A workshop in Smart Contract Programming

Shelly Grossman & Mooly Sagiv

Tel Aviv University

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

http://www.cs.tau.ac.il/~msagiv/courses/scontracts.html
msagiv@acm.org
shellygr151@gmail.com

Acknowledgments

- Slides by Andrew Miller (UMD)
- Spring 2018 (2nd half): ECE398SC: Smart Contracts and Blockchain Security at UIUC
- **CSCI-GA.3033-019: Cryptocurrencies and decentralized ledgers**
- **http://cs251crypto.stanford.edu/18au-cs251**

Tentative Schedule

- March 3: Overview and Introduction and division into 2 person teams
- March 17: Project Selection by email/separate meetings
- TBD: Project Presentation

Course Goals

- Learn how to program dapps
- Understand one interesting application
- Improve programming skills
  - Critical thinking
- Learn about formal verification

The Times 03/Jan/2009 *Chancellor on brink of second bailout for banks.*

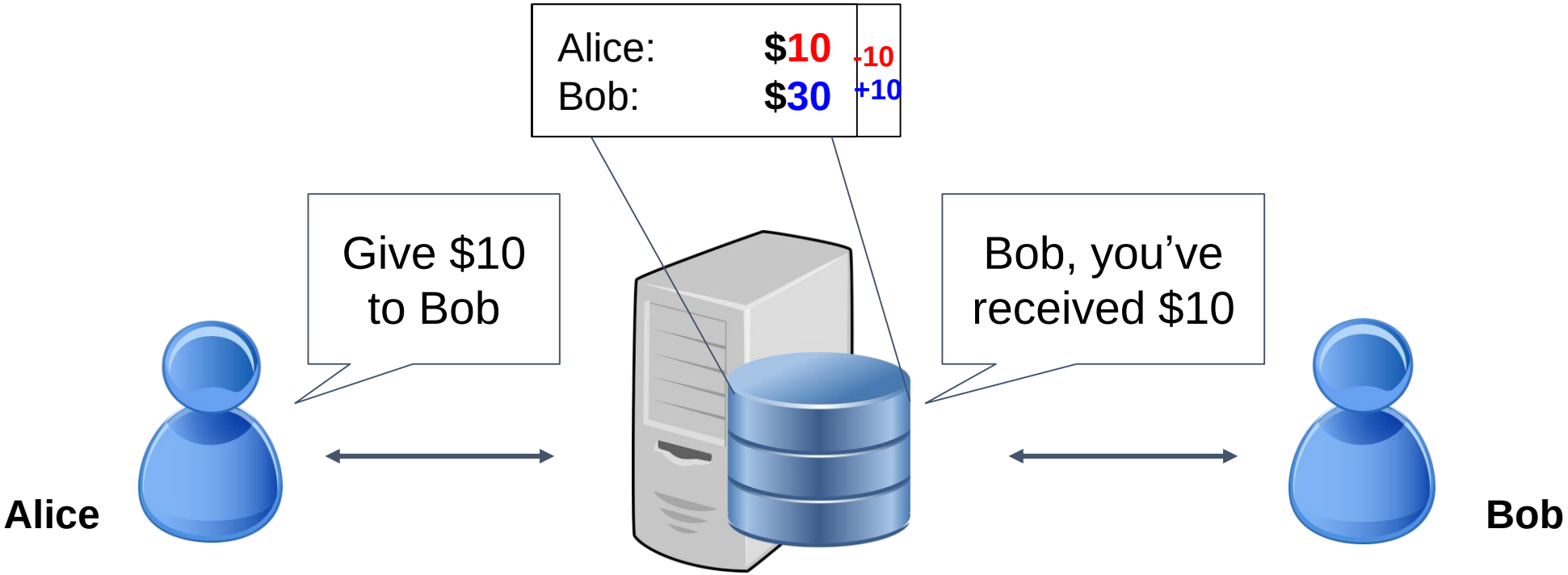
Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of

```
bitcoin-0.1.0.rar
bitcoin-0.1.0.tgz
```

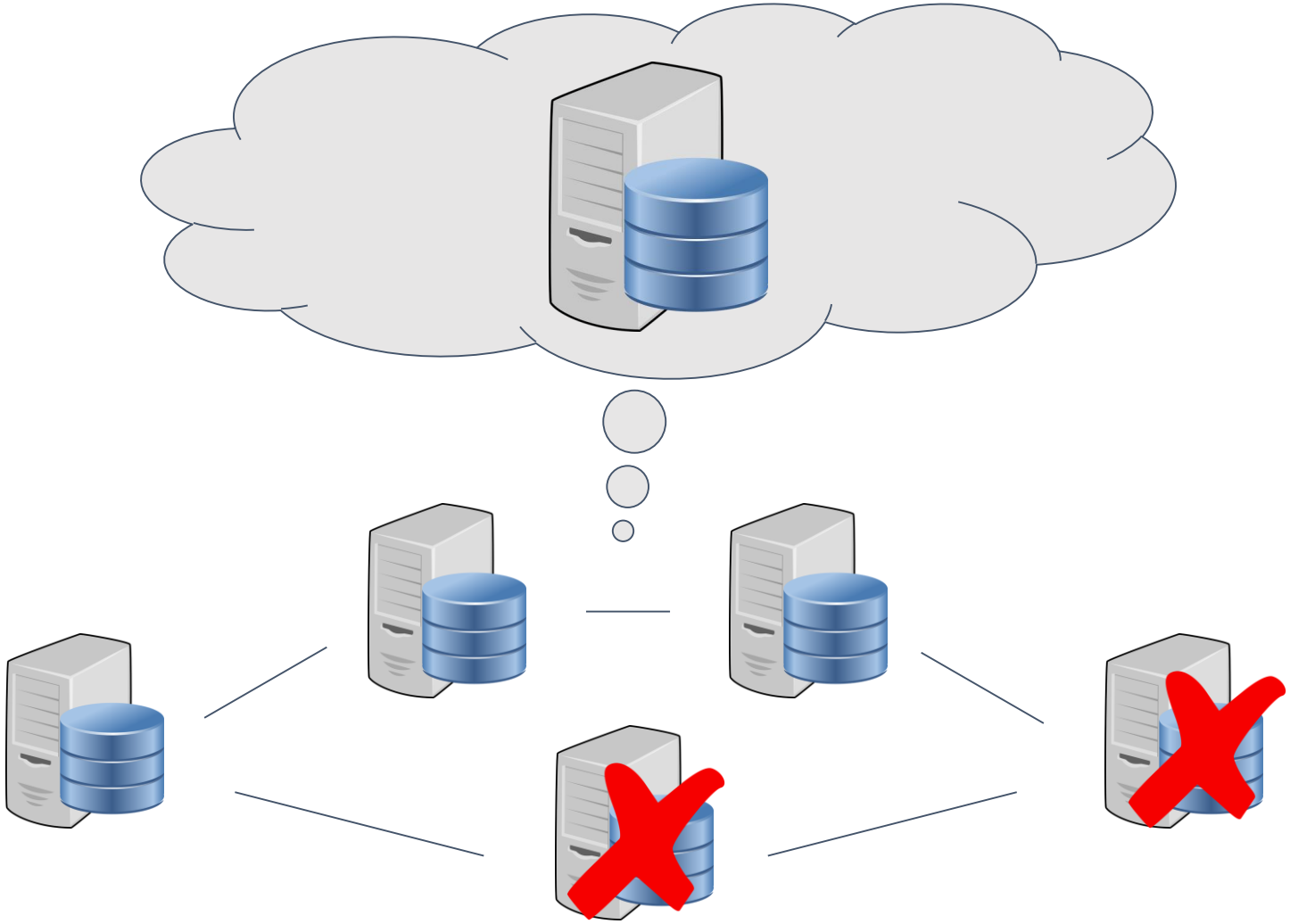
Virtual currencies are implemented on top of a ***shared database***



