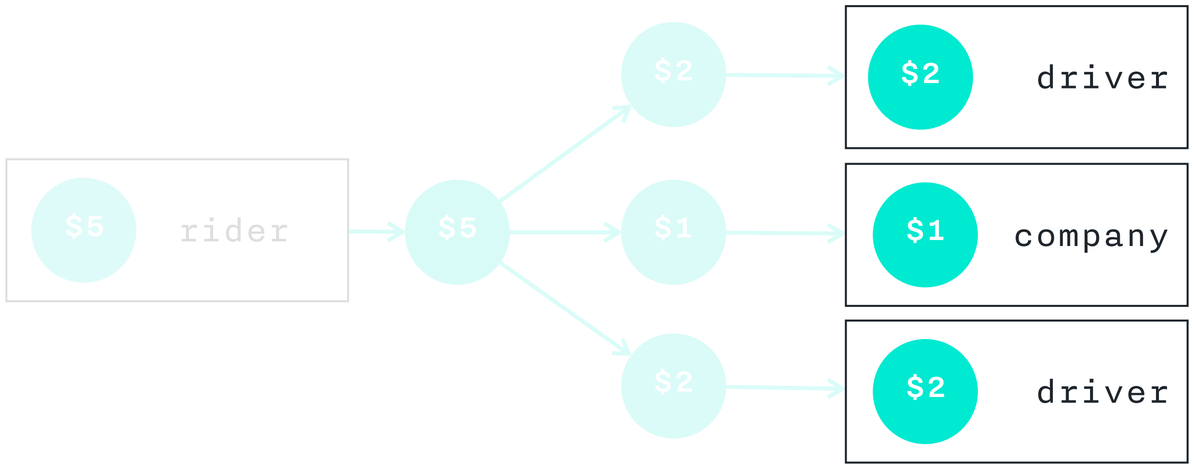
Invalid transaction violating rule #2




1. Operations preserve "law of conservation”
2. All objects must be cleared from the VM

# First-class Values

Invalid transaction violating rule #2




✔ 1. Operations preserve "law of conservation”

✔ 2. All objects must be cleared from the VM

# TxVM internals

1. VM parts

stacks

tx log

runlimit

code

# TxVM internals

1 VM parts

stacks

tx log

runlimit

code

2 VM rules

empty stacks

tx is finalized

runlimit not exceeded

no failures

# TxVM internals

1. VM parts
   - stacks
   - tx log
   - runlimit
   - code
2. VM rules
   - empty stacks
   - tx is finalized
   - runlimit not exceeded
   - no failures
3. Blockchain updates
   - all effects in tx log
   - remove inputs
   - add outputs

