chaincode

Do Bitcoin Devs Dream of Script ?

A talk about Covenants & Advanced Contracts

What's Bitcoin is good for ?

Reserve of value !

Payment system...

Commerce infrastructure ?

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must

What's commerce in real-world ?

Alice wants a new laptop sold by Bob. Alice sends 1 BTC to Bob.

But laptop never comes…

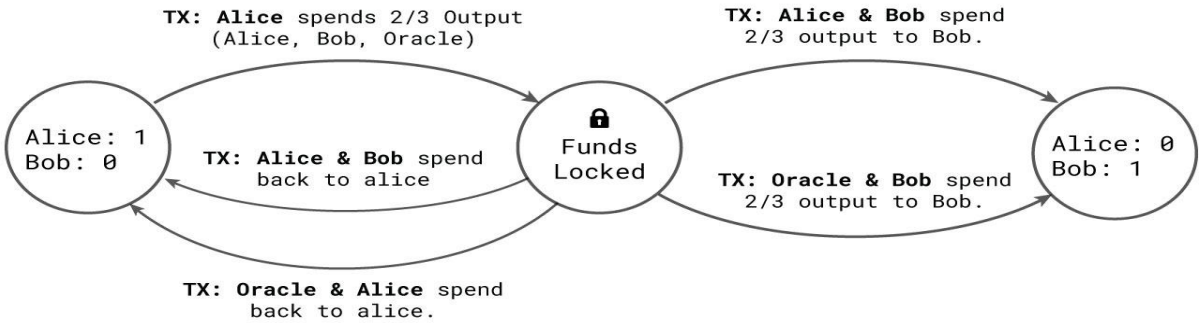
What's a Finite State Machine ?

An abstract machine that ...

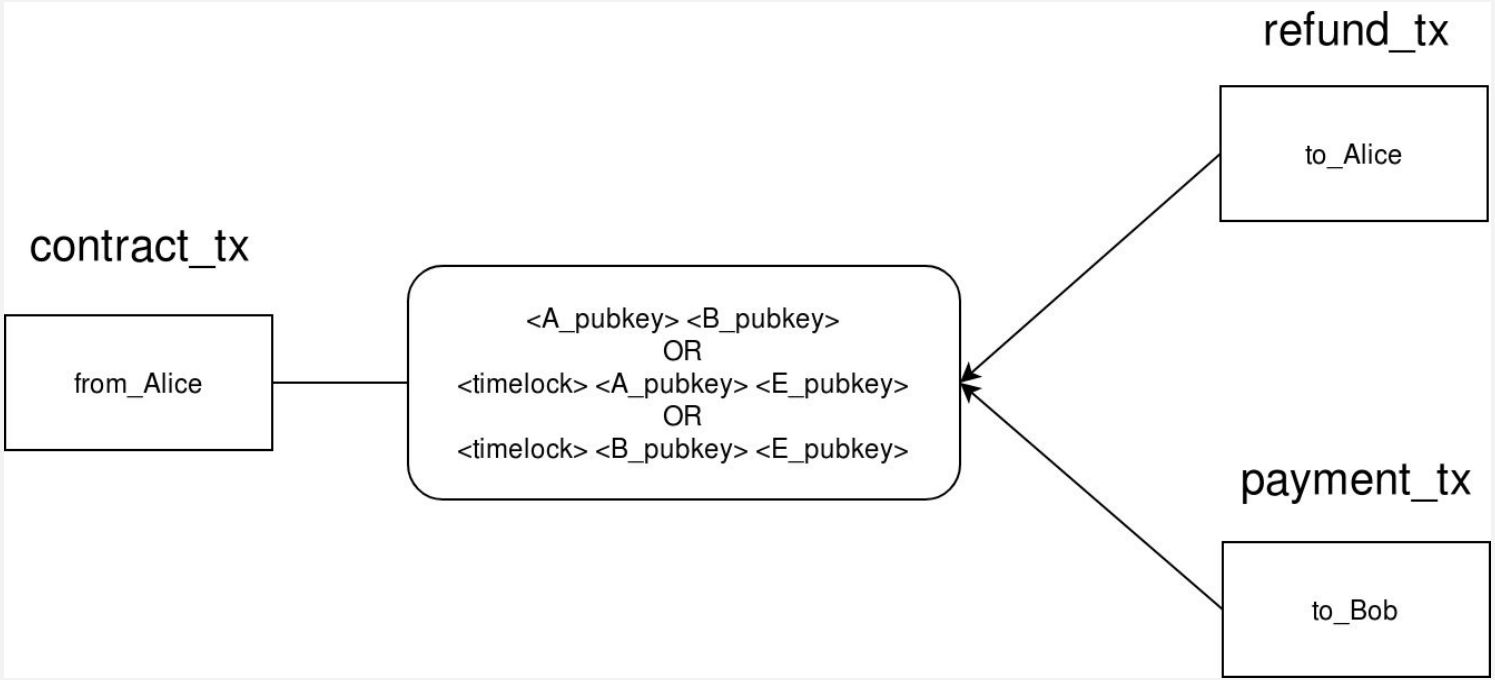
- Can have exactly **one** among a finite number of states at a given time
- Can change states in response to external **events** called a **transition**