Desired security properties:

- Consistency
- Availability
- Application-defined access controls (for writes/updates)
- Privacy

A blockchain is a *Distributed Trusted Computer*

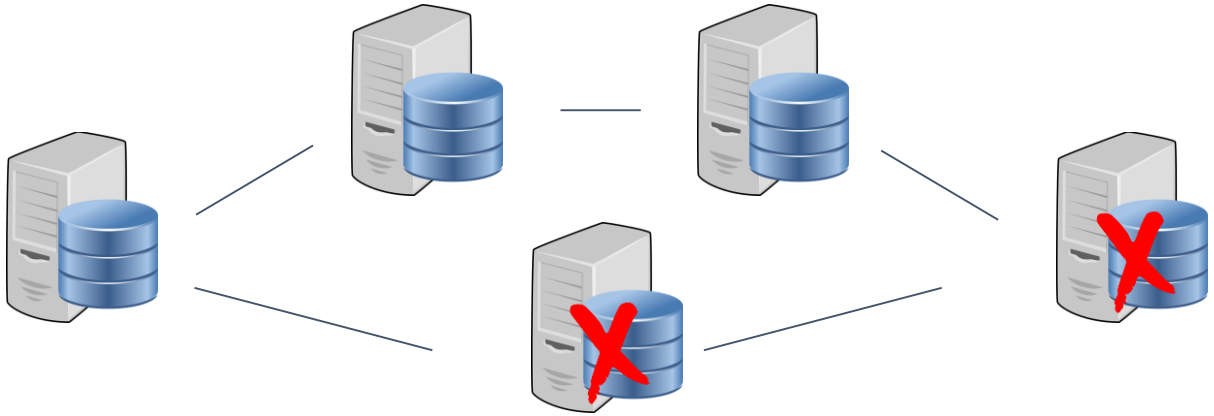


Ordinary databases:
- distributed within one company
- distributed for performance and availability

Blockchain databases:
- distributed across multiple entities
- distributed for privacy and security against attacks

Permissioned Blockchains

AKA “Consortium blockchain”
Nodes are run by well-known,
mostly trusted entities

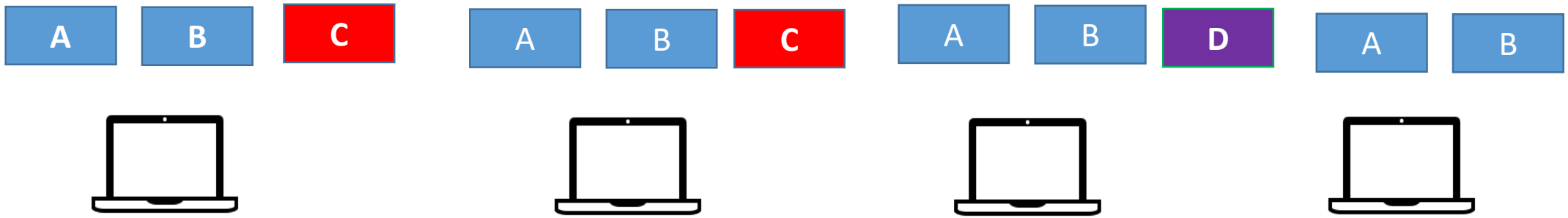


Public Blockchains

Open for participation by anyone

Replicated Transaction Log

- The information is stored at every node
- A consensus algorithm builds the log
- Distributed database builds replicated logs



More generally: “programmable money”

Etherscan
The Ethereum Block Explorer

Search for Account, Tx

Contract Accounts

A total Of 253479 contracts found (11,485,845.474 Ether)

Displaying the last 100000 records only

Digital currency is just one application on top of a blockchain

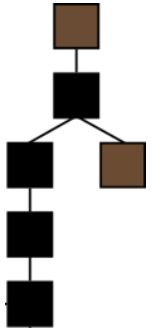
Decentralized Consensus
"Blockchain"