# TxVM walkthrough

Alice's $10 `input call`

Alice's $5 `input call`

```
merge_values
```

Carol's account `contract call`

```
finalize

<signatures>... call
```

# TxVM walkthrough

Alice's $10

Alice's $5

```
input call

input call

merge_values
```

Carol's account

```
contract call

finalize

<signatures>... call
```



stack



tx log

# Claim and open an input

Alice’s $10 `input call`

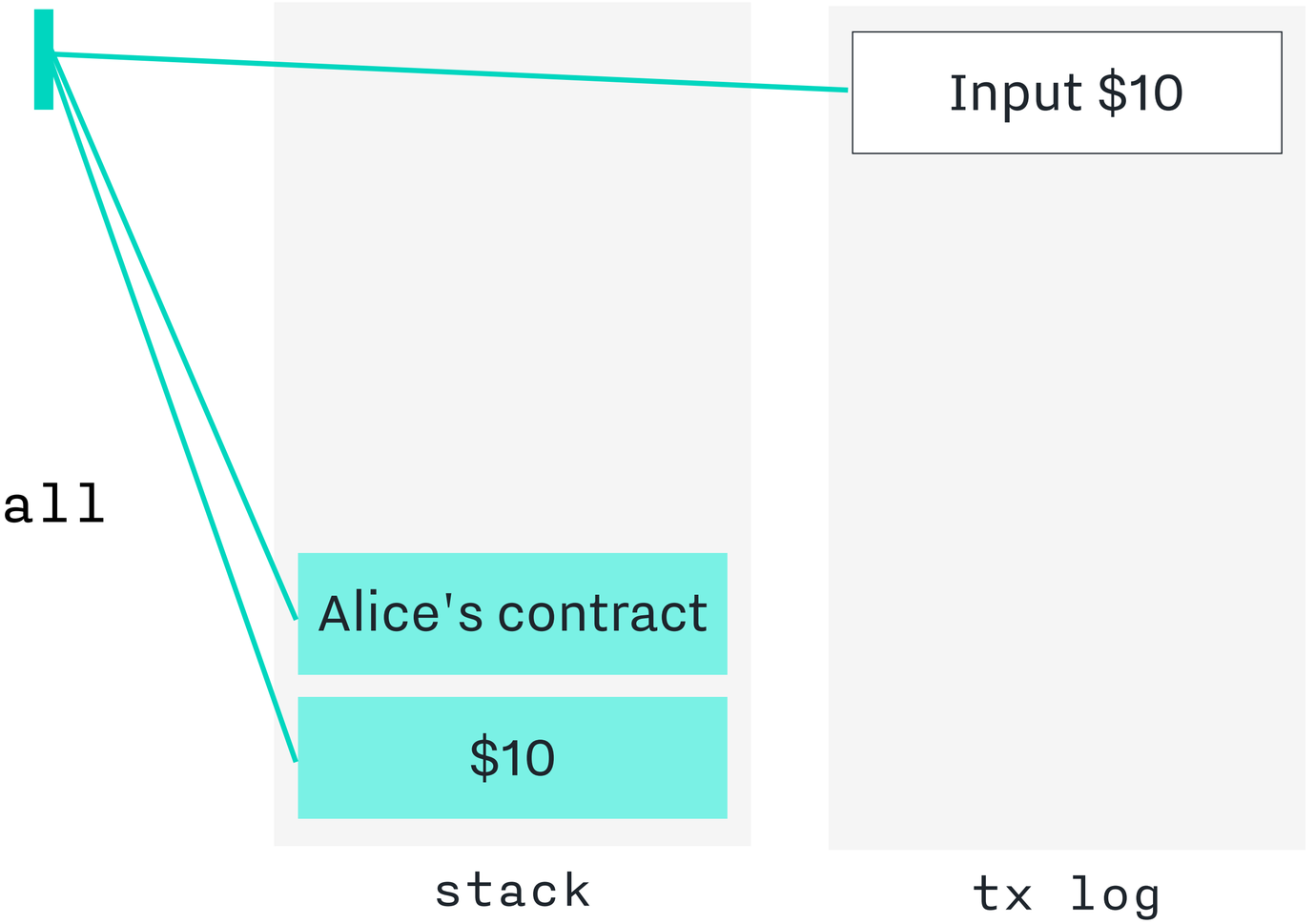
Alice’s $5 `input call`

`merge_values`

Carol’s account `contract call`

`finalize`

`<signatures>... call`



# Claim a second input

Alice’s $10 `input call`

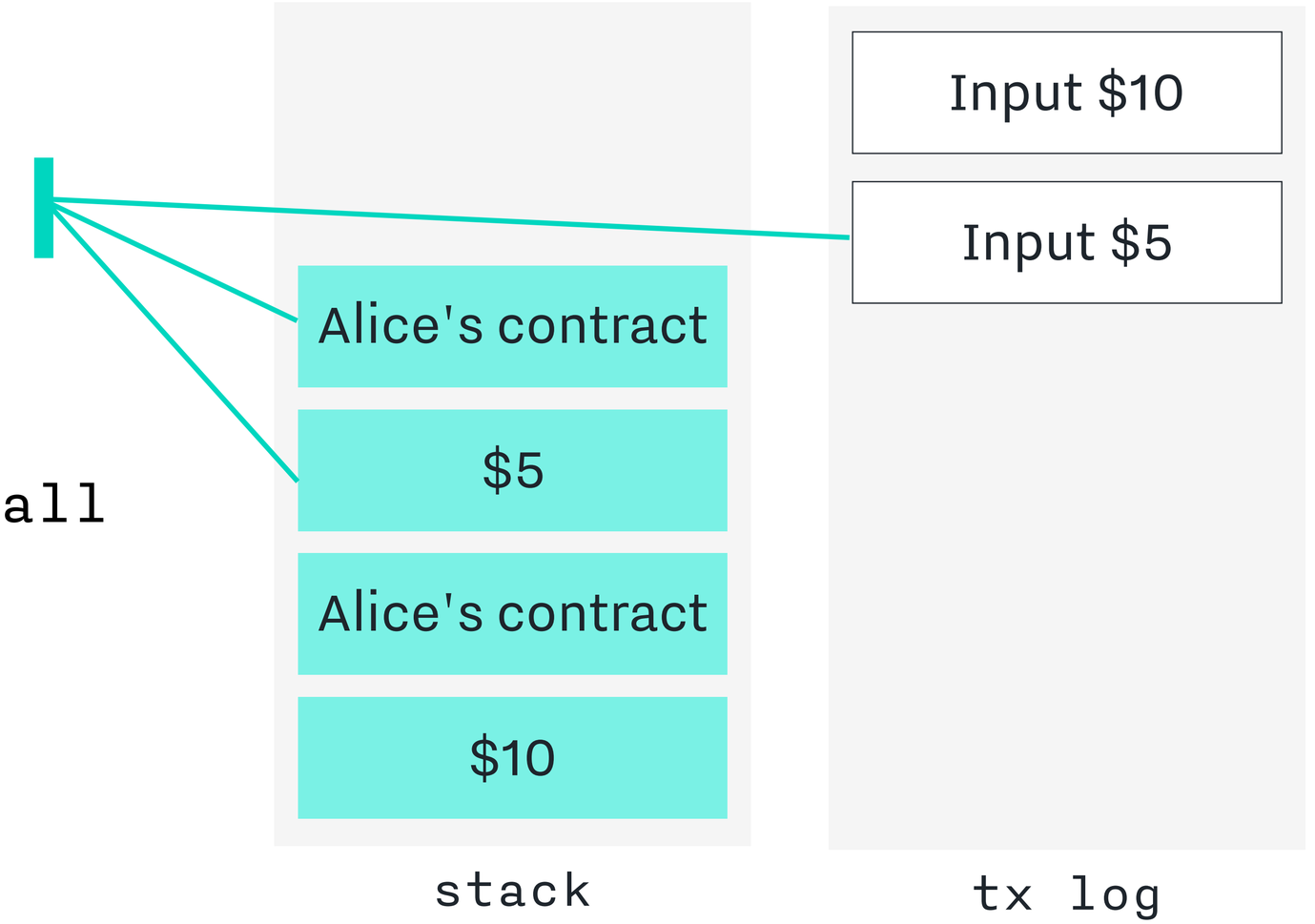
