TÉCNICO LISBOA

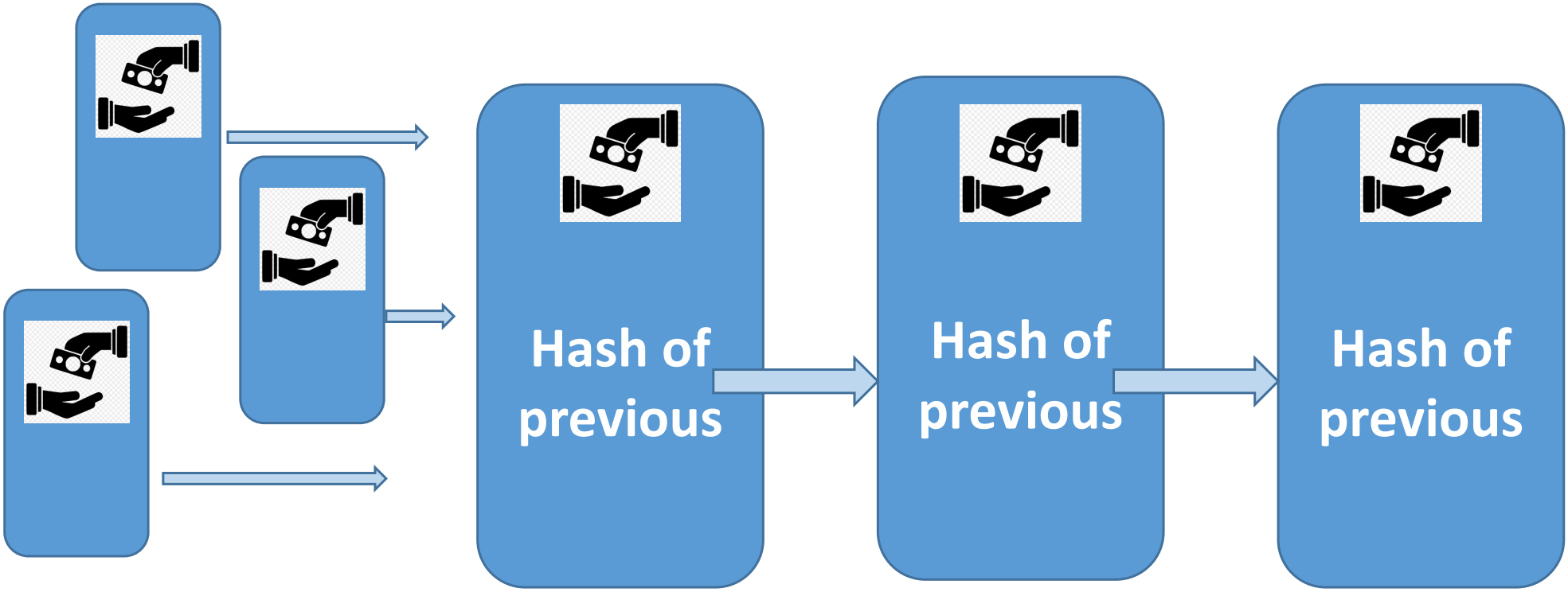
Ethereum Virtual Machine

Highly dependable systems – 2022/23

Lecture 8

Lecturers: Miguel Matos and Rodrigo Miragaia Rodrigues

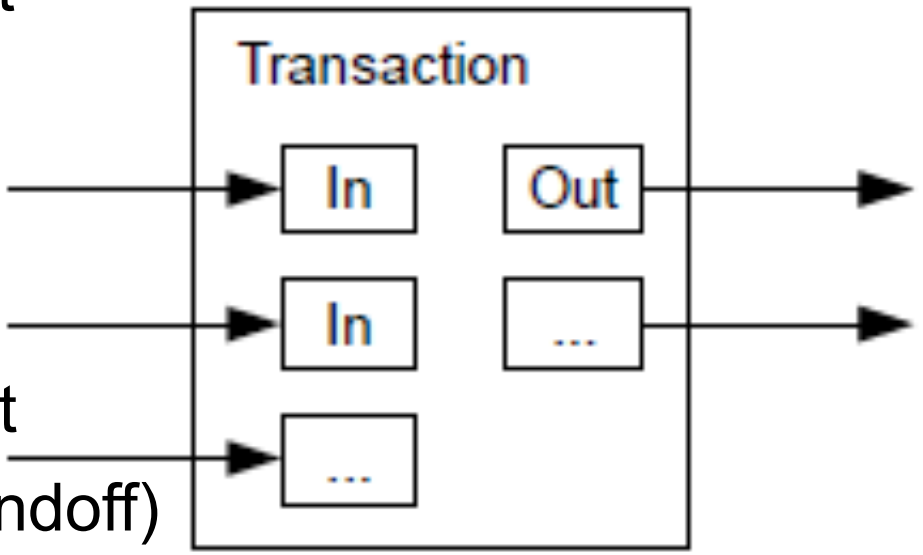
Last lecture: Bitcoin



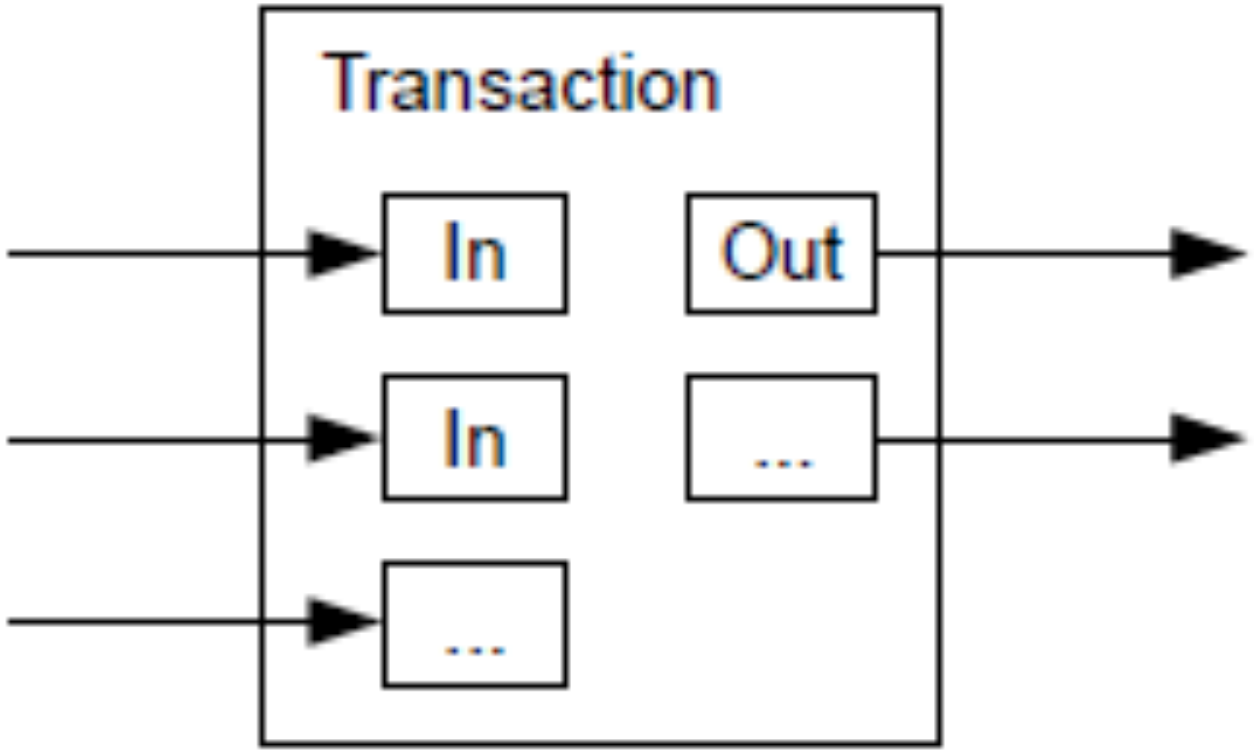
- PoW → miners compete to append next record
- In some cases may lead to fork → longest chain rule
- Shortcomings: studied several attacks + energy waste

Each block in the blockchain is a set of transactions

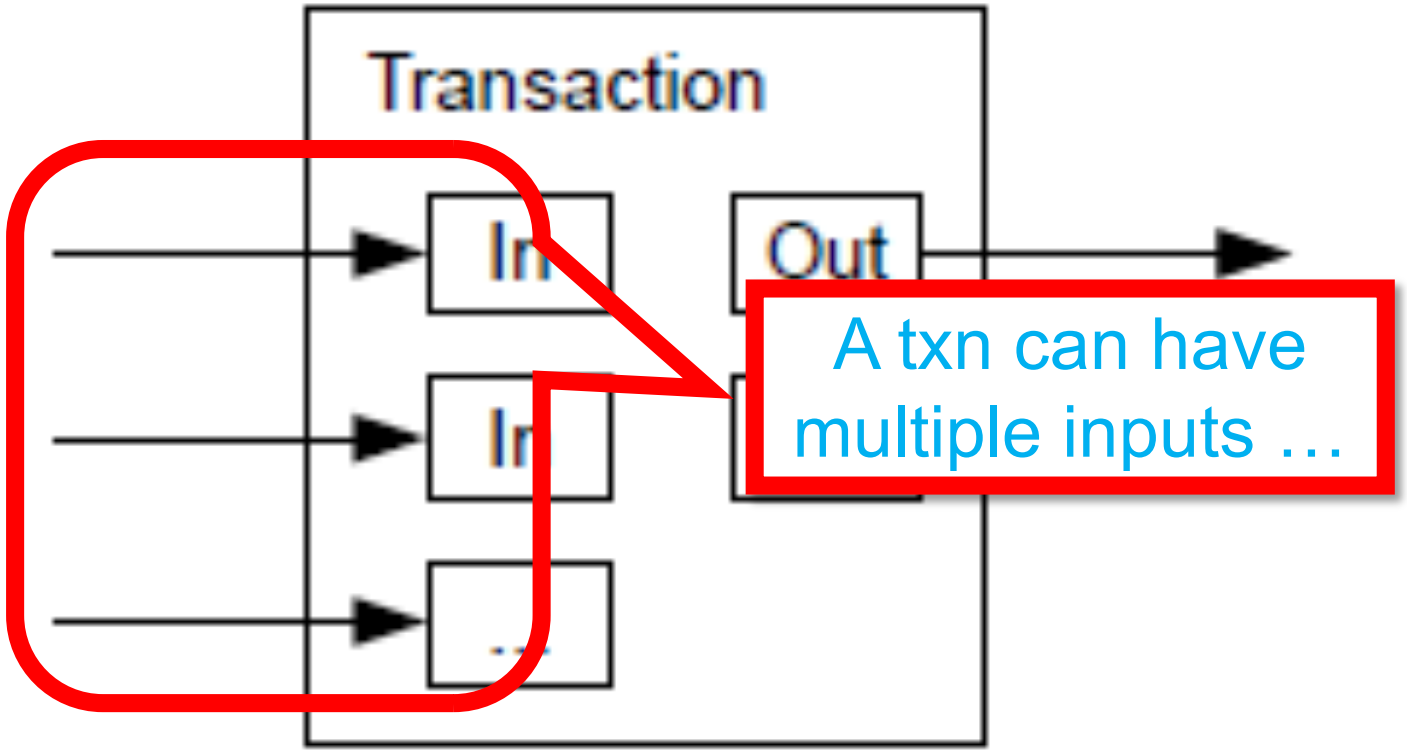
- Transactions can have multiple inputs and <=2 outputs
- Each input describes debit
  - Sender → signs handoff of amount (signed with private key)
- Each output describes credit
  - Recipient → public key of entity (corresponding to private key that later can sign the subsequent handoff)



