SMART CONTRACTS ARE NEITHER

SMART NOR CONTRACTS

FRITZ HENGLEIN
UNIVERSITY OF COPENHAGEN

Cyber Security, Privacy and Blockchain
High Tech Summit, DTU
2017-09-21

FRITZ HENGLEIN

- Professor of programming languages and systems
  - Foundations, techniques, algorithmics, language design
  - Enterprise systems, healthcare, finance, blockchain, contract management
- Director, Research center for high-performance computing for finance (HIPERFIT.dk)
- Steering committee chair, Innovation network for Finance IT (CFIR.dk)
- Mostly academic, some industrial lab/start-up experience

Technische Universität München, Rutgers University;
New York University, Utrecht University, University of Copenhagen, IT University of Copenhagen;
IBM Research, Cornell University, University of New South Wales, University of Oxford;
Hafnium Research ApS, Deon Digital AG/Deon Digital Denmark A/S, …

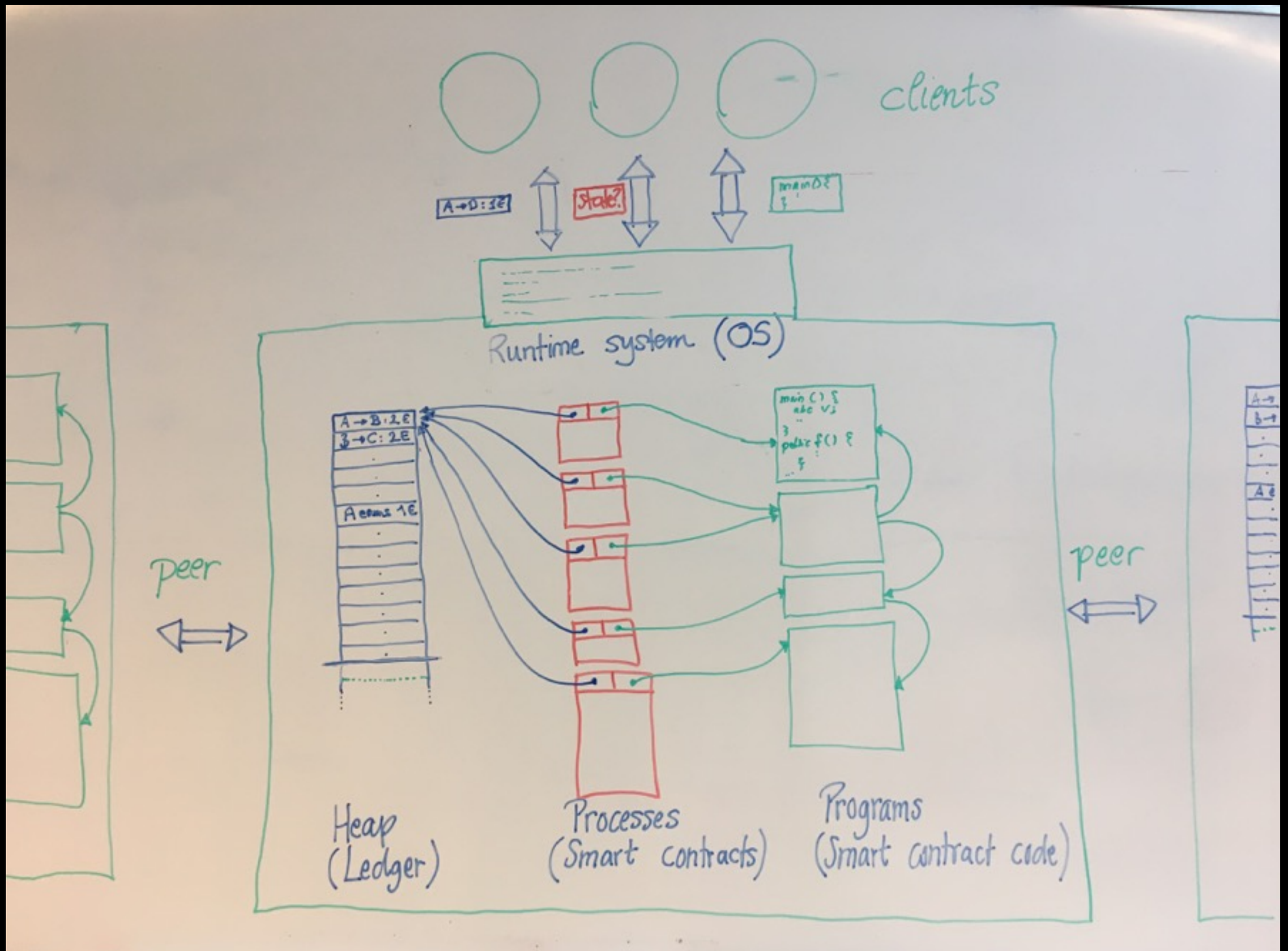
A CRASH SLIDE ON BLOCKCHAIN AND SMART CONTRACTS

SMART TERM	WHAT IT ACTUALLY MEANS
BLOCKCHAIN	**DISTRIBUTED** APPEND-ONLY TRANSACTION LOG (LEDGER)
SMART CONTRACT (CODE)	CLASS (IN JAVA-LIKE LANGUAGE)
SMART CONTRACT (EXECUTING)	PROCESS (OBJECT [= CLASS INSTANCE])
OBJECT MESSAGES	ORDINARY MESSAGES **LINEAR RESOURCE TRANSFERS**

A CRASH SLIDE ON BLOCKCHAIN AND SMART CONTRACTS

SMART TERM	WHAT IT ACTUALLY MEANS
BLOCKCHAIN	**DISTRIBUTED** APPEND-ONLY **LINEAR RESOURCE TRANSFER** LOG
SMART CONTRACT (CODE)	CLASS (IN JAVA-LIKE LANGUAGE)
SMART CONTRACT (EXECUTING)	PROCESS (OBJECT [= CLASS INSTANCE])
OBJECT MESSAGES	ORDINARY MESSAGES **LINEAR RESOURCE TRANSFERS**

