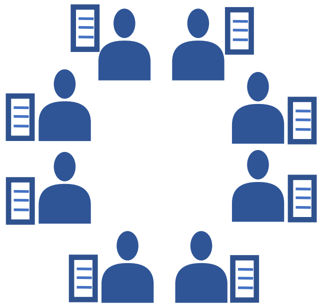
Rensselaer

why not change the world?®

Ontology Aided Smart Contract Execution for Unexpected Situations

Farhad Mohsin, Xingjian Zhao, Zhuo (Robin) Hong, Geeth de Mel, Lirong Xia, and **Oshani Seneviratne**

Blockchain and Smart Contract



- Blockchain enables **trustworthy** data sharing between untrusting parties in a **tamper-proof** manner
- Smart contracts enables us to add **logic** to govern updates via transactions
- Once the smart contracts are set in motion, they cannot be changed!

Can we predict, detect, and fix unexpected situations in smart contracts?

Limitations of Smart Contracts

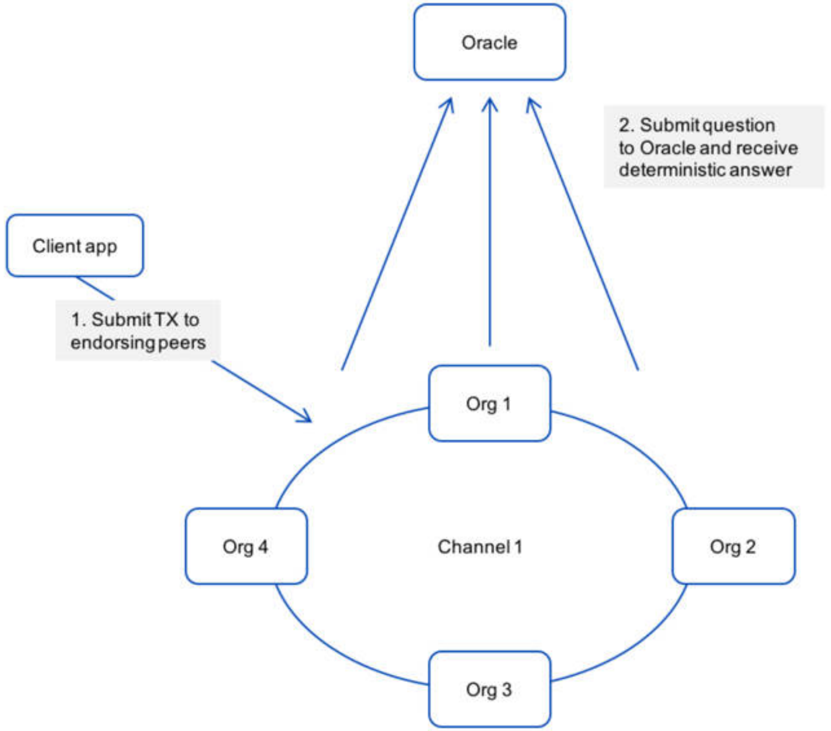
- Immutable
- No way out for a *break-glass-in-case-of-emergency* scenarios
- Need to foresee all unexpected situations
- We need a solution when smart contracts aren't as smart as they need be

Our Proposal

Use *Oracles* to change how smart contracts execute, so *unexpected situations* may be resolved

Oracles in Blockchain

- Trusted system for information transfer
- Good for extending smart contracts with off-chain complex logic
  - To integrate volatile knowledge, e.g., stock price
  - Complex business rules