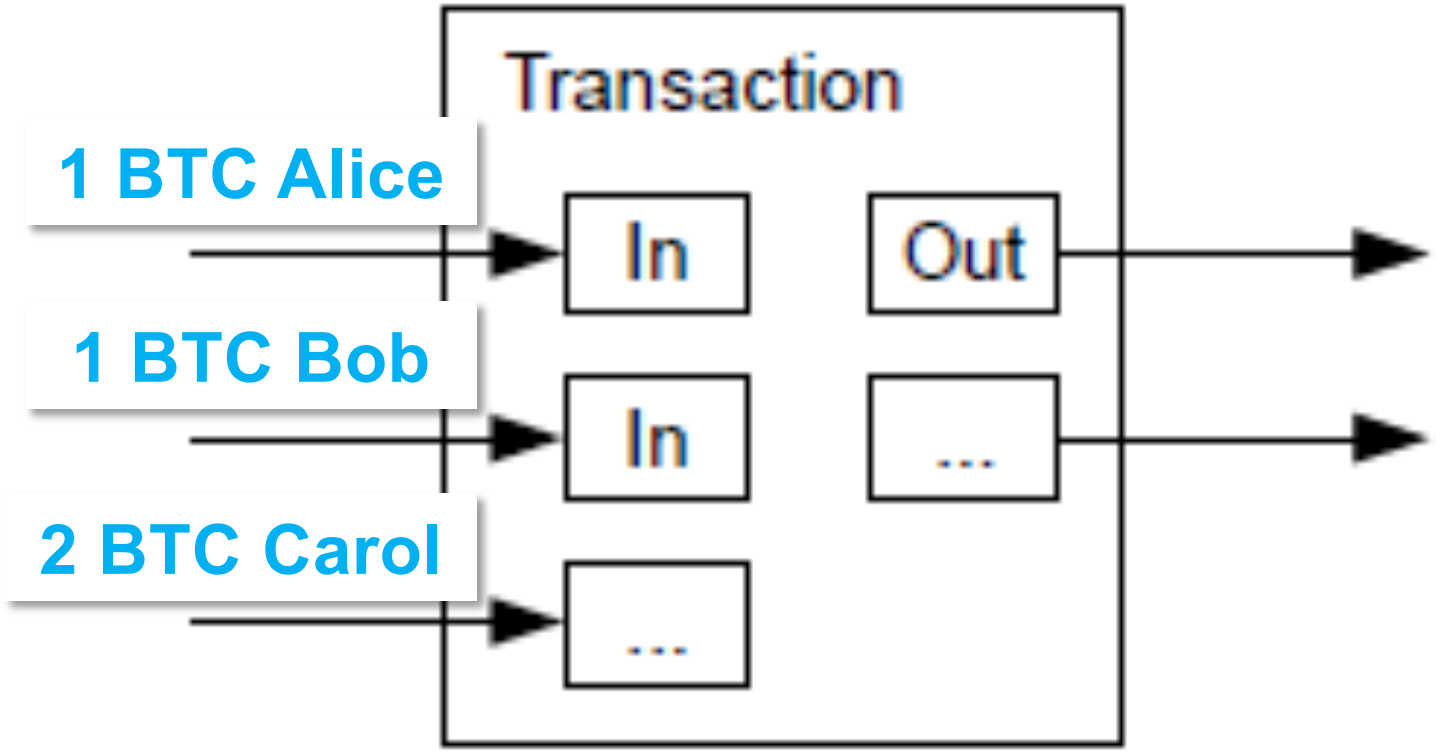
Bitcoin UTXO Model



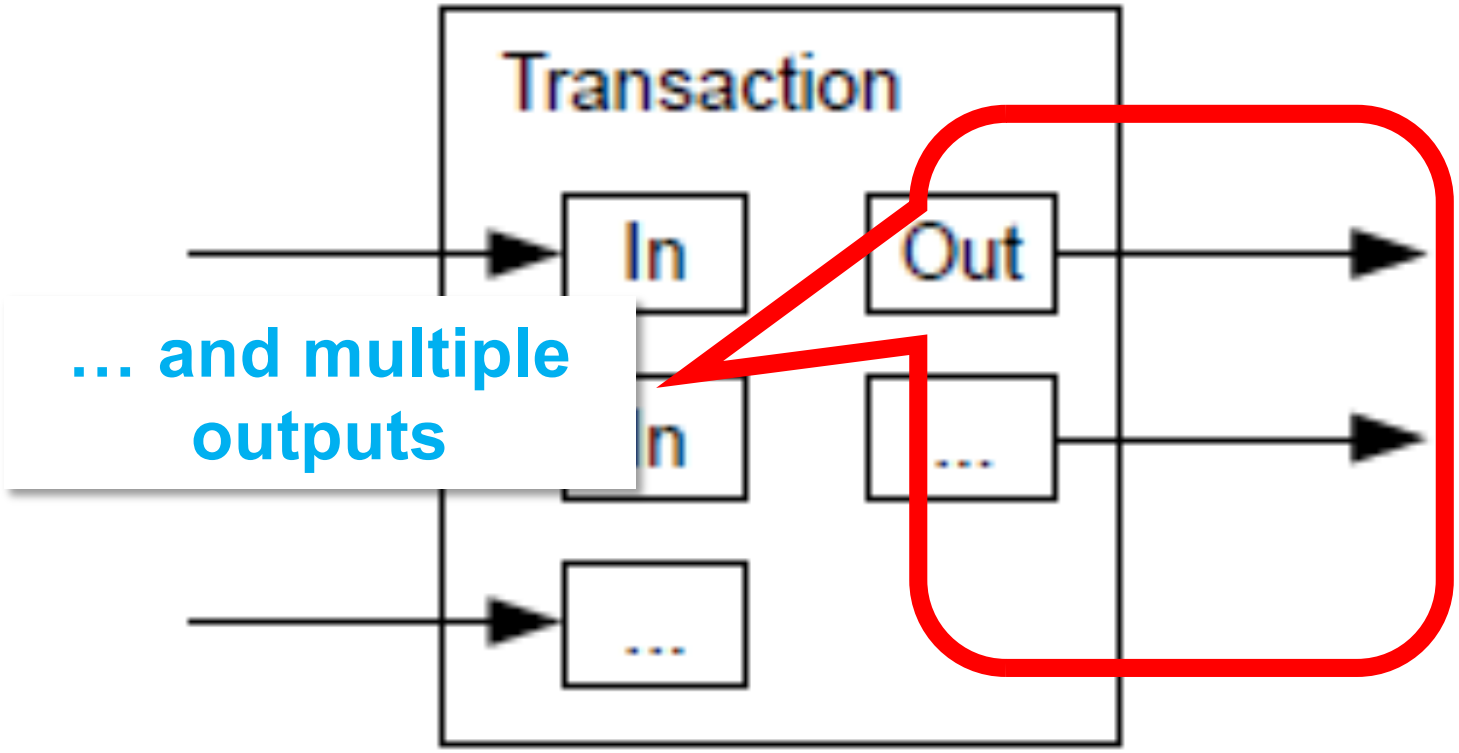
Bitcoin UTXO Model



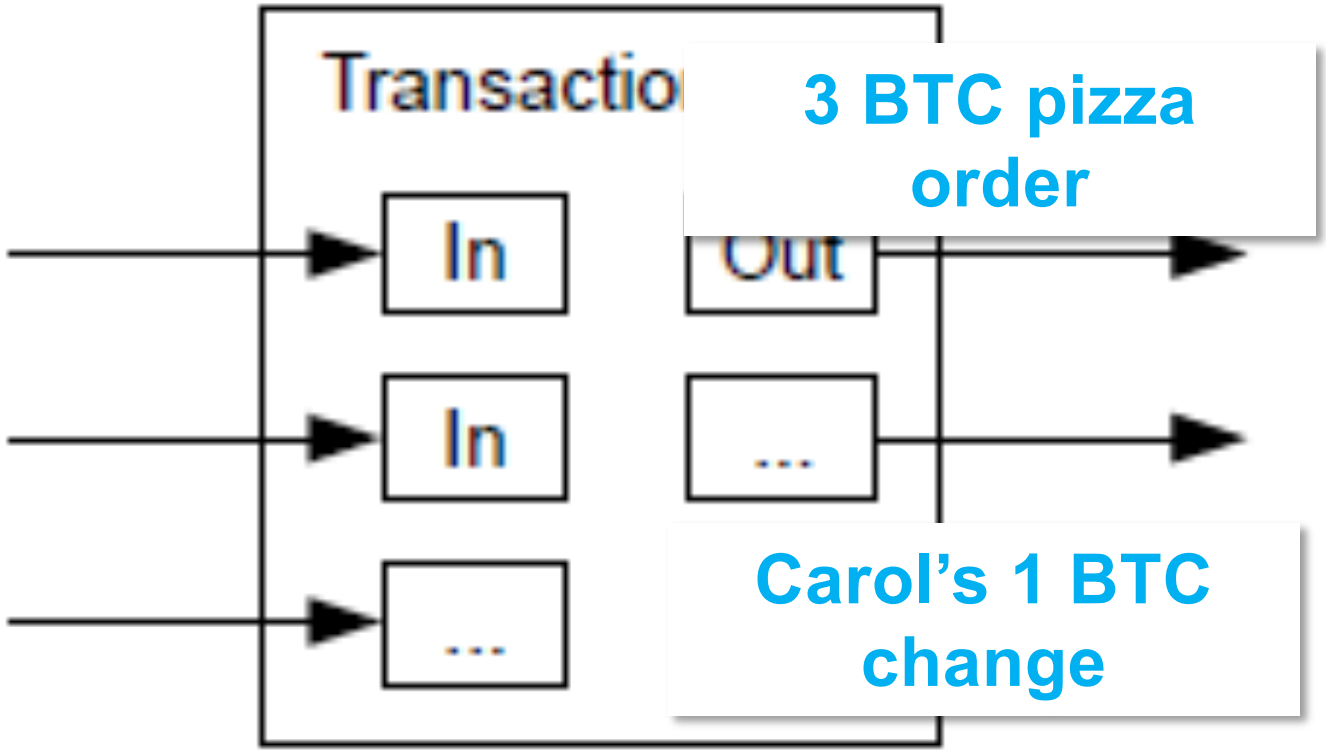
Bitcoin UTXO Model



Bitcoin UTXO Model

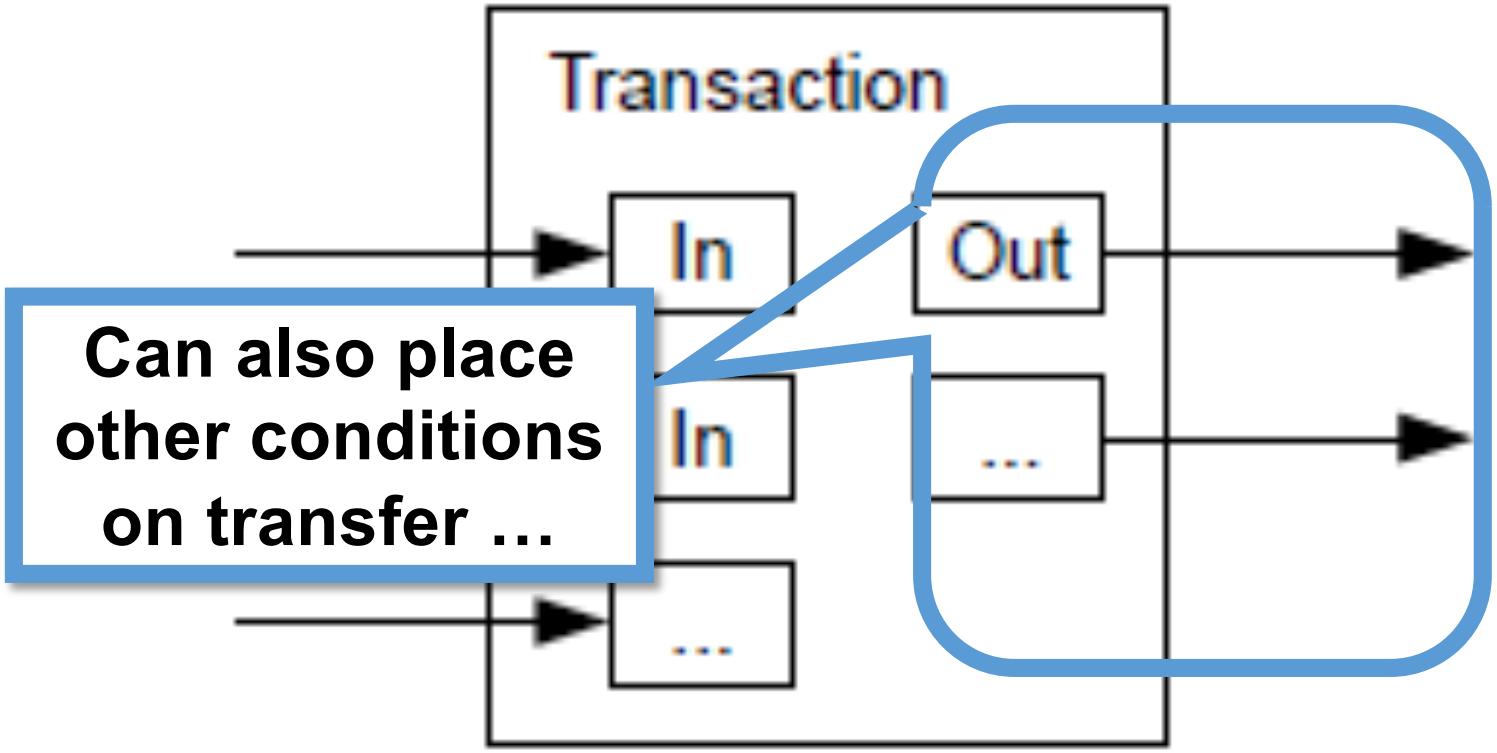


Bitcoin UTXO Model



Bitcoin UTXO Model

"smart contracts" (bitcoin → simple script with conditions for spending UTXOs)



State maintained by bitcoin nodes

- Blockchain stores all transaction since beginning of bitcoin (>500 GB of information, increases as more txns occur)
- Enables keeping track of outputs that haven't been spent yet (about 4 GB of information, increases as more users join)
  - Called "unspent transaction output" (UTXO)
  - "Soft state" → can be reconstructed and verified from blockchain
  - Alternative would be to store account balances (has pros and cons)
- New transactions remove UTXOs from this set, and create new ones that are added to the set

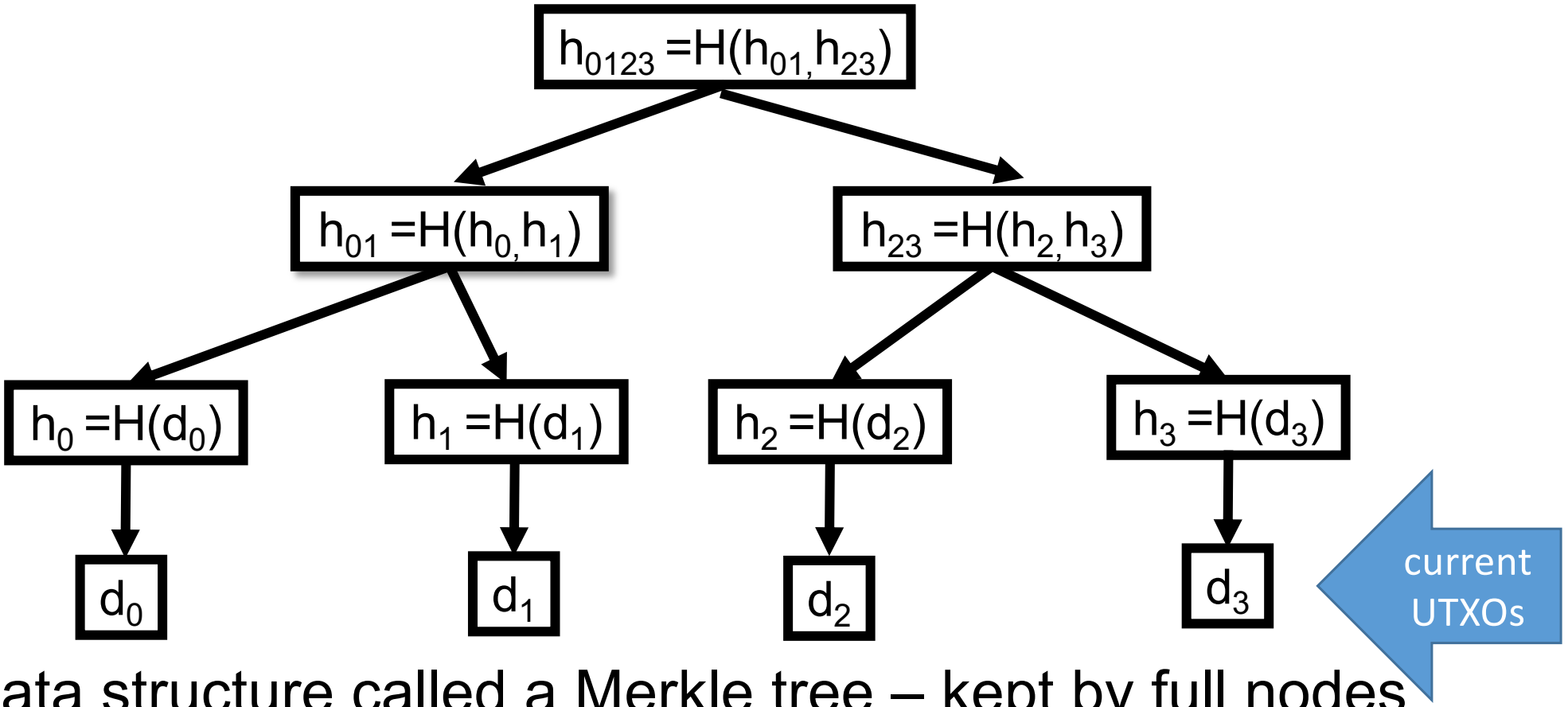
Bitcoin scalability and Utreexo

- Resources required for running a full node are a problem (barrier to entering the system)
- Initial block download can take 6 hours (SSD) to 1 week (HDD), but this is only done once per node
- Continuously maintaining a (growing) set of all UTXOs is more of a concern
- Initially there were light clients → do not store state nor validate transactions
  - At the cost of weaker security properties than full nodes
- Led to the proposal of Utreexo
  - Make it easier to run full nodes
  - New set of data structures, and new type of node (compact state node)

Compact state nodes using Utreexo

- Observation: most UTXOs aren't needed for years (from creation until being spent)
  - No need to store all the UTXOs
- Store only a summary (in crypto terms, an accumulator)
- Require transaction issuer to send additional information (proofs) to verify transactions
- Drastically reduces storage requirements, trading for higher bandwidth (mostly) and CPU (not much)

How to compute a summary of UTXOs?
(simplified)



- Data structure called a Merkle tree – kept by full nodes
- Compact nodes **only store root of the tree**

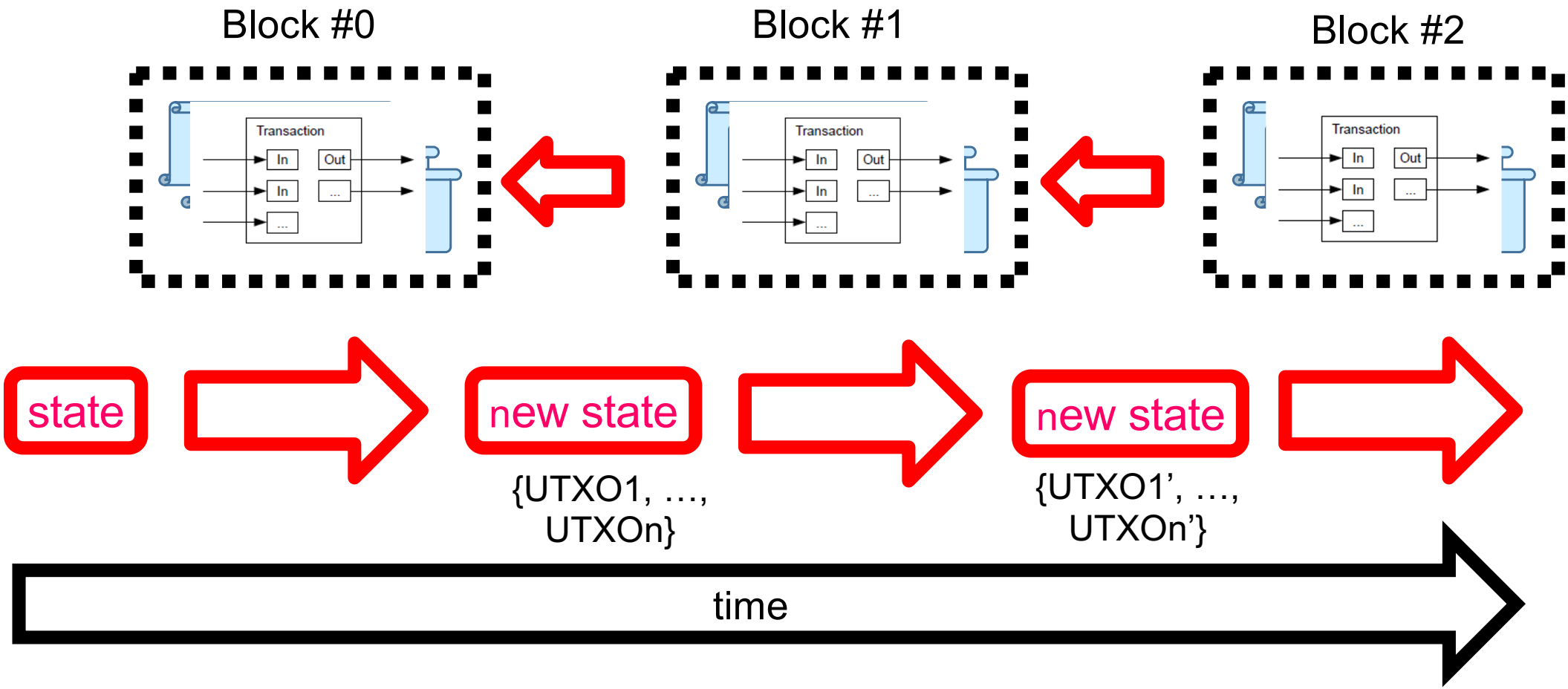
Upon receiving a UTXO, how to check it is part of the current set?

- Transaction issuer must send proof
- E.g., proof that d_2 is in the set of UTXOs?
- "d_2, h_3, h_{01}" → This must requested from a full node
- From this info, receiver computes h_2, h_{23}, h_{0123}
- Then checks whether h_{0123} matches the currently held root of the tree
- (Tree maintenance after adds and deletes + supporting legacy transactions gets a bit more intricate – see paper for details)

Bitcoin as a state machine

- State is current set of UTXOs
  - (when a node boots, this is derived from all transactions since inception)
- Transactions cause the following state transition:
  - Remove all UTXOs in “In” from set
  - Add all UTXOs in “Out” to the set

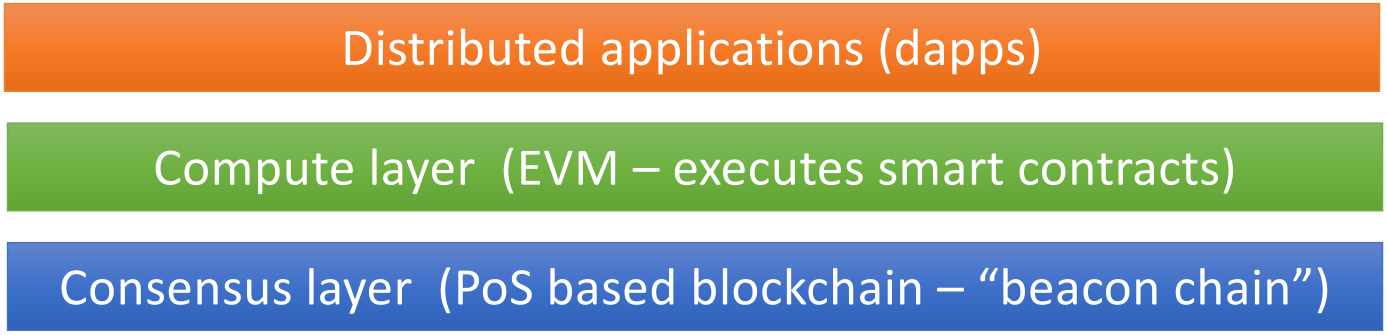
Bitcoin’s state machine



Today: Ethereum

- Ethereum is the second largest cryptocurrency after Bitcoin
- Very generic platform, supporting myriad of distributed applications (dapps), with several currencies, NFTs, DAOs, etc.
- Introduced broad support for “smart contracts”
  - Programs that run in the blockchain
  - Ability to read and modify the blockchain state
  - Turing complete
- Adversarial environment without centralized trust
  - Guaranteeing safety is key

Ethereum's layered design



Ethereum's PoS blockchain

- Moved from PoW to PoS in December 2020 ("beacon chain")
- One block every 12 seconds (100-200 txns/block)
- Participants (validators) must stake 32 ETH (~3500 EUR today)
  - Large amount, some systems allow for pooling
- Block proposer is chosen at random* among validators
  - Broadcast new block; collects transaction fees
- Remaining validators check correctness and sign correct blocks
  - Block is finalized after a certain threshold of signatures
  - Validators that behave correctly → rewarded (every 32 blocks)
  - Behave incorrectly → part of stake is slashed