BLOCKCHAIN SYSTEM ARCHITECTURE



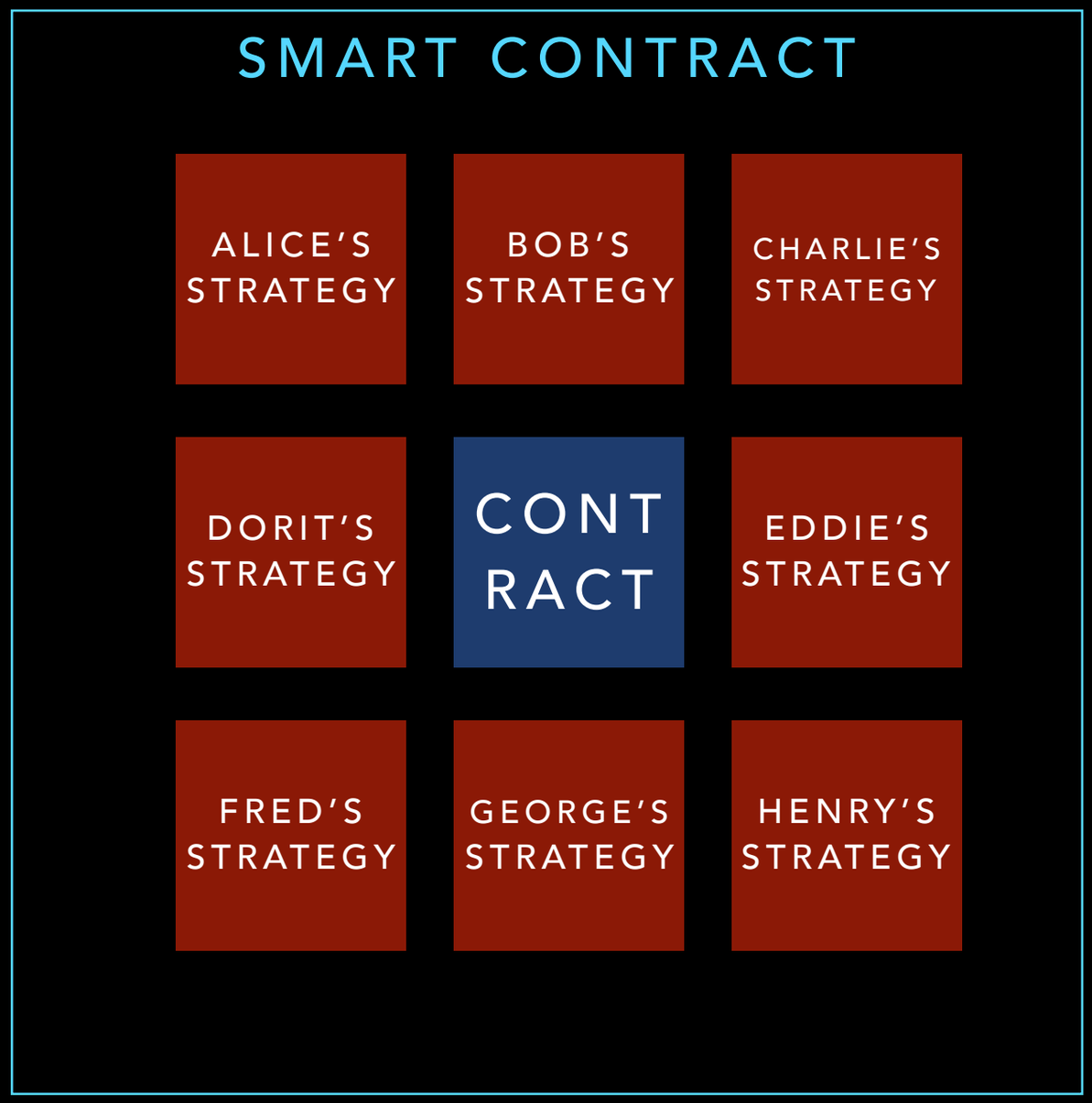
CONTRACT VERSUS SMART CONTRACT

Contract:
Obligations and permissions
(rules)

Example 2 (FX American Option)**.** Party X may, within 90 days, decide whether to (immediately) buy 100 US dollars for a fixed rate 6.5 of Danish kroner from party Y.

$$\begin{aligned}&\textbf{if } \textbf{obs}(X \text{ exercises option}, 0) \textbf{ within } 90\\ &\textbf{then } 100 \times (\mathsf{USD}(Y \to X)\,\&\,6.5 \times \mathsf{DKK}(X \to Y))\\ &\textbf{else } \emptyset\end{aligned}$$