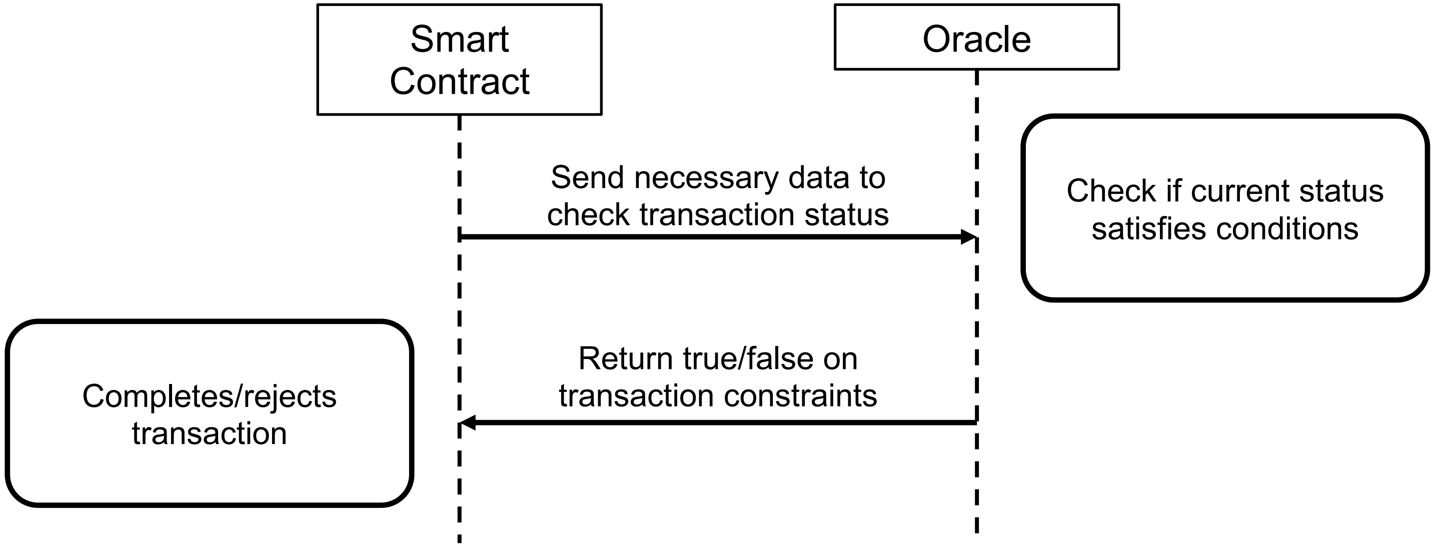


Oracle
2. Submit question to Oracle and receive deterministic answer
Client app
1. Submit TX to endorsing peers
Org 1
Org 4
Channel 1
Org 2
Org 3

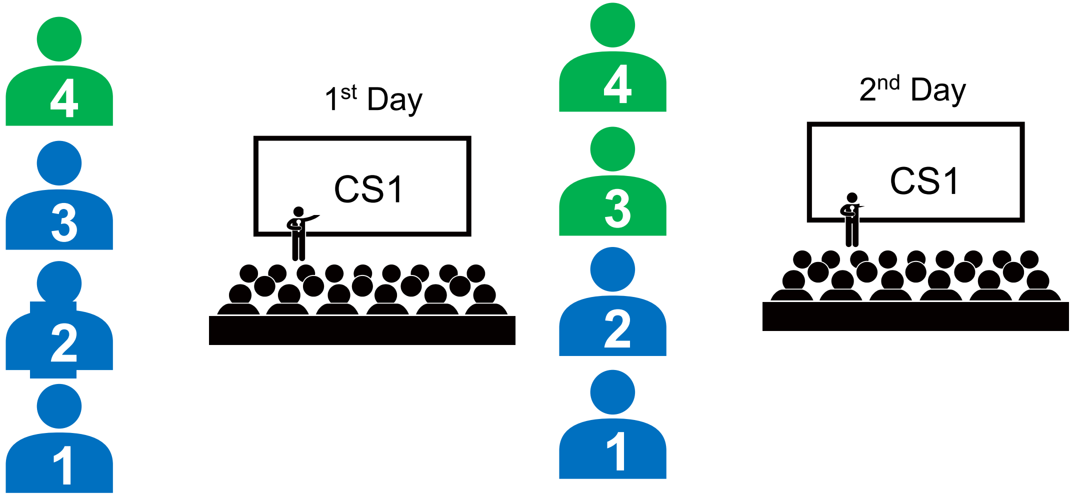
https://developer.ibm.com/articles/cl-extend-blockchain-smart-contracts-trusted

Ontology based Oracle for Smart Contract Execution

- Blockchain to will act as a verifiable data structure
- Logic for each transaction will be performed off-chain


Smart Contract
Oracle
Send necessary data to check transaction status
Check if current status satisfies conditions
Return true/false on transaction constraints
Completes/rejects transaction