E.g. Traffic Lights

Finite State Machine to model contracts



Implementing FSM in Bitcoin Transactions



What's a predicate ?

"The script is actually a predicate. It's just an equation that evaluates to true or false. Predicate is a long and unfamiliar word so I called it script."

-Satoshi, BitcoinTalk 2010

- Definitions:
  - Lock script is the expression *y*
  - The unlock script is the expression *x*
  - The predicate is the assertion *x* = *y*

- y may be a hashlock, a pubkey, a timelock, … x is the signature, the preimage, the block height, …
- You may create tree of expressions (Taproot <3)

Bitcoin contract = state machine

Where *events* are successful predicate evaluations, *transitions* are transactions leading to new *states*, i.e balances between parties.

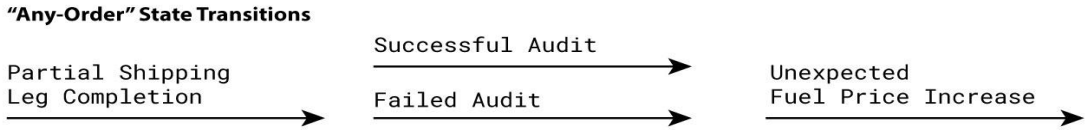
State machines are good cognitive tools to design complex contracts.

How to model advanced contracts ?

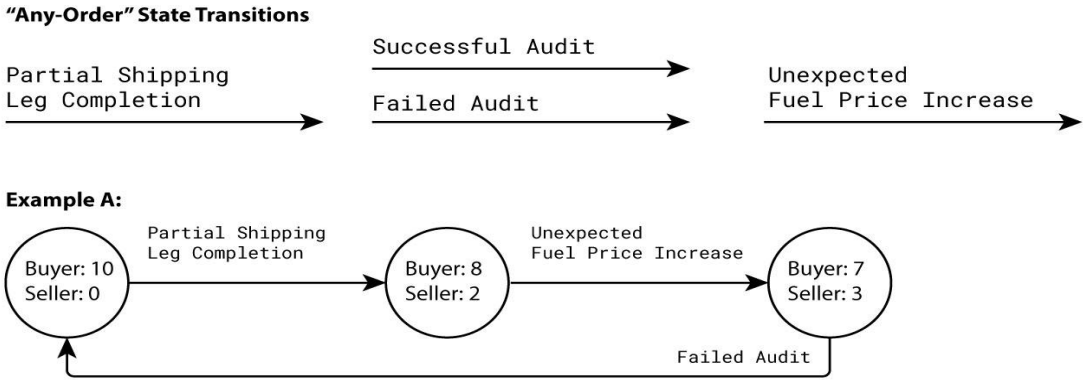
Alice is in Rotterdam and wants to buy 100_000 oil barrels to Bob in Brazil. They want to add some clauses to the contract, eg:

- *Audit clause*
- *Delay penalty*
- *Hardship clause*
- *Force majeure clause*
- *Final payment clause*