CONTRACT VERSUS SMART CONTRACT

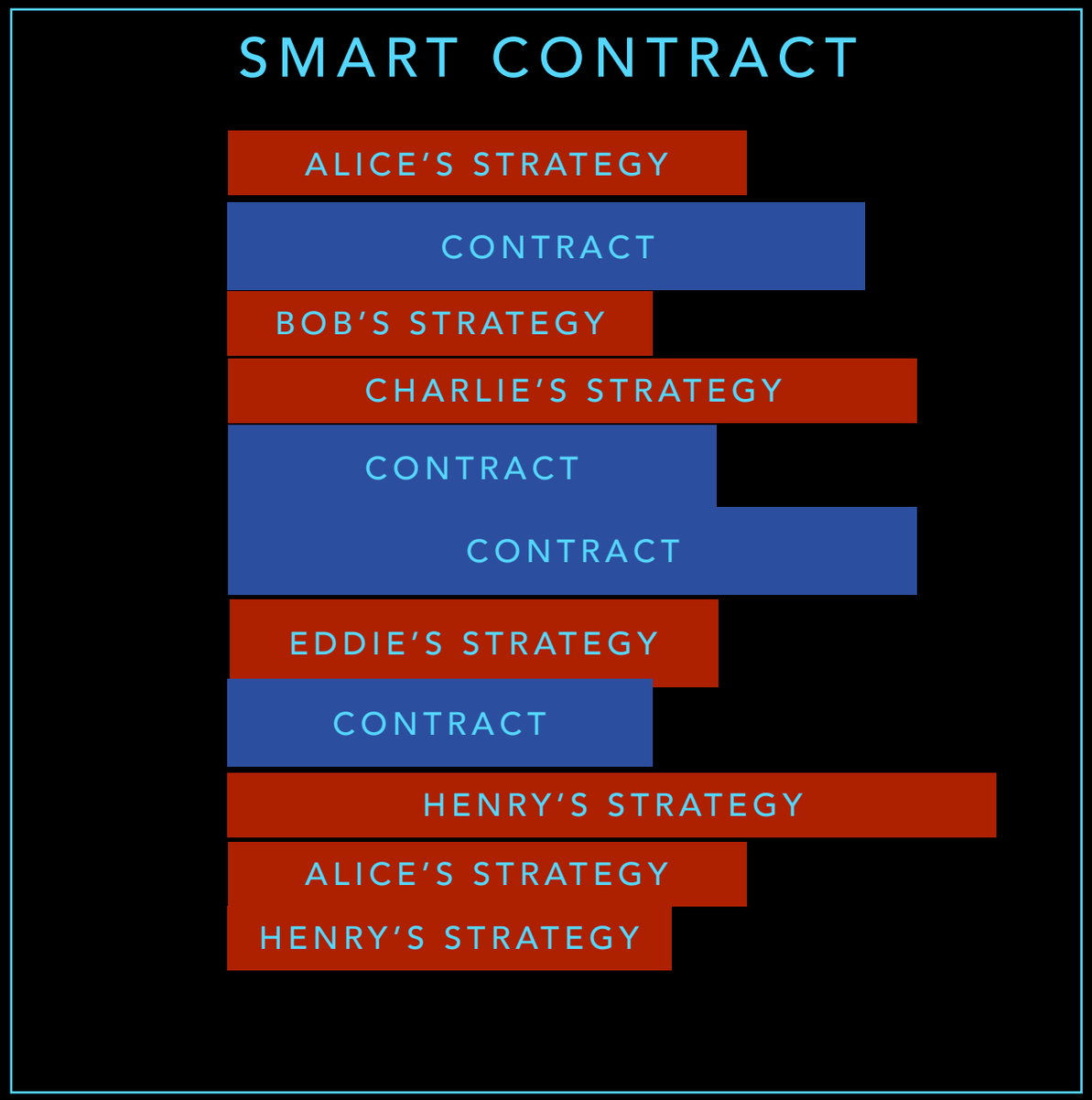


Contract:
Obligations and permissions (rules)

Strategy:
A single party's actions (actions)