Money

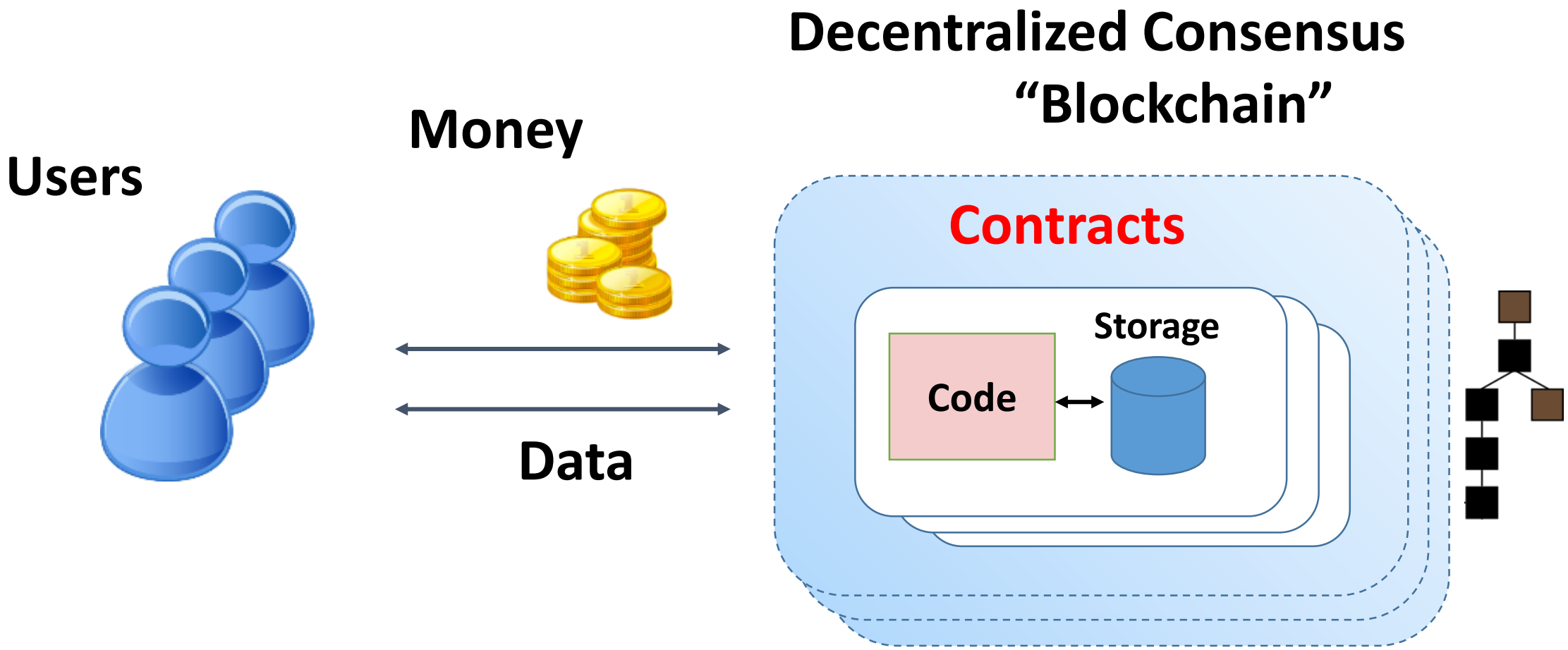
Users

Account Balances

Alice:	฿10
Bob:	฿15
Carol:	฿120



Smart Contracts: user-defined programs running on top of a blockchain



Smart Contract Example (very high level)

If GOOG rises to $1,000 by 30 June 2015, assign 10 shares from Alice to Bob and pay Alice $10,000

Other examples abound:

Auctions, elections, lotteries, escrow, ...

“Smart contracts” conceptualized by Szabo in 1994

A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs.

-Nick Szabo “The Idea of Smart Contracts”

A “dumb contract” example

Alice will reveal to Bob a value x such that SHA-256(x) = 0x2a...

In exchange, Bob will give Alice $10 in cash.

If Alice does not give Bob by July 1, 2018, then she will pay a penalty of US$1 per day that she is late, up to US$100.

Signed: Alice Bob

Traditional contracts vs. smart contracts

	Traditional	Smart
specification	Natural language + “legalese”	Code
identity & consent	Signatures	Digital signatures
dispute resolution	Judges, arbitrators	Decentralized platform
nullification	By judges	????
payment	Carried out by parties separately	built-in
escrow	Trusted third party, settled in $	built-in

Ordinary contracts between two parties

- Civil courts are available to mediate, but typically don't get involved

- Parties responsible for their own accounting:
  "Double entry accounting"

Third-party services

E.g., Paypal, Kickstarter, ….Piazza?

- Transactions are automatically processed by the third party

- Formal intermediary to finance network (help with taxes)

- Settles disputes through mediation

Smart Contract dApps "decentralized apps"

- Transactions typically go through the blockchain

- App developers do not receive custody of funds

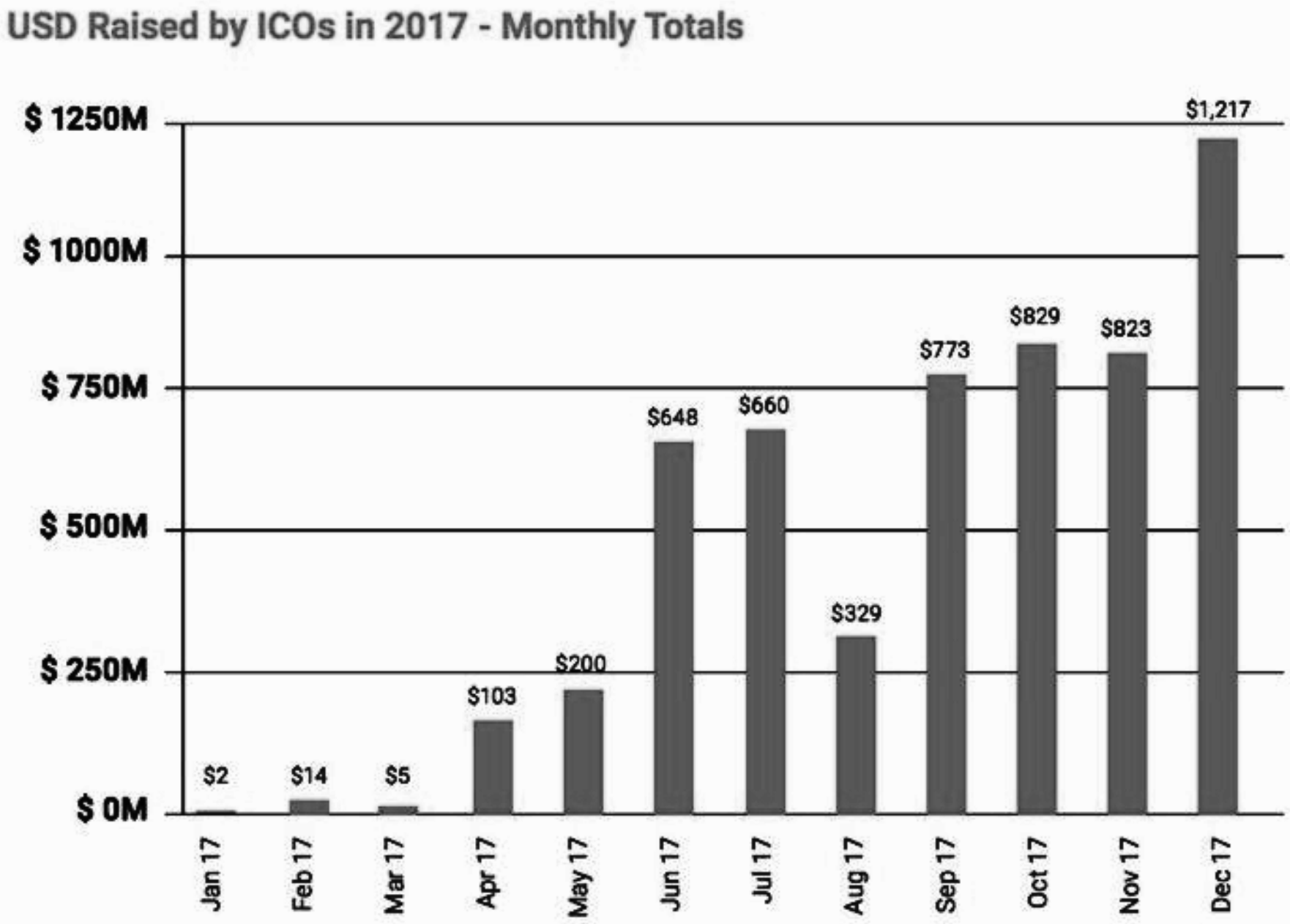
- Developers limited in ability to change the rules