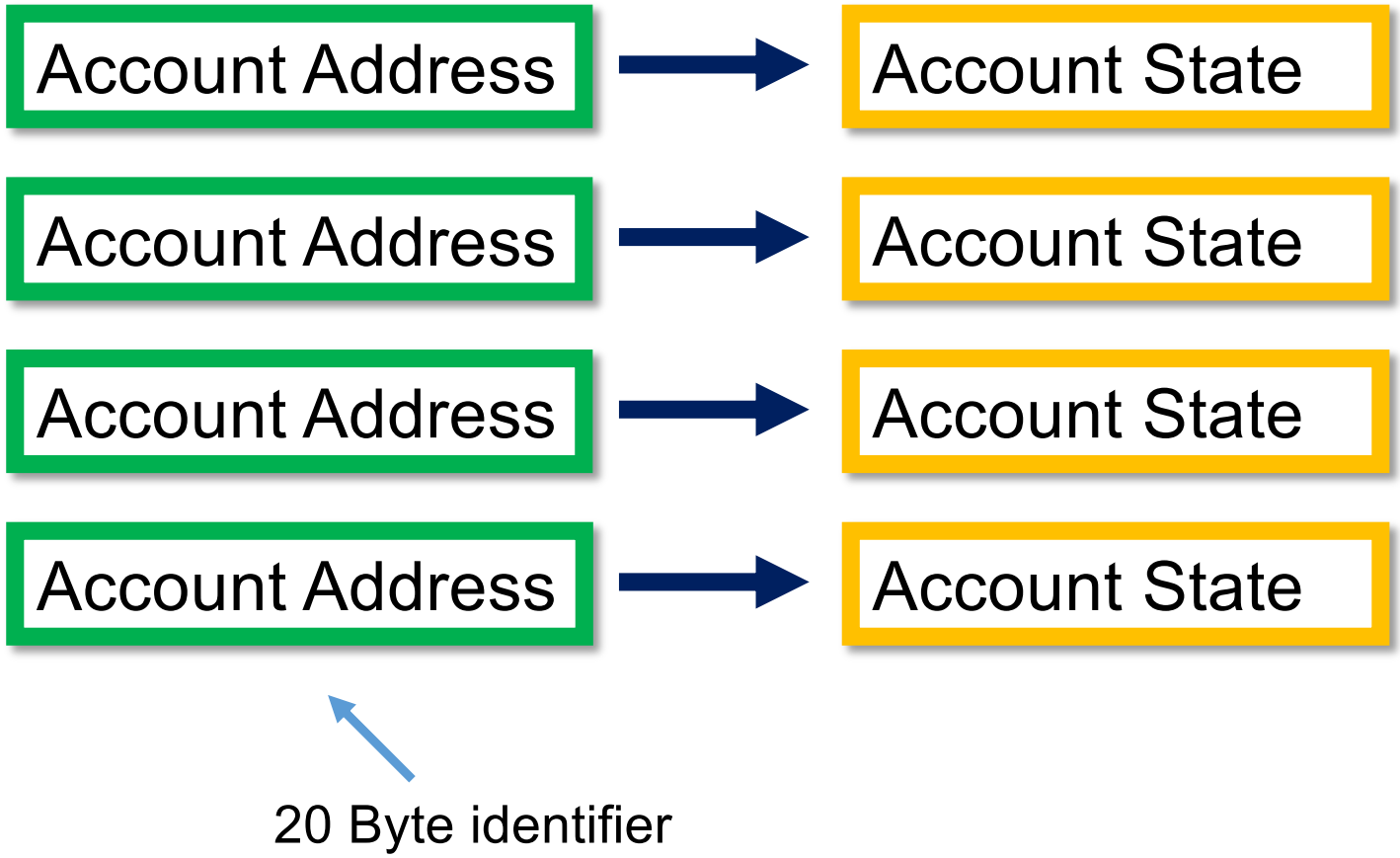
Epoch

* probability is proportional to stake size

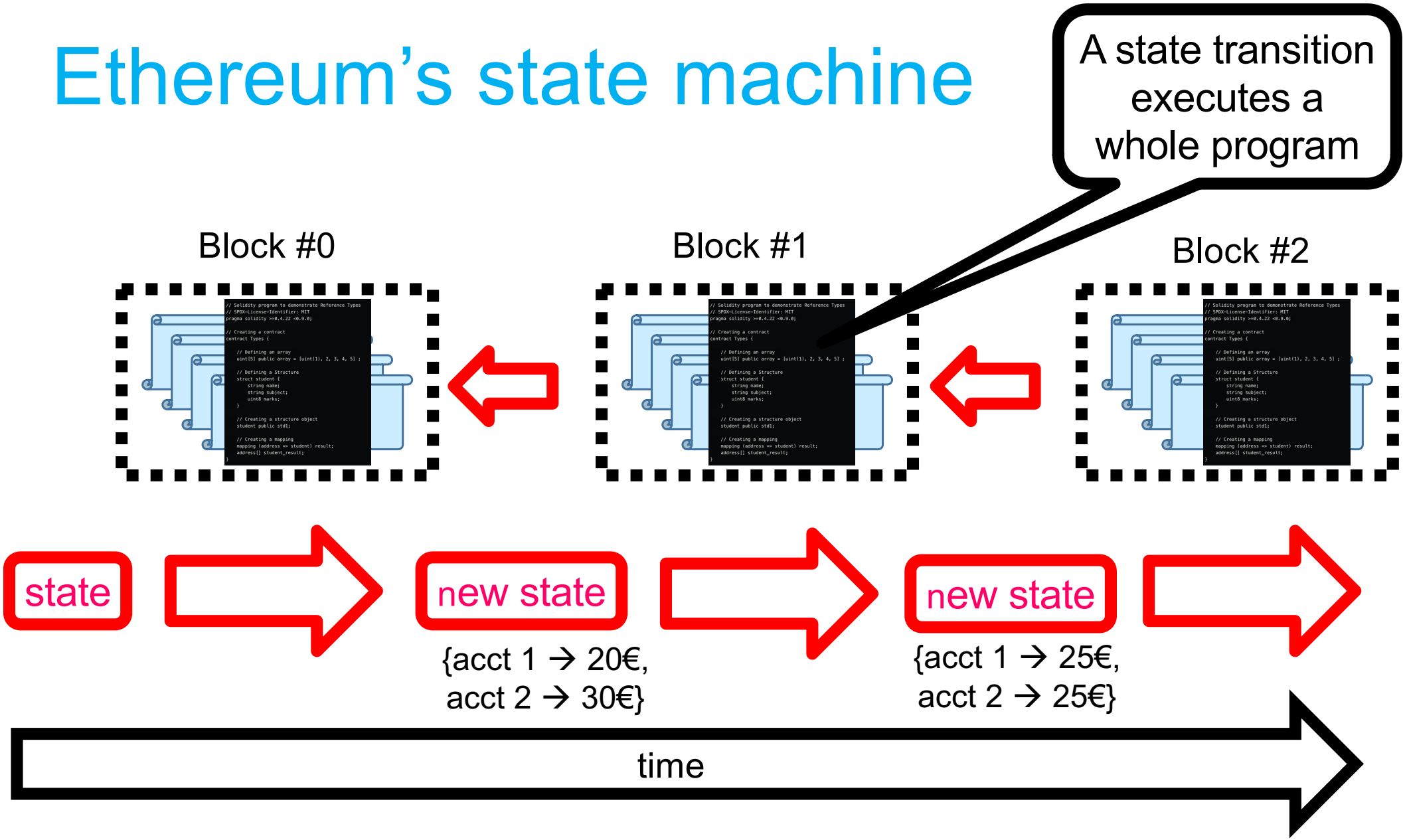
Ethereum has an account model

- Instead of UTXOs, the state of Ethereum is a mapping from a set of accounts to their respective state
- At a basic level, the state is the account balance (but it gets more sophisticated)

Ethereum State



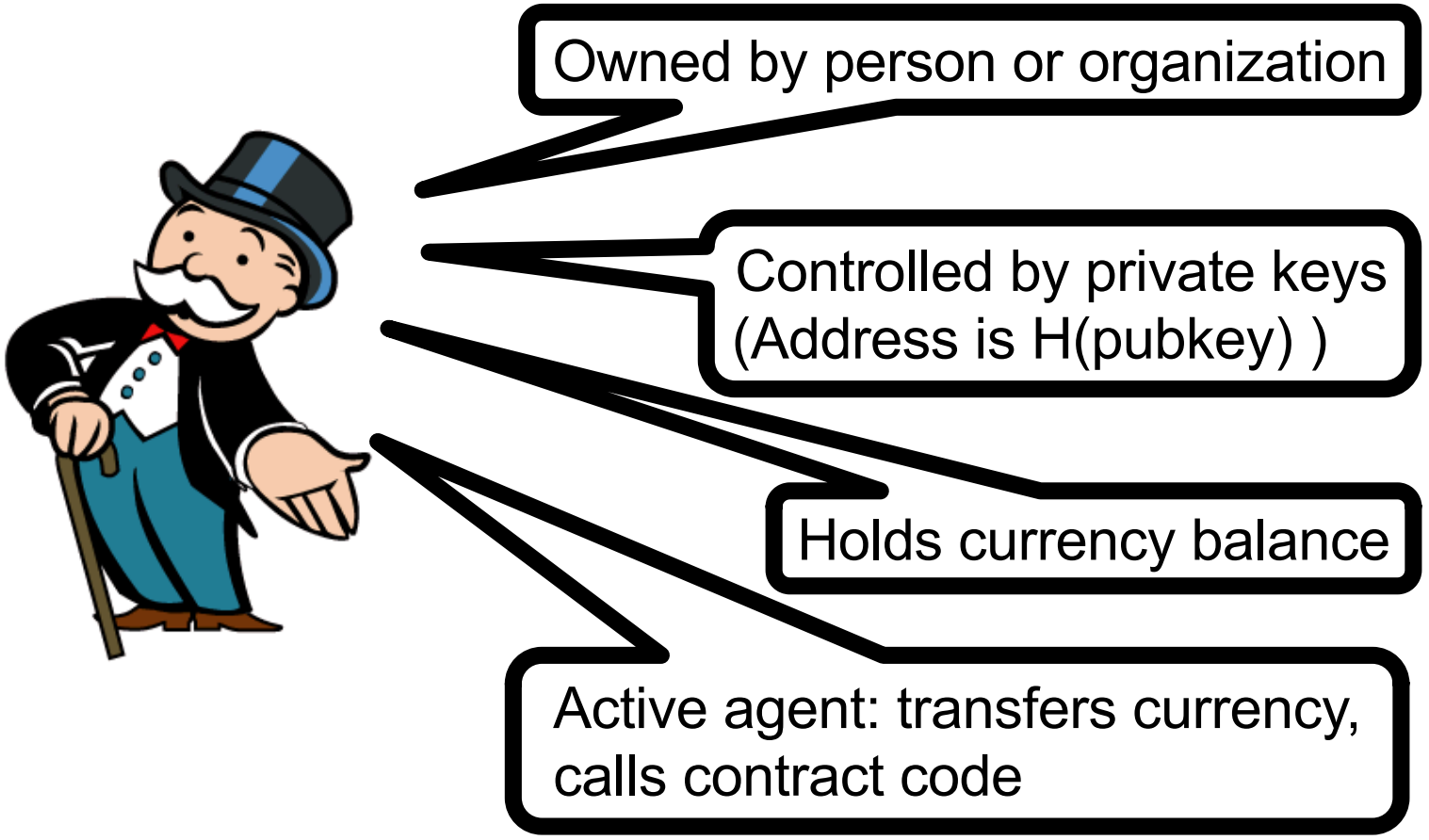
Ethereum's state machine



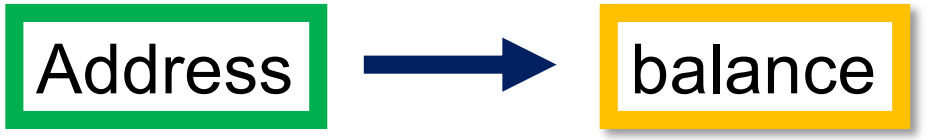
Two types of accounts in Ethereum

- External (owned accounts)
- Contract

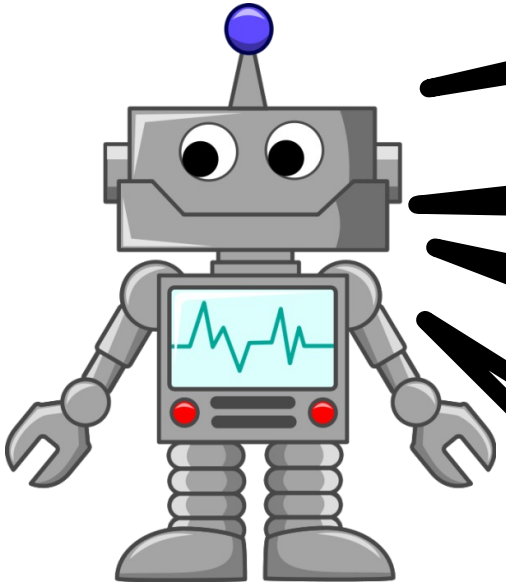
External Account (“owned”)



External Account (“owned”)



Contract Account



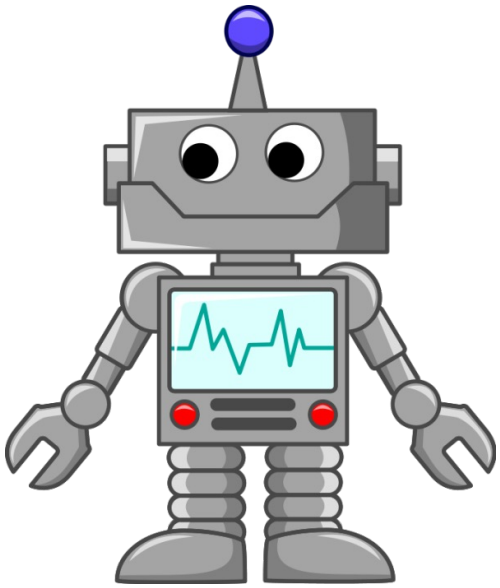
Holds currency balance

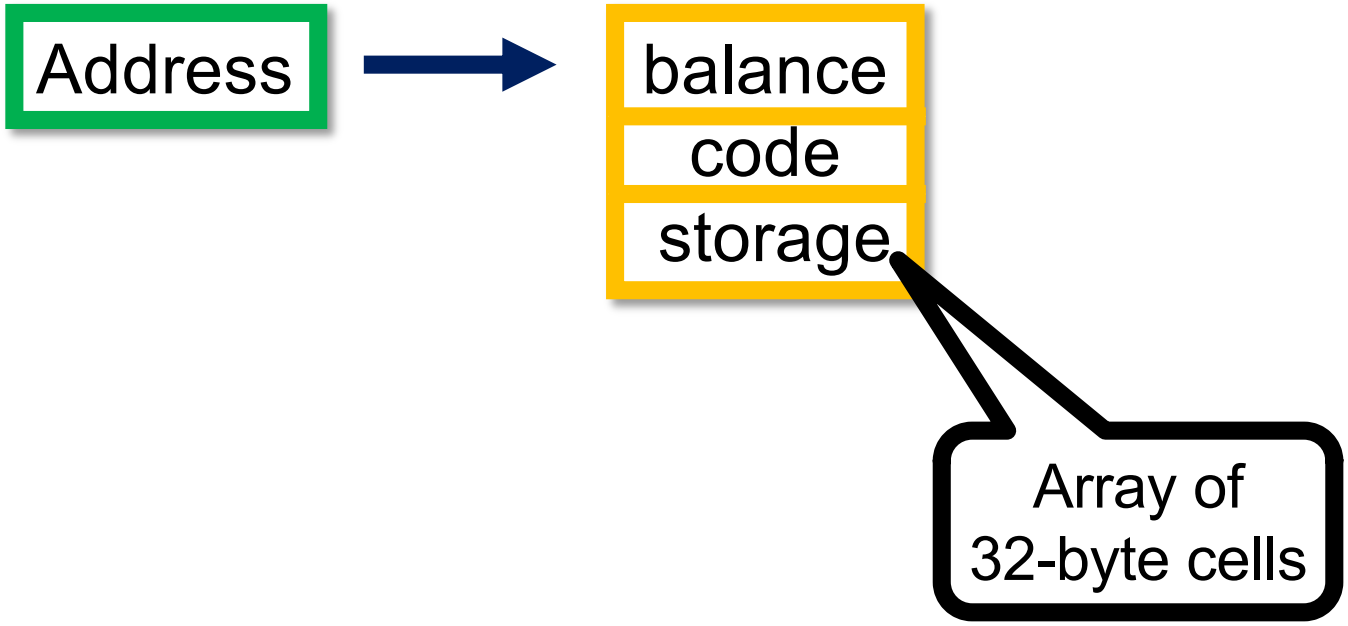
Associated with code
(set at creation time)

Passive Agent:
code called by other accounts

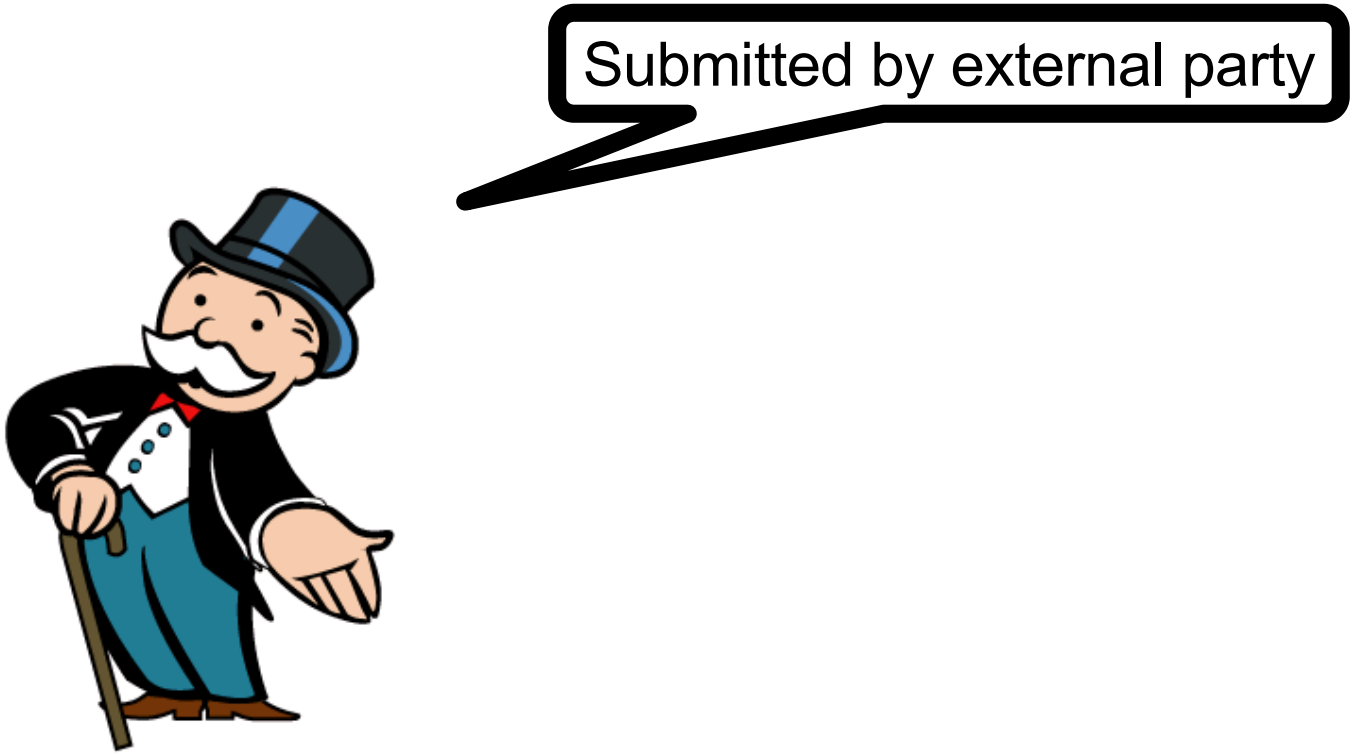
code can transfer money,
call other contracts

