All Smart Contracts Are Ambiguous

James Grimmelmann

Cornell Tech

November 29, 2018

In this talk

- Legal contracts are often ambiguous.
- Are smart contracts ambiguous too?
- Yes they are.
- It’s okay.

Contracts and ambiguity

An ambiguous contract

"US Fresh Frozen Chicken, Grade A, Government Inspected, Eviscerated, 2½-3 lbs. and 1½-2 lbs. each, all chicken individually wrapped in cryovac, packed in secured fiber cartons or wooden boxes, suitable for export

75,000 lbs. 2½-3 lbs.....@$33.00

25,000 lbs. 1½-2 lbs.....@$36.50

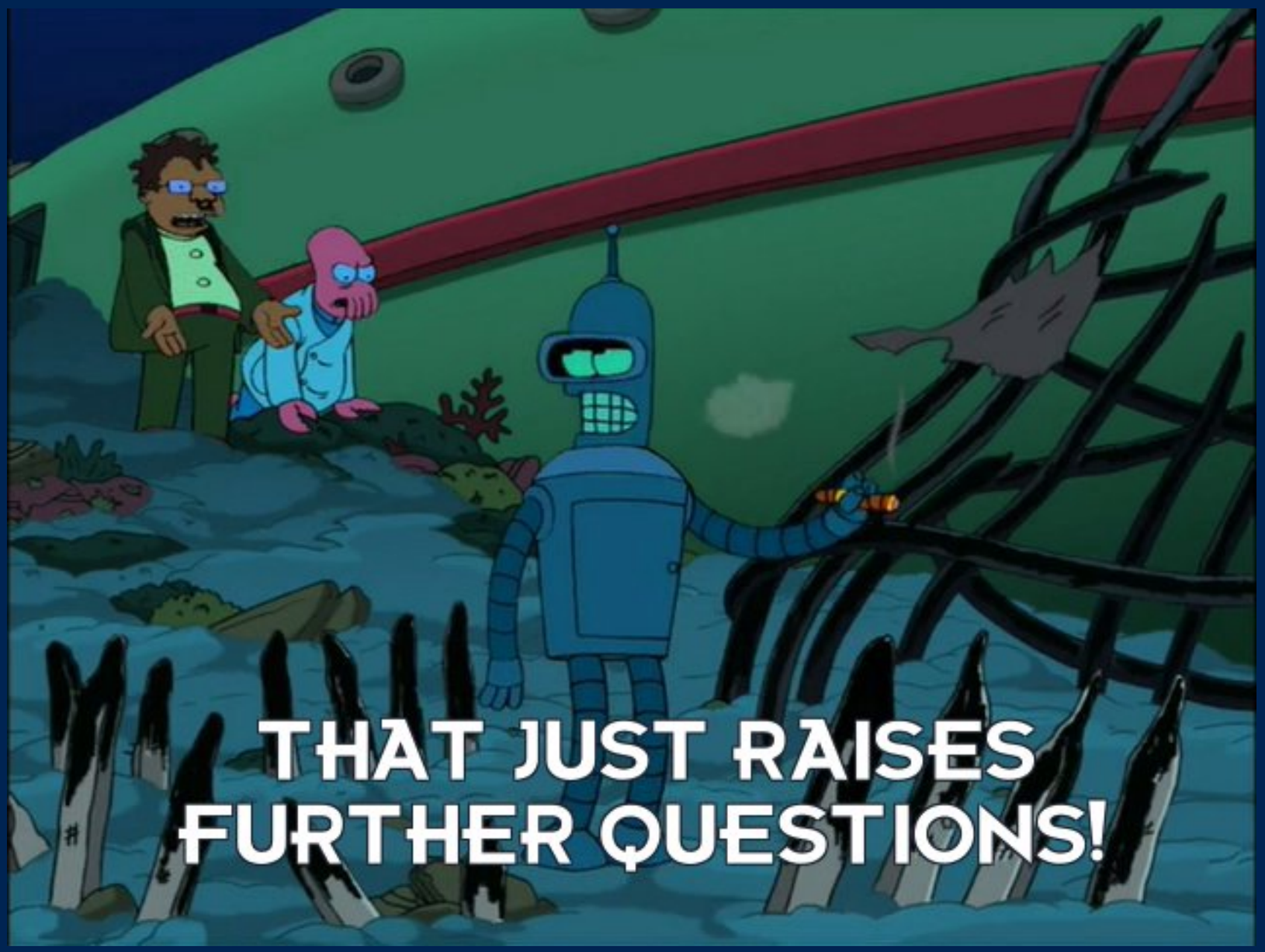
per 100 lbs. FAS New York, scheduled May 10, 1957 pursuant to instructions from Penson & Co., New York."