Smart contract:
Rules and all parties' codified actions intermixed

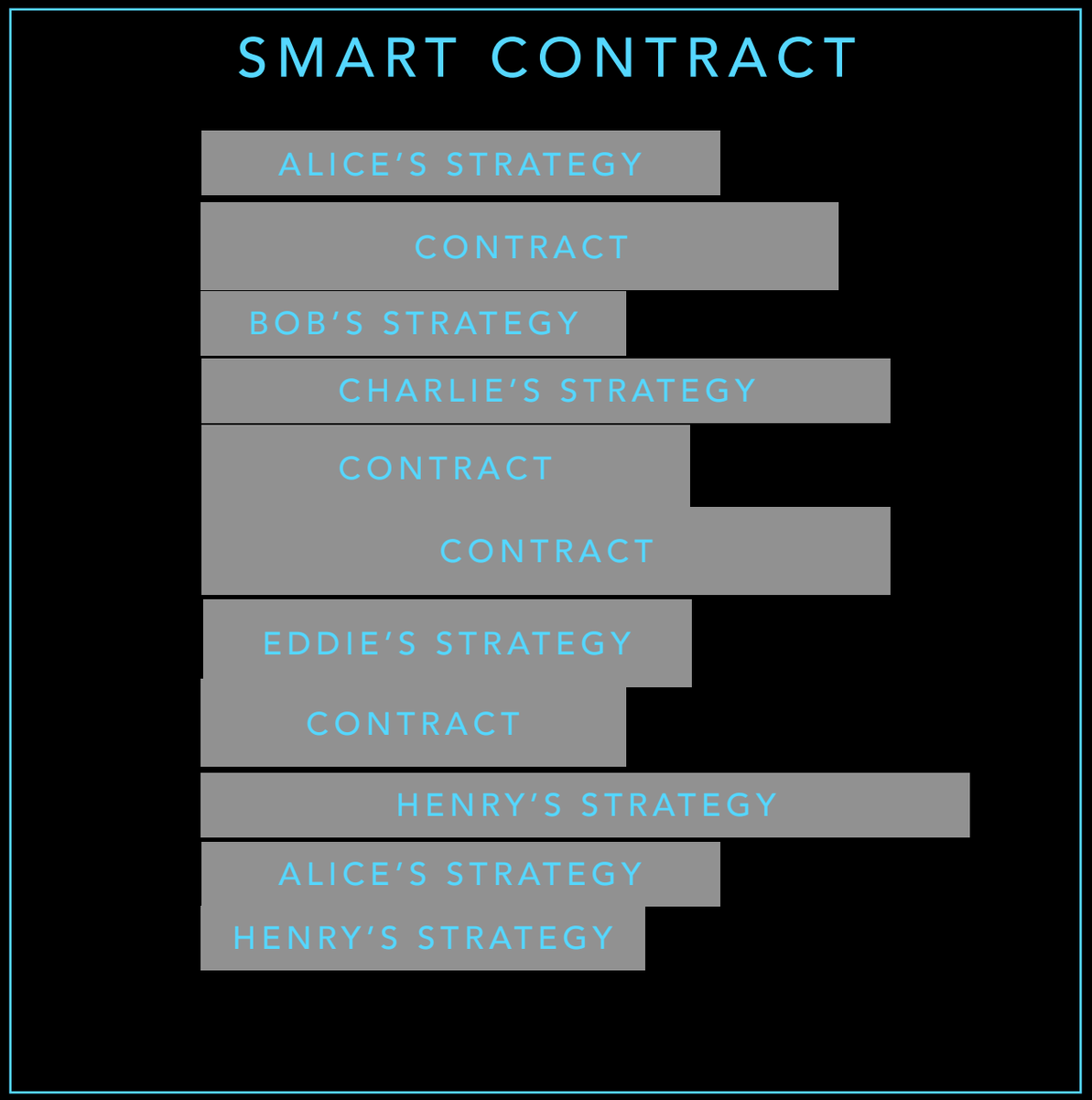
CONTRACT VERSUS SMART CONTRACT



Actually…

Contract checking and
actions (strategy) mixed together
in the source code

CONTRACT VERSUS SMART CONTRACT

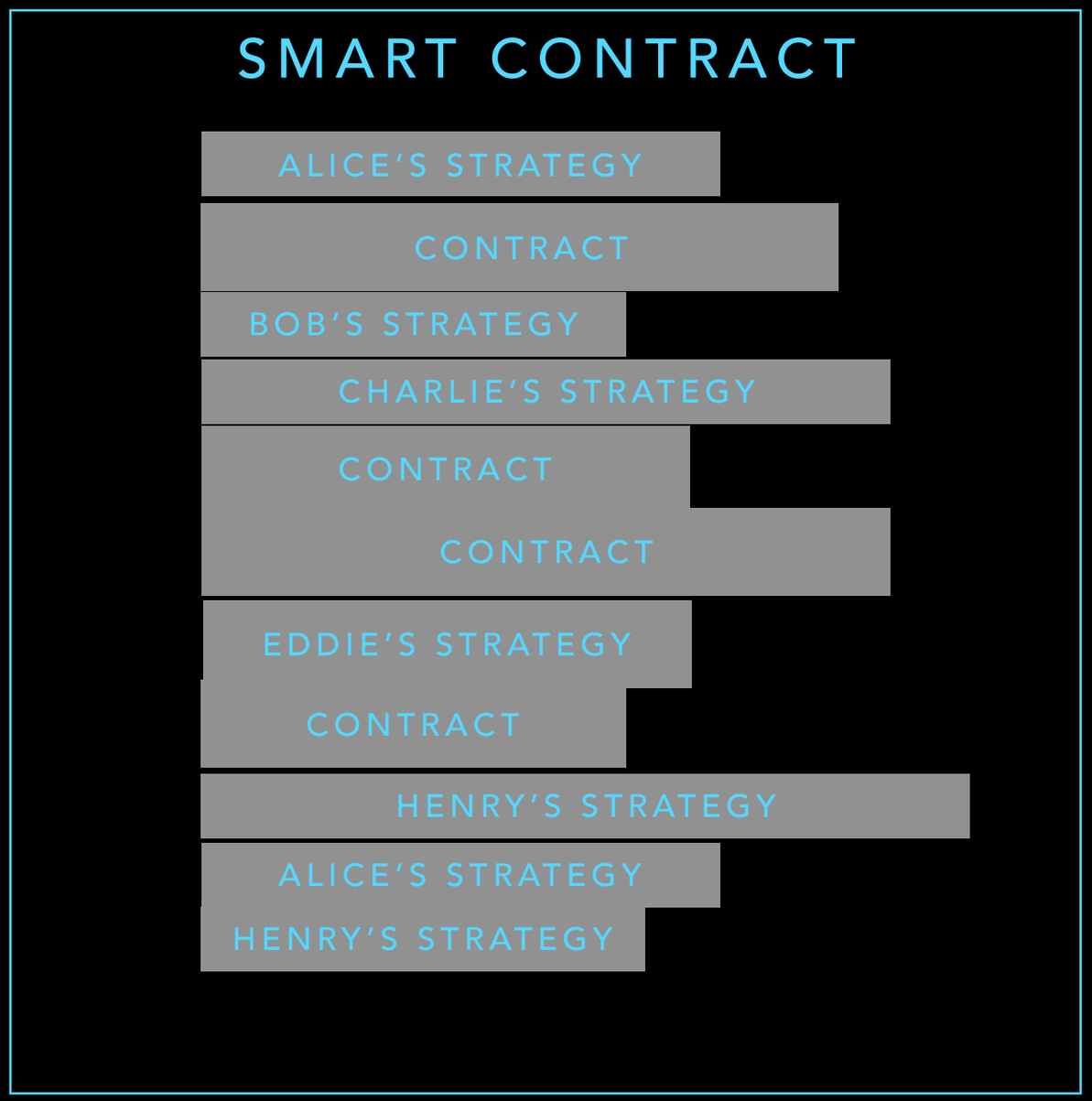


Actually…