Contract Account



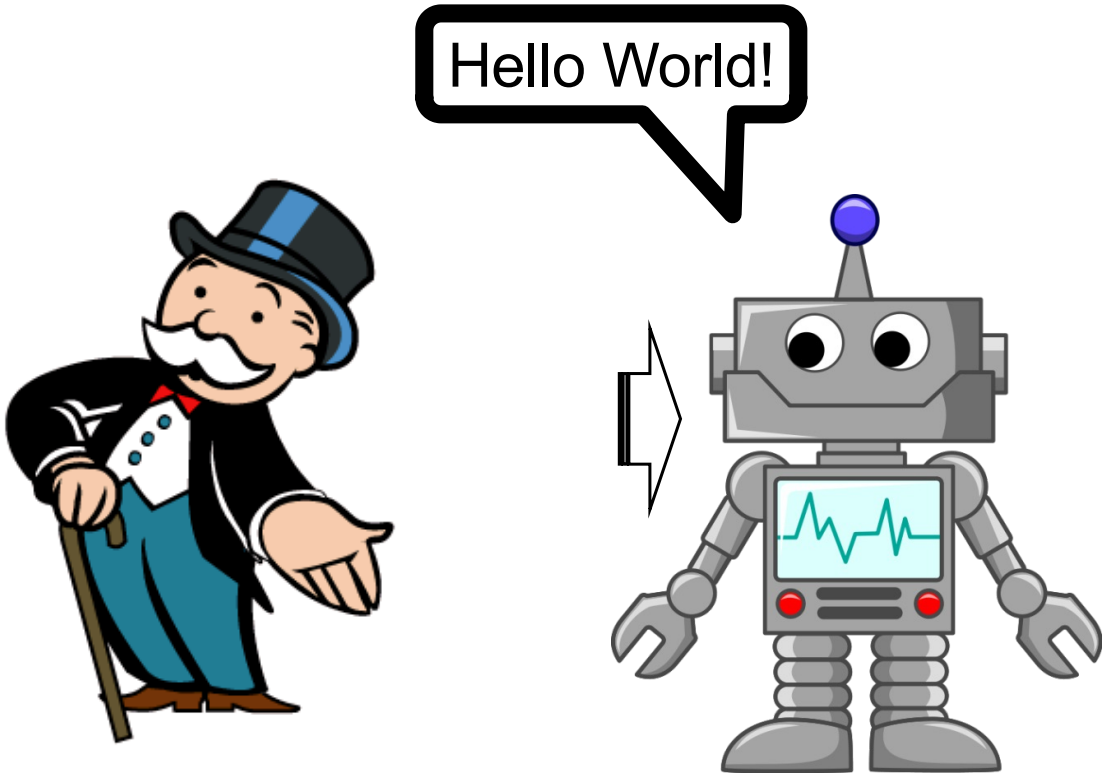


Transaction Creation

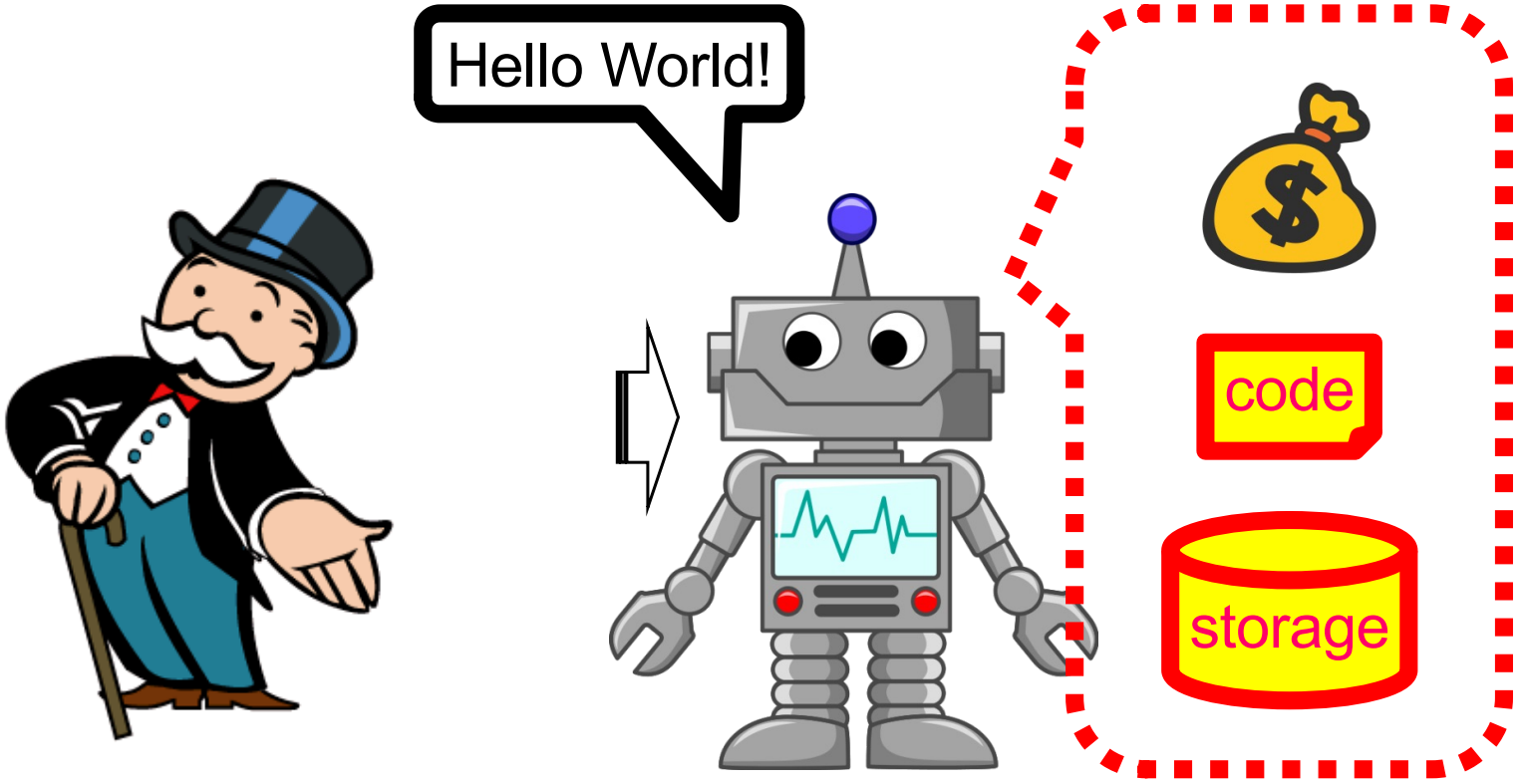


Contract Creation Transaction

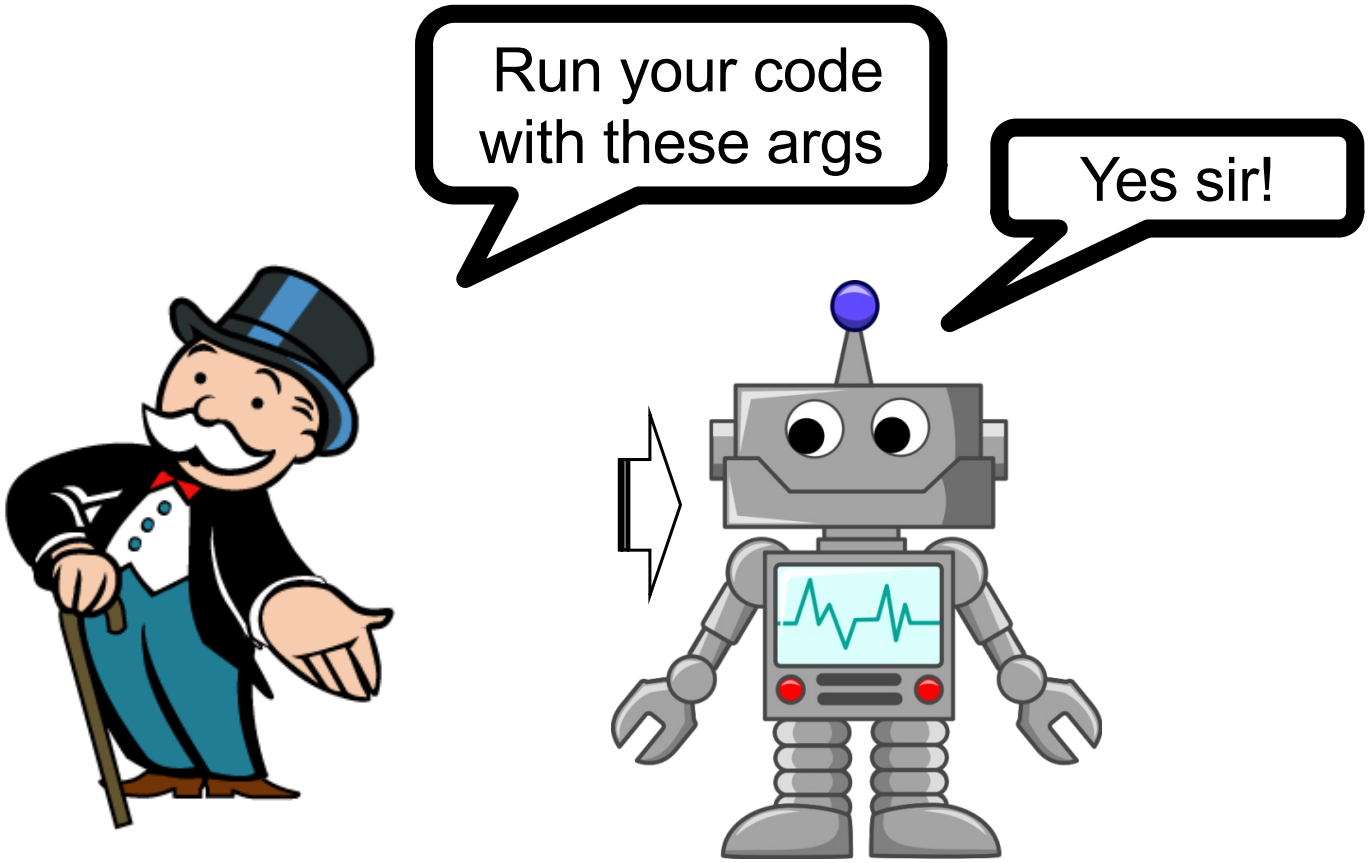
Contract Creation Transaction



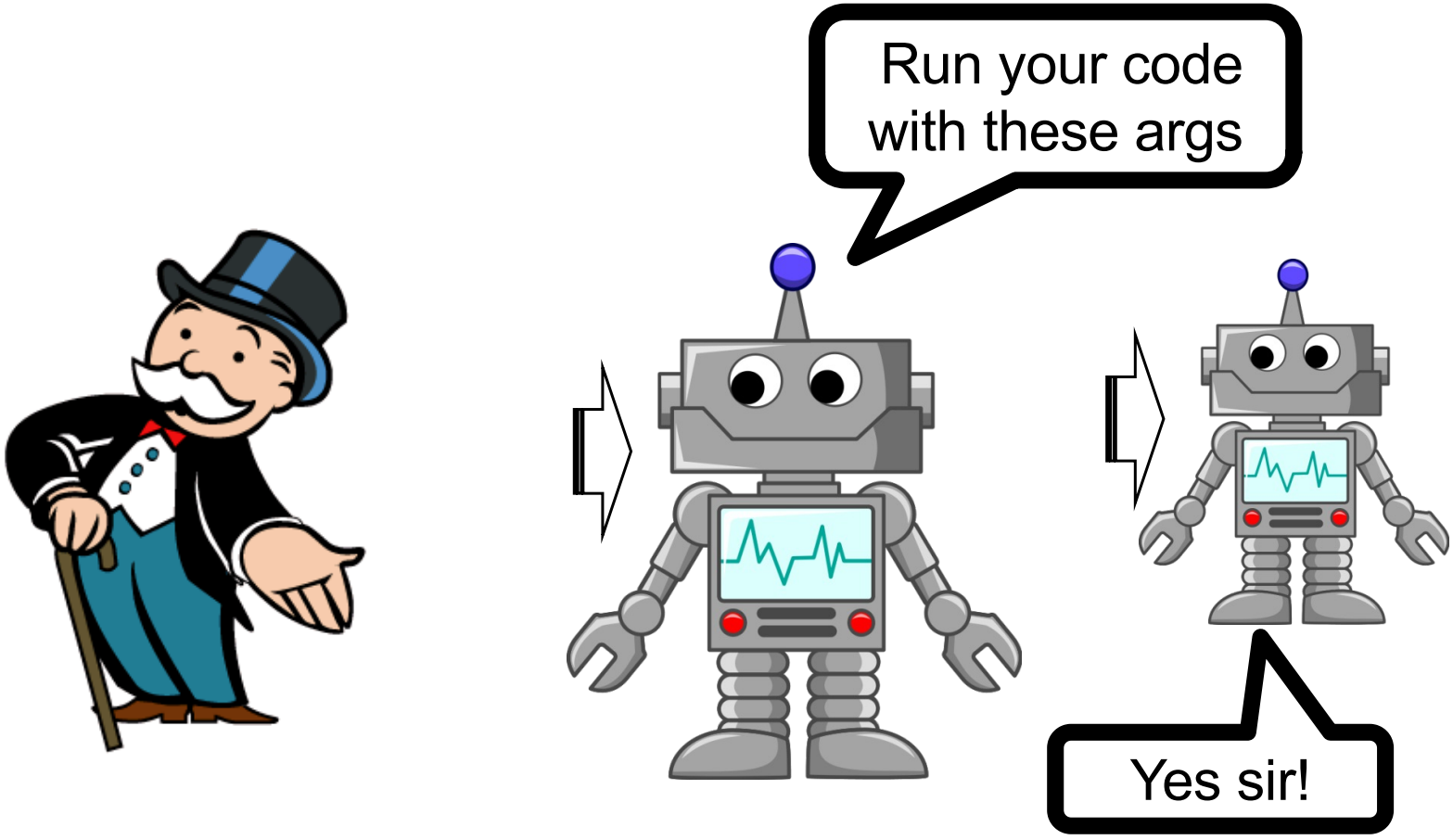
Contract Creation Transaction



Message Call Transaction

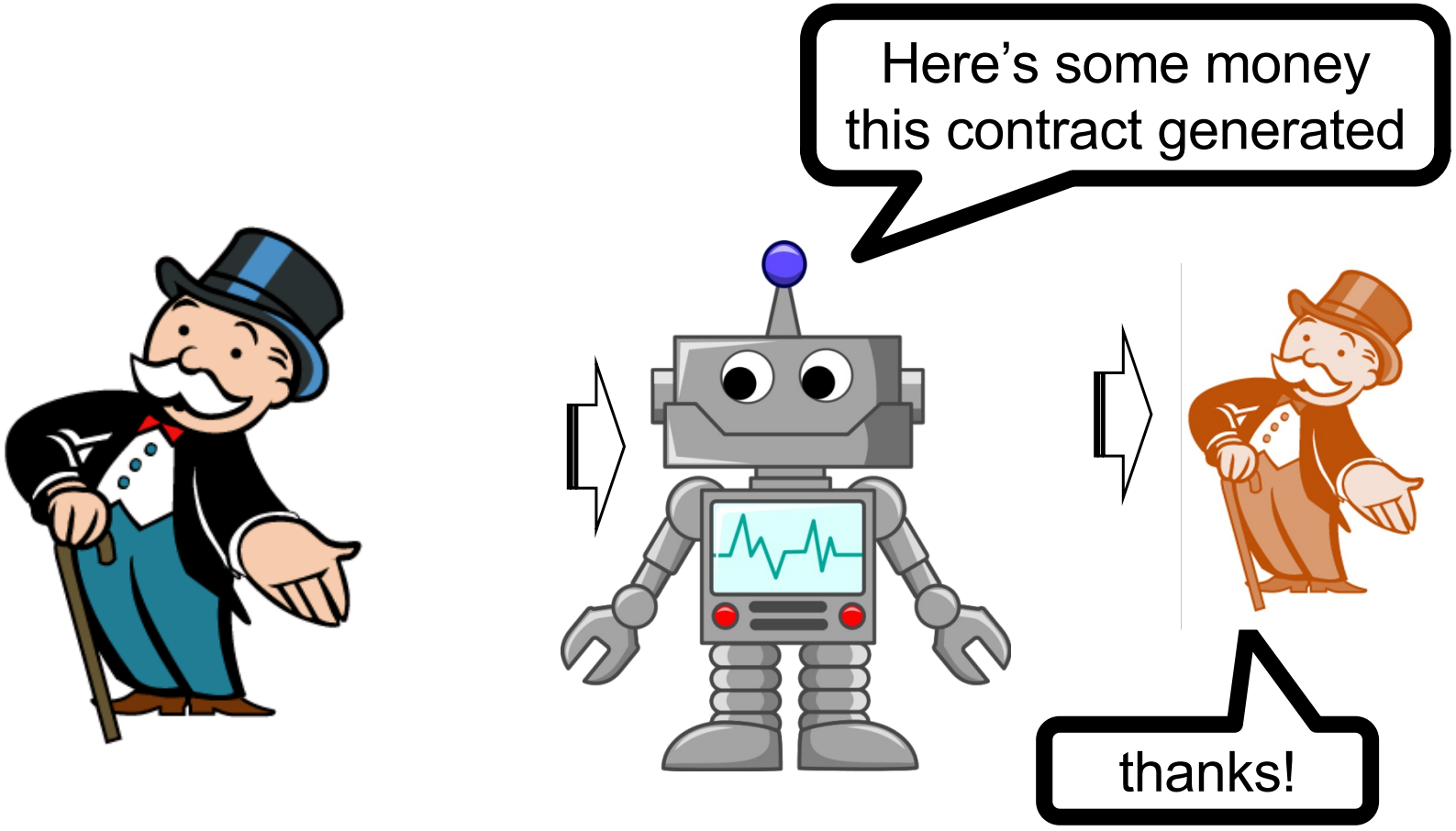


Message Call Transaction



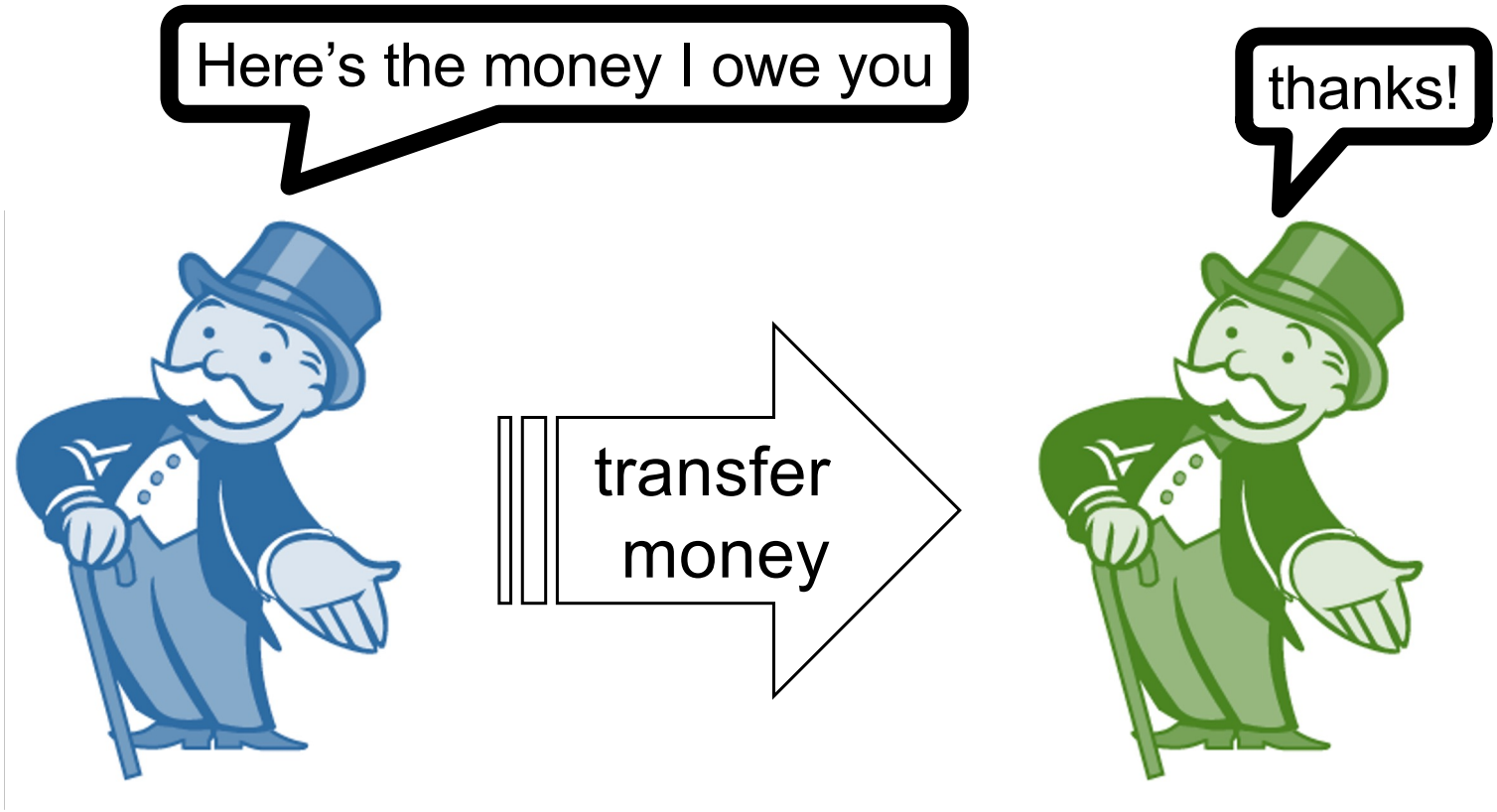
Contracts can call other contracts

Message Call Transaction

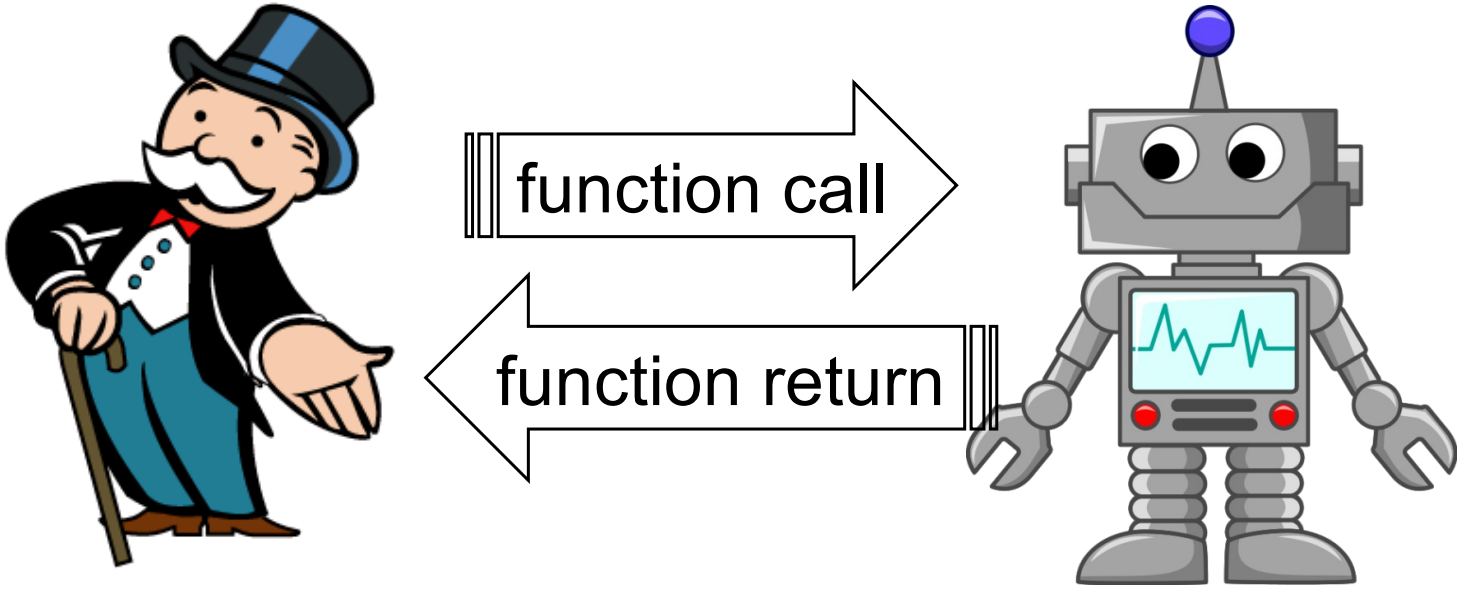


Contracts can also send funds to users

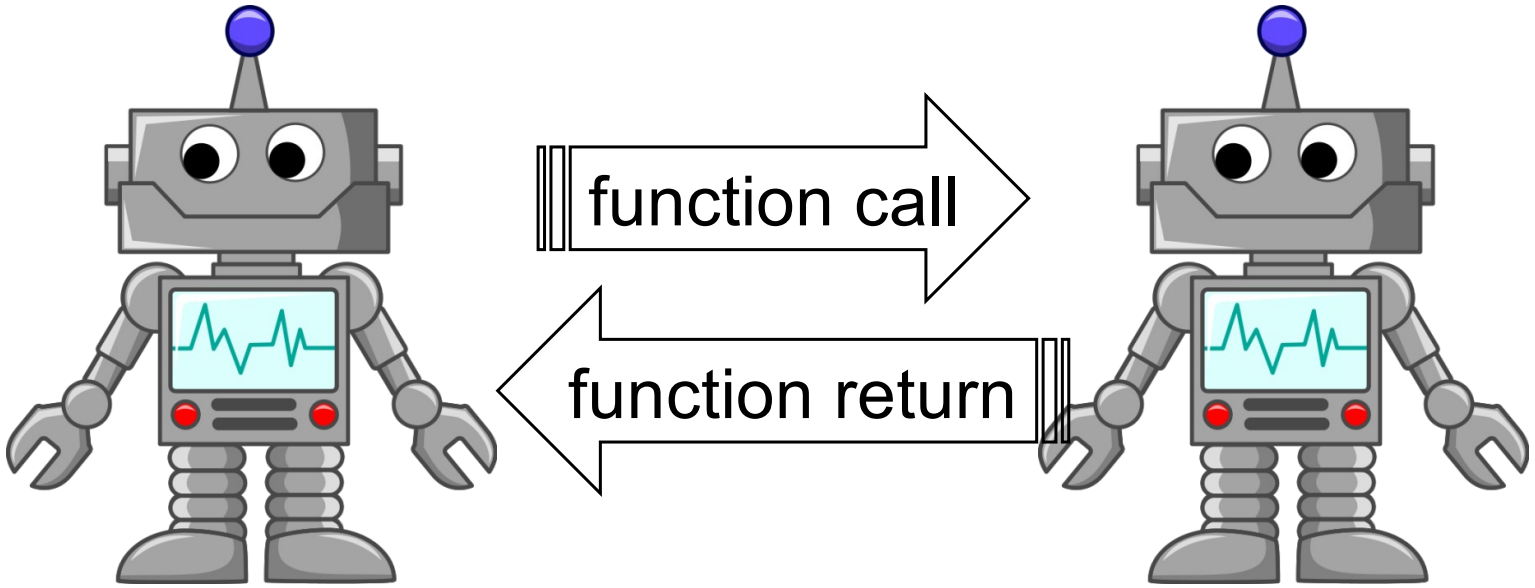
External to External Message



External to Contract & Vice-Versa



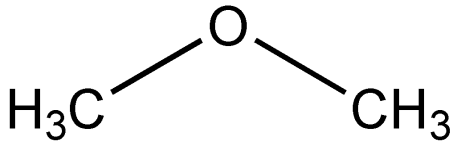
Contract to Contract Message



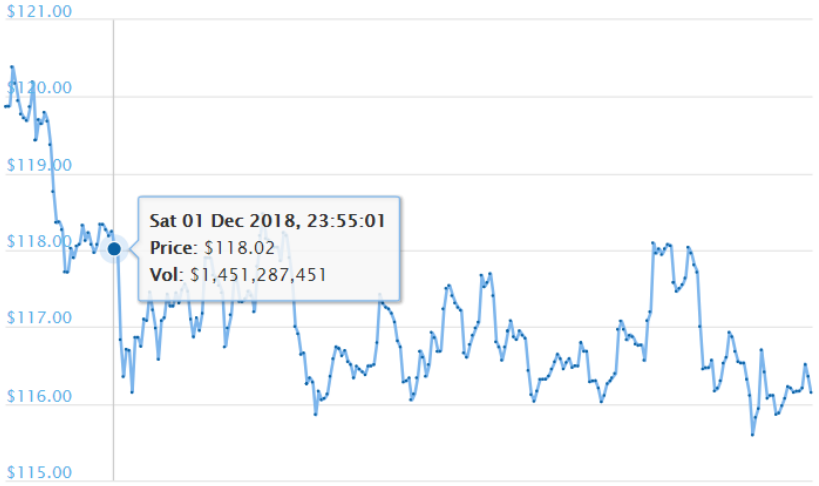
Money, Honey

Native currency called *ether*

not this



but this



Gas

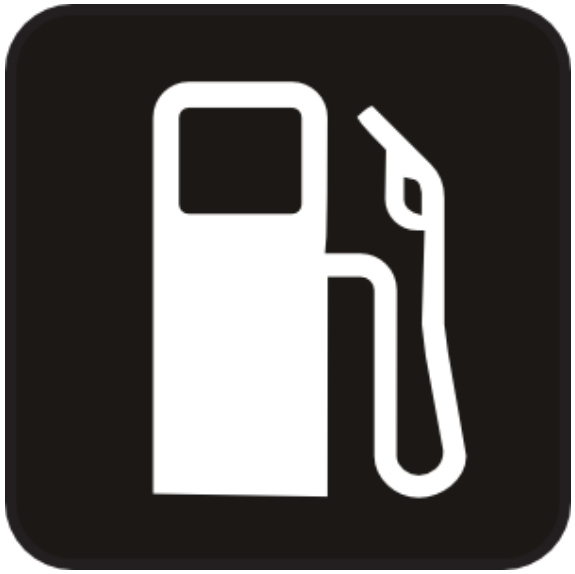
By calling a contract we are essentially running code, written on a Turing complete language, on thousands of nodes worldwide