Frigaliment Importing Co. v. B.N.S. Int'l Sales Corp., 190 F. Supp. 116 (S.D.N.Y. 1960)

What is “chicken”?

- Buyer (plaintiff): “a young chicken suitable for broiling and frying”
- Seller (defendant): “any bird of that genus”
- The court examines the parties’ negotiations (in English and German), dictionaries, trade usage, and USDA regulations; finds evidence on both sides, and concludes, “For plaintiff has the burden of showing that ‘chicken’ was used in the narrower rather than in the broader sense, and this it has not sustained.”

What if the contract said “young chicken suitable for broiling”?



What’s “suitable”? What’s “broiling”?

An inescapable problem

- The meaning of natural language is social
  - Not just what the speaker intended
    - Even "objective" sources like dictionaries depend on how people actually use words
- Since the legal effect of a contract depends on the interpretation of its terms …
  - The meaning of a contract is a *social fact*

Smart contracts

An escrow contract

SAMPLE COPY

ESCROW AGREEMENT

Agreement made this _____ day of _____, 19___, by and between _____ and _____ as Escrow Agent (the "Escrow Agent").

WITNESSETH:

Pursuant to Section 513( ) of the Delaware Insurance Code, _____ is required to maintain on deposit in the State of Delaware for the protection of all its policyholders wherever located, except the deposits required by Delaware statute to be maintained solely for the benefit of Delaware policyholders, cash or cash equivalents in an amount and in a manner specified by the said Code and the Insurance Commissioner of the State of Delaware. It is intended that the deposit made and maintained pursuant to this Escrow Agreement satisfy the requirements of said Code and said Insurance Commissioner.

Now, THEREFORE, in consideration of the premises and the covenants herein contained, the parties hereto, intending to be legally bound, agree as follows:

(1) _____hereby agrees to deposit with the Escrow Agent cash or cash equivalents of a kind and of a value or amount sufficient to satisfy the applicable deposit requirements of the Delaware Insurance Code. This deposit (hereinafter referred to as the "Escrow Deposit") shall consist initially of the cash or cash equivalents listed on Schedule A attached hereto. Thereafter, the Escrow Deposit shall consist of such other cash or cash equivalents as _____ may deliver to the Escrow Agent with written instructions that the same shall be part of the Escrow Deposit. the Escrow Agent shall hold the Escrow Deposit subject to the terms and conditions of this Escrow Agreement. In no event, shall the Escrow Agent have any responsibility to ascertain whether the Escrow Deposit satisfies the applicable deposit requirements of the Delaware Insurance Code.

(2) All cash or cash equivalents delivered by _____ as part of the Escrow Deposit shall be registered in the name of _____ and shall be accompanied by irrevocable transfer powers executed in blank, sufficient to allow the Escrow Agent to sell or deliver such cash or cash equivalents in accordance with this Escrow Agreement. The Escrow Agent may cause said cash or cash equivalents to be registered in the name of the Escrow Agent or its nominee. The cash or cash equivalents so deposited shall at all times be kept separate and distinct from all other deposits, so that at all times they may be identified as belonging to _____.

(3) _____ shall receive from time to time payments of any interest payments or other distributions upon any government obligations, corporation obligations included as a part of said

An escrow “contract”

escrow.sol — Raw

```
pragma solidity ^0.4.21;

import "../../node_modules/zeppelin-solidity/contracts/token/ERC20/ERC20.sol";
import "../../node_modules/zeppelin-solidity/contracts/ownership/Ownable.sol";
import "../webshop/Webshop.sol";

contract Escrow is Ownable {
    enum PaymentStatus { Pending, Completed, Refunded }

    event PaymentCreation(uint indexed orderId, address indexed customer, uint value);
    event PaymentCompletion(uint indexed orderId, address indexed customer, uint value, PaymentStatus status);

    struct Payment {
        address customer;
        uint value;
        PaymentStatus status;
        bool refundApproved;
    }

    mapping(uint => Payment) public payments;
    ERC20 public currency;
    address public collectionAddress;
    Webshop public webshop;

    function Escrow(ERC20 _currency, address _collectionAddress) public {
        currency = _currency;
        collectionAddress = _collectionAddress;
        webshop = Webshop(msg.sender);
    }
```