- Settles automatically via cryptocurrency

- Auditing may be simplified by public transparent log

Ethereum

- A decentralized platform that runs smart contracts
- Proposed in late 2013 by Vitalik Buterin
- Released 2015
- Supports Turning complete smart contracts (Solidity)
- A virtual machine for cryptocurrency (Ethereum Virtual Machine)
  - Creating new currencies
  - Guaranteeing certain currency consistency
- But has all bad features of computer programs (DAO, Parity, …)



Source: Business insider

U.S. SECURITIES AND EXCHANGE COMMISSION

https://www.sec.gov/ICO

Contract programming model

```
pragma solidity ^0.4.19;
contract TestContract {

    // declare storage
    int[] myStorage1;

    // define functions
    function testFunction() {
        // Code goes here
    }

    // ...
}
```

- Contract class

  Create an object of this class by making a transaction

- Define functions you can call

Ethereum's timeline has been pocked by failures caused by buggy and insecure smart contracts

'$300m in cryptocurrency' accidentally lost forever due to bug

User mistakenly takes control of hundreds of wallets containing cryptocurrency Ether, destroying them in a panic while trying to give them back

KLINT FINLEY BUSINESS 06.18.16 04:30 AM

A $50 MILLION HACK JUST SHOWED THAT THE DAO WAS ALL TOO HUMAN

CRYTPO ENTOMOLOGY

A coding error led to $30 million in ethereum being stolen

THE PROBLEM

Massive Losses due to Bugs

How to steal $50M – the DAO bug

```
DAO::withdraw(to) {
 if shares[to] > 0 {
  transferTo(to, shares[to]);
  shares[to] = 0;
 }
}
```

coins[Thief]=7

shares[Thief]=100

How to steal $50M – the DAO bug

```
DAO::withdraw(to) {
  if shares[to] > 0 {
    transferTo(to, shares[to]);
    shares[to] = 0;
  }
}
```

coins[Thief]=7

shares[Thief]=100

How to steal $50M – the DAO bug

```
DAO::withdraw(to) {
 if shares[to] > 0 {
  transferTo(to, shares[to]);
  shares[to] = 0;
 }
}
```

```
Thief::uponTransfer(a) {
 DAO::withdraw(Thief)
}
```

coins[Thief]=107

shares[Thief]=100

How to steal $50M – the DAO bug

```
DAO::withdraw(to) {
  if shares[to] > 0 {
   transferTo(to, shares[to]);
   shares[to] = 0;
  }
}


Thief::uponTransfer(a) {
 DAO::withdraw(Thief)
}
```

coins[Thief]=107

shares[Thief]=100

How to steal $50M – the DAO bug

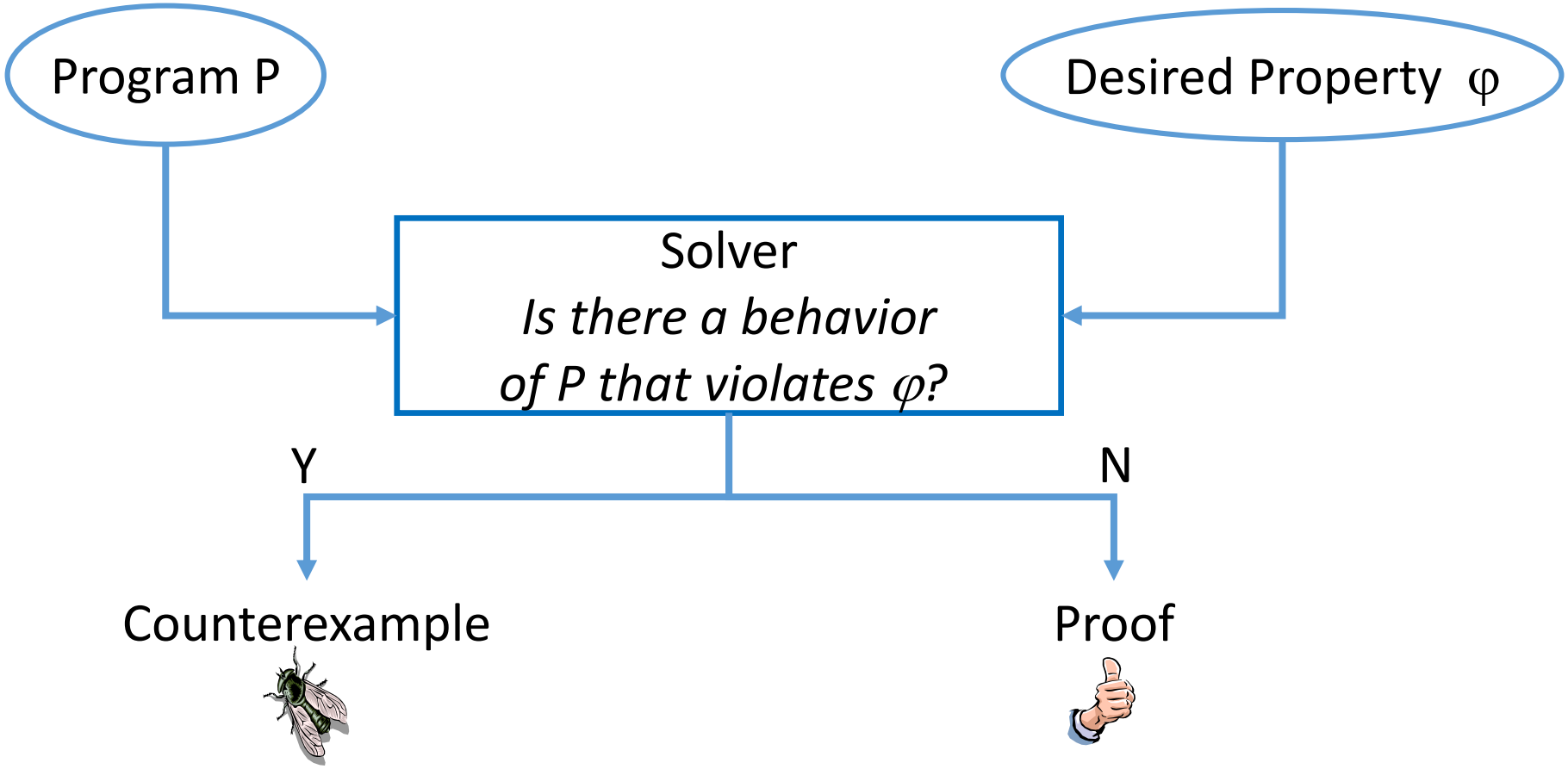
```
DAO::withdraw(to) {
  if shares[to] > 0 {
   transferTo(to, shares[to]);
   shares[to] = 0;
  }
}


Thief::uponTransfer(a) {
 DAO::withdraw(Thief)
}
```