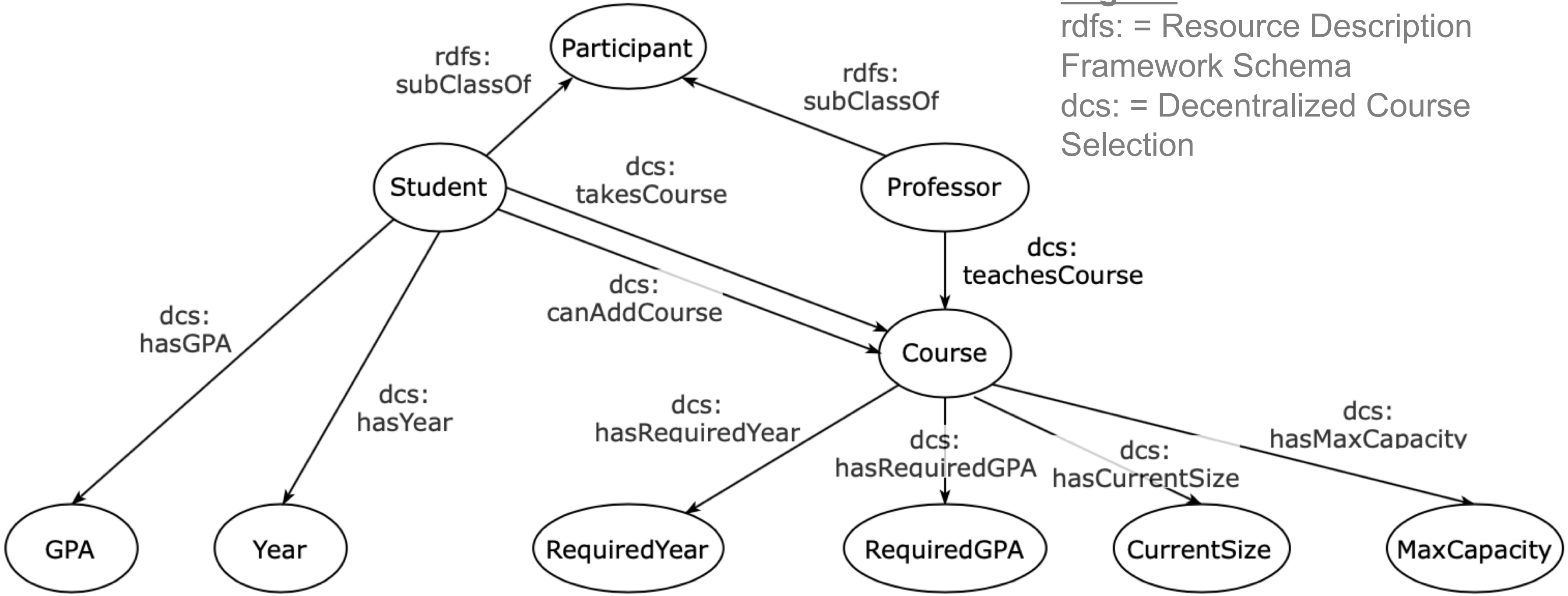
Example: Decentralized Course Selection



Unexpected Situation
A freshman student with a very good GPA gets a special permission to enroll in an already full course.

But, no proper function in the original Smart Contract!

Decentralized Course Selection (DCS) Ontology



Legend
rdfs: = Resource Description Framework Schema
dcs: = Decentralized Course Selection