Alice’s $5 `input call`

`merge_values`

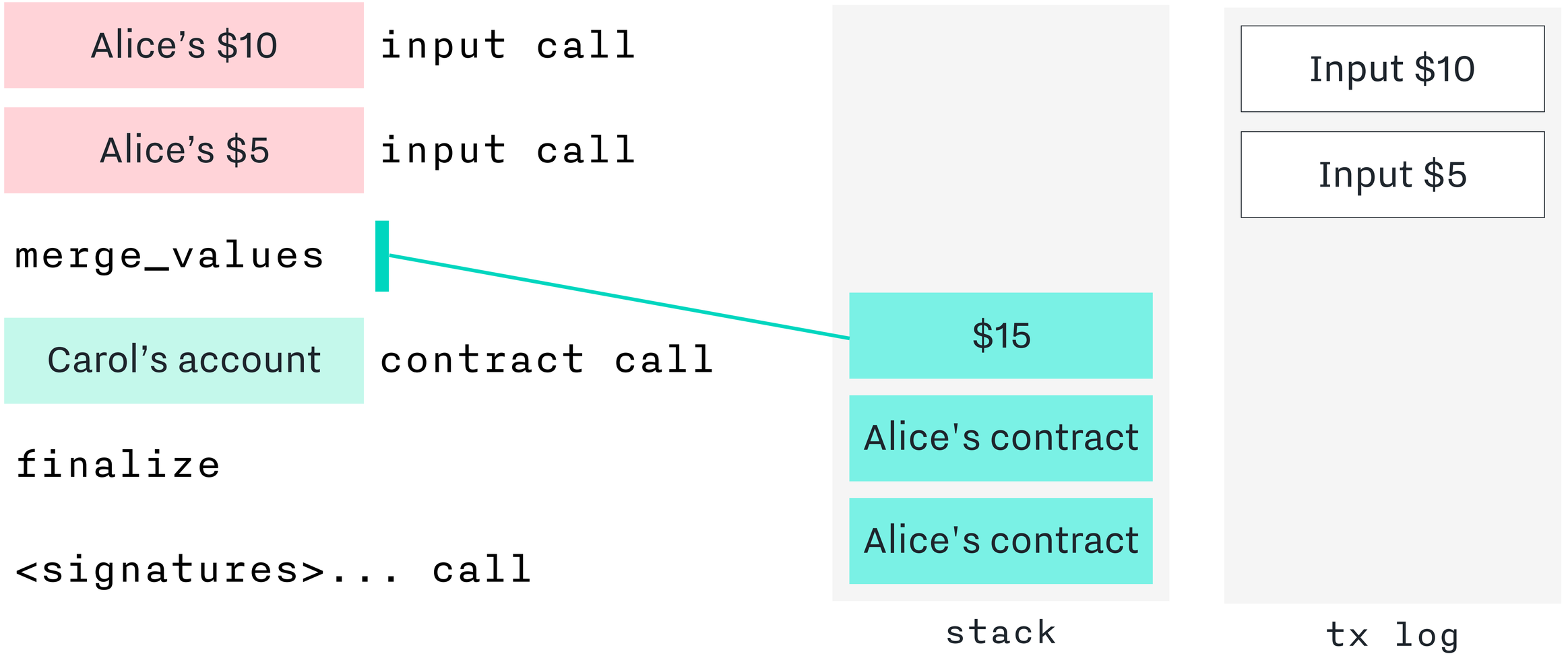
Carol’s account `contract call`

`finalize`

`<signatures>... call`



# Merge and split values



# Lock value in a new contract

Alice's $10 `input call`

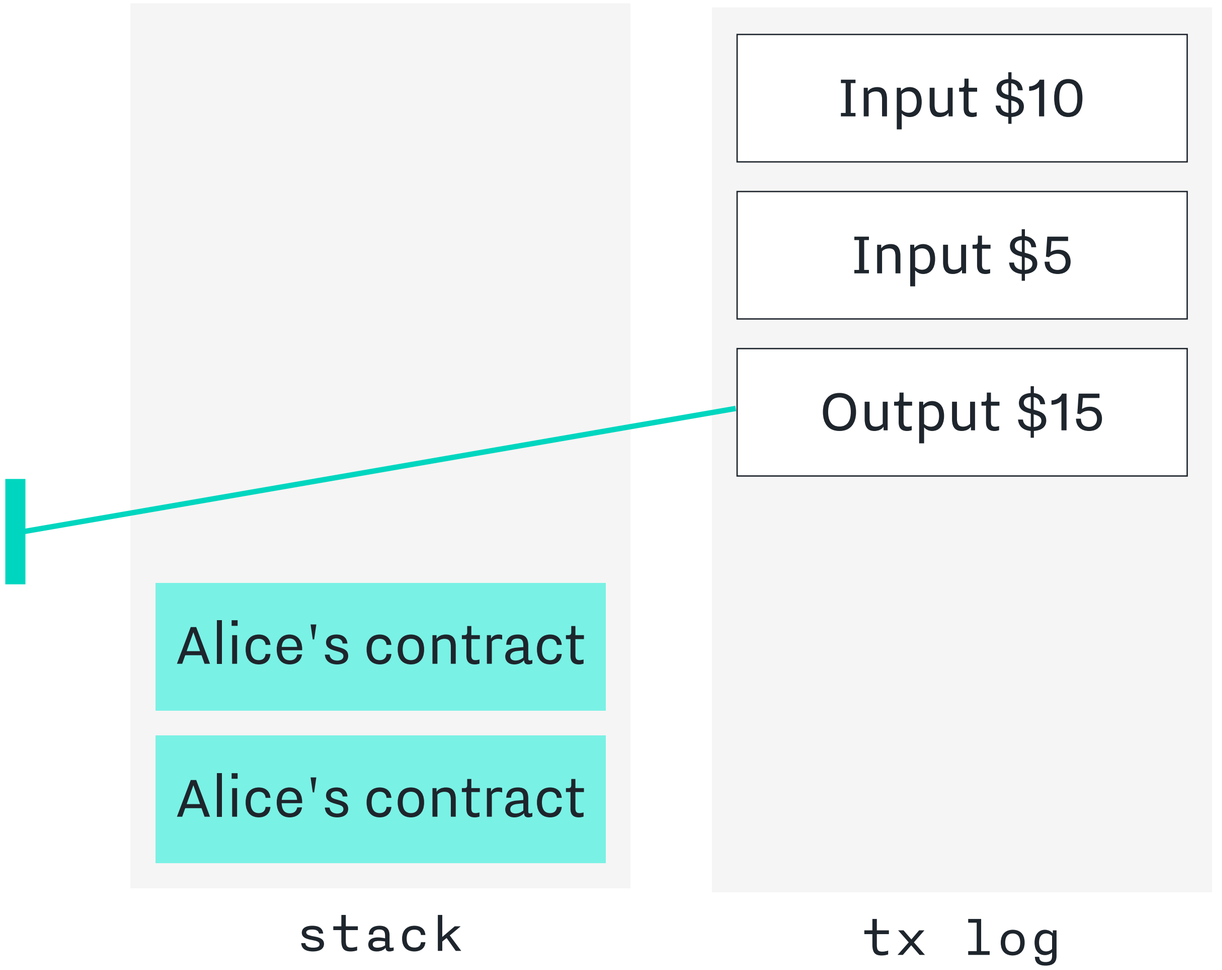
Alice's $5 `input call`

`merge_values`

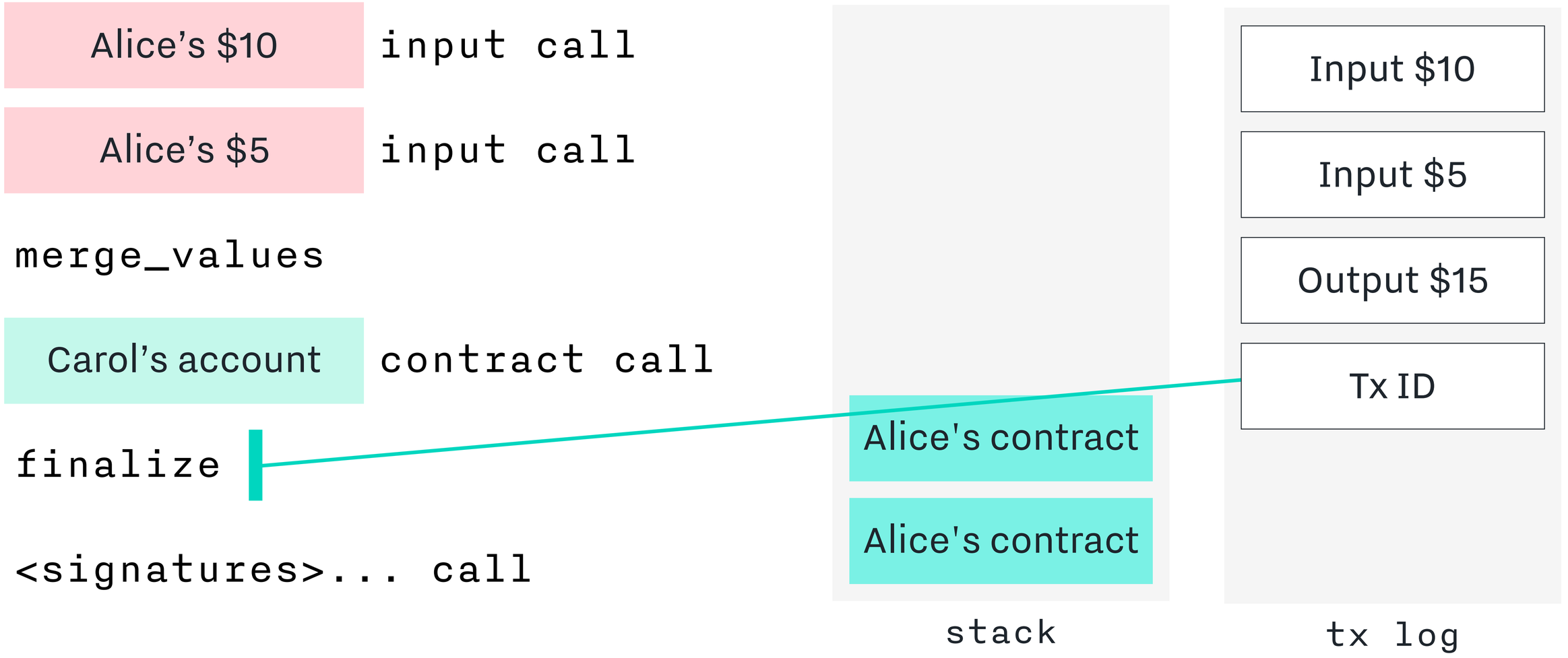
Carol's account `contract call`

`finalize`

`<signatures>... call`



# Freeze modifications, compute transaction ID



# Clear remaining contracts

Alice’s $10 `input call`

Alice’s $5 `input call`

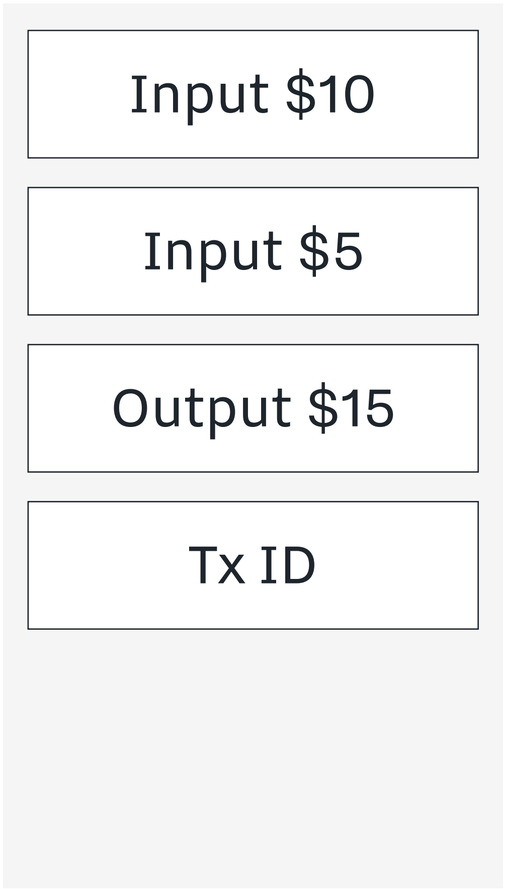
`merge_values`

Carol’s account `contract call`