Contract checking and actions (strategy) mixed together in the source code

and one cannot even see which is which

CONTRACT VERSUS SMART CONTRACT

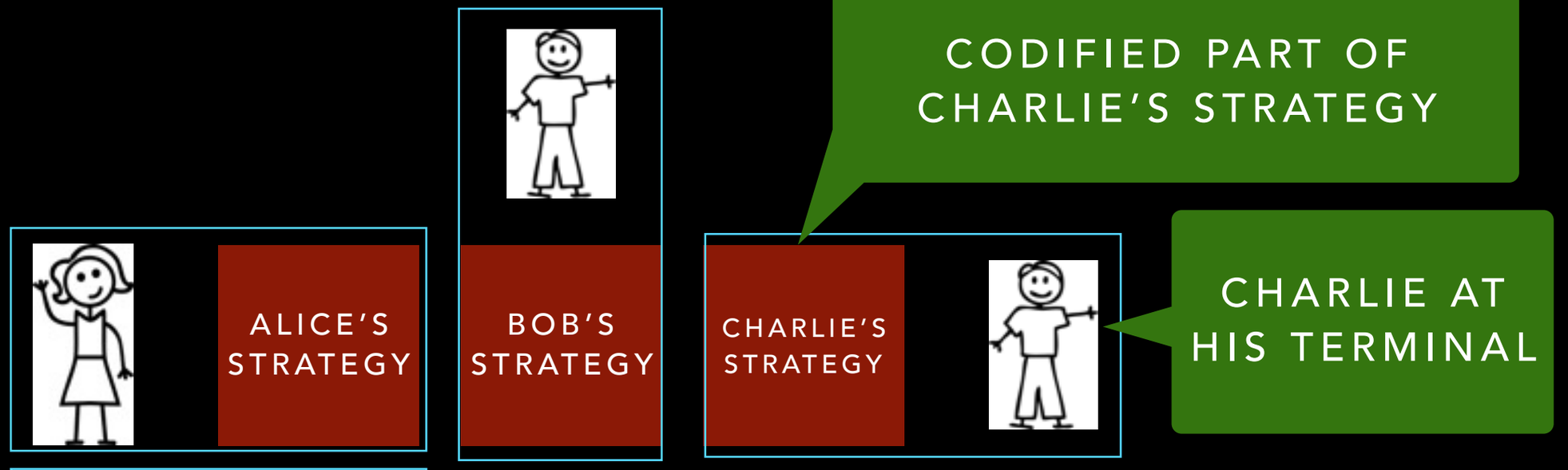


What is the **contract** and what is **strategy**?

How do you **compose** contracts (by themselves)?

How do you **analyze** contracts?

