Chapter 4

BLONDiE Ontology

This chapter gives a comprehensive description of the development of the BLONDiE ontology. The first section starts with explaining the representation requirements needed for the development of the ontology. This is followed by a section that explains the domain capture and the identified classes and properties. The next section explains the implementation principles used.

As we saw in Chapter 3: Related work, there is a very limited work in the semantic web and ontology research communities that stresses the potential of using ontologies for cryptocurrencies like Bitcoin and Ethereum. Currently there is no ontology that covers the complete structure of the blockchain of these new frameworks rendering some of these efforts incomplete. I aim to fill this gap by proposing an open source collaborative ontology called BLONDiE (Blockchain ontology with dynamic extensibility). It is specifically focused on Bitcoin and Ethereum although it can be extended to cover other altcoins based on Bitcoin and Ethereum, since they may share the same or very similar structure.

4.1 Representational Requirements

Goals: (1) Developed a schema for a queriable knowledge base that stores information from the Bitcoin and Ethereum blockchain and other related information.

(2) include business intelligence in the knowledge repository that runs on powerful semantics to answer queries from the users about the Bitcoin and Ethereum framework.

There is a natural need to browse the content of existing blockchain frameworks like Bitcoin and Ethereum. Information as: Checking address balances, tracking coin transfer histories, watching for transaction acceptance, monitoring the network hash rate, etc.

The analysis is made on 2 parts:

i) The documented Blockchain Structure(including Ethereum accounts)

ii) Additional elements

4.2 Domain Capture

Goals:

In the following, we give an overview of the relevant conceptual entities and types of relationships. We split our capture in 2 phases: 1) First we express all the structural information of Bitcoin and Ethereum blockchains and related classes as they are expressed in their official documentation. 2) Later we gather other information according to our analysis.

Identified classes will be expressed with: and properties with:

A Block can be of two types Bitcoin Block, and Ethereum Block. A Block is composed by a Blockheader and a Payload. A Blockheader can be of two types Bitcoin Blockheader and Ethereum Blockheader. A Payload can be of two types Bitcoin Blockheader and Ethereum Blockheader. A Transaction can be of two types Bitcoin Transaction and Ethereum Transaction. An Ethereum transaction can be of two types Normal Ethereum Transaction and Contract Creation Transaction. An Ethereum account can be of two types Normal Ethereum Account and Contract Ethereum Account.

a) Structure of the blockchain

- Block.
  Given class name: Block.

- Bitcoin block.
  Given class name: BitcoinBlock.

- Ethereum block.
  Given class name: EthereumBlock.

- Blockheader.
  Given class name: Blockheader

  A blockheader has the following general properties.

  - Previous block hash: Hash of previous block header.
    Given property name: hashPreviousBlockheader

  - Merkle root hash: Top hash of the Merkle tree built from all transactions. ***Given property name: merkleRootBlockheader***

  - Timestamp: Timestamp in UNIX-format of approximate block creation time. ***Given property name: timestampBlockheader***

  - Difficulty target: Target T for the proof of work problem in compact format. ***Given property name: difficultyBlockheader***

  - Nonce: Nonce allowing variations for solving the proof of work problem. ***Given property name: nonceBlockheader***

- Bitcoin Blockheader
  Given class name: BitcoinBlockheader
  A Bitcoin blockheader has the following specific properties:

  – Version: The block version number indicates which set of block validation rules to follow. ***Given property name: versionBitcoinBlockheader***

- Ethereum Blockheader
  Given class name: EthereumBlockheader
  An Ethereum blockheader has the following specific properties:

  – Omners hash: Hash of the ommers list portion of the block
  Given property name: omnersHashEthereumBlockheader

  – Beneficiary: The address to which all fees collected from the successful mining of this block be transferred;
  Given property name: beneficiaryEthereumBlockheader

  – Transaction root: Hash of the root node of the trie structure populated with each transaction in the transactions list portion of the block
  Given property name: transactionRootEthereumBlockheader

  – Receipts root: hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list portion of the block
  Given property name: receiptsRootEthereumBlockheader

  – Logs bloom: The Bloom filter composed from indexable information (logger address and log topics) contained in each log entry from the receipt of each transaction in the transactions list
  Given property name: logsBloomEthereumBlockheader

  – number of ancestors: A scalar value equal to the number of ancestor blocks. The genesis block has a number of zero
  Given property name: numberAncestorsEthereumBlockheader

  – Gas limit: A scalar value equal to the current limit of gas expenditure per block.
  Given property name: gasLimitEthereumBlockheader

  – Gas used: A scalar value equal to the total gas used in transactions in this block.
  Given property name: gasUsedEthereumBlockheader

  – Extra data: An arbitrary byte array containing data relevant to this block.
  Given property name: extraDataEthereumBlockheader

  - Mix hash: hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block
    Given property name: mixHashEthereumBlockheader

- Payload
  Given class name: Payload.
  A payload has the following general properties:

  - List of transactions:
    Given property name: hastransactionPayload
  -

- Bitcoin Payload
  Given class name: BitcoinPayload.
  A Bitcoin payload has the following specific properties:

  - Number of transactions: Number of transaction entries.
    Given property name: numberTransactionsBitcoinPayload

- Ethereum Payload
  Given class name: EthereumPayload.
  An Ethereum payload has the following specific properties:

  - list of ommer block header:
    Given property name: hasOmmerEthereumPayload

- Transaction.
  Given class name: Transaction.