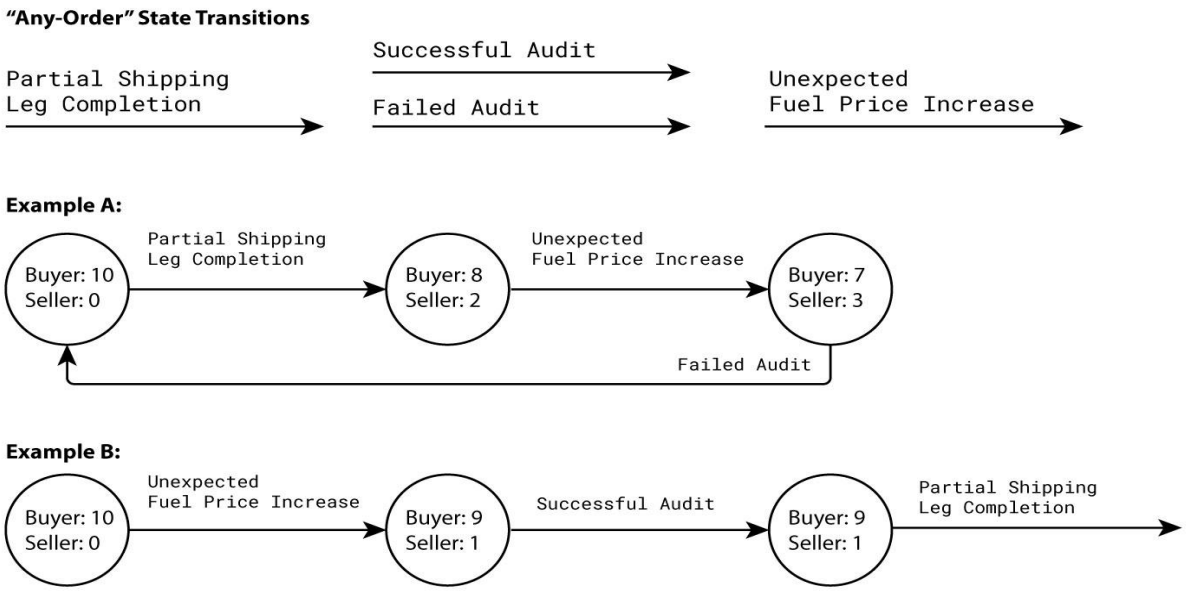
A simple FSM for trade contract 1/3



A simple FSM for trade contract 2/3



A simple FSM for trade contract 3/3



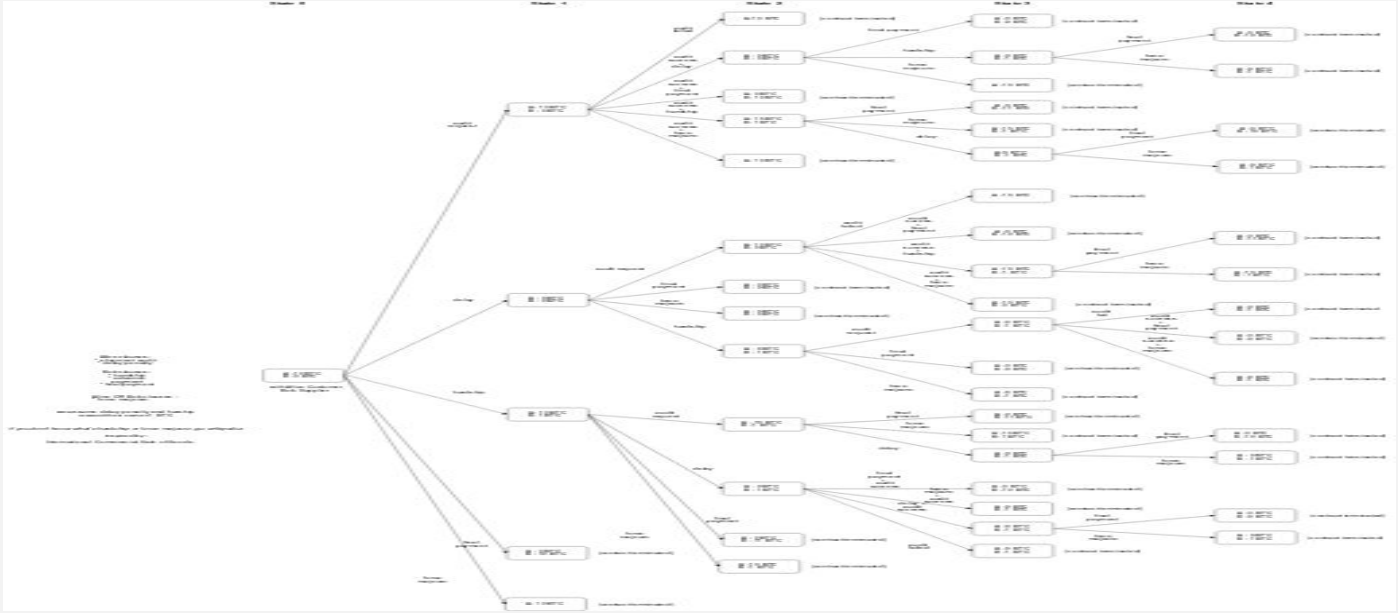
How do we design any-order contract ?

