Extending blockchain with data quality assessment

Marco Comuzzi

December 3, 2019

School of Management Engineering
Ulsan National Institute of Science and Technology (UNIST)
Ulsan, Republic of Korea

Outline

- Data quality in information systems
- Data quality in blockchain: Why?
- Solutions
    - Quality-aware transaction validation
    - Data quality controls as smart contracts
- Ongoing and future work

Data quality assessment in information systems

- Data Quality (DQ) as *fit for use*
  - Ensure that data stored in an information system are of sufficient quality for its purpose
- Users and client applications may push low quality data into information systems
  - On purpose
  - By mistake

Data quality assessment: online v. offline

- Offline: assessing quality of data stored in a system
  - Data profiling
  - Anomaly detection
- **Online**: assessing quality of a data value before it is stored
  - DQ Controls: DQ assessment expressed as a formula, implemented by a sw program

Temperature of container cannot exceed 25% of average temperature in previous 30 mins

Data quality and cryptocurrencies (Bitcoin, Ethereum)

- All data are produced on-chain
- Online data quality assessment means to prevent double spending
- Data quality assessment through:
  - Combination of validation (UTXO) and consensus (PoW) in Bitcoin
  - Transaction nonce in Ethereum

Data quality in 2nd generation blockchains

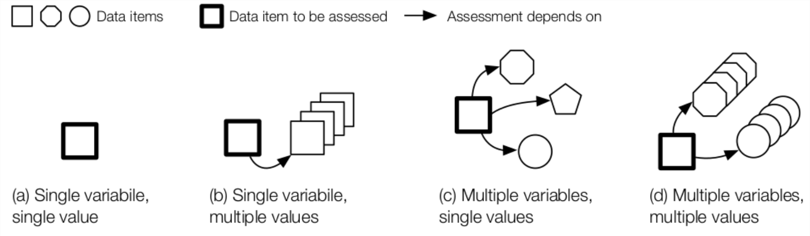
- Smart contract-enabled blockchains
- Transactions payload include all type of data (generated on-/off-chain)
- Transaction payloads currently treated as black box (quality of data not assessed)

How to assess quality of transaction payloads?

- Modelling data quality concerns
  - Information requirements for DQ control
  - Data access options in blockchains
  - Reaction policies
- Approach 1: Quality-aware transaction validation
- Approach 2: Quality controls as smart contracts

Information requirements for DQ control

WHICH data values do we need to run data quality control?



(a) Single variabile, single value

(b) Single variabile, multiple values

(c) Multiple variables, single values

(d) Multiple variables, multiple values

(a) A temperature T must be between 25 and 32 °C

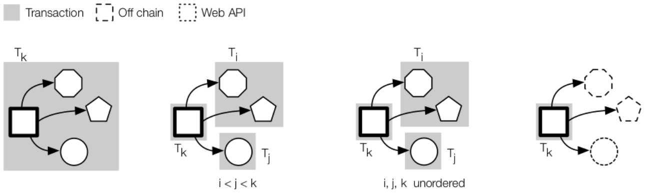
(b) T is accurate if does not exceed avg values recorded in the last hour of more than 25%

(c) Patient name accuracy checked against a number of public records

(d) T accuracy checked against current and historical series

Data access options for DQ control in blockchain

WHERE do the data required for quality control come from?



(a) A transaction has all required values in its payload

(b) Values in ordered transactions
  - The data item of which we want to control quality is last
  - Feasible if transactions are sufficiently distant in time

(c) Multiple transactions, no order guaranteed
  - Data value correlation required

(d) External data sources

Reaction policies

WHAT do we do when low quality is discovered?

- ✚ *Accept value*: in some cases, quality alerts can be simply ignored.
- ⊘ *Do not accept value*: quality alerts can be critical and low quality values must not be stored in the blockchain.
- 🛢 *Log violation*: accept low quality value, but flag it to make applications using it aware of its low quality.
- ⚡ *Raise event*: low quality value signals a critical situation that must be addressed immediately.
- ↻ *Defer decision*: single quality violation not enough to take a decision.

Approach 1: Quality-aware transaction validation

- Extend blockchain protocol (transaction validaton) to support DQ controls
- Solution must be specific to blockchain protocol

  Ethereum-centric (for now)

Overall framework

- Nodes receive transactions
- Transaction validated in respect of data access requirements of data quality controls
    - As a result, transaction validation order may differ from the one determined by transaction nonces
- Transaction annotated with DQ control result
    - Rejected/failed if quality too low
    - Validated, quality annotations handled by application

Model

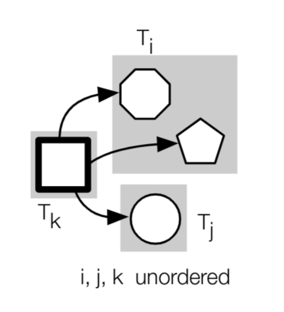
- Transactions carry data d_i as key-value pairs
- Quality control qc_j requires N data values to be executed

$$qc_j = f(d_1, \ldots, d_N)$$

- Data values d_n carried by multiple transactions that can be received in any order by nodes
- Examples:

  Precision of IoT readings from multiple sensors (e.g., pressure, temperature, volume)