Three motivations

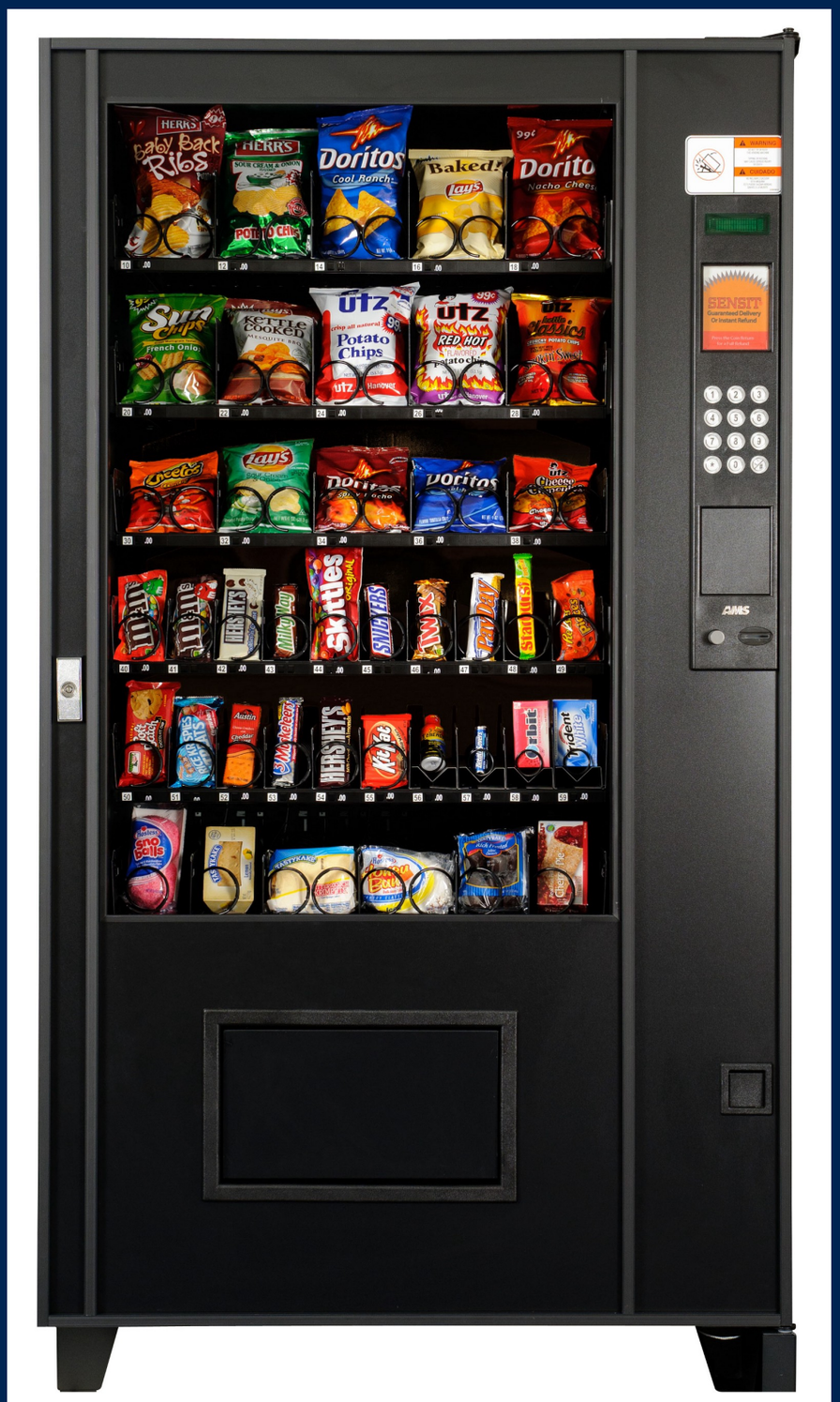
- *Ambiguity*: legal contracts are written in ambiguous natural languages
- *Corruption*: judges who interpret legal contracts can be threatened or bribed
- *Enforcement*: parties might be able to ignore a court's judgment against them

The argument for smart contracts

- Programming languages are *unambiguous*
- Computers are *incorruptible*
- Enforcement is *automatic*

Standard toy example

- The "dispense Skittles" logic is completely specified in code
- Threats literally mean nothing to the machine
- No money, no Skittles