Let's define a **any-order contract** where we can't know before the fact the order of clause executions.

If one of them is executed and is not a TERMINATED state, we want to inherit validity of other clauses on subsequent states.

Problem ?

A static tree of transactions ? O(n!)



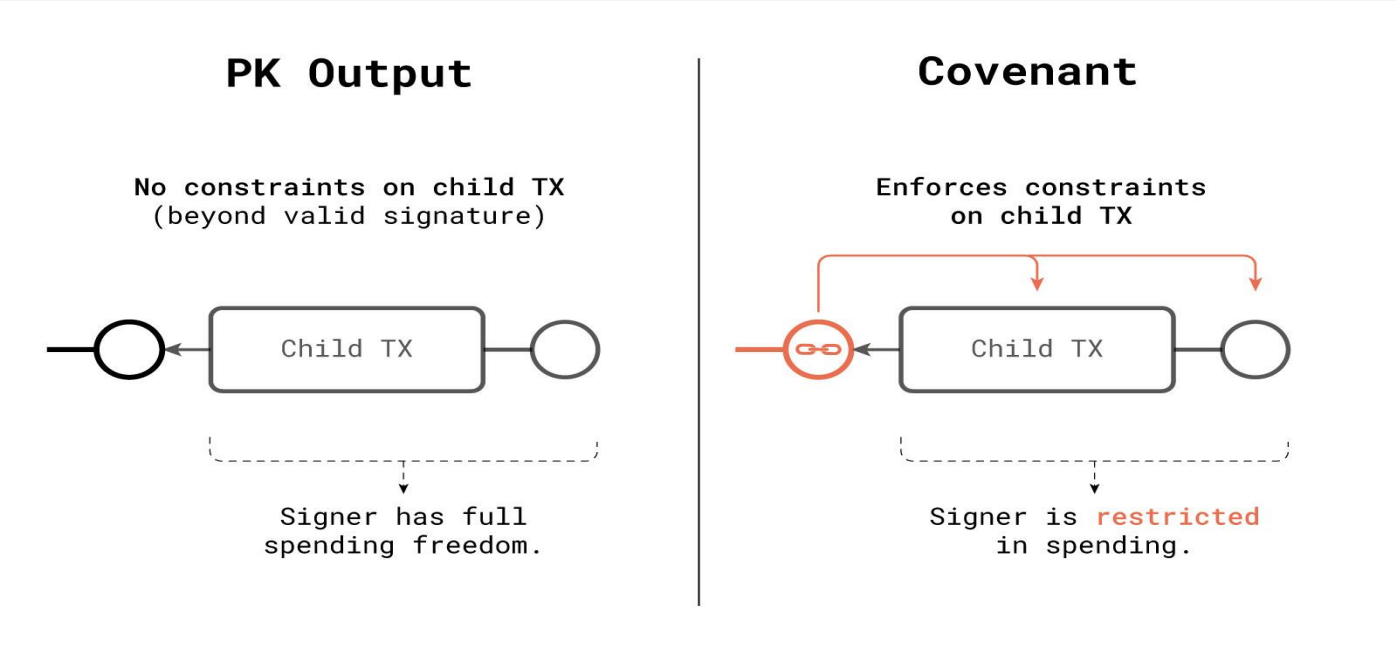
chaincode

What's a covenant ?

Transactions that are able to enforce restrictions on the composition of subsequent transactions.

Recursive covenant ?

Covenants on Bitcoin transactions



SIGHASH_NOINPUT + multisig

With SIGHASH_NOINPUT, don't commit to *outpoint*...

You can rebind a transaction dynamically on any output if witnesses match.

Multisig + _NOINPUT on transactions set = weak covenant

OP_PUSHTXDATA

script : <TXDATA_WEIGHT> OP_PUSHTXDATA
output: <transaction_weight>

script: <index> <TXDATA_VOUT> OP_PUSHTXDATA
output: <scriptPubkey> <amount>

script: <TXDATA_LOCKTIME> OP_PUSHTXDATA
output: <nLockTime>

To do a covenant you just have to compare pushed *scriptPubkey* against a predefined value.