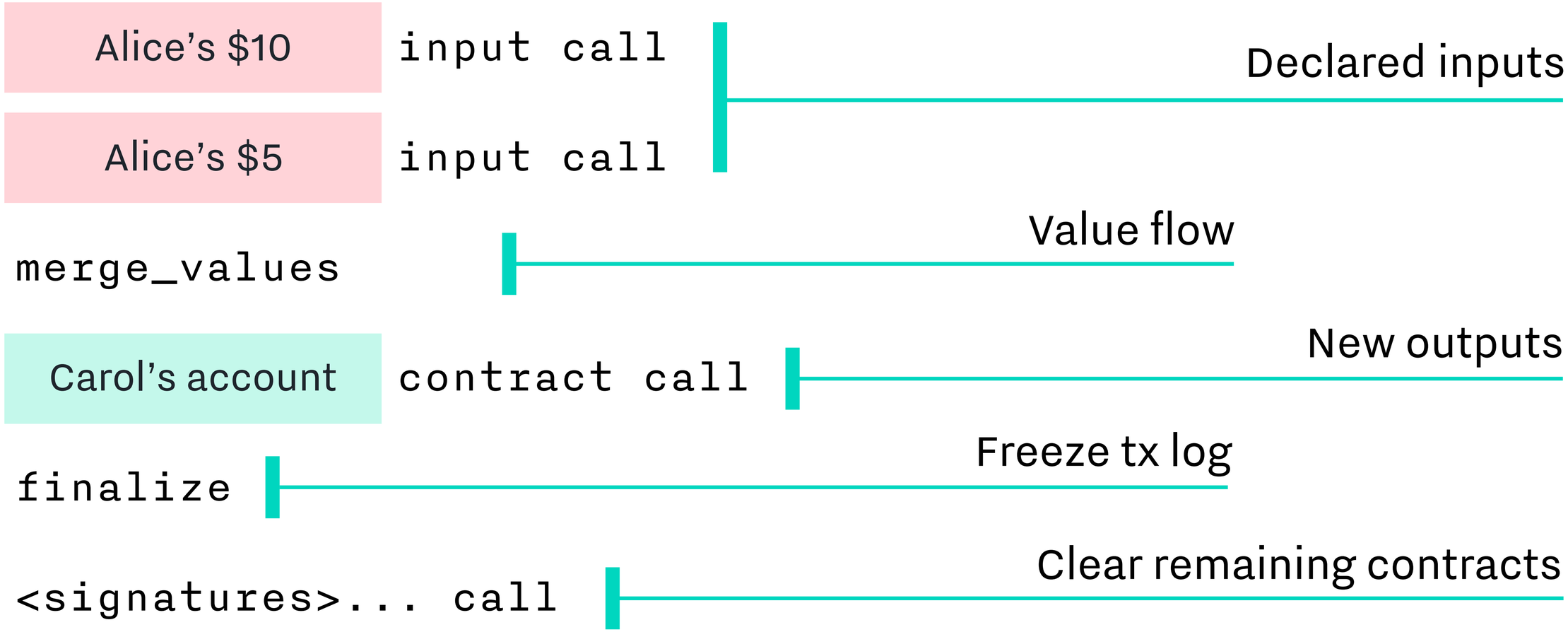
`finalize`

`<signatures>... call`



# TxVM recap


Alice’s $10
input call
Alice’s $5
input call
Declared inputs
merge_values
Value flow
Carol’s account
contract call
New outputs
finalize
Freeze tx log
<signatures>... call
Clear remaining contracts

# Diving into contracts

C2

Alice’s $10 `input call`

Alice’s $5 `input call`

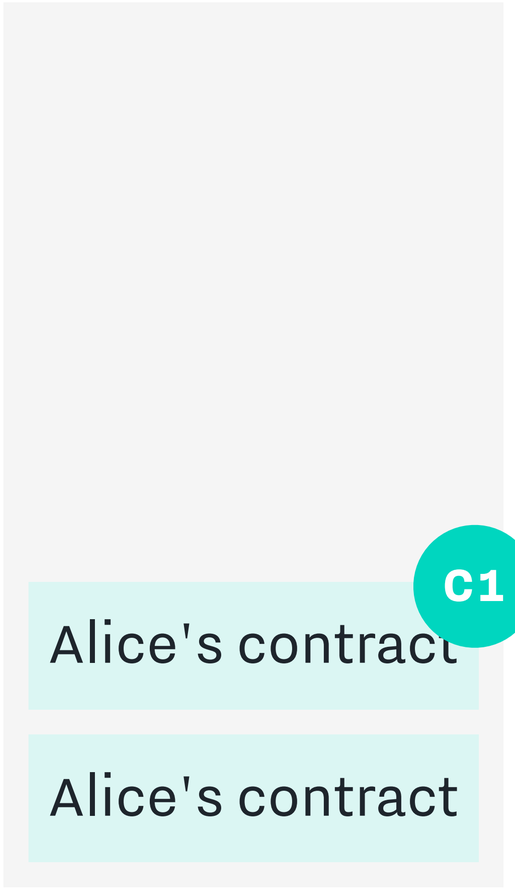
`merge_values`

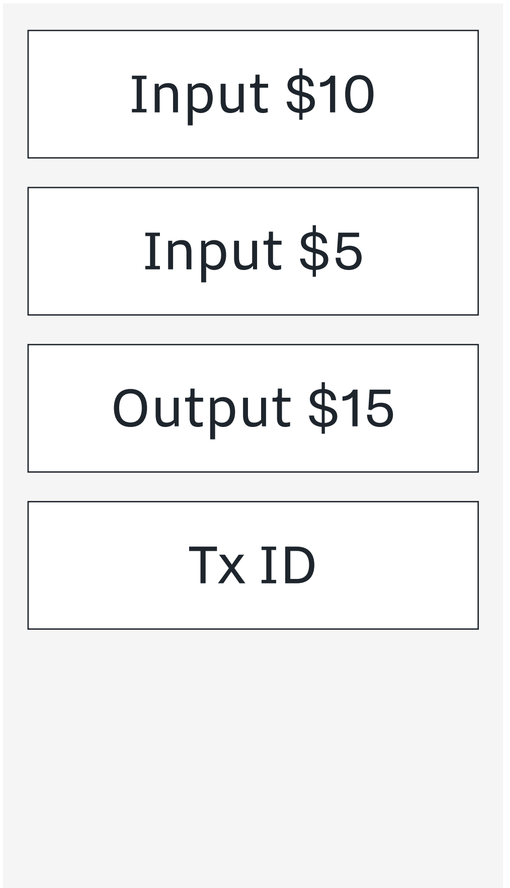
C3

Carol’s account `contract call`

`finalize`

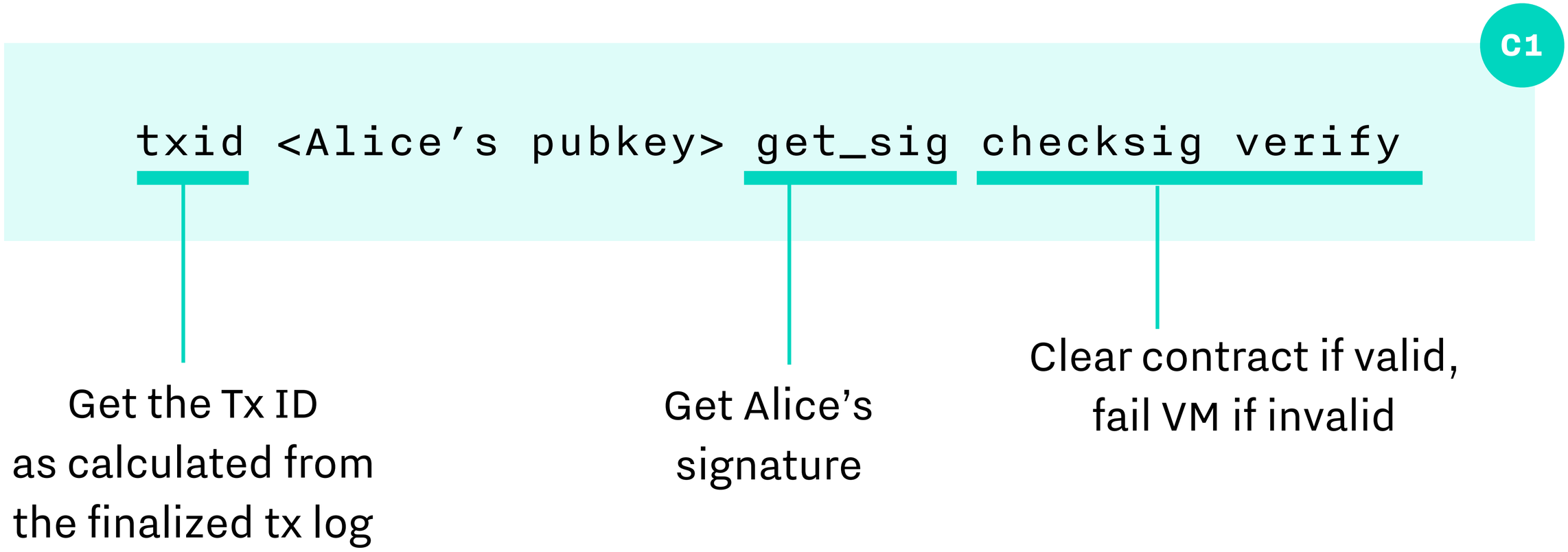
`<signatures>... call`



# Signature contract

Checks a simple signature on transaction ID.