Off-Chain Rule Update

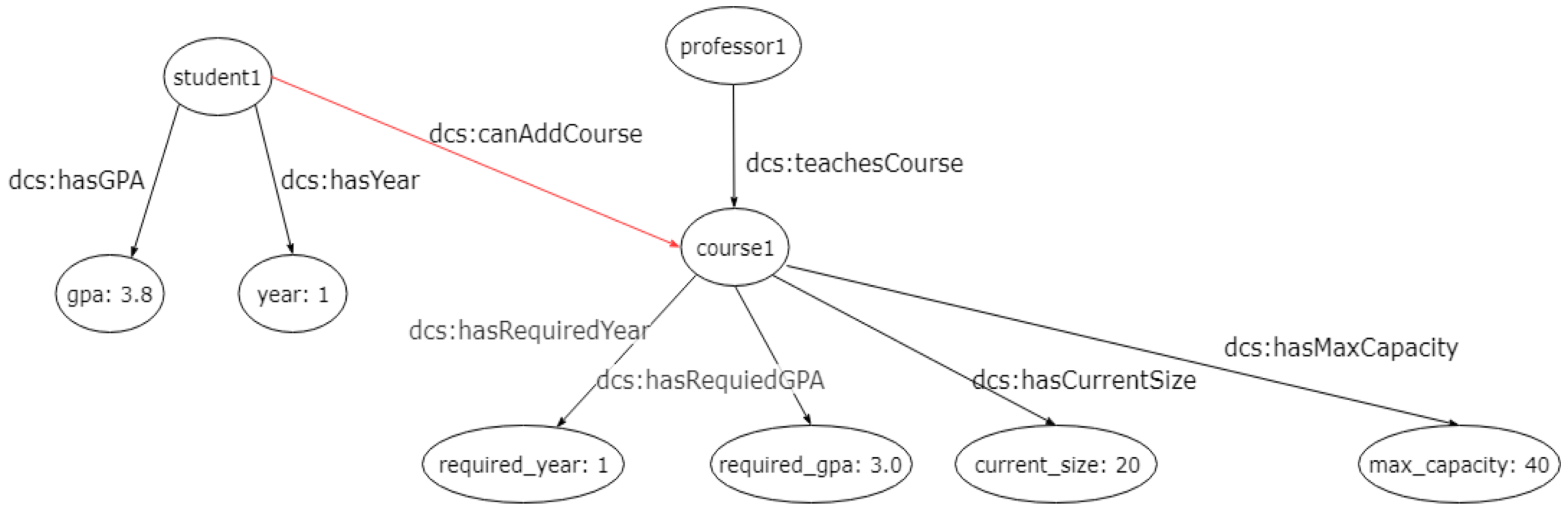
Initial Rule

```
Student(?s) ∧
hasYear(?s,?y) ∧
Course(?c) ∧
hasRequiredYear(?c, ?ry) ∧
hasMaxCapacity(?c, ?mc) ∧
hasCurrentSize(?c, ?curr) ∧
swrlb:greaterThanOrEqual(?y, ?ry) ∧
swrlb:lesserThan(?curr, ?mc)
→
canAddCourse(?s, ?c)
```

Updated Rule

```
Student(?s) ∧
hasGPA(?s, ?g) ∧
hasRequiredGPA(?c, ?rg) ∧
hasYear(?s,?y) ∧
Course(?c) ∧
hasRequiredYear(?c, ?ry) ∧
hasMaxCapacity(?c, ?mc) ∧
hasCurrentSize(?c, ?curr) ∧
swrlb:greaterThanOrEqual(?g, ?rg) ∧
swrlb:greaterThanOrEqual(?y, ?ry) ∧
swrlb:lesserThan(?curr, ?mc)
→
canAddCourse(?s, ?c)
```