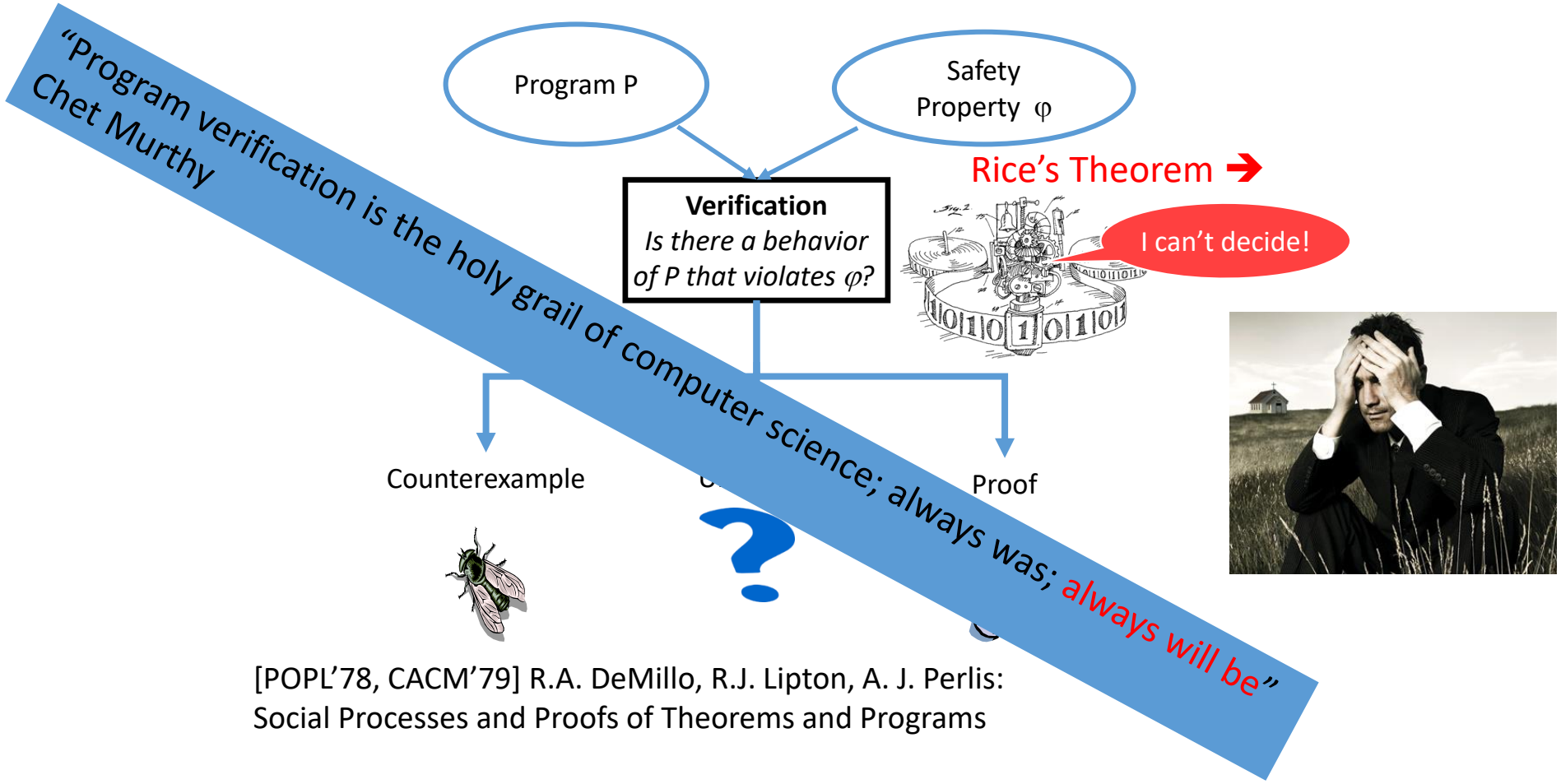
coins[Thief]=207

shares[Thief]=100

Automatic software verification



Disillusionment in program verification 80's



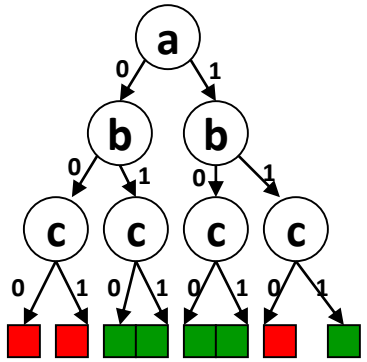
[POPL'78, CACM'79] R.A. DeMillo, R.J. Lipton, A. J. Perlis: Social Processes and Proofs of Theorems and Programs