# Diving into contracts

C2

Alice’s $10 `input call`

Alice’s $5 `input call`

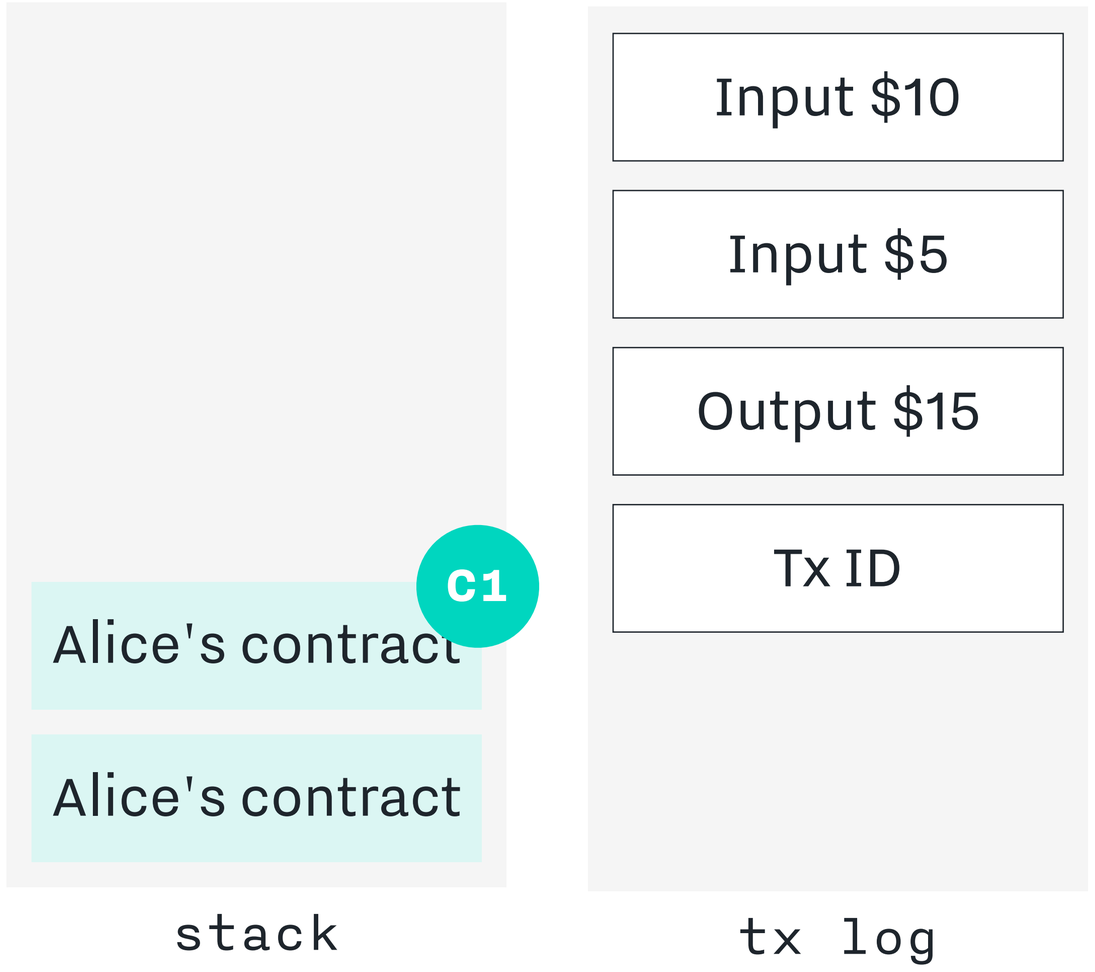
`merge_values`

C3

Carol’s account `contract call`

`finalize`

`<signatures>... call`



# Unspent output contract

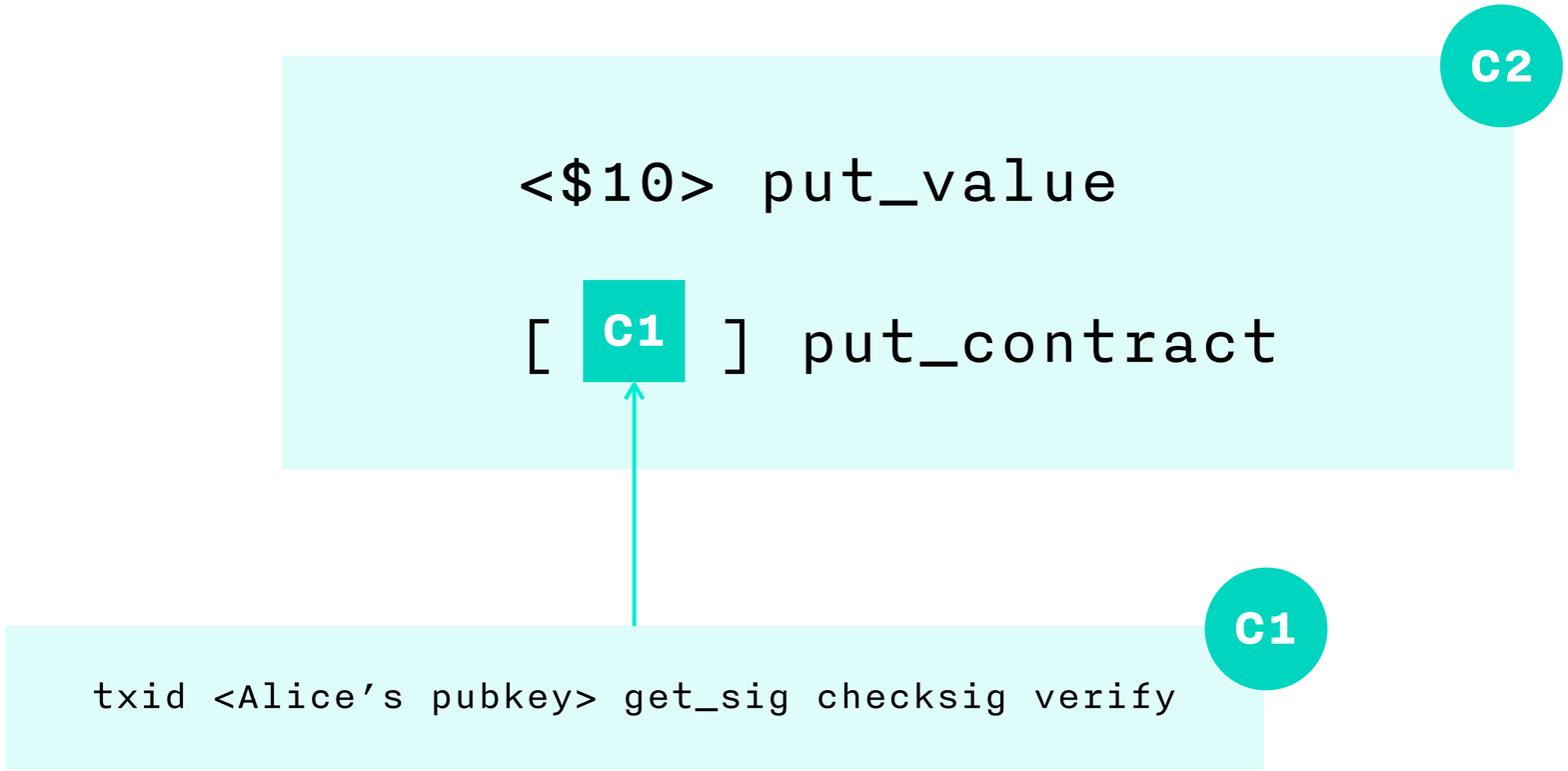
Unlocks the value and defers a checksig contract.