CONTRACT VERSUS SMART CONTRACT

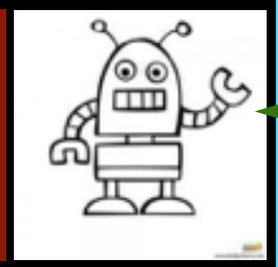


CONTRACT VERSUS SMART CONTRACT

ALICE'S STRATEGY

BOB'S STRATEGY

CHARLIE'S STRATEGY



CHARLIE AUTOMATES HIMSELF

DORIT'S STRATEGY

CONT RACT

EDDIE'S STRATEGY

FRED'S STRATEGY

GEORGE'S STRATEGY

HENRY'S STRATEGY

CONTRACT VERSUS SMART CONTRACT

ALICE'S STRATEGY

BOB'S STRATEGY

CHARLIE'S NEW, FULLY AUTOMATED STRATEGY

DORIT'S STRATEGY

CONT RACT

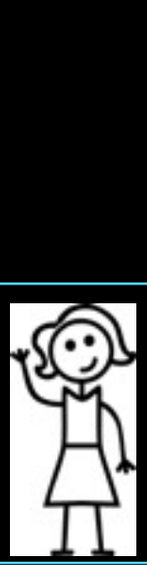
EDDIE'S STRATEGY

FRED'S STRATEGY

GEORGE'S STRATEGY

HENRY'S STRATEGY

CONTRACT VERSUS SMART CONTRACT



ALICE'S STRATEGY

BOB'S STRATEGY

CHARLIE'S NEW, FULLY AUTOMATED STRATEGY

DORIT'S NEW SEMI-AUTOMATED STRATEGY

CONT RACT

EDDIE'S STRATEGY

FRED'S STRATEGY

GEORGE'S STRATEGY

HENRY'S STRATEGY

DORIT AUTOMATES PARTS OF HER TERMINAL INTERACTIONS

CONTRACT VERSUS SMART CONTRACT

CONTRACT IS UNCHANGED!