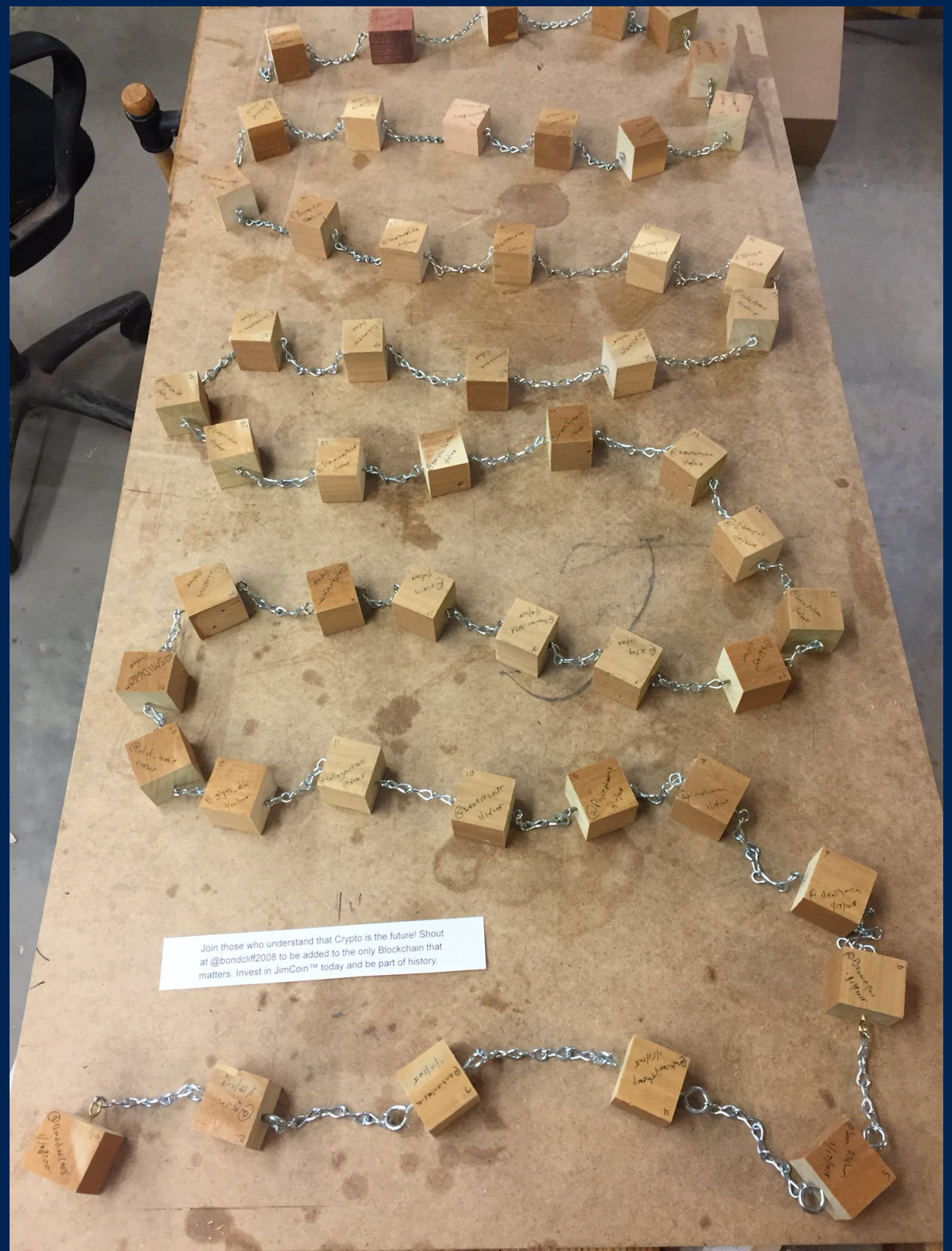


Scaling up the vending machine

- You can't inspect the machine's code
  - Smart contracts should be open source …
- You can threaten the machine's owner
  - Smart contracts should be decentralized …
- You can cut a hole through the window
  - Smart contracts should directly control resources …
- … solution: use a blockchain

Smart contracts on a blockchain

Blockchain?


Join those who understand that Crypto is the future! Shout at @bondcliff2008 to be added to the only Blockchain that matters. Invest in JimCoin™ today and be part of history.