C2

```
<$10> put_value
  [txid <Alice’s pubkey> get_sig checksig verify]
                                              put_contract
```

# Unspent output contract

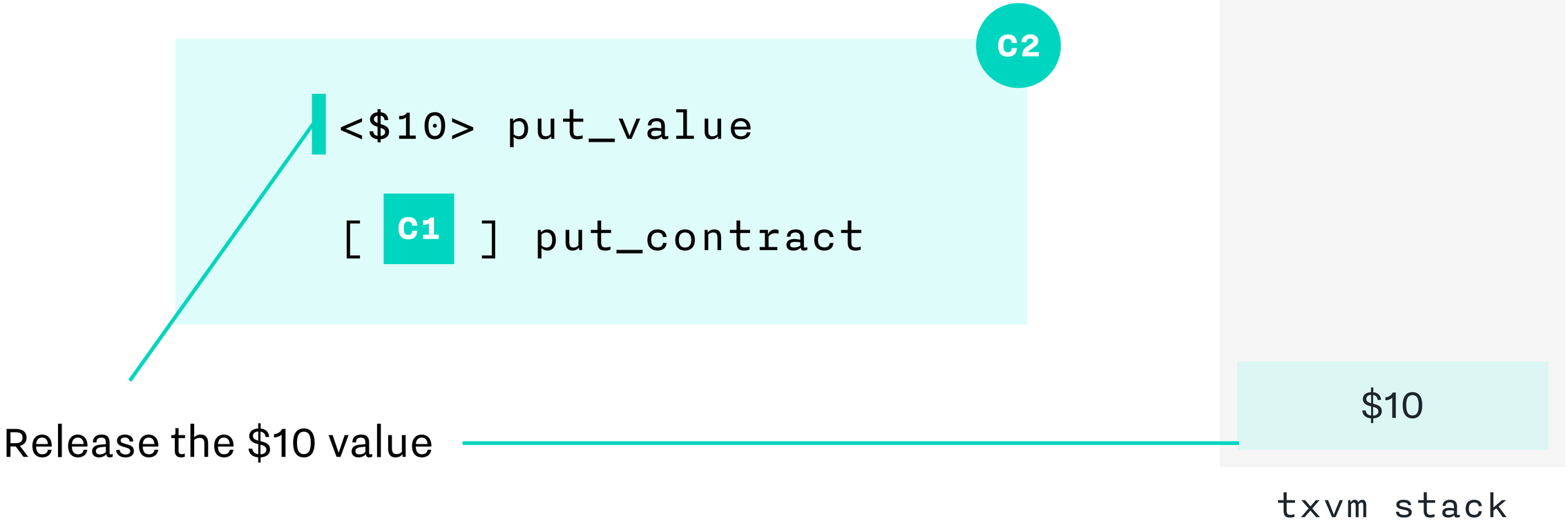
Unlocks the value and defers a checksig contract.




txvm stack

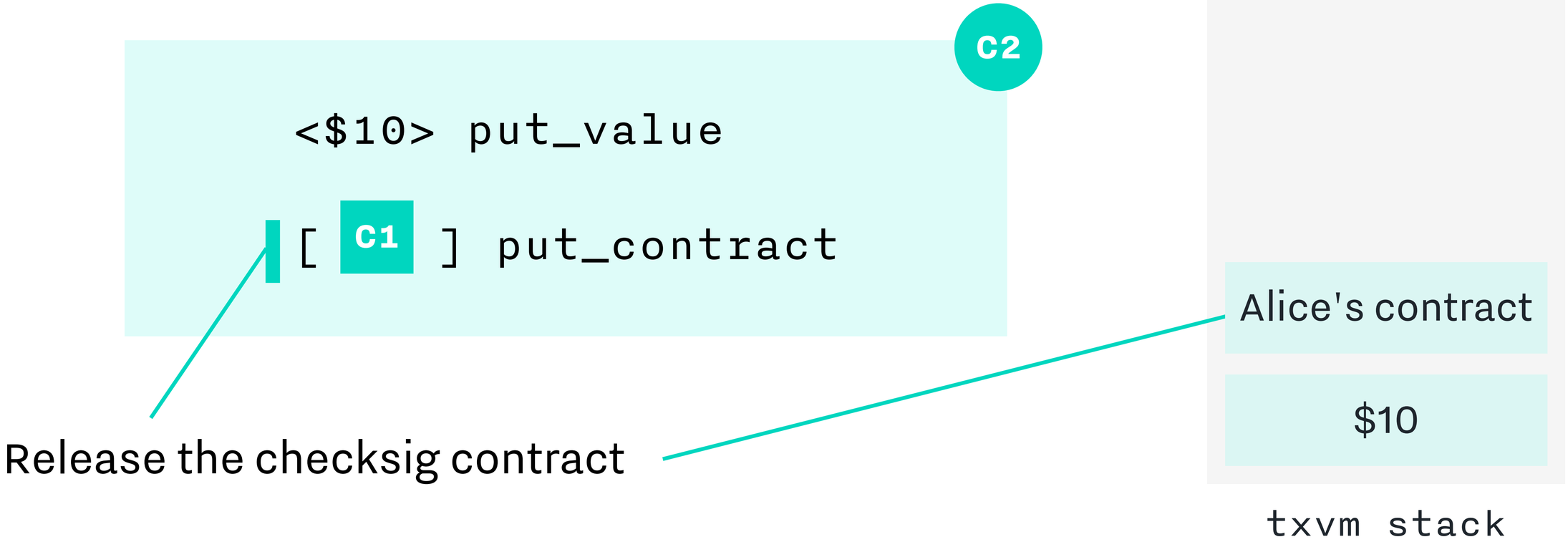
# Unspent output contract

Unlocks the value and defers a checksig contract.



# Unspent output contract

Unlocks the value and defers a checksig contract.