Challenges in program verification

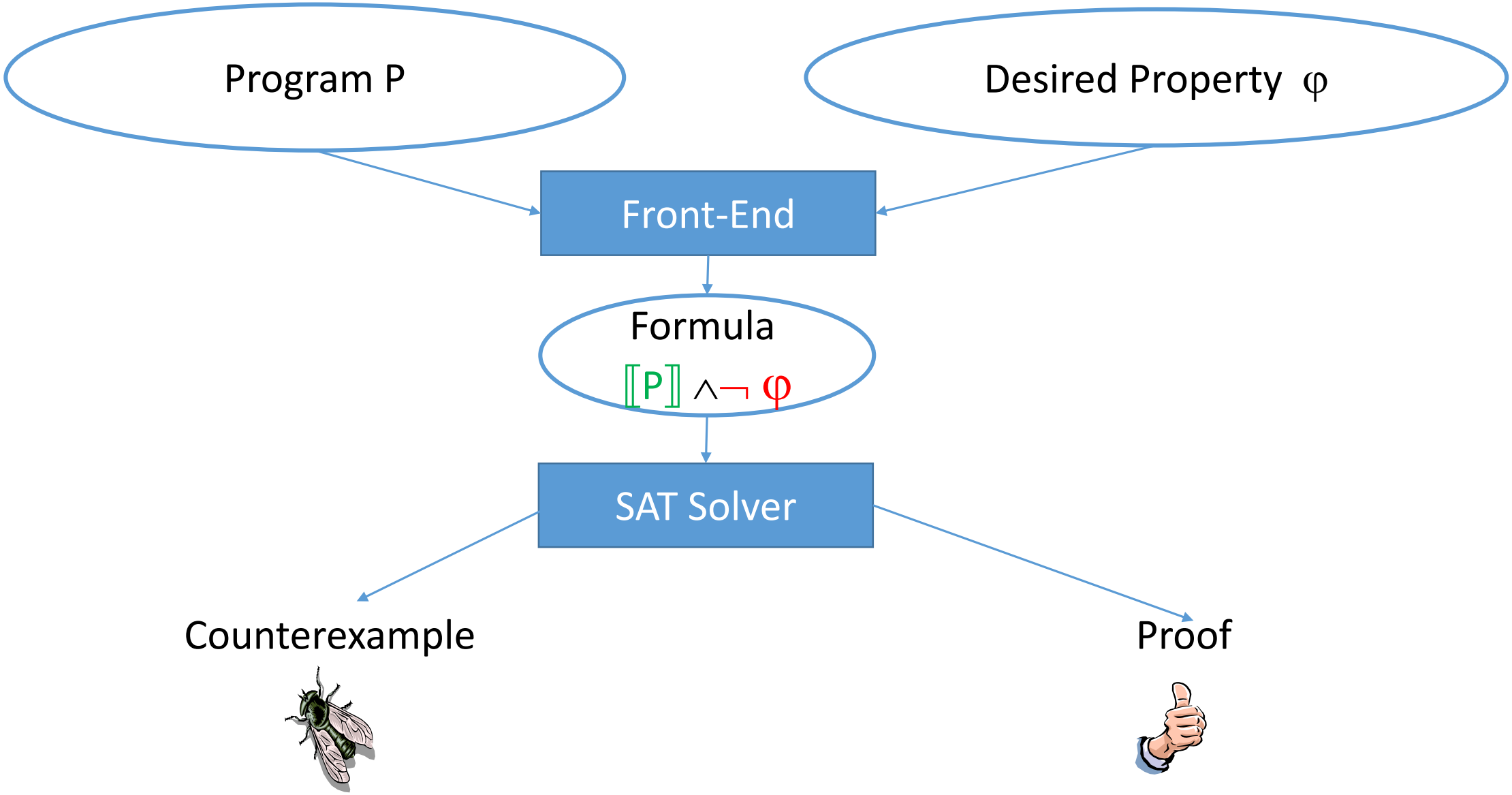
- Specifying program behavior
- Complexity of program verification
  - The halting problem
  - Rice theorem
  - The ability of simple programs to represent complex behaviors
- Complexity of realistic systems
  - Huge code
  - Heterogeneous code
  - Missing code

The SAT Problem

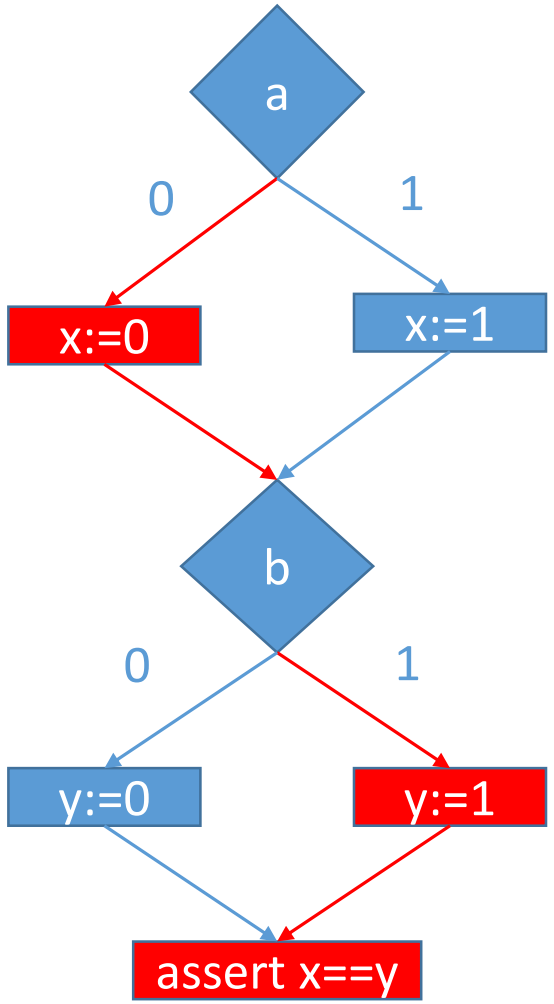
- Given a propositional formula (Boolean function)
  - φ **= (a $\vee$ b) $\wedge$($\neg$ a$\vee$ $\neg$b $\vee$ c)**
- Determine if φ is satisfiable
  - Find a satisfying assignment or report that such does not exist
- For *n* variables, there are 2^n possible truth assignments to be checked
- Tools exist: Z3, Yices, CVC4, …



Verification by reductions to SAT



Verification by reduction to SAT



SAT Query:

$((a \wedge x) \vee (\neg a \wedge \neg x))$

$\wedge$

$((b \wedge y) \vee (\neg b \wedge \neg y))$

$\wedge$

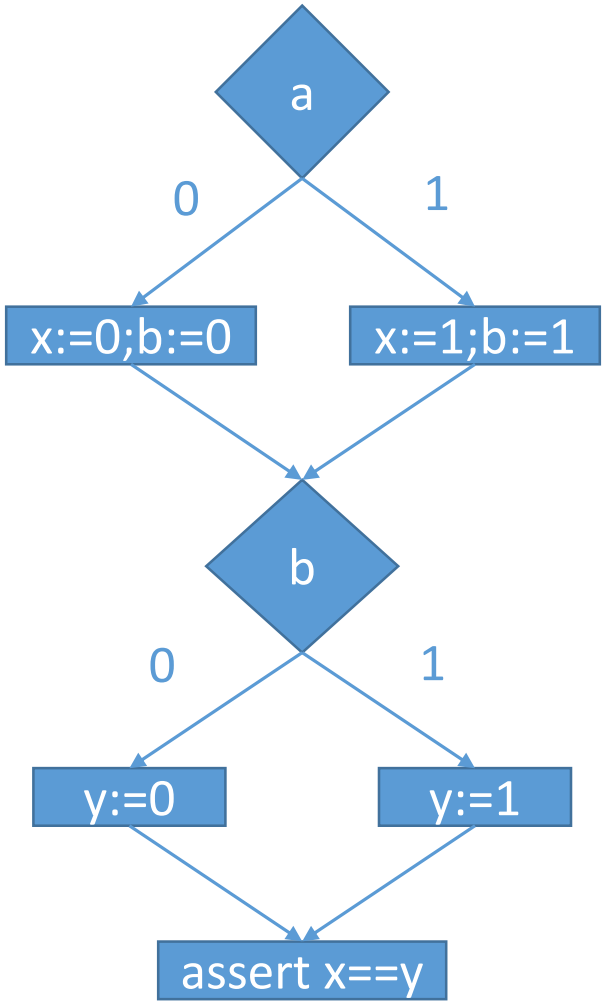
$((x \wedge \neg y) \vee (\neg x \wedge y)$

?

SAT Answer:

Satisfiable by a=0, b = 1

Verification by reduction to SAT



SAT Query:

$((a \wedge x \wedge b) \vee (\neg a \wedge \neg x \wedge \neg b))$

$\wedge$

$((b \wedge y) \vee (\neg b \wedge \neg y))$

$\wedge$

$((x \wedge \neg y) \vee (\neg x \wedge y)$

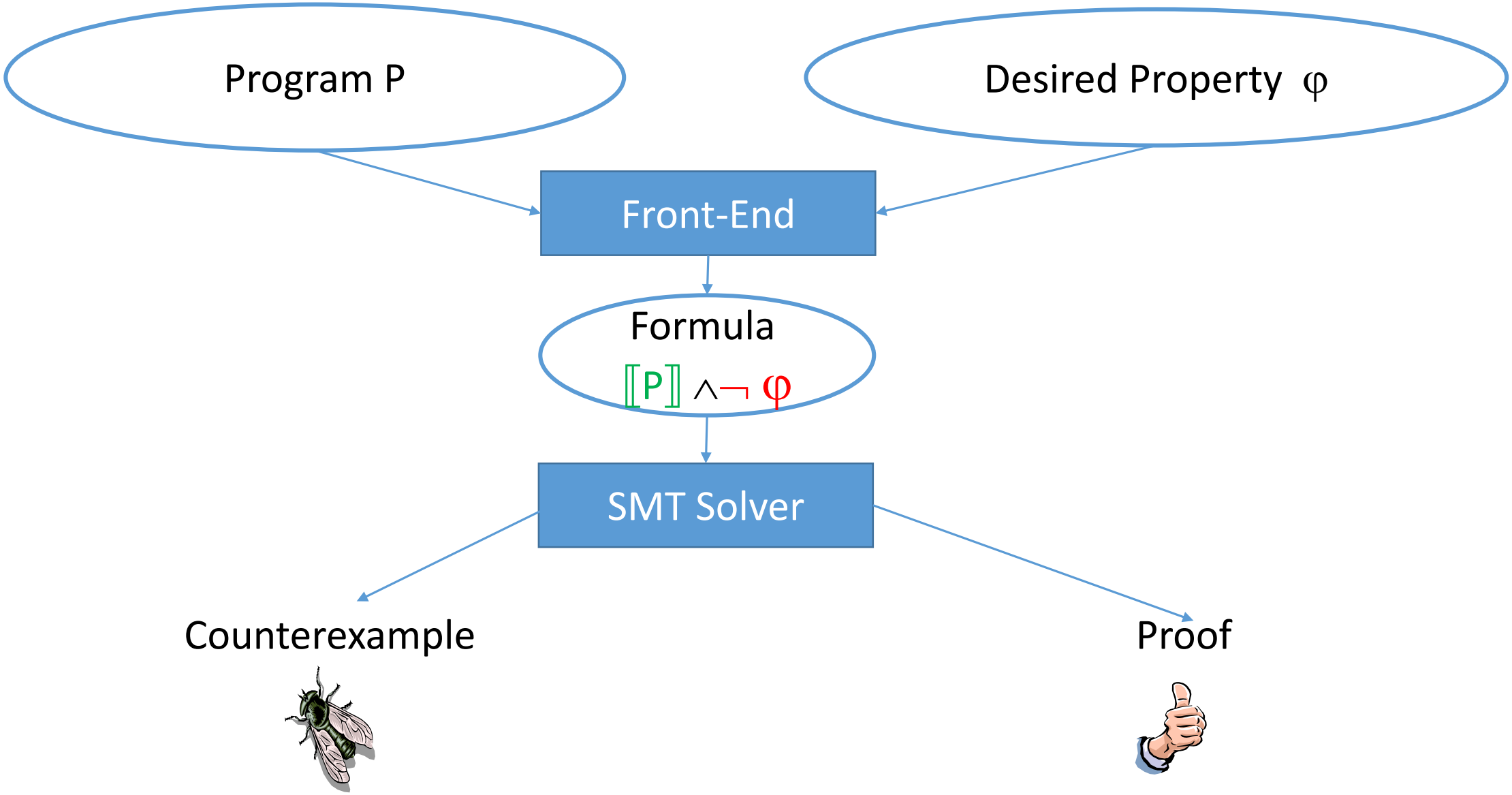
?