Key blockchain ideas

- A transactional ledger that …
- … is cryptographically secure
- … has no centralized recordkeeper
- … uses incentives to ensure consensus

Bitcoin as a distributed ledger

- Every participant has copy of a shared ledger
  - Records transfers of Bitcoin among users
- New transactions accepted only if:
  - Cryptographically signed by the sender
  - Consistent with previous transactions

Bitcoin: achieving consensus

- Miners compensated with (a) mining rewards and (b) transaction fees
- Proof of work: rewards proportional to effort
  - Consequence: userbase too large to corrupt
- Participants must agree on transactions
  - Penalty for disagreement is incompatibility
  - Strong incentive to accept valid blocks

Smart contracts on a blockchain

- Specify a virtual machine (VM)
- Blockchain transactions update the VM
- Blockchain protocol forces VM consistency
- VM primitives can affect shared resources
- VM specification provides a programming model with desired security properties

Ethereum-style smart contracts

- To create: write a program (e.g. in Solidity)
  - Compile it to EVM bytecodes
  - Submit (and pay for) a transaction setting up the program as a smart contract on the shared VM
- To use: send a transaction *to the program*
  - This triggers state changes, resource distribution, more transactions, etc.

Smart contract utopia?

(1) Write the contract as a computer program

- The program is *unambiguous*

(2) Put the program on a blockchain

- Miners are collectively *incorruptible*

(3) Give it control of the relevant assets

- Results are enforced *automatically*

Or not

Other critiques

- Real-world contracting parties don't actually *want* perfect unambiguity
- One person's "corruption" is another person's "democracy" or "rule of law"
- Automated enforcement may be too efficient and have other bad consequences
- Mining is an environmental catastrophe
- There is a massive blockchain bubble

The claim for unambiguity

- The meaning of "chicken" is a *social fact*
  - There are dictionaries, patterns or speech, usage in multiple trades, etc.
  - Its meaning can vary and be misunderstood
- The meaning of `2+2` in Python is a *technical fact*
  - This expression will always evaluate to `4`
  - Its meaning never changes, and if you think it evaluates to `5` that is your mistake

Where does program meaning come from?

- Why *doesn't* `2+2` in Python evaluate to `5`?
- Not because that's what "`2+2`" inherently means
  - Any more than "chicken" inherently means any *gallus gallus domesticus*, even one that is wholly unsuitable for cooking
- In 1991, GvR picked + as the addition operator
  - He could have picked ++ instead

Usual sources of program meaning

- Use a program: a *reference implementation* whose behavior is by stipulation treated as correct
- Use natural language: a *specification* that defines the behavior of a correct implementation
- Use mathematics: a *formal semantics* that identifies programs with abstract entities

Three questions

- Where do these come from?
  - Some people got together to write them
- What makes one of them definitively correct?
  - Because people agree that it is
- What language are we running?
  - “Python” 2.7 is different from “Python” 3.6
- These questions can be answered only by reference to a community of programmers and users

Program meaning is a social fact, too

- Yes, `2+2` in Python is unambiguously `4`
- But that's only because Python users have already agreed on what "Python" is
- If they agreed differently, "Python" would be different, and so might `2+2`
- This happens *every time* there's a new version
- Technical facts depend on social facts!

Fixing program meaning

- A technical community agrees on a process for deriving a functional meaning from texts
- Developers implement that process on different computers, with different tools, etc.
- Most of the time, running a program on most implementations yields the same result
- We perceive as fixed technical facts the *successful* result of coming to a social consensus

Blockchain governance

Does this matter?

- We might be able to ignore all of this if smart-contract blockchains never had trouble
- But in fact, there are fights over the meanings of blockchain programs *all the time*

Example 1: Oracles

- How does a smart contract observe the world?
  - An *oracle* has to tell it what happened
  - E.g., a trusted party or a fixed data feed
- This is also a problem of ambiguity
  - The world is complex
  - Contract terms map ambiguously onto the world
  - The oracle resolves the ambiguity

Oracles and consensus

- What if the oracle is … corrupt?
  - Go back in time and choose a better oracle!
  - Consensus oracles seek correct agreement by multiple participants about the world
    - The *truth* is unobservable by the contract, so protocols typically give incentives to *agree*

Two takeaways

- The obvious one:
  - An oracle's *resistance to corruption* is only as good as its consensus mechanism
- The subtle one:
  - An oracle's *ability to resolve ambiguity* is only as good as its consensus mechanism

Example 2: Upgrades

- In 2017, Bitcoin upgraded to implement "segregated witness"
  - Some data moved out of the blockchain, effectively allowing more transactions
- The blockchain before the upgrade and the blockchain after have *different semantics*
  - Some transactions that were valid under the old rules are invalid under the new ones

What do you mean "Bitcoin upgraded?"