# Diving into contracts

C2

Alice’s $10 `input call`

Alice’s $5 `input call`

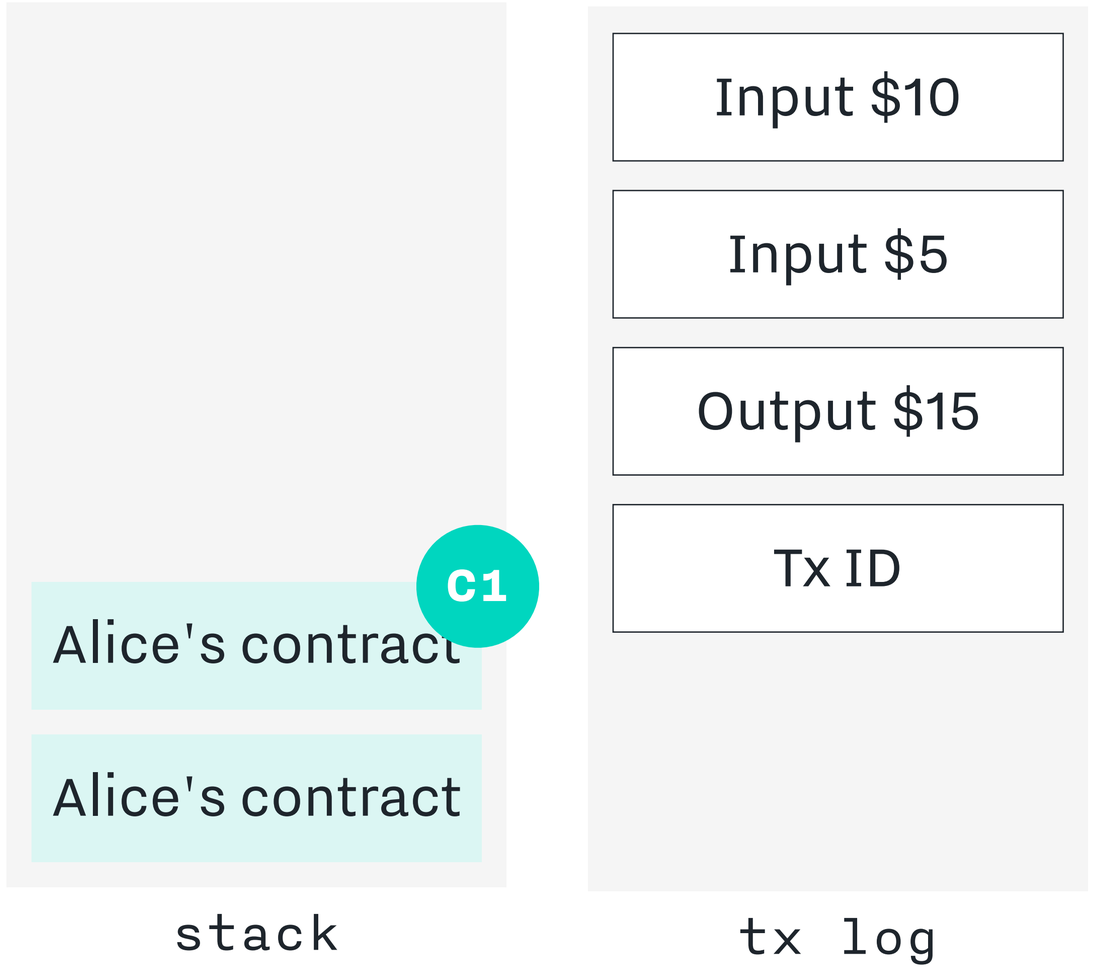
`merge_values`

C3

Carol’s account `contract call`

`finalize`

`<signatures>... call`



# New output contract

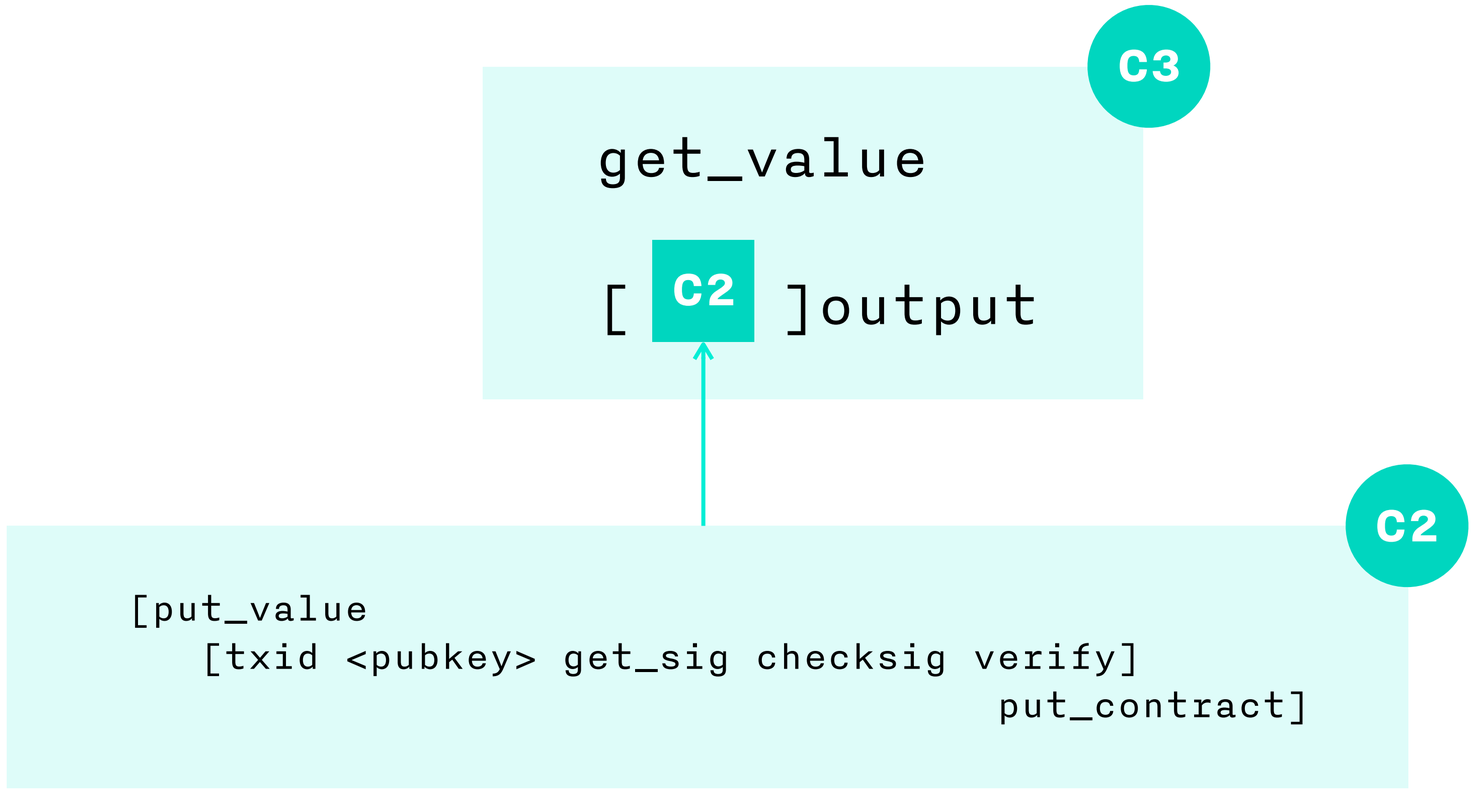
Locks the value and creates an output.

C3

```
get_value
    [put_value
        [txid <pubkey> get_sig checksig verify]
                                        put_contract]
                                                output
```

# New output contract

Locks the value and creates an output.



# New output contract

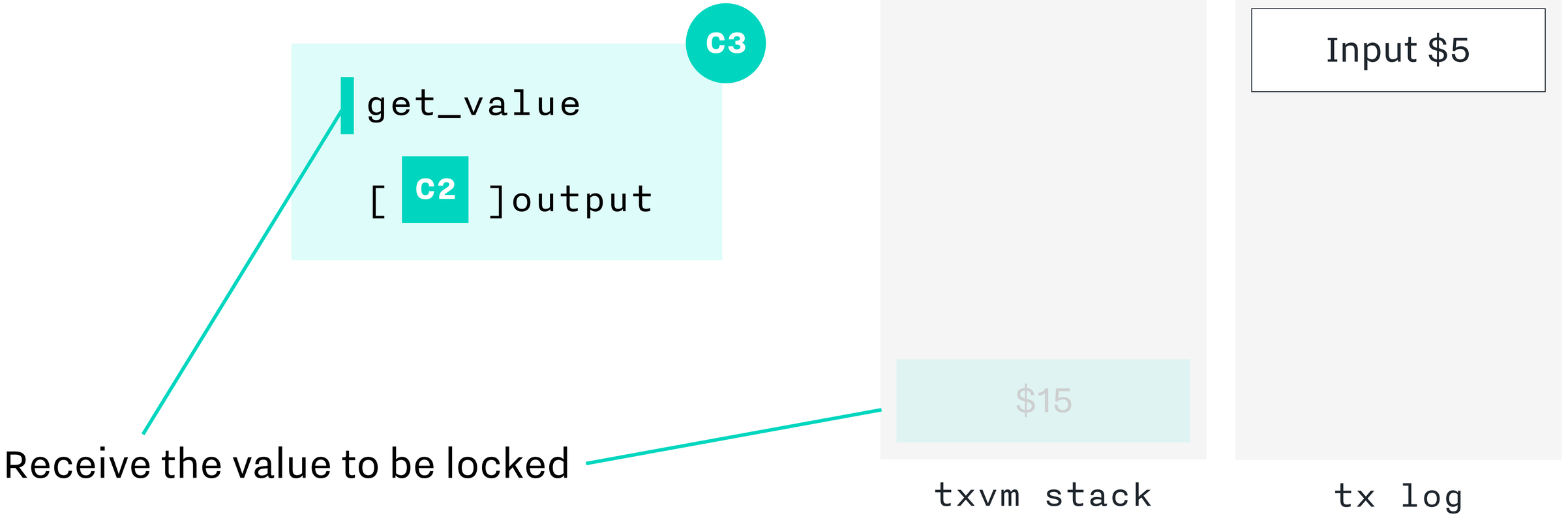
Locks the value and creates an output.