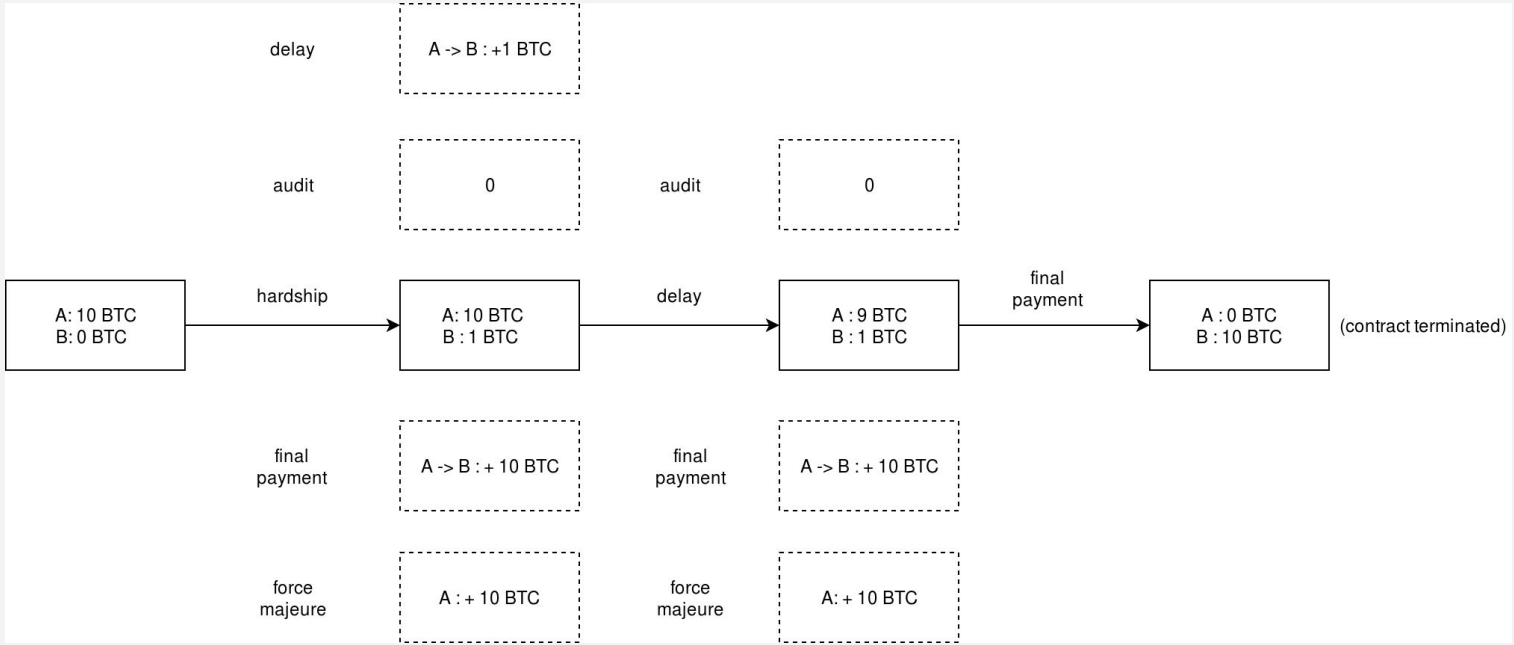
OP_SECURETHEBAG

script: OP_SECURETHEBAG <BagHash>

If <BagHash> != hash(nVersion, nLocktime, hashOutputs, hashSequence, ScriptSig, number_of_inputs>, execution fails.

Wrapped in a Taproot script path.

A dynamic tree of transactions (MAST + COV)



Open issues

- Protocol setup
- Engineering cost
- User cost
- Fungibility
- Security
- Computation
- ...

What's Bitcoin is good for ?

> The design supports a tremendous variety of possible transaction types that I designed years ago. Escrow transactions, bonded contracts, third party arbitration, multi-party signature, etc. If Bitcoin catches on in a big way, these are things we'll want to explore in the future, but they all had to be designed at the beginning to make sure they would be possible later.

- Satoshi, BitcoinTalk 2010

May we carefully extend it ?

Thank you and questions ?