- Bitcoin doesn't upgrade itself
  - Bitcoin's *users* collectively upgraded
    - A critical mass activated segregated witness, and everyone else went along
- Like moving from Python 3.6 to Python 3.7

Consensus all the way down

- The "Bitcoin blockchain" exists only because and only insofar as people agree on what it is
- Bitcoin's consensus protocols help coordinate and incentivize that agreement
- But the protocols cannot establish their own rule of recognition … a user community can always collectively change or ignore them
- This is what happens in an upgrade

Example 3: Bitcoin Cash

- A long-running dispute over Bitcoin block size caused some users to fork Bitcoin Cash
  - Bitcoin has ~1MB blocks
  - Bitcoin Cash had 8MB blocks (now 32MB)
- The two blockchains have different semantics
  - Is a block valid? The question can't be answered without specifying *by whom*

Forks and ambiguity

- Forks are consensus failures
  - Each blockchain by itself achieves local consensus, but there is no global consensus
- Forks create explicit ambiguity
  - Each blockchain by itself is “unambiguous” but the *choice of blockchains* creates ambiguity
- These are inextricably linked

Upgrades and forks

- Literally anything on a blockchain is subject to the latent ambiguity that the blockchain itself could be upgraded out from underneath it
- Whether this happens is inherently political
  - The anti-change faction yields: upgrade
  - The pro-change faction yields: status quo
  - Neither faction yields: fork

Example 4: The DAO

"The terms of The DAO Creation are set forth in the smart contract code existing on the Ethereum blockchain at `0xbb9bc244d798123fde783fcc1c72d3bb8c189413`. Nothing in this explanation of terms or in any other document or communication may modify or add any additional obligations or guarantees beyond those set forth in The DAO's code."

The DAO

- April 2016: The DAO begins crowdfunding for a democratic online venture capital fund
- May 2016: 11,000+ investors put $150M+ of assets into The DAO
- June 2016: An anonymous hacker drains $50M of the assets into their own account

Ethereum Classic

- Following the DAO hack, Ethereum upgraded to a new version that *specifically unwound the DAO transactions*
- Not everyone was happy with this, and some users were unhappy enough to fork Ethereum Classic, which didn't have this "upgrade"
- The two blockchains have different semantics

The DAO (legal) contract

- The English phrase “the smart contract code existing on the Ethereum blockchain at `0xbb9bc244d798123fde783fcc1c72d3bb8c18941`” is ambiguous
- “the Ethereum blockchain” does not uniquely refer: do you mean ETH or ETC?
- It uniquely referred when the contract was drafted, but no longer

Where to go from here?

All is not lost

- Smart contracts are based on social facts
  - Social facts are empirically contingent: they are always open to contestation and change
- Legal contracts are based on social facts, too
  - And a lot of the time, they work just fine!
- Smart contracts *cannot* be perfectly unambiguous
  - But they can be unambiguous enough

Focus on the consensus

- Blockchains make consensus explicit
  - The mechanism that holds them together is the protocol for agreeing on the next block
- Put another way, every smart contract is vulnerable to a “51%” attack
  - Where the “attack” could happen through persuasion as well as raw computational power

The contractual is political

- A blockchain whose governance fails will collapse, fork, be hijackable, etc. —
  - All these threaten the smart contracts on it
- Contract law depends on social institutions that establish and limit government
  - Smart contract code depends on social institutions that establish and limit blockchain governance
- There is no escape from politics

Good blockchain citizenship

- (Practically, not perfectly) unambiguous smart contracts require correct, stable blockchains
- Blockchain correctness and stability require a good blockchain community
  - Correctness comes from making good changes
  - Stability comes from not making bad ones
- Not just consensus protocols — it's also the mailing lists, the depth of developer knowledge, user commitment to long-term health, etc.

Conclusion


Non est potestas Super Terram quæ Comparetur ei Iob. 41. 24.
LEVIATHAN
Or
THE MATTER, FORME
and POWER of A COMMON-
WEALTH ECCLESIASTICALL
and CIVIL.
By THOMAS HOBBES
of MALMESBVRY.
London
Printed for Andrew Crooke
1651.

Blockchains are made out of people

Questions