ALICE'S STRATEGY

BOB'S STRATEGY

CHARLIE'S NEW, FULLY AUTOMATED STRATEGY

DORIT'S NEW SEMI-AUTOMATED STRATEGY

CONTRACT

EDDIE'S STRATEGY

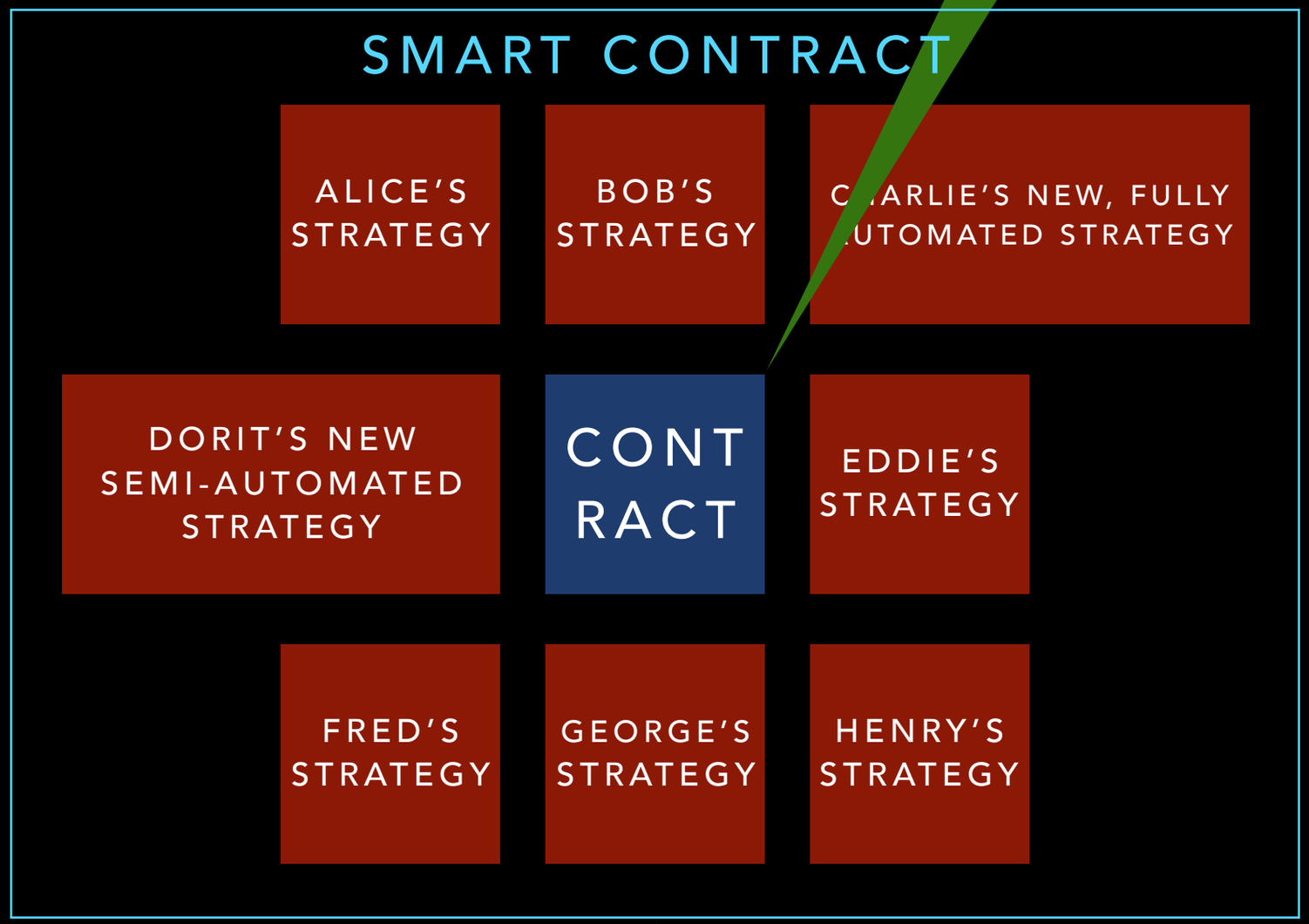
FRED'S STRATEGY

GEORGE'S STRATEGY

HENRY'S STRATEGY

CONTRACT VERSUS SMART CONTRACT

CONTRACT IS UNCHANGED!



BUT SMART CONTRACT SHOULD BE CHANGED!?

THE PRICE OF EXPRESSIVENESS: RICE'S THEOREM

Rice (1953)

- Smart contract: usually program, written in **Turing-complete** programming language (Ethereum, Corda, Fabric, …)

- **+** : Expressive

- **-** : **Undecidable properties** even with *full access to the source code*

  - Smart contracts are ultimately **unanalyzable**

ETHEREUM VULNERABILITIES

LUU, CHU, OLICKEL, SAXENA, HOBOR, MAKING SMART CONTRACTS SMARTER (2016)

- Transaction-order dependence: Messages may have different effect depending on their order of arrival
  - Who controls the process scheduler (= message sequencer)? Some *miner*: Front-running
- Time-stamp dependence: Smart contracts may have different executions depending on the time stamp on a transaction block
  - Who controls the time stamping of transaction blocks? Some miner: Clock manipulation
- Exception handling, gas management fragility: Subtle differences in exception semantics, limited run-time stack
  - Provoking out-of-stack and gas exhaustion exceptions: Any user
- Programming language subtleties:
  - Exception handling subtleties (send vs. call)
  - Reentrancy vulnerability (DAO hack)