Problems?

Gas

Caller pays fee for each transaction step

Denial of Service attacks expensive



Gas

Each step has fixed “gas” fee

But gas price in Ether up to caller!

Low price means low priority …

And vice-versa

Gas

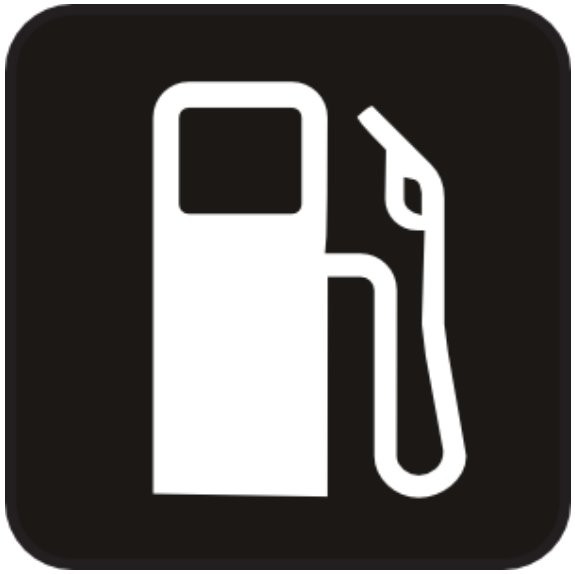
If a call runs out of gas …

Effects discarded

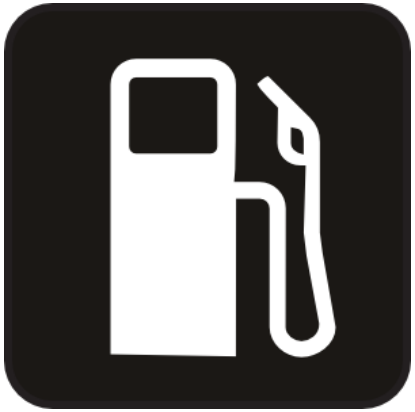
Gas not refunded

If a call has leftover gas …

Unused gas refunded



Block Gas Limit



Bitcoin has limit on block *size*

Ethereum has limit on block *gas*

Block full when transactions' gas costs reach limit

We will see how this can be exploited later

Transaction Fields

Gas price

Value

Gas limit

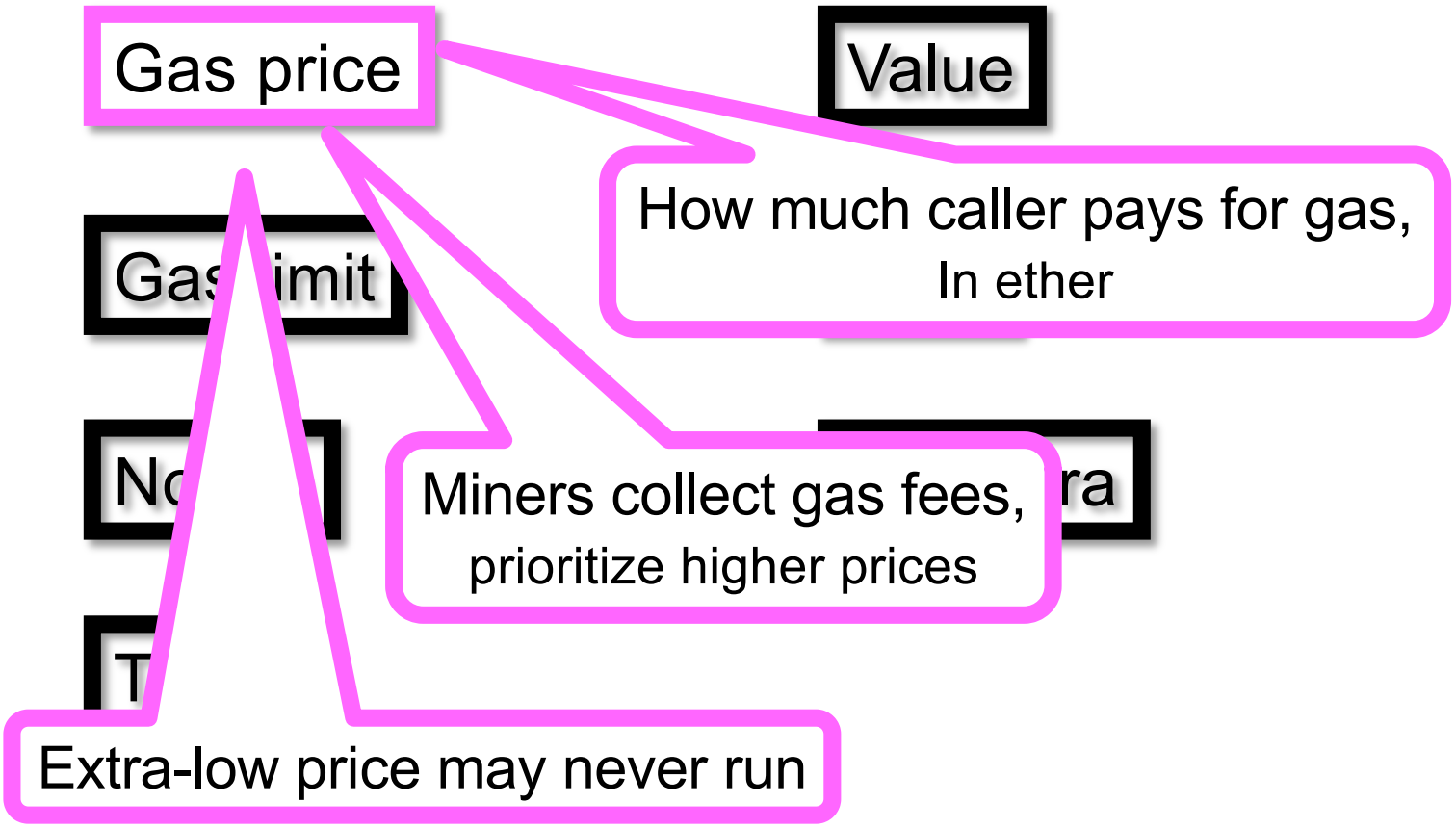
Data

Nonce

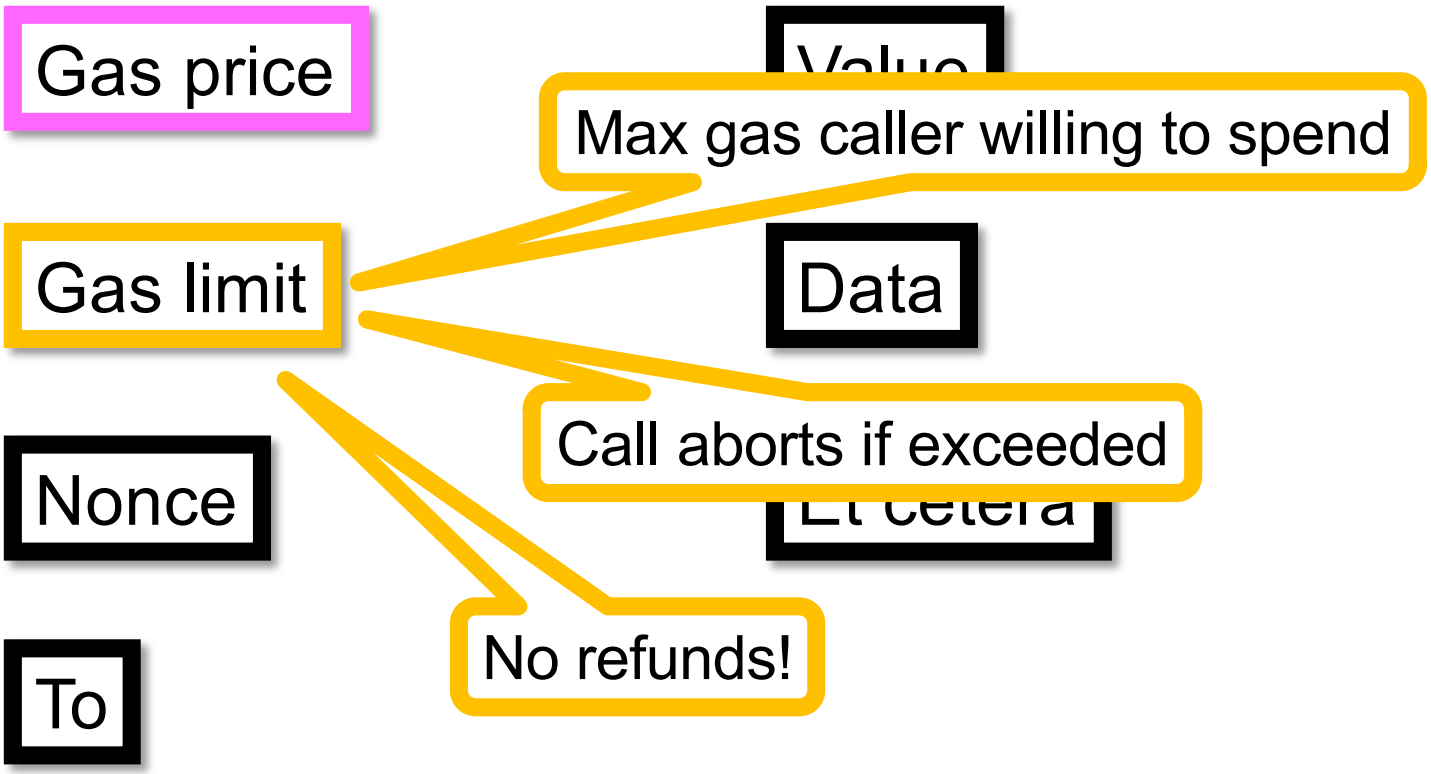
Et cetera

To

Transaction Fields

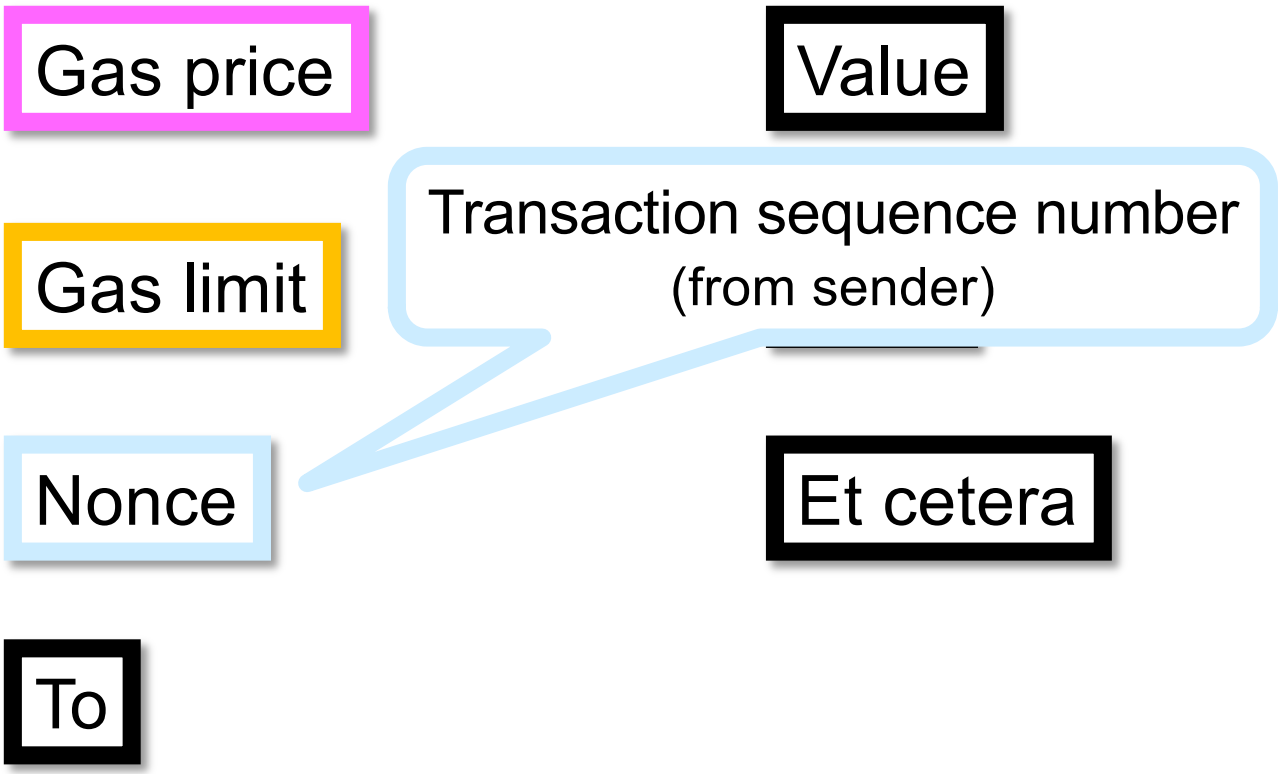


Transaction Fields

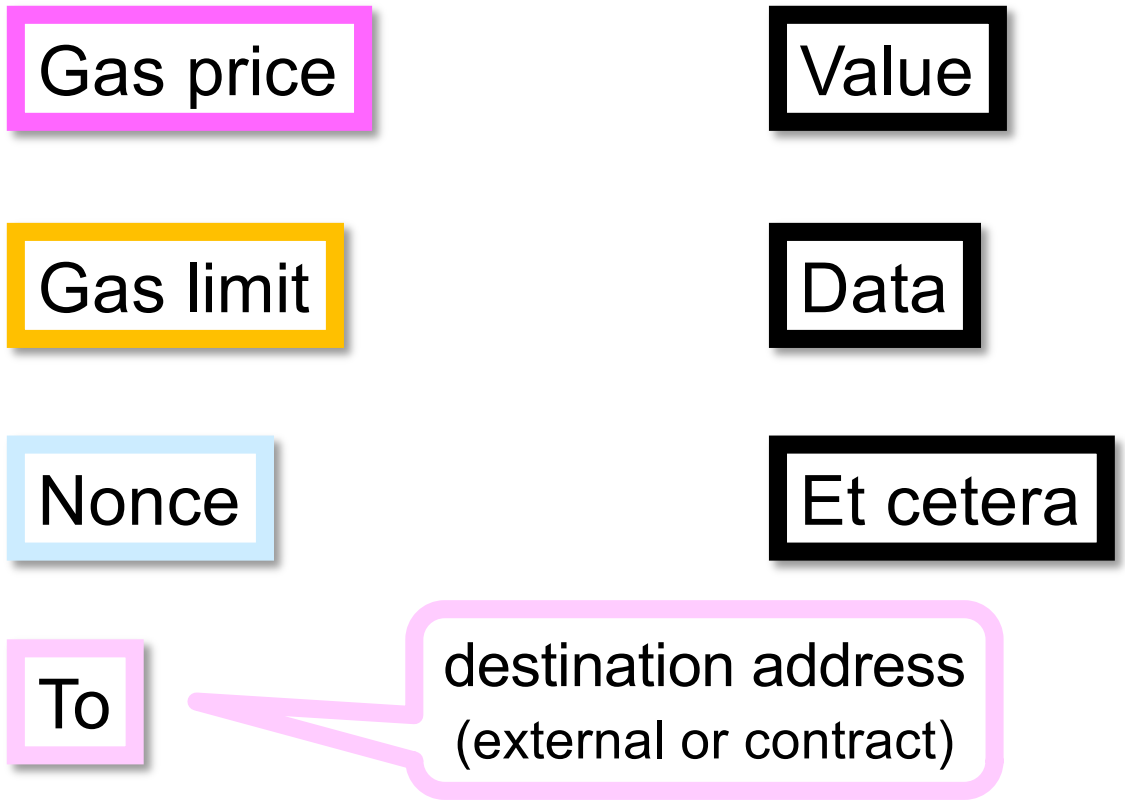


Can a caller be sure of the gas that is going to be spent?