C3

```
get_value
[ C2 ]output
```

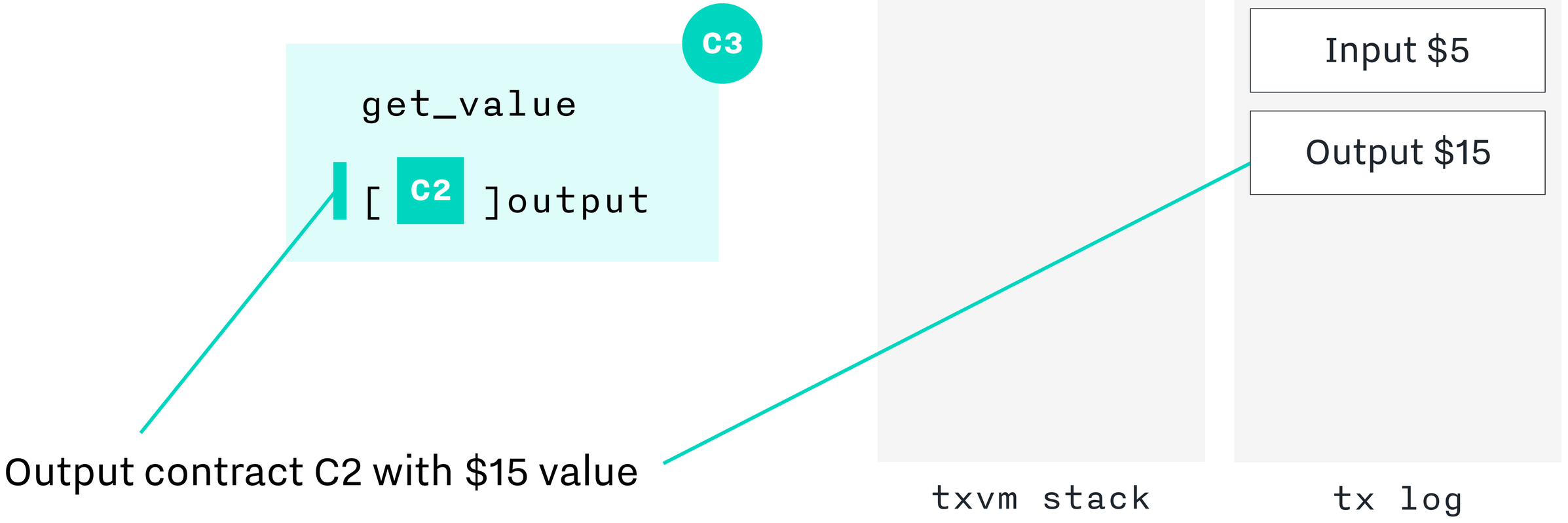
# New output contract

Locks the value and creates an output.



# New output contract

Locks the value and creates an output.



# Diving into contracts

C2

Alice’s $10 `input call`

Alice’s $5 `input call`

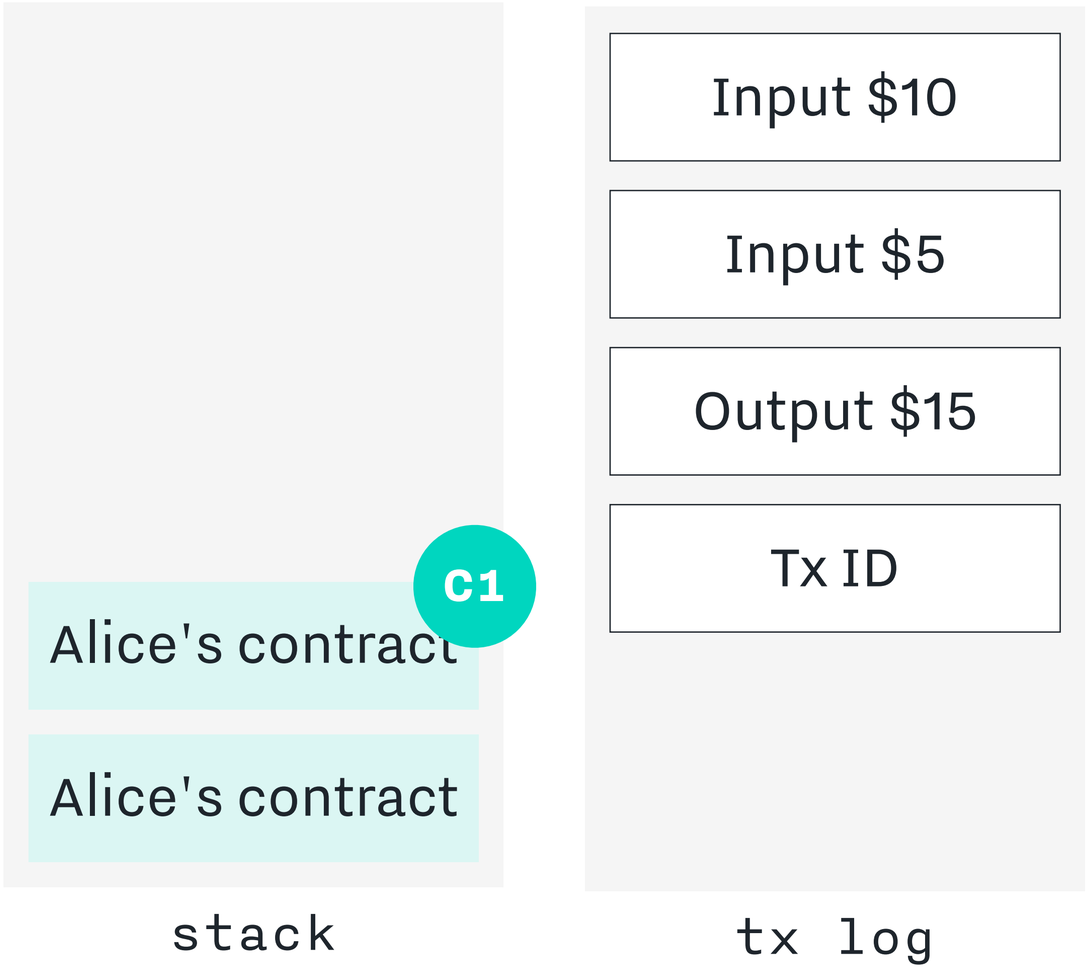
`merge_values`

C3

Carol’s account `contract call`

`finalize`

`<signatures>... call`



# TxVM instruction set

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 (false) | 16 | int | nonce | verify | eq | 1 byte | 17 bytes |
| 1 (true) | 17 | add | merge | jumpif | dup | 2 bytes | 18 bytes |
| 2 | 18 | neg | split | exec | drop | 3 bytes | 19 bytes |
| 3 | 19 | mul | issue | call | peek | 4 bytes | 20 bytes |
| 4 | 20 | div | retire | yield | tuple | 5 bytes | 21 bytes |
| 5 | 21 | mod | amount | wrap | untuple | 6 bytes | 22 bytes |
| 6 | 22 | gt | assetid | input | len | 7 bytes | 23 bytes |
| 7 | 23 | not | anchor | output | field | 8 bytes | 24 bytes |
| 8 | 24 | and | vmhash | contract | encode | 9 bytes | 25 bytes |
| 9 | 25 | or | sha256 | seed | cat | 10 bytes | 26 bytes |
| 10 | 26 | roll | sha3 | self | slice | 11 bytes | 27 bytes |
| 11 | 27 | bury | checksig | caller | bitnot | 12 bytes | 28 bytes |
| 12 | 28 | reverse | log | contractprog. | bitand | 13 bytes | 29 bytes |
| 13 | 29 | get | peeklog | timerange | bitor | 14 bytes | 30 bytes |
| 14 | 30 | put | txid | prv | bitxor | 15 bytes | 31 bytes |
| 15 | 31 | depth | finalize | ext | 0 bytes | 16 bytes | 32 bytes |

| | |
|---|---|
| 0x00-0x1f | smallints |
| 0x20-0x29 | ints |
| 0x2a-0x2f | stack |
| 0x30-0x37 | values |
| 0x38-0x3b | crypto |
| 0x3c-0x3f | tx |
| 0x40-0x4d | control flow |
| 0x4e-0x4f | extension |
| 0x50-0x5e | data |
| 0x5f+ | pushdata |

# Conclusion

- TxVM is a **new model** for blockchain transactions.
- Borrows **good parts** from both Bitcoin and Ethereum.
- Powerful abstractions: first-class **values** and **contracts**.
- Safe and straightforward **language** for writing smart contracts.
- **Open source & whitepaper** soon.

Chain

Cathie Yun
cathie@chain.com

Bob Glickstein
bob@chain.com

Dan Robinson
@danrobinson

Oleg Andreev
@oleganza