  - Implicit method forwarding (multi-sig exploit)

REENTRANCY VULNERABILITY

LUU, CHU, OLICKEL, SAXENA, HOBOR, MAKING SMART CONTRACTS SMARTER (2016)

```
1 contract SendBalance {
2  mapping (address => uint) userBalances;
3  bool withdrawn = false;
4  function getBalance(address u) constant returns(uint){
5   return userBalances[u];
6  }
7  function addToBalance() {
8   userBalances[msg.sender] += msg.value;
9  }
10 function withdrawBalance(){
11  if (!(msg.sender.call.value(
12    userBalances[msg.sender])())) { throw; }
13  userBalances[msg.sender] = 0;
14  }}
```

Figure 7: An example of the reentrancy bug. The contract implements a simple bank account.

OTHER BLOCKCHAIN SYSTEM ASPECTS

- Performance
- Availability
- Partition tolerance
- Security
- Privacy
- (Trade-offs between above, some inherent, some not)
- …

SUMMARY

- Smart contracts = self-executing contracts (programs) in complex Turing-complete programming language
- Rules and actions intermixed:
  Not contracts
- Hard to analyze, low-level programs:
  Not smart

MORE INFORMATION

- hiperfit.dk: Functional high-performance computing for finance
  - Domain-specific languages for **compositional and verifiable contracts**

GOING LIVE ANY TIME NOW…

- plan-x.org: Functional programming language technology for **high-performance blockchain systems**

FUNCTIONAL PROGRAMMING = PROGRAMMING WITH ~~IMMUTABLE~~ TAMPER-PROOF DATA

Thank you!

henglein@diku.dk