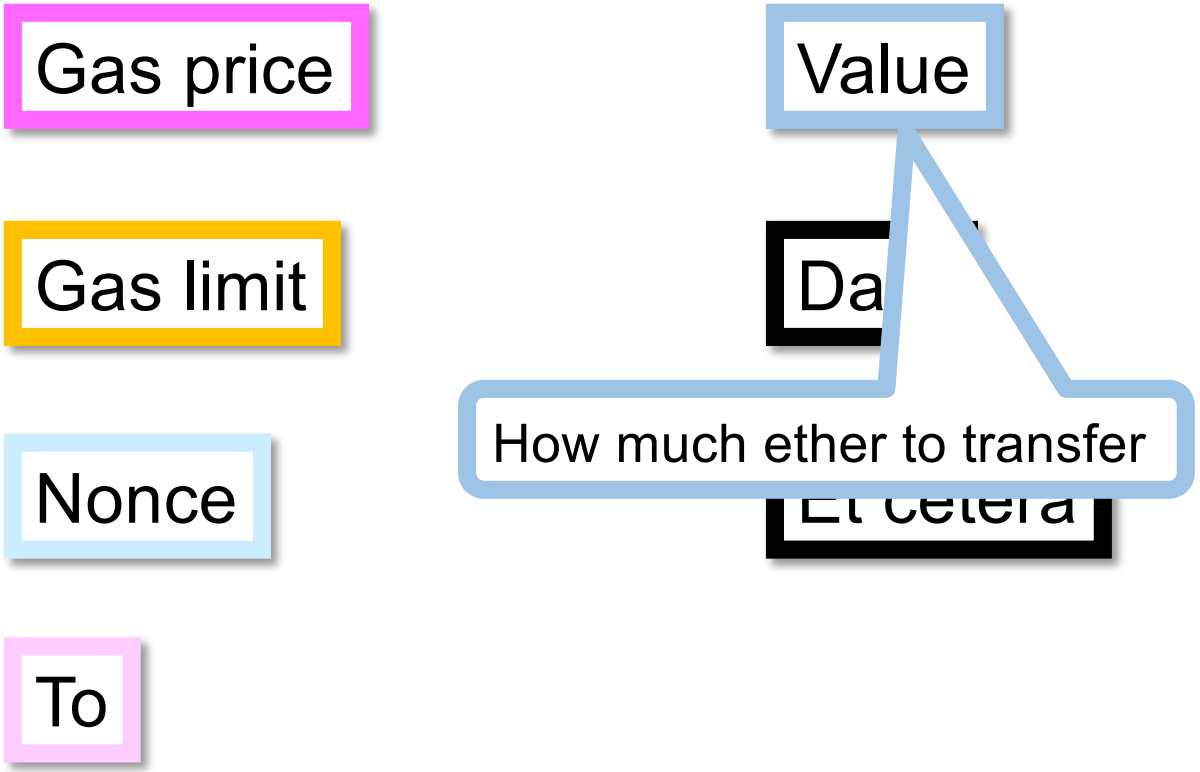
Transaction Fields



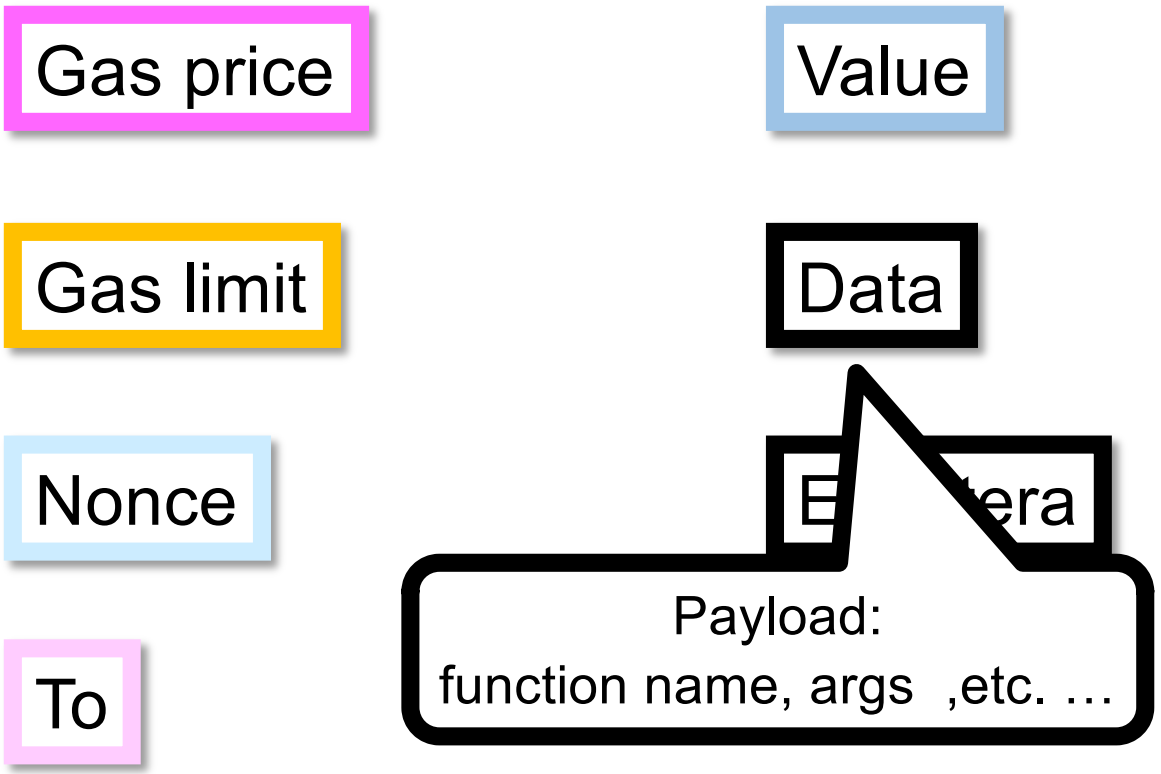
Transaction Fields



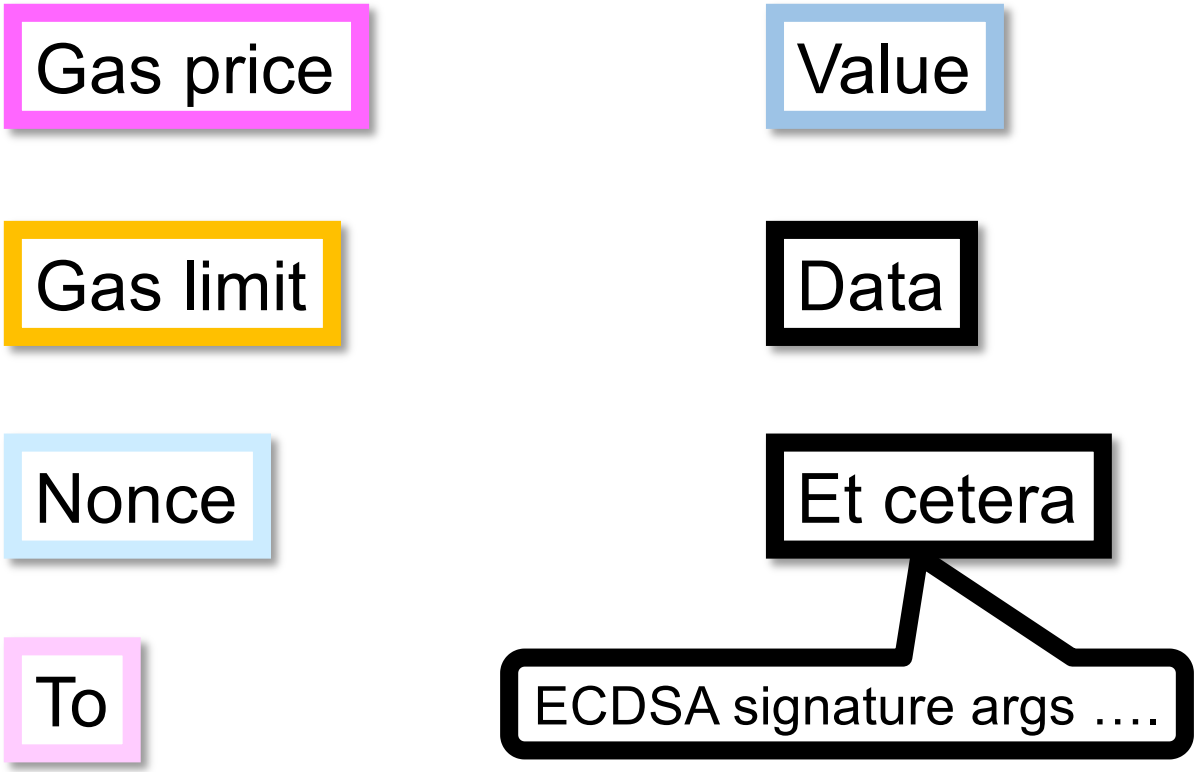
Transaction Fields



Transaction Fields



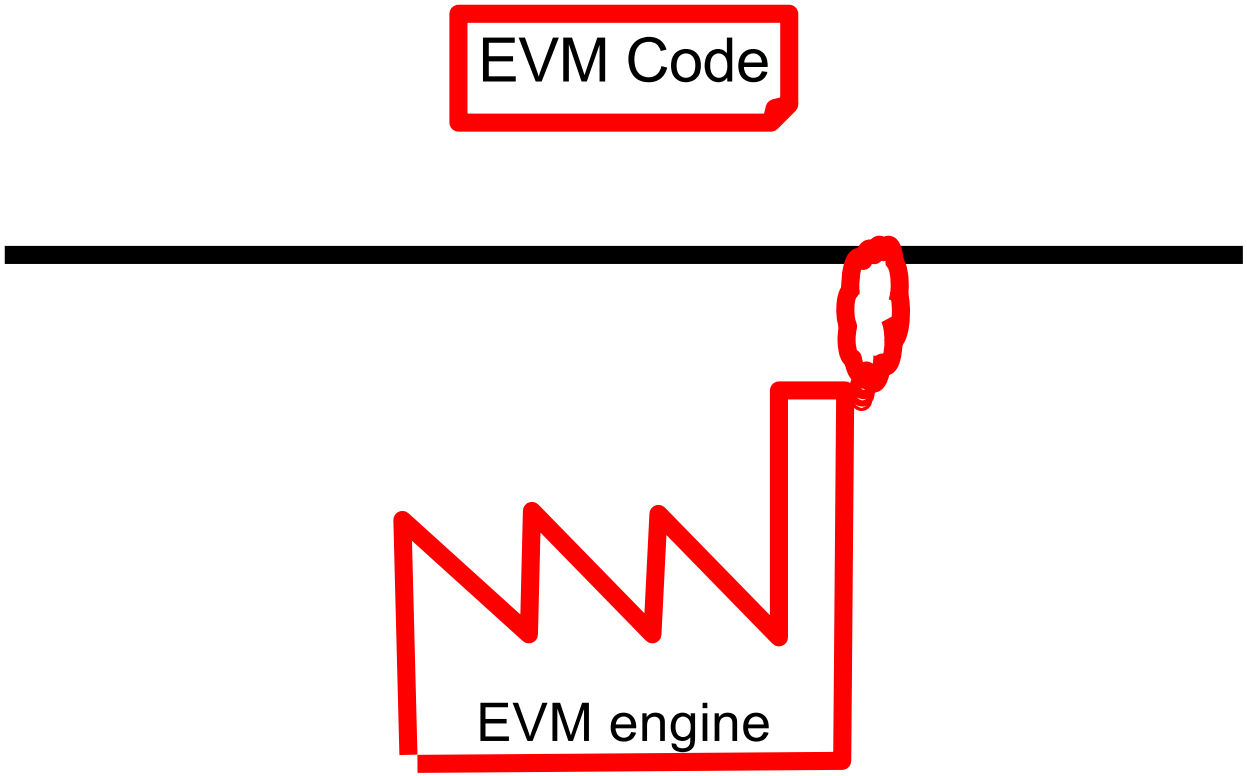
Transaction Fields



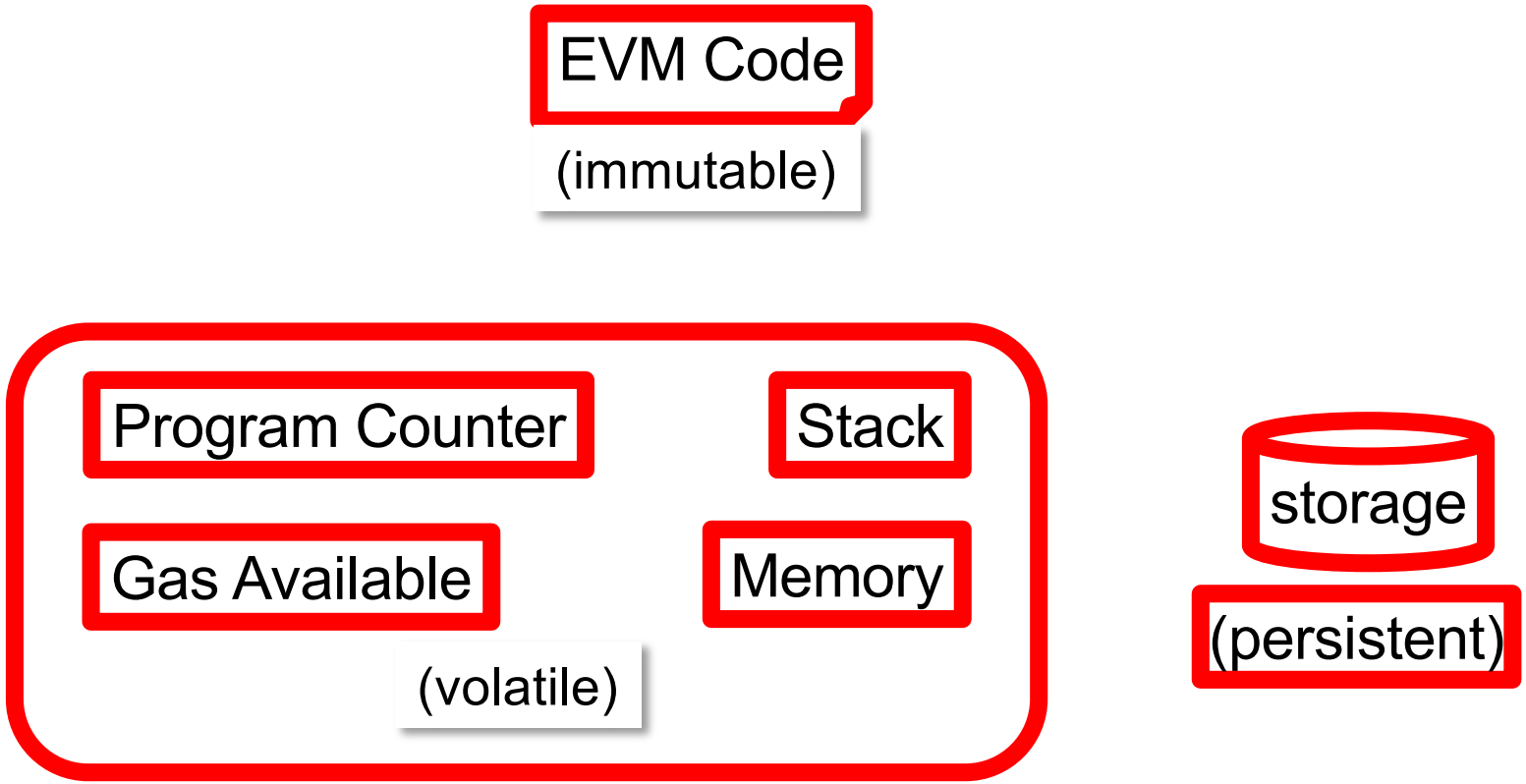
Recent changes in gas pricing

- EIP-1559 (part of “London hard fork” – Aug. 2021) defines new policies for gas pricing
- Idea: replace a first price auction with a base fee mechanism (though users can add a tip for their txns to be prioritized by validators)
- Base fee depends on how full recent blocks are:
  - Blocks do not fill up → base fee decreases
  - Blocks are completely full → base fee increases