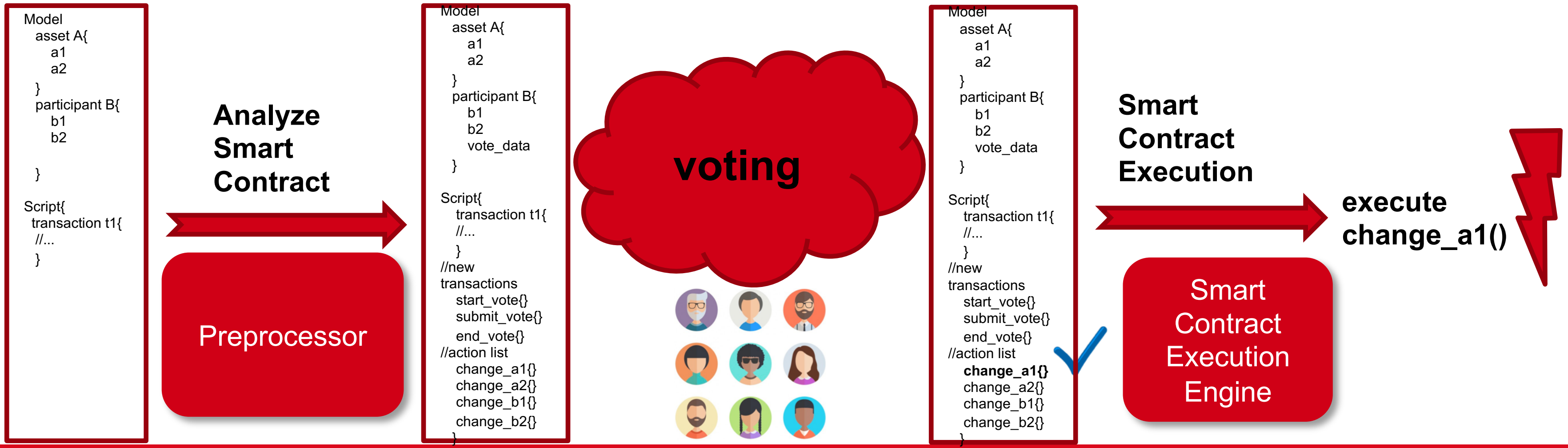
DCS Instance Graph



Governance Structure

- Pre-processor determines an action list
- Smart Contract Execution Engine executes the action that was selected by the peers

Strengthening Smart Contracts to Handle Unexpected Situations; *Shuze Liu, Farhad Mohsin, Lirong Xia, Oshani Seneviratne;* International Conference on Decentralized Applications and Infrastructures 2019



```
Model
  asset A{
    a1
    a2
  }
  participant B{
    b1
    b2

  }

Script{
 transaction t1{
  //...
  }
```

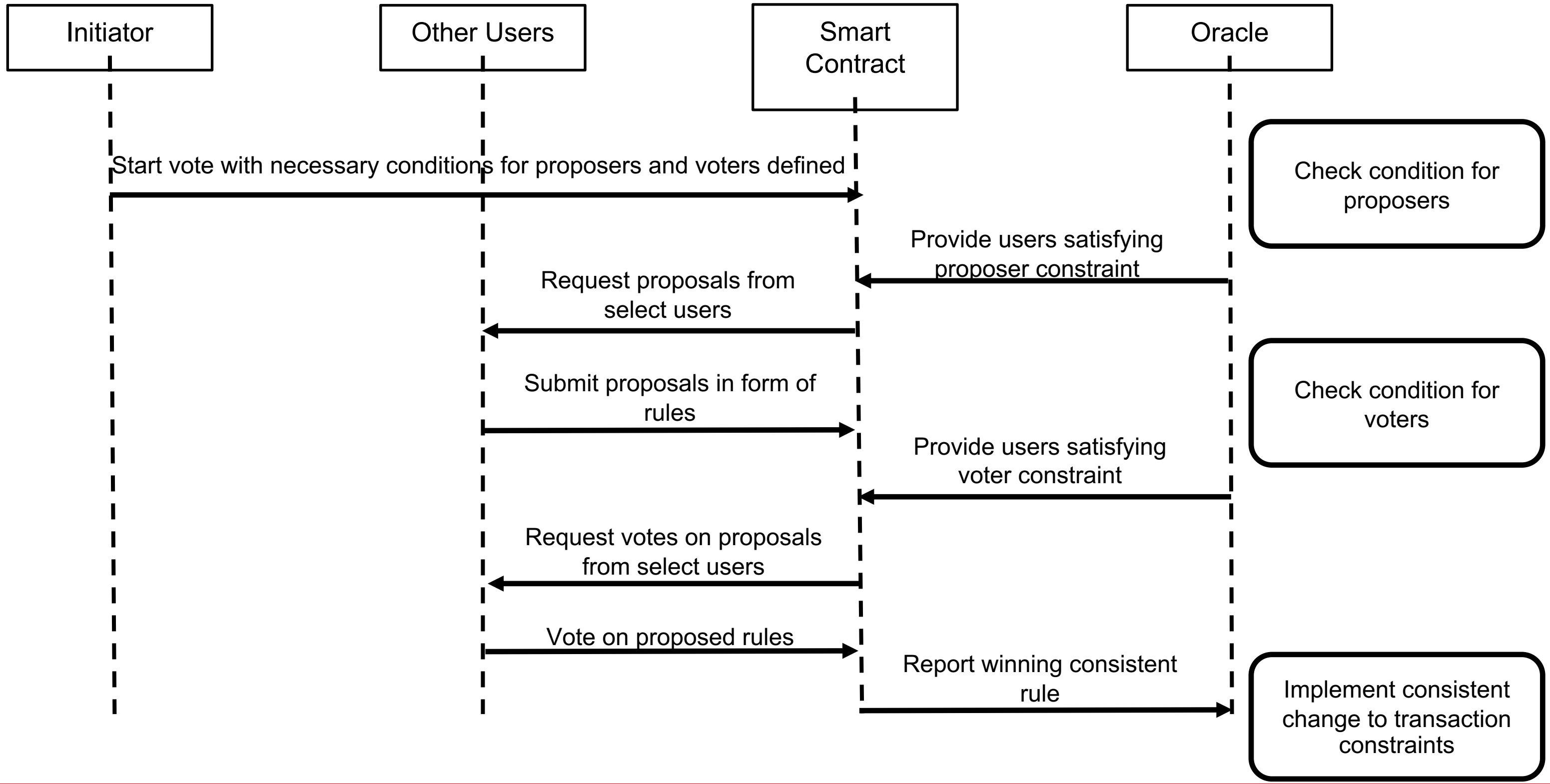
```
Model
  asset A{
    a1
    a2
  }
  participant B{
    b1
    b2
    vote_data
  }

Script{
  transaction t1{
  //...
  }
//new
transactions
  start_vote{}
  submit_vote{}
  end_vote{}
//action list
  change_a1{}
  change_a2{}
  change_b1{}
  change_b2{}
  }
```

```
Model
  asset A{
    a1
    a2
  }
  participant B{
    b1
    b2
    vote_data
  }

Script{
  transaction t1{
  //...
  }
//new
transactions
  start_vote{}
  submit_vote{}
  end_vote{}
//action list
  change_a1{} ✓
  change_a2{}
  change_b1{}
  change_b2{}
  }
```