- Bitcoin transaction.
  Given class name: BitcoinTransaction.

  A Bitcoin transaction has the following specific properties:

  - Version: Transaction format version.
    Given property name: versionBitcoinTransaction
  - Lock time: Timestamp past which transactions can be replaced before inclusion in block.
    Given property name: lockTimeBitcoinTransaction
  - Number of input transactions: Number of transaction input entries in vin.
    Given property name: nunberInputTransactionsBitcoinTransaction
  - Number of output transactions: Number of transaction output entries in vout.
    Given property name: numberOutputTransactionsBitcoinTransaction
  - list of input transactions: List of input transactions.
    Given property name: hasBitcoinTransactionInput

  - list of output transactions: List of output transactions.
    Given property name: hasBitcoinTransactionOutput

- Bitcoin transaction Input.
  Given class name: BitcoinTransactionInput.
  A Bitcoin transaction Input has the following specific properties:
  - hash: Double-SHA256 hash of a past transaction.
    Given property name: hashBitcoinTransactionInput
  - index: Index of a transaction output within the transaction specified by hash.
    Given property name: indexBitcoinTransactionInput
  - scriptSigLenBitcoinTransactionInput: Length of scriptSig field in bytes.
    Given property name: scriptSigLenBitcoinTransactionInput
  - scriptSigBitcoinTransactionInput : Script to satisfy spending condition of the transaction output (hash,n).
    Given property name: scriptSigBitcoinTransactionInput
  - nSequenceBitcoinTransactionInput: Transaction input sequence number.
    Given property name: nSequenceBitcoinTransactionInput

- Bitcoin transaction Output.
  Given class name: BitcoinTransactionOutput.
  A Bitcoin transaction Input has the following specific properties:
  - nValueBitcoinTransactionOutput: Amount of 10-8 BTC.
    Given property name: nValueBitcoinTransactionOutput
  - scriptPubkeyLenBitcoinTransactionOutput: Length of scriptPubkey field in bytes.
    Given property name: scriptPubkeyLenBitcoinTransactionOutput
  - scriptPubkeyBitcoinTransactionOutput: Script specifying conditions under which the transaction output can be claimed.
    Given property name: scriptPubkeyBitcoinTransactionOutput

- Ethereum transaction.
  Given class name: EthereumTransaction.

  An Ethereum transaction has the following specific properties:
  - Nonce: A scalar value equal to the number of transactions sent by the sender
    Given property name: nonceEthereumTransaction
  - Gas price: A scalar value equal to the number of Wei to be paid per unit of gas for all computation costs incurred as a result of the execution of this transaction.
    Given property name: gasPriceEthereumTransaction
  - Gas limit: A scalar value equal to the maximum amount of gas that should be used in executing this transaction. This is paid up-front, before any computation is done and may not be increased later
    Given property name: gasLimitEthereumTransaction

  - Recipient: The 160-bit address of the message call's recipient or, for a contract creation transaction
    Given property name: recipientEthereumTransaction
  - Value: A scalar value equal to the number of Wei to be transferred to the message call's recipient or, in the case of contract creation, as an endowment to the newly created account
    Given property name: valueEthereumTransaction
  - v: Values corresponding to the signature of the transaction and used to determine the sender of the transaction
    Given property name: vEthereumTransaction
  - r: Values corresponding to the signature of the transaction and used to determine the sender of the transaction
    Given property name: rEthereumTransaction
  - s: Values corresponding to the signature of the transaction and used to determine the sender of the transaction
    Given property name: sEthereumTransaction

- Normal Ethereum transaction.
  Given class name: NormalEthereumTransaction.

- Contract creation Ethereum transaction.
  Given class name: ContractCreationEthereumTransaction.

  A contract creation Ethereum transaction has the following specific properties:

  - Init: An unlimited size byte array specifying the EVM-code for the account initialisation procedure
    Given property name: initContractCreationEthereumTransaction

- Message call Ethereum transaction.
  Given class name: MessageCallEthereumTransaction.

  A message call Ethereum transaction has the following specific properties:

  - data: An unlimited size byte array specifying the input data of the message call
    Given property name: dataMessageCallEthereumTransaction

- Ethereum account.
  Given class name: EthereumAccount.

  An Ethereum account has the following general properties:

  - nonce: A scalar value equal to the number of transactions sent from this address or, in the case of accounts with associated code, the number of contract-creations made by this account.
    Given property name: nonceEthereumAccount

  - balance: A scalar value equal to the number of Wei owned by this address.
    Given property name: balanceEthereumAccount

- Normal Ethereum account.
  Given class name: NormalEthereumAccount.

- Contract Ethereum account.
  Given class name: ContractEthereumAccount.

  A contract Ethereum account has the following general properties:

  - storageRoot: A hash of the root node of a Merkle Patricia tree that encodes the storage contents of the account
    Given property name: storageRootContractEthereumAccount
  - codeHash: The hash of the EVM code of this account.
    Given property name: codeHashContractEthereumAccount

b) More elements:

- On Block.
  Identified additional properties:

  - Block height: The height of the block on the chain
    Given property name: heightBlock

- On Transaction.
  Identified additional properties:

  - sender: The address of the sender of the transaction
    Given property name: senderTransaction

4.3 Implementation

In BLONDiE vocabulary I defined 70 classes, and 67 properties.