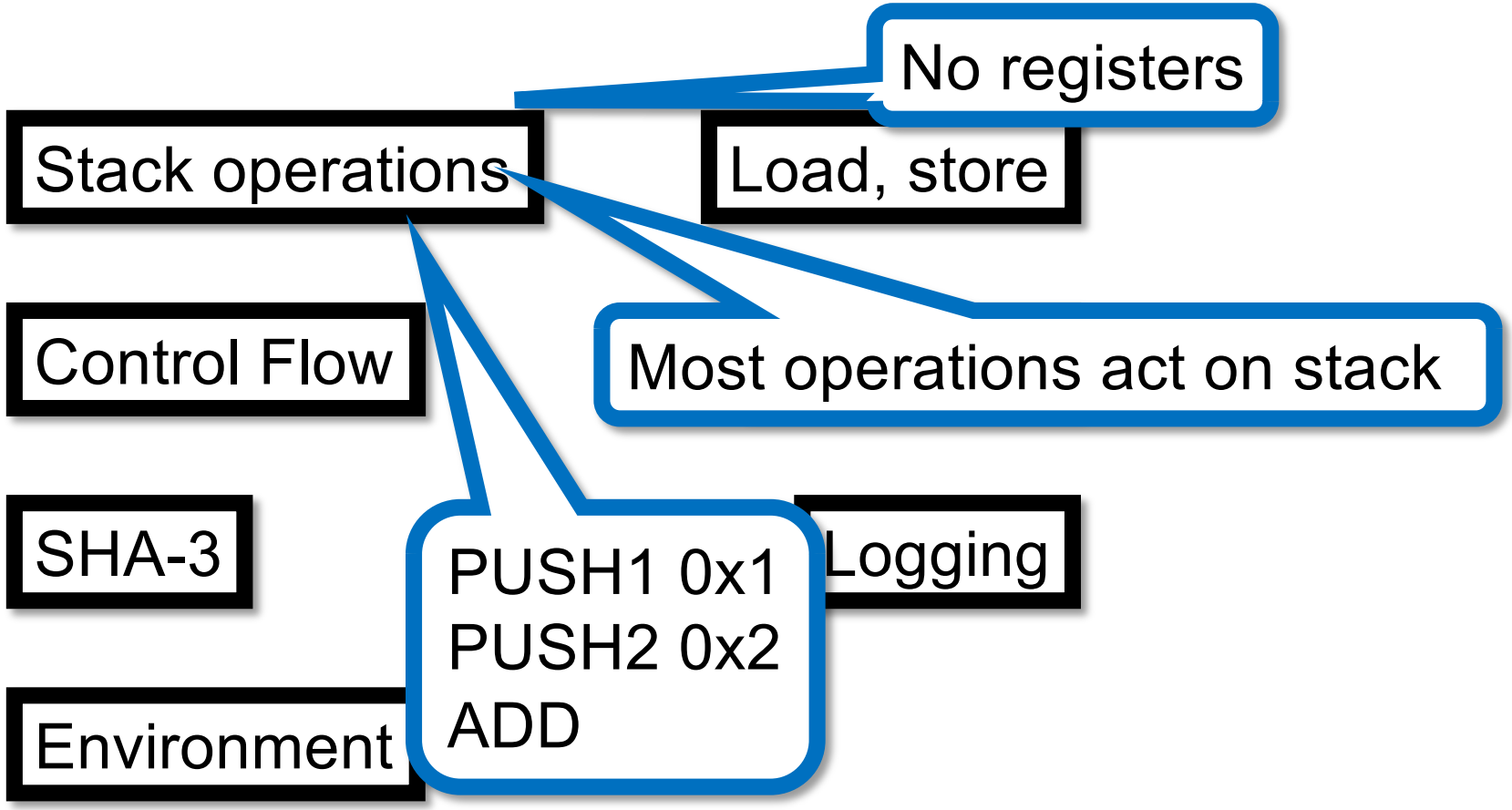
Ethereum Virtual Machine



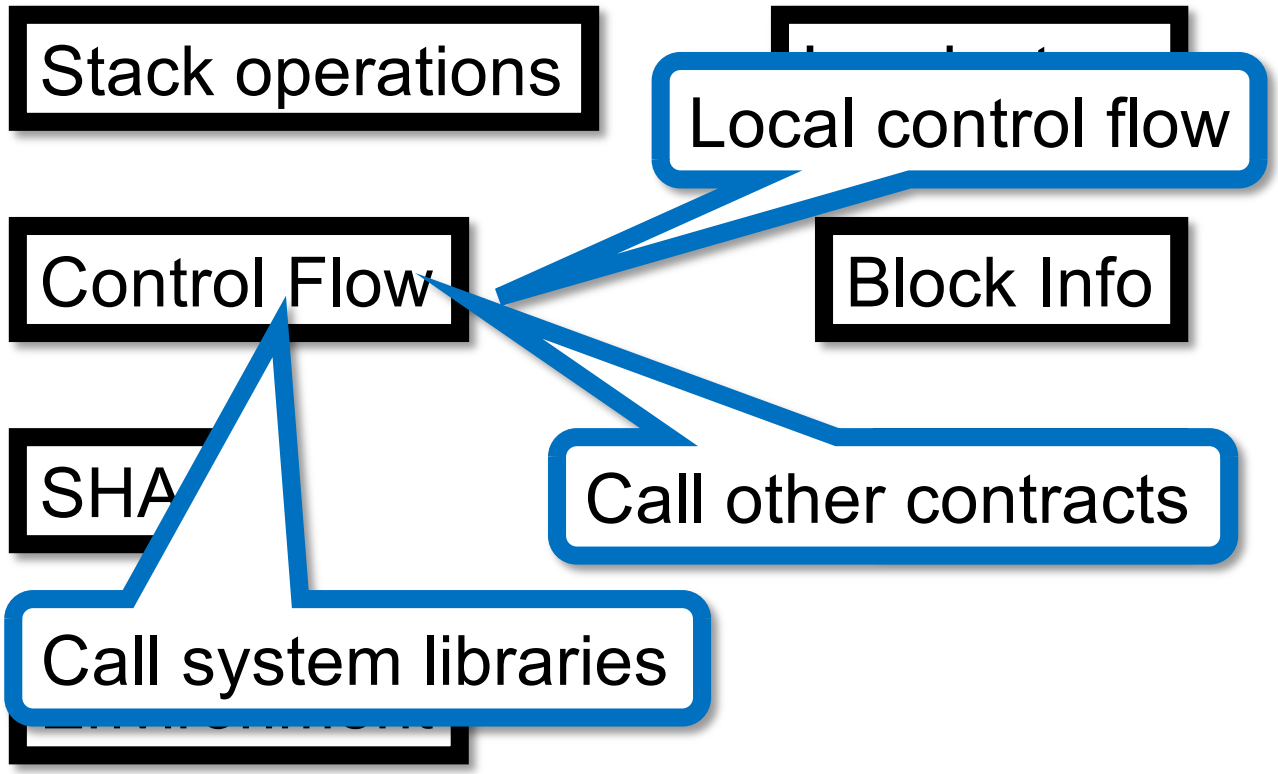
Ethereum Virtual Machine



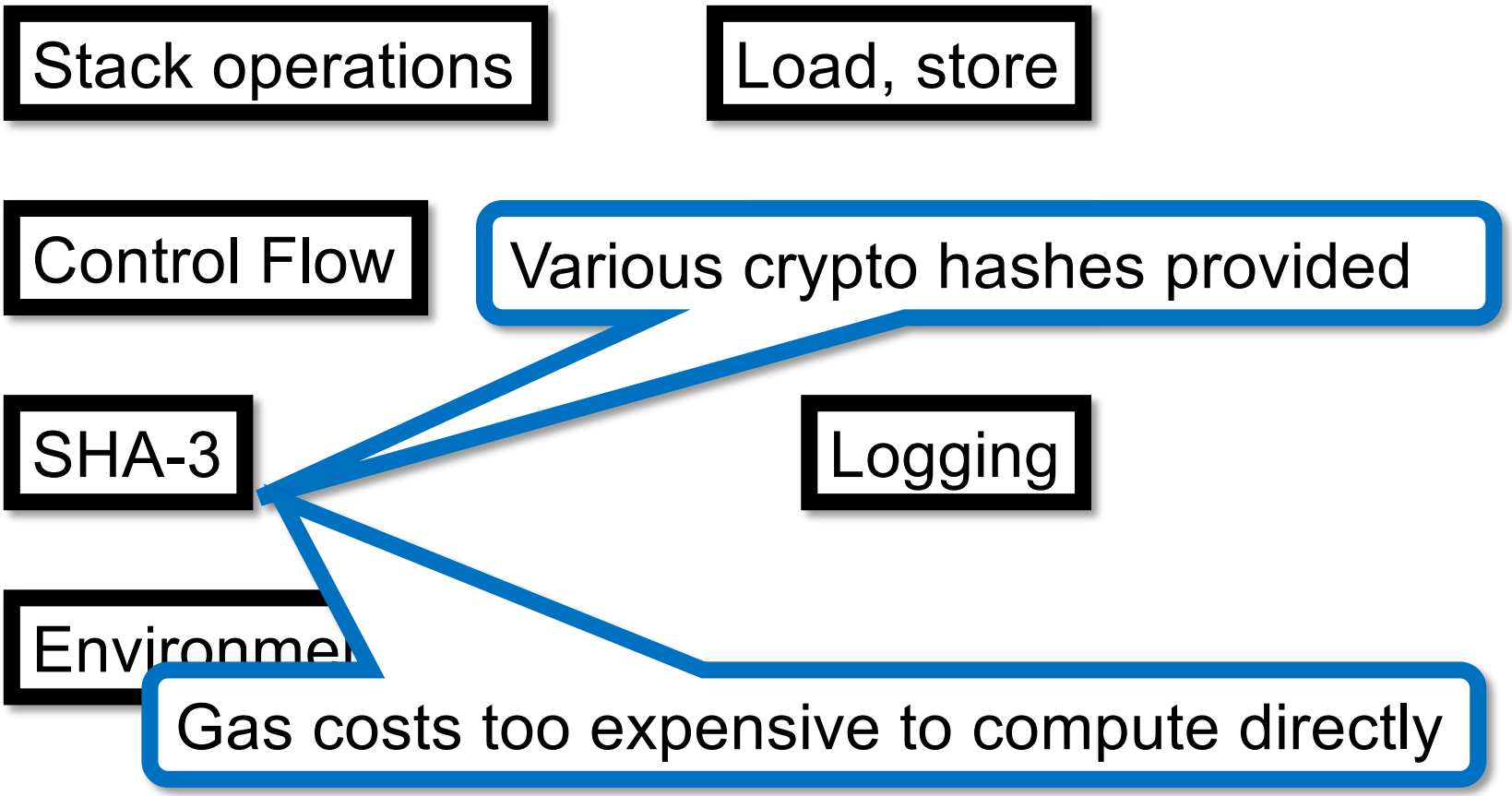
Types of Instructions



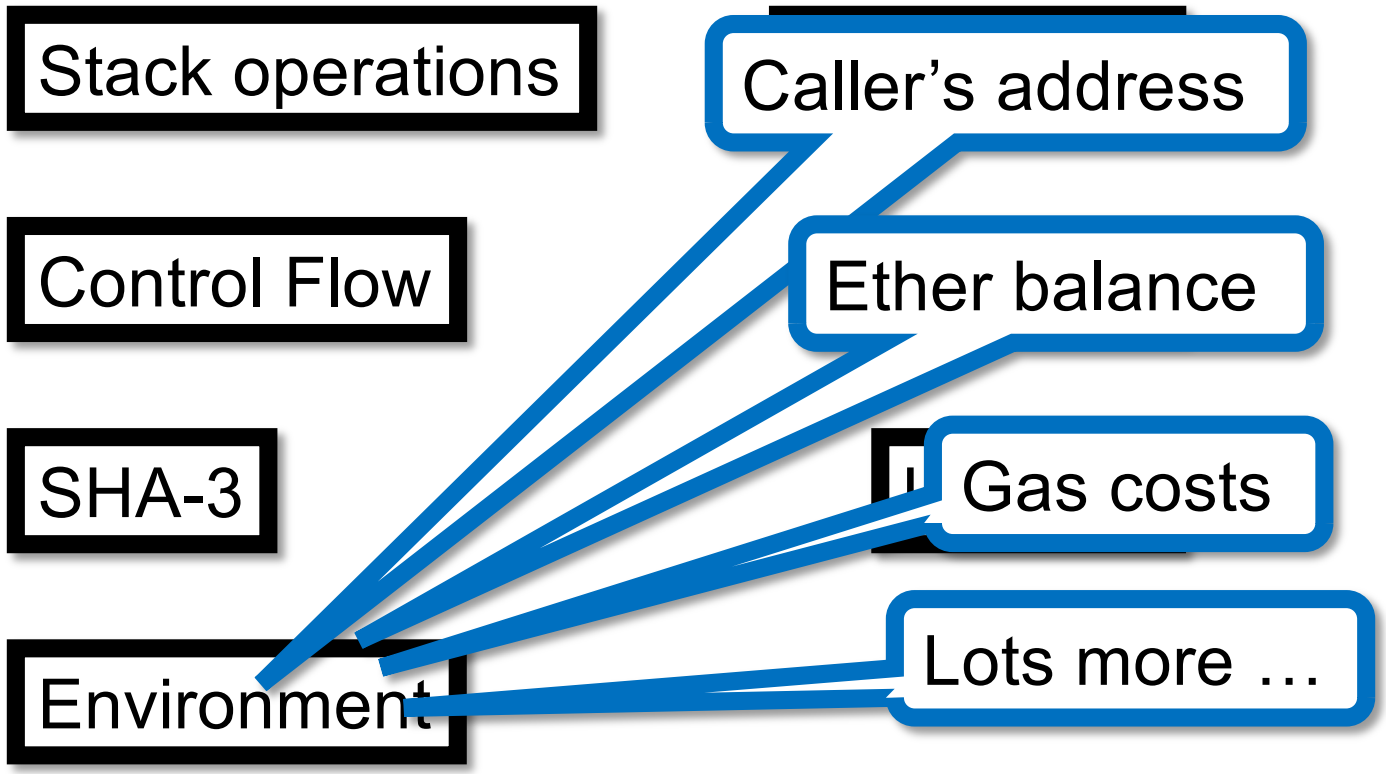
Types of Instructions



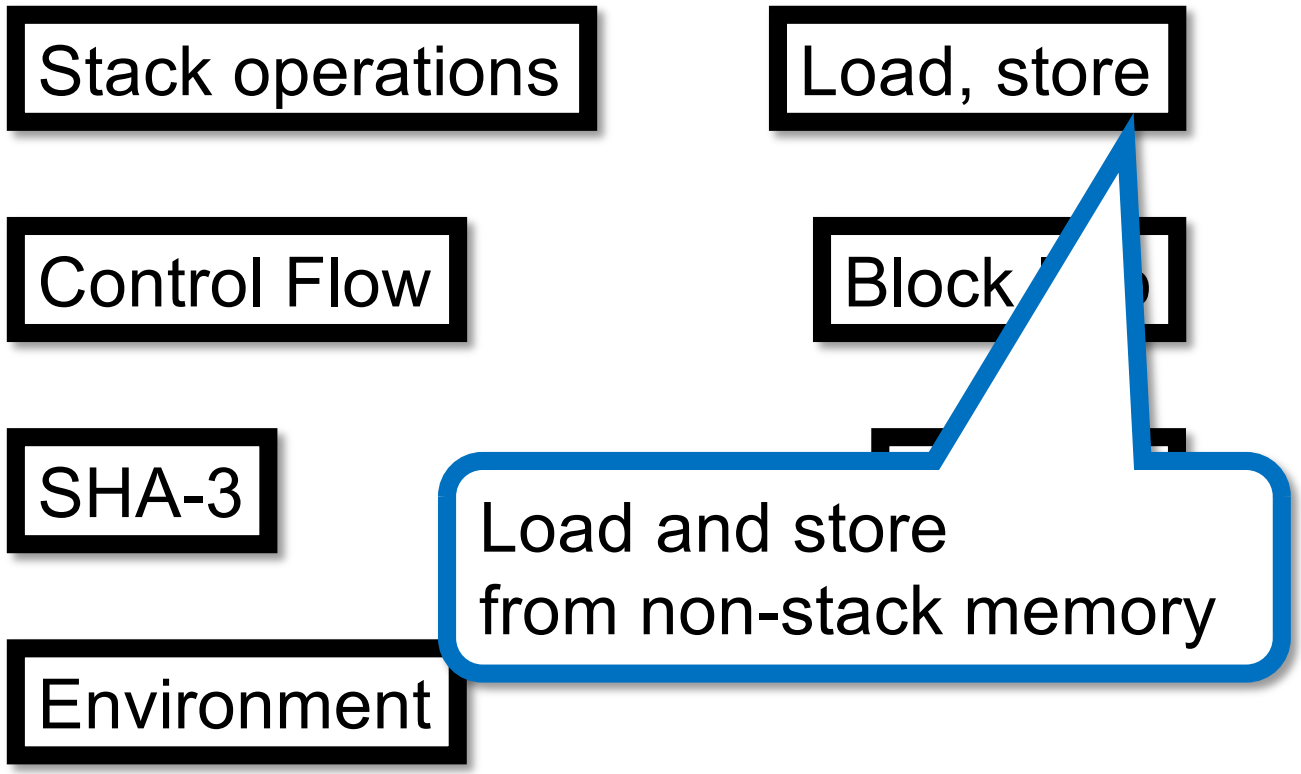
Types of Instructions



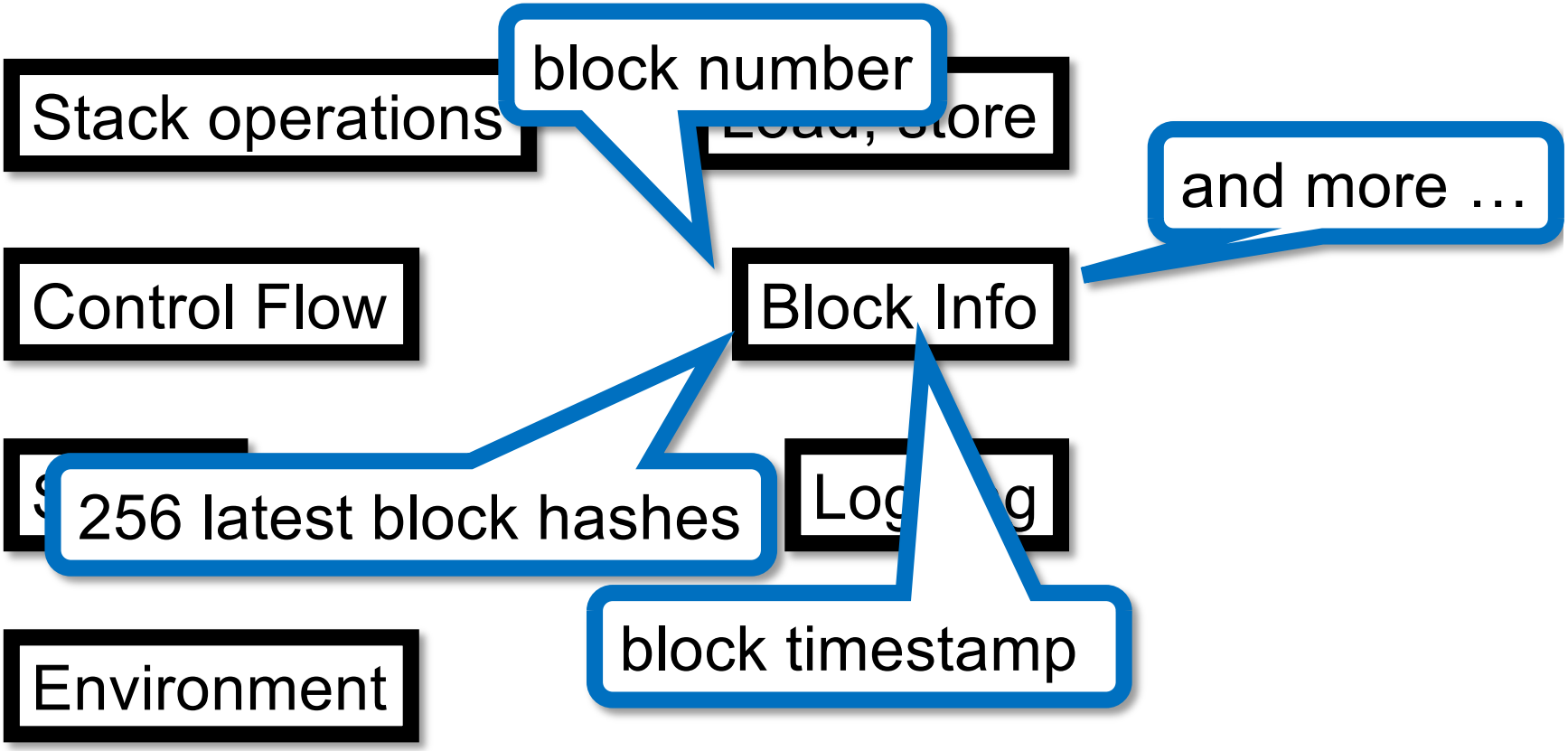
Types of Instructions



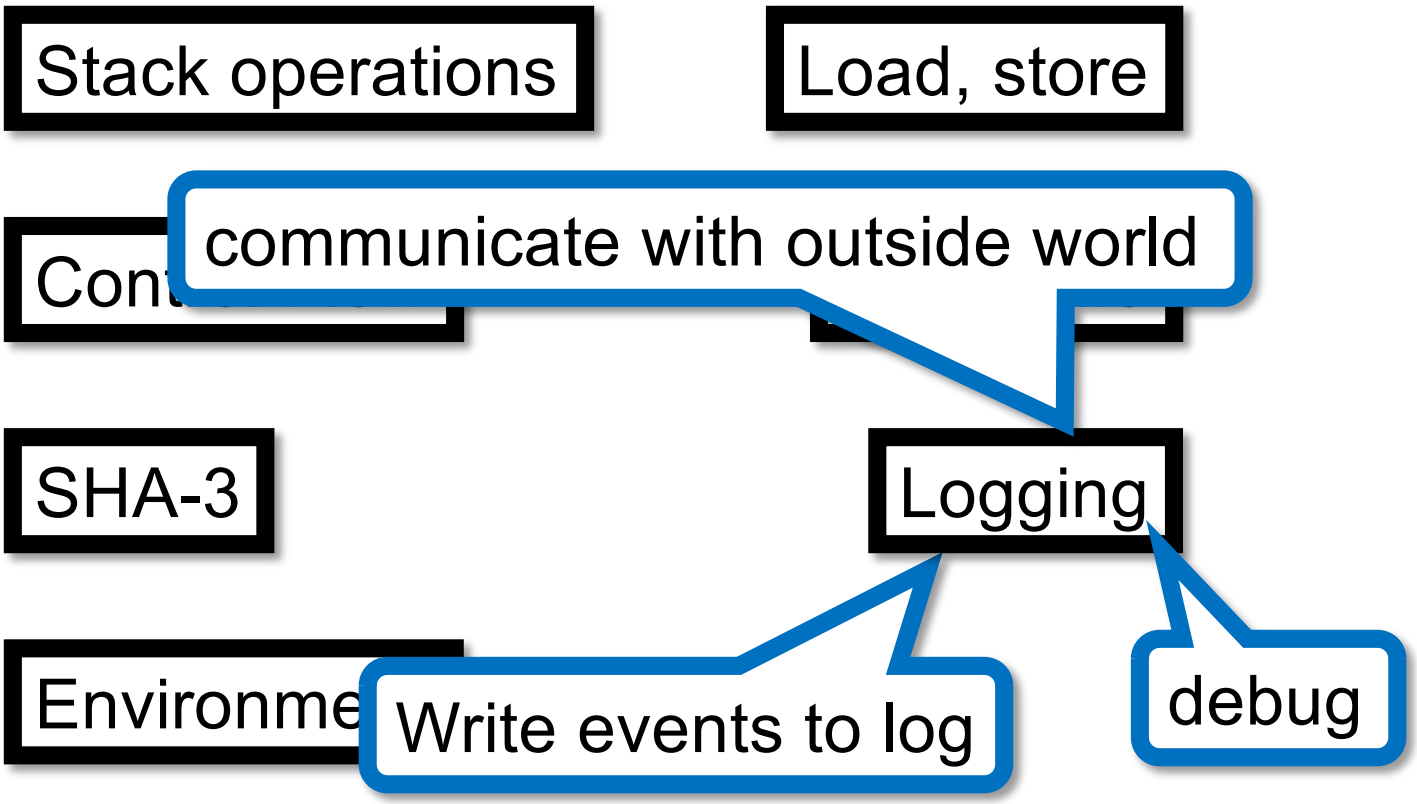
Types of Instructions



Types of Instructions



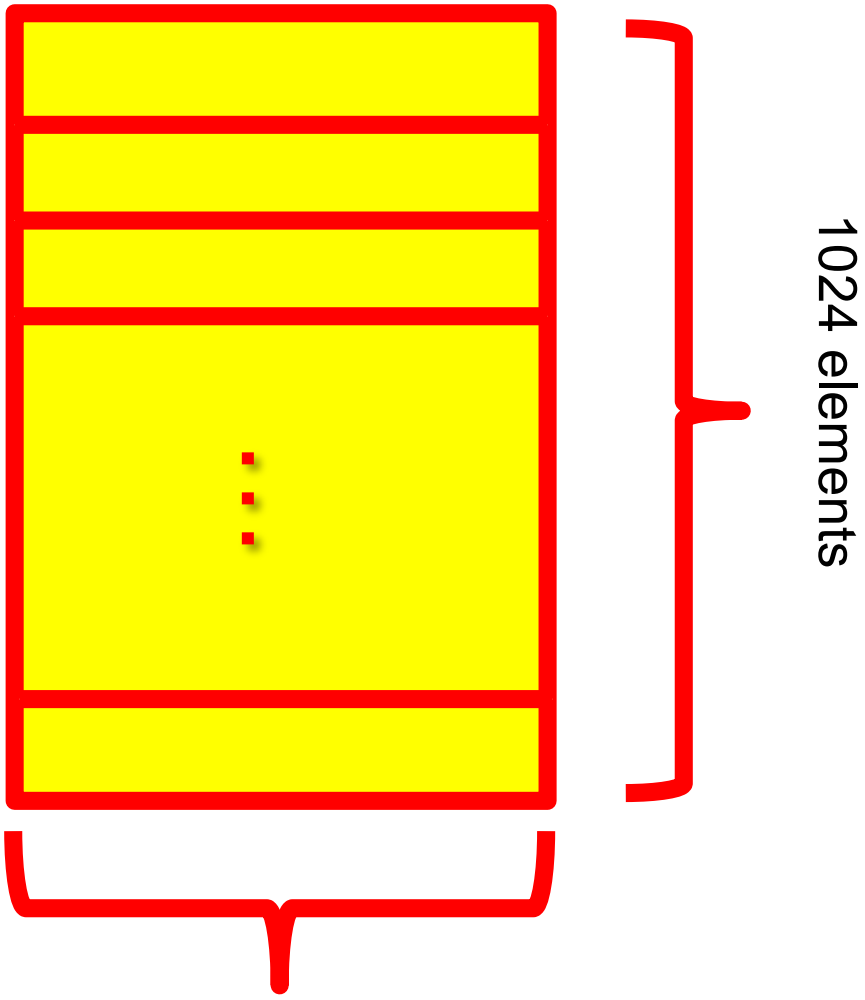
Types of Instructions



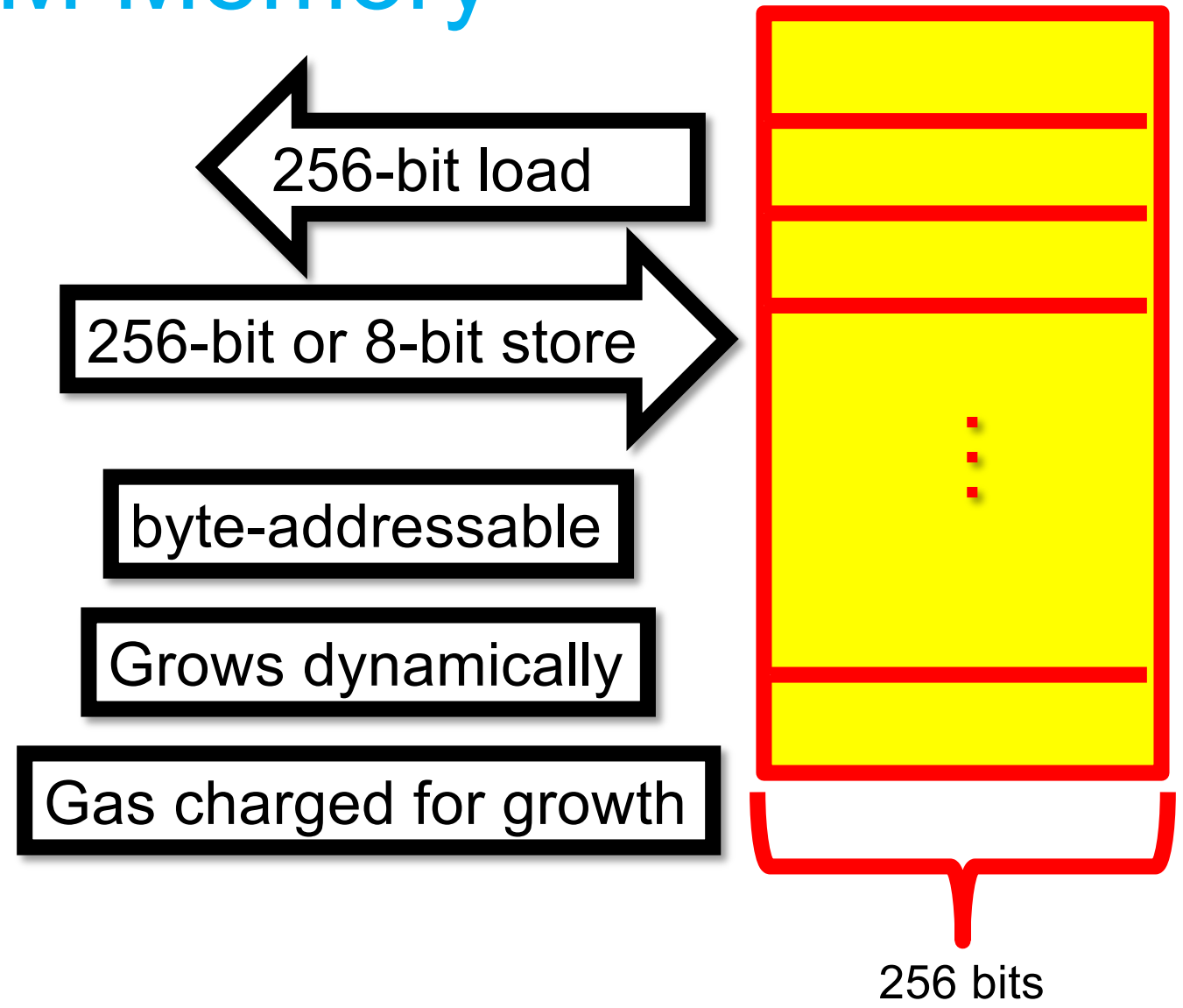
EVM Stack

All operations act on stack (e.g., arithmetic Logic, etc.)

PUSH, POP, SWAP …



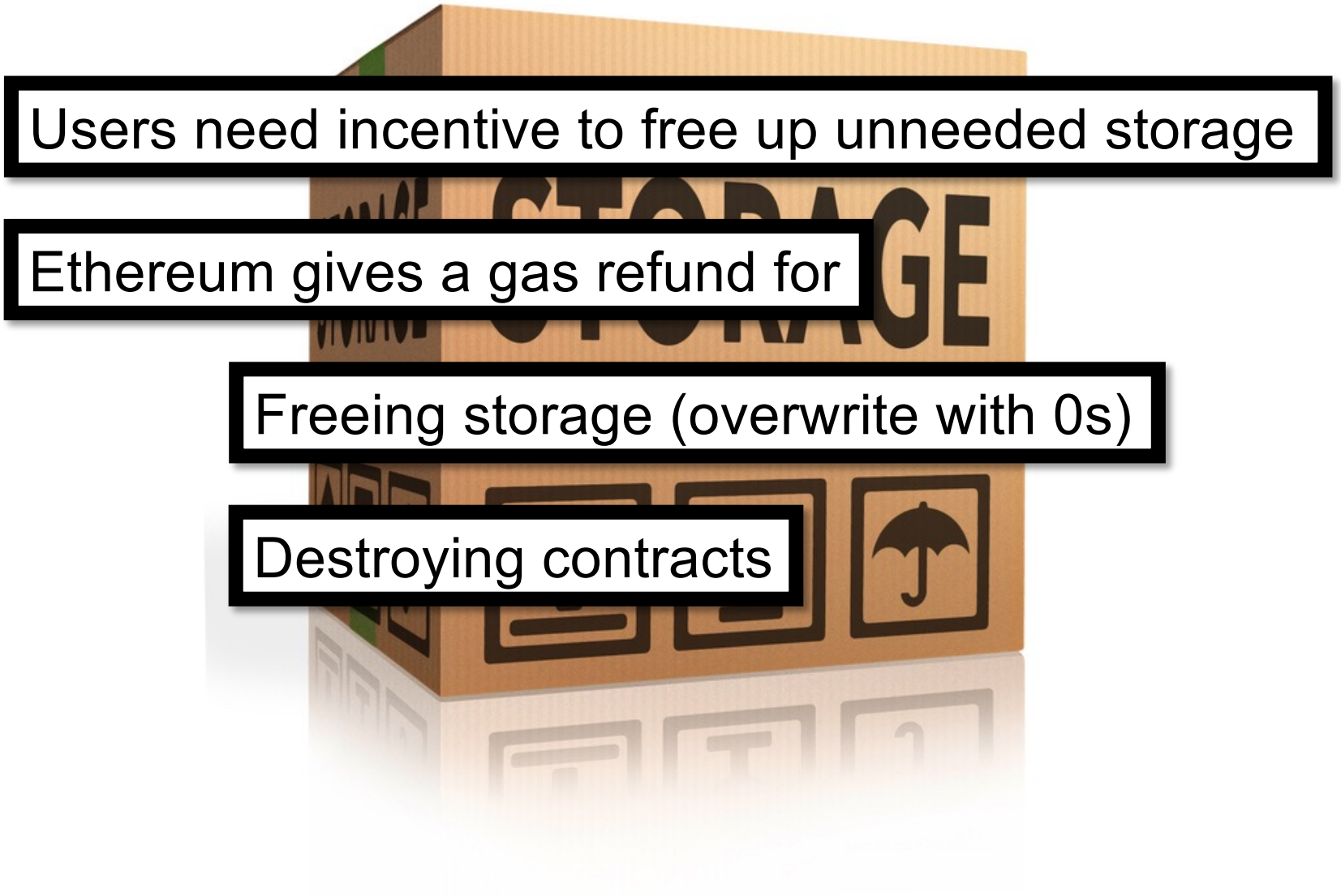
EVM Memory


256-bit load
256-bit or 8-bit store
byte-addressable
Grows dynamically
Gas charged for growth
256 bits

EVM Storage

- API/abstraction is identical to memory, but values persist across transactions
- Storage is persisted in the blockchain itself