SAT Answer:
Unsatisfiable

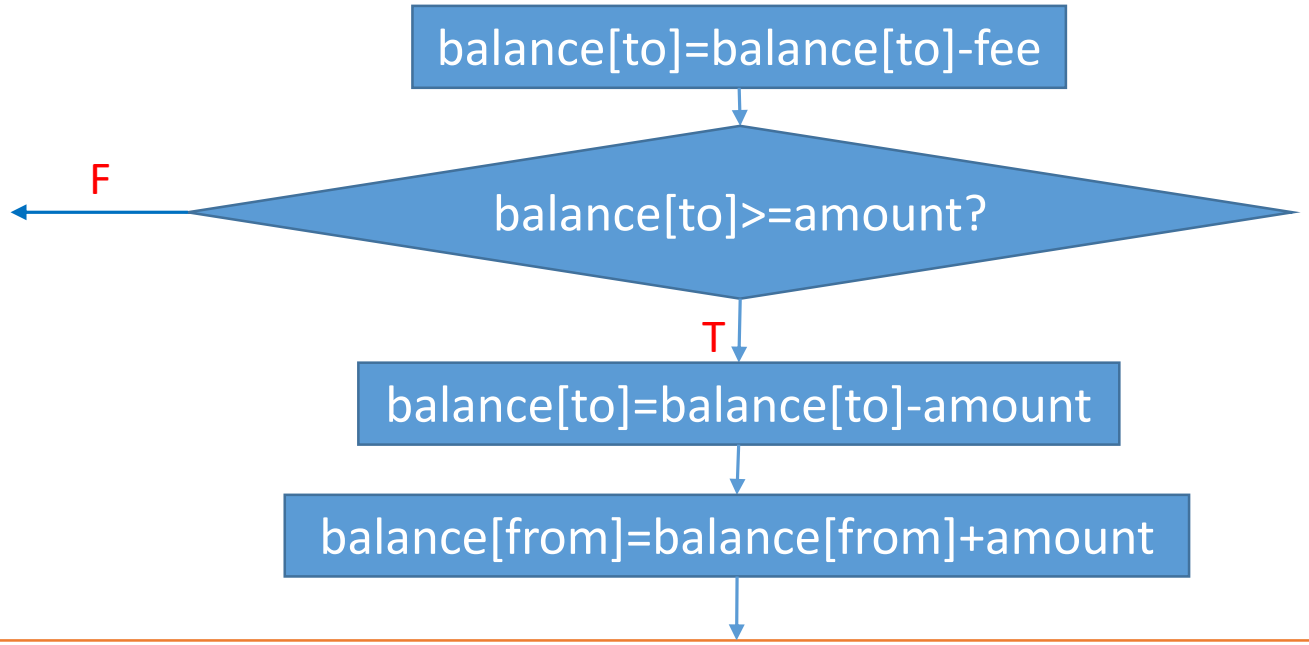
The SMT(Sat Modulo Theory) Problem

- Given a ground first order formula over theories(Boolean function)
  - $\varphi = \exists x, y: 2x + y \geq 5 \wedge y < 3$
- Determine if φ is satisfiable
  - Find a satisfying assignment or report that such does not exist
- Satisfiability becomes harder
- But tools exist: Yices, Z3, CVC, …

Verification by reductions to SMT



Simple Example Token (buggy)

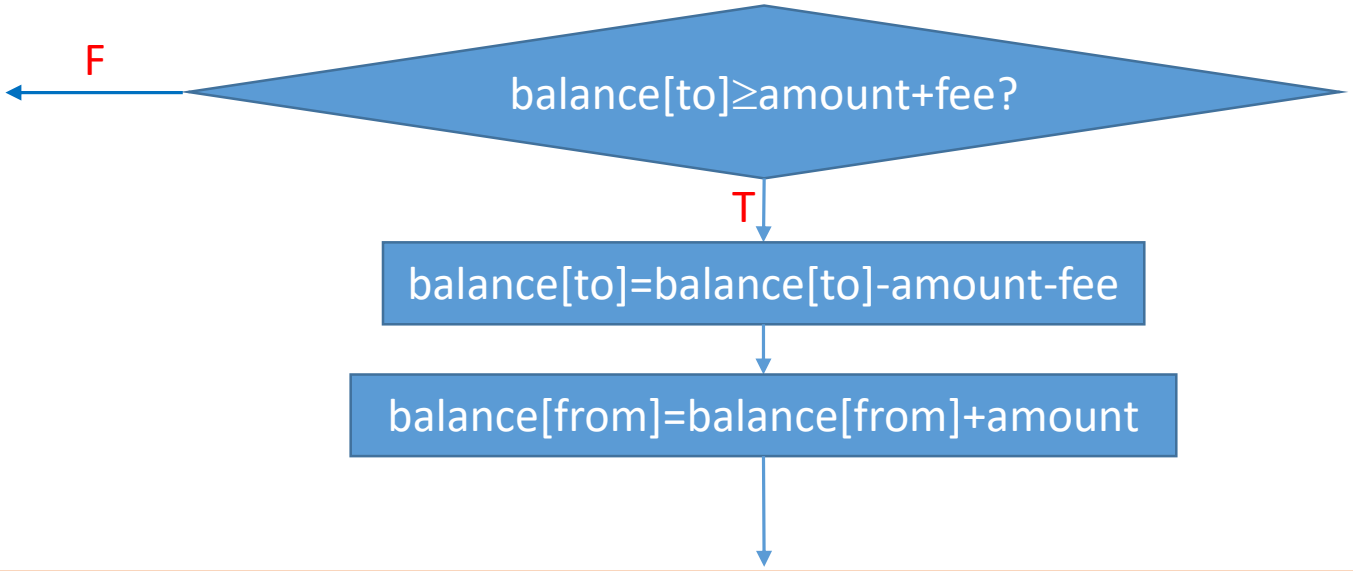


SMT Answer:
Satisfiable by
balance[to]=10,
fee=5,
amount=6
balance[from]=100

assert

$$\forall x.\ (x \neq to \wedge x \neq from) \Rightarrow b'[x]=b[x] \wedge$$
$$b[to] \geq amount+fee \Rightarrow (b'[to]=b[to]-amount-fee \wedge b'[from]=b[from]+amount) \wedge$$
$$b[to]<amount+fee \Rightarrow (b'[to]=b[to] \wedge b'[from]=b[from])$$

Simple Example Token (corrected)



SAT Answer:
Unsatisfiable

assert

$$\forall x.\ (x \neq to \wedge x \neq from) \Rightarrow b'[x]=b[x] \wedge$$
$$b[to] \geq amount+fee \Rightarrow (b'[to]=b[to]-amount-fee \wedge b'[from]=b[from]+amount) \wedge$$
$$b[to]<amount+fee \Rightarrow (b'[to]=b[to] \wedge b'[from]=b[from])$$

Homework

- https://solidity.readthedocs.io/en/v0.5.4/solidity-by-example.html
- Select an open source project
  - https://github.com/bkrem/awesome-solidity
  - https://github.com/Neufund/ico-contracts
  - https://github.com/OpenZeppelin/openzeppelin-solidity
  - https://github.com/gnosis/pm-contracts
  - https://github.com/ColuLocalNetwork/CLN-solidity
  - https://github.com/trending/solidity
  - https://github.com/aragon/aragonOS
  - https://github.com/trusttoken/trueUSD
  - https://github.com/decentraland/mana
  - https://github.com/DRI-network/RICO

Selecting a project

Potential Ethereum Applications

https://github.com/ethereum/wiki/wiki/White-Paper#applications

- Tokens
- Lotteries
- Cryptocurrency exchanges
- Marketplaces
- Insurance / hedging
- Supply-chain management
- “Self-sovereign” identity management
- Sharing economy

And many more!