Future Work: Proposed Voting Mechanism for Updating the Ontology



Initiator
Other Users
Smart Contract
Oracle
Start vote with necessary conditions for proposers and voters defined
Check condition for proposers
Provide users satisfying proposer constraint
Request proposals from select users
Submit proposals in form of rules
Check condition for voters
Provide users satisfying voter constraint
Request votes on proposals from select users
Vote on proposed rules
Report winning consistent rule
Implement consistent change to transaction constraints

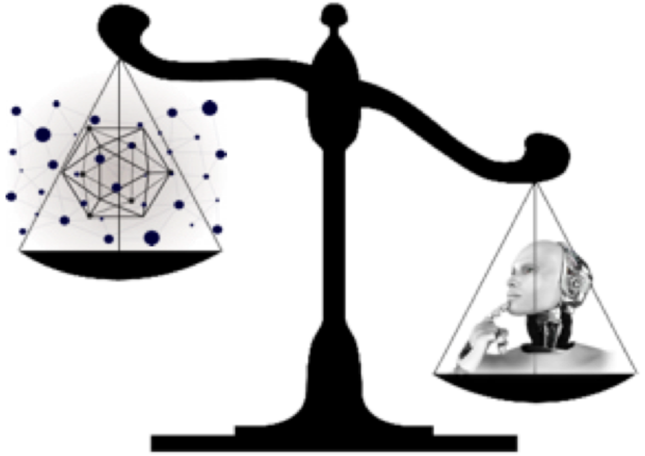
Implementation Concerns

- Rules and attributes should only be changed to an extent.
  - E.g. course.MaxCapacity may be changeable, student.GPA should probably not be changed
- For privacy concerns, the oracle should receive data necessary for forming instances for each transaction and never store a complete knowledge graph
- Update on the rules should only occur from the smart contract and protected against external tampering

Summary

- Utilization of external rules to augment the smart contract logic
- If there is a gap in the logic, the external oracle could be updated

Questions?



senevo@rpi.edu

SCALES – Smart Contracts Augmented with LEarning and Semantics

https://idea.tw.rpi.edu/projects/scales