Storage is Expensive



Gas Token

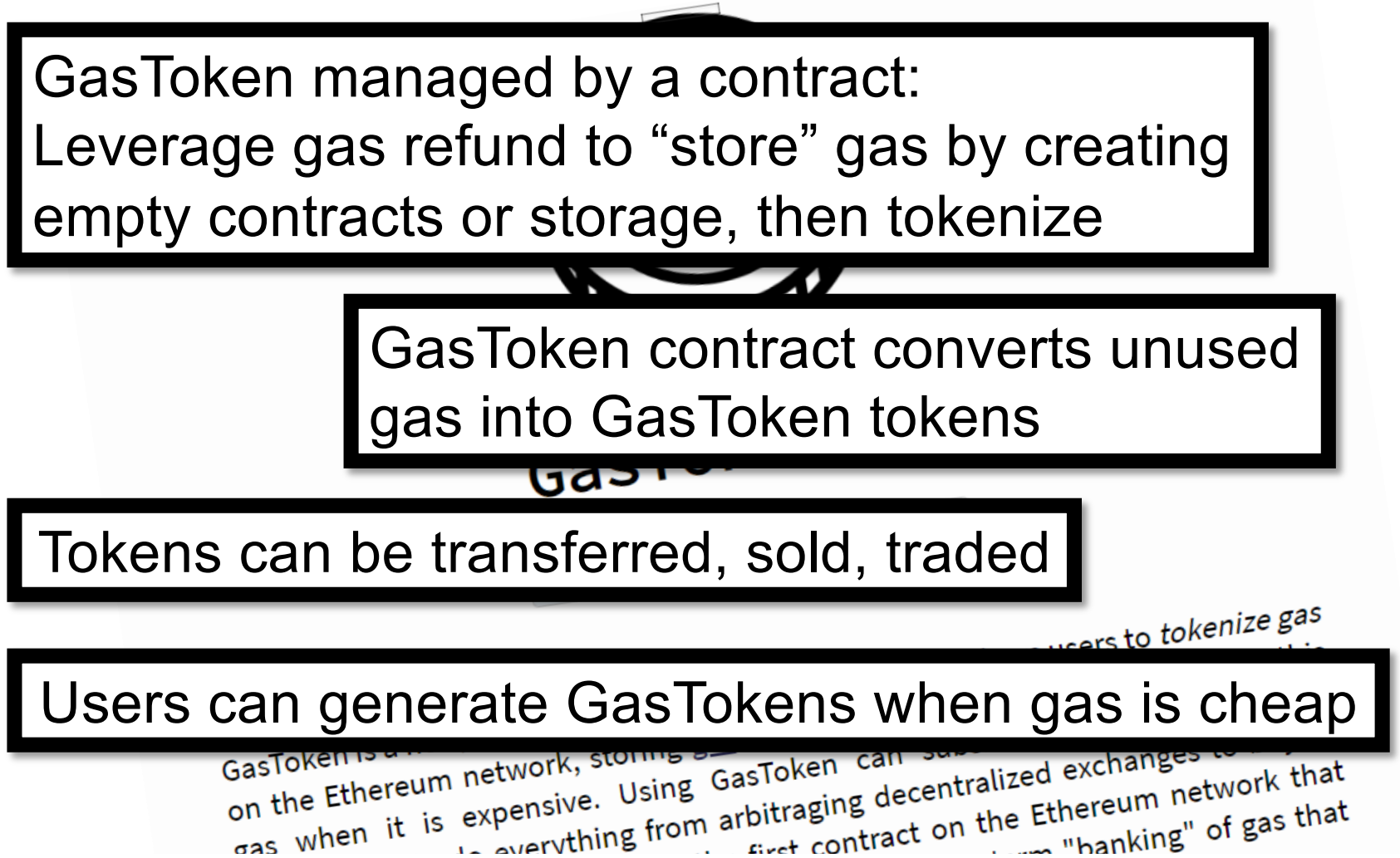
(Ethereum token that represents a storage of gas)

GasToken managed by a contract:
Leverage gas refund to “store” gas by creating empty contracts or storage, then tokenize

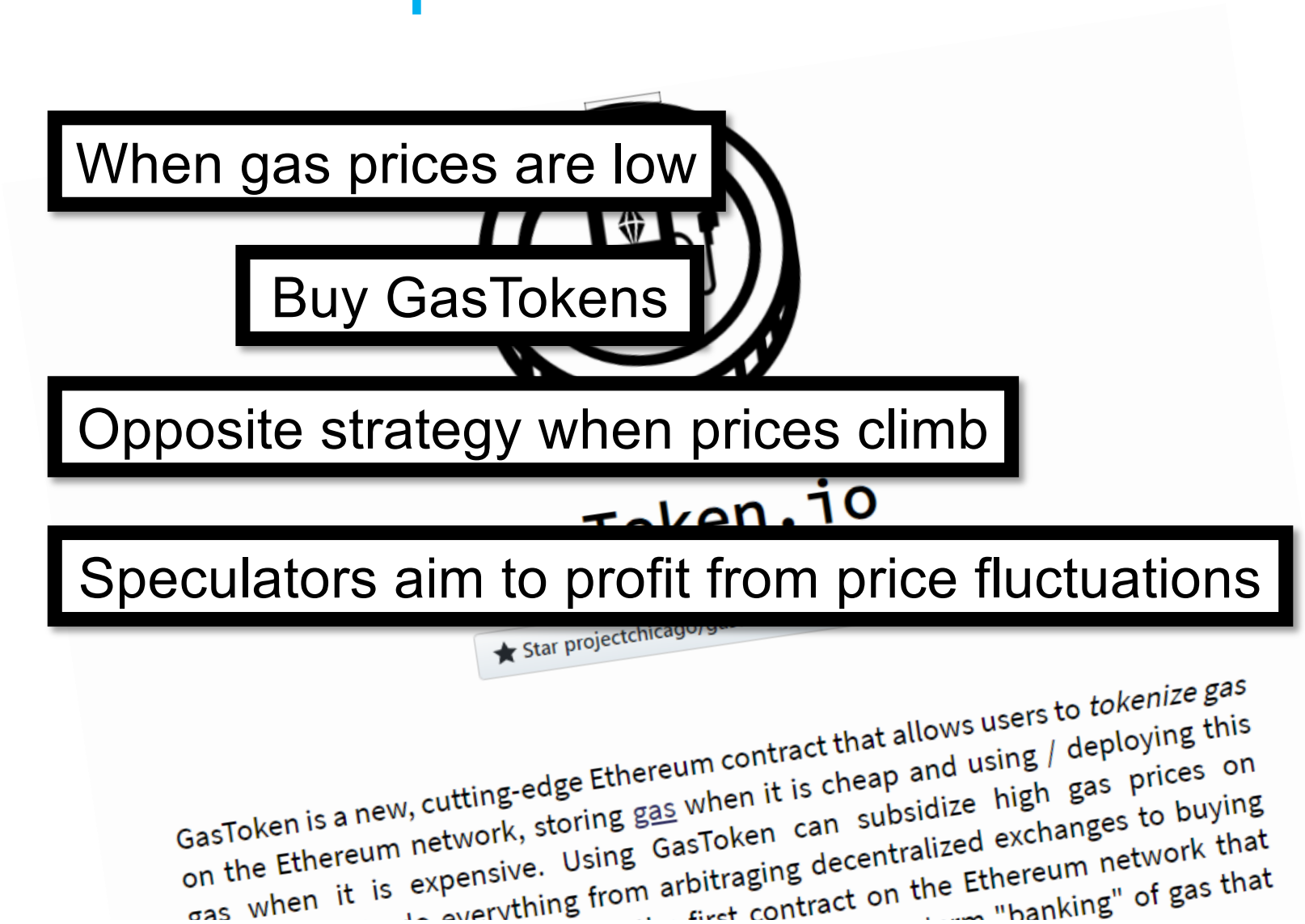
GasToken contract converts unused gas into GasToken tokens

Tokens can be transferred, sold, traded

Users can generate GasTokens when gas is cheap



Gas Token Speculation



Is That Ethical?

Negative effects:

- Uses memory only for speculation
- GasToken might drive up prices
- Unethical behavior can arise

Is That Ethical?

- Positive Effects
  - Gas-banking service for users
  - Hedge against increases
  - Predictability in face of volatility

Further Reading and Acknowledgements

- Ethereum: A Secure Decentralised Generalized Transaction Ledger
- Slides adapted from Maurice Herlihy, Brown University, available under CC BY-NC 4.0 (https://creativecommons.org/licenses/by-nc/4.0/)