  Consistency of customer information (e.g., city + zip code) delivered from multiple nodes

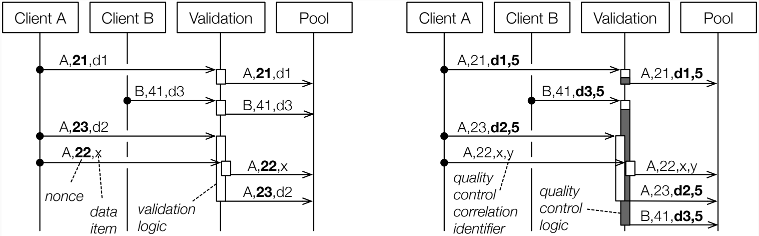


Solution

- Controls qc_j embedded in protocol
- Data items carry a correlation identifier to match qc_j *instances*
    - A new set of temperature readings
    - A different patient
- Instances of qc_j activated by node when first d_n received
- Result of qc_j instance calculated by node when last d_n received
- Once a qc_j instance active, validation of transactions carrying d_n should occur only after all d_n have been received for that instance

Modified transaction validation order

- $qc_1 = f(d1)$
- $qc_2 = f(d_2, d_3)$



(a) **Standard** validation and ordering of transactions (arrows) based on transaction nonce (in bold)

(b) **Quality-aware** validation and ordering of transactions based on transaction nonce and quality controls (in bold data items to be checked and correlation identifier of control)

Handling quality control results

- Reject/fail transactions when qc_j result not acceptable
    - Blockchain protocol embeds transaction rejection logic in qc_j definition
    - Need to handle possibly conflicting results if transaction carries d_n relevant to multiple qc_j
- Write qc_j result in transaction, mine it, and let applications decide
    - Transaction structure must be extended with fields to register qc_j results

Approach 2: Quality control as smart contracts

- Online DQ controls, implemented as smart contracts
- DQ smart contract templates addressing:
  - Information requirements for DQ control
  - Data access options and reaction policies in DQ controls
  - Reaction policies

Implementation options

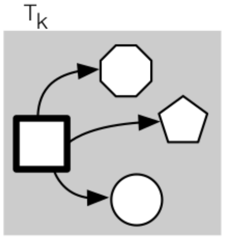
- Stateless smart contract
- Stateful smart contract
- Stateful smart contract + correlation
- Stateful smart contract + oracle

1) Stateless smart contract

- Stateless smart contract if all data items available in one single transactions
- Two sub-options:
  - Ad-hoc SC accepting values to be checked and returning the assessment, invoked using a standard message call
  - Reusable library (e.g. `SafeMath`) attached to data types in application SC

    using *lib_name* for *data_type*

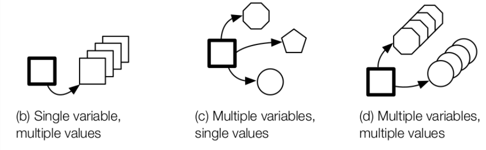
(a) Single variable, single value



(a) Single transaction

3) Stateful smart contract + correlation

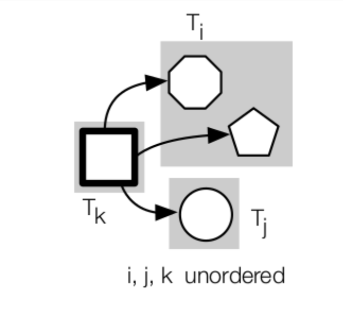
- Extend stateful SC with correlation logic
- Necessary if ordering of transactions carrying values cannot be guaranteed



(b) Single variable, multiple values

(c) Multiple variables, single values

(d) Multiple variables, multiple values



(c) Interleaved transactions

Stateful smart contract + correlation: template

```
contract FlaggingDQContract {
    uint16 varA;        // monitored variable
    bool   isUpdatedA;  // update flag
    uint32 varB;        // variable the control depends on
    bool   isUpdatedB;  // update flag

    function check() returns (int){
      if (isUpdatedA && isUpdatedB) { // if both variables are up to date
          isUpdatedA = isUpdatedB = false; // reset flags
          ... // TODO: apply quality control logic and return result
      } else return -1; // return if check not applicable yet
    }

    function setA(uint _varA) public returns (int){
      varA = _varA;
      isUpdatedA = true;
      return check(); // control quality if applicable
    }

    function setB(uint32 _varB) public returns (int){
      varB = _varB;
      isUpdatedB = true;
      return check(); // control quality if applicable
    }
    ... // TODO: implementation of getters
}
```

Comparing approaches

Approach 1 (Transaction validation)

- Requires modification of blockchain protocol
- Allows to reject low quality transactions
- Permissioned/controlled blockchains in controlled scenarios
- Cannot handle off-chain data

Approach 2 (Smart contracts for DQC)

- No need to extend existing blockchain protocol
- Low quality transactions always registered
- Decoupling of functional and quality assessment logic
- Can handle off-chain data through oracles

Ongoing and future work

- Approach 1: quality-aware transaction validation
  - Model and solution needs fine tuning
  - Implementation is challenging
  - Need to find *killer* use cases
- Approach 2: data quality controls as smart contracts
  - Implementation on toy examples, preliminary evaluation
  - Need real world use cases

Thank you! Any questions?