Trust in Smart Contracts is a Process, As Well

Firas al-Khalil, Tom Butler, Leona O'Brien, and Marcello Ceci

Governance, Risk, and Compliance Technology Center
University College Cork
Cork, Ireland

1st Workshop on Trusted Smart Contracts
In Association with Financial Cryptography 2017
April 07, 2017

The Problem

1. The financial industry is showing more interest in DLTs
2. Reluctance in the adoption of DLTs and Smart Contracts:
3. Too much attention is given to *developers*, i.e. programmers
4. What about *"traditional developers"* of contracts?
   - 4.1 The financial industry is heavily regulated (remember 2008?)
   - 4.2 How to *"explain"* the operations in Smart Contracts to regulators?
   - 4.3 How to make sure that operations carried by Smart Contracts are compliant with current regulation?

Outline

2 Tools for Lawyers

1 Tools for Developers

An Overview

- `coinmarketcap.com`[1] tracks market capitalisation of different cryptocurrencies, lists 719 platforms
- Bitcoin: scripting language is not Turing-complete, and very limited in expressivity
- Nxt: RESTful APIs, and uses PoS instead of PoW
- Rootstock: a sidechain of Bitcoin, provides a Turing-complete RVM
- Ethereum: maybe the most popular platform, provides a Turing-complete EVM
  1. Opcode
  2. Serpent and Solidity
- τ—Chain: ...

[1] At the time we wrote the paper, ~3rd of February, 2017.

Writing Smart Contracts is Hard

- Undergraduate students at the University of Maryland were taught to develop (Ethereum) smart contracts [1]
- Smart contract programming requires an "*economic thinking*" perspective
  - Money leaks
  - Failure to use cryptographic primitives to secure the contracts from attackers
  - Mistakes directly related to Ethereum
  - …

[1] Delmolino, K., et al.: **Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab**. (2016)

[2] Luu, L., et al.: **Making smart contracts smarter**. (2016)

Writing Smart Contracts is Hard

- Undergraduate students at the University of Maryland were taught to develop (Ethereum) smart contracts [1]
- Smart contract programming requires an "*economic thinking*" perspective
  - Money leaks
  - Failure to use cryptographic primitives to secure the contracts from attackers
  - Mistakes directly related to Ethereum
  - …
- A class of security related bugs [2] in smart contracts are due to the gaps in the understanding of the distributed semantics of the underlying platform

[1] Delmolino, K., et al.: **Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab**. (2016)

[2] Luu, L., et al.: **Making smart contracts smarter**. (2016)

Smarter Contracts

- Masters Thesis by Pettersson and Edström [3] to help said programmers to develop safer smart contracts
- Their aim is to prevent 3 kinds of mistakes contract developers fall in:
  1. unexpected states
  2. failure to use cryptography
  3. overflowing the EVM's stack
- They developed a code generator:
  `Idris` $\rightarrow$ `Serpent` $\rightarrow$ `EVM bytecode`

[3] Pettersson, J., Edström, R.: Safer smart contracts through type-driven development. (2015)

And the list goes on …

- The business process language BPMN can be mapped [4] into executable smart contracts on the Ethereum.
- Logic-based smart contracts [5]
  1. Contract negotiation, formation, storage/notarizing, enforcement, and monitoring and activities related to dispute resolution
- Business Collaboration Language (BCL) [6] proposed by IBM: *"SQL for shared ledgers, regardless of implementation-specific detail"*

[4] García-Bañuelos, L., Ponomarev, A., Dumas, M., Weber, I.: **Optimized Execution of Business Processes on Blockchain**. ArXiv e-prints (Dec 2016)
[5] Idelberger, F., Governatori, G., Riveret, R., Sartor, G.: **Evaluation of Logic-Based Smart Contracts for Blockchain Systems** (2016)
[6] Hull, R., Batra, V.S., Chen, Y.M., Deutsch, A., Heath III, F.F.T., Vianu, V.: **Towards a Shared Ledger Business Collaboration Language Based on Data-Aware Processes** (2016)

Outline

1 Tools for Developers

2 Tools for Lawyers

- Existing Work
- The Web Ontology Language OWL
- Smart Contracts and Regulatory Compliance

Existing Work

- Frantz and Nowostawski [7] propose a semi-automated method for the translation of human readable contracts to smart contracts on Ethereum
  `(DSL for Smart Contracts)` $\rightarrow$ `Solidity`
  - It is not clear how extensible such a DSL is
  - It doesn't cover the legal language that a lawyer would be accustomed to.
- Clack et al. [8] rightly identify two semantics of contracts:
  - Denotational Semantics, that capture the *"legal meaning"* of the contract, as understood by a lawyer.
  - Operational Semantics, concerned with the execution of the contract on a specific platform

  They propose the use of Ricardian Contracts.

[7] Frantz, C.K., Nowostawski, M.: **From institutions to code: Towards automated gen- eration of smart contracts**. (2016)
[8] Clack, C.D., Bakshi, V.A., Braine, L.: **Smart Contract Templates: essential re- quirements and design options**. ArXiv e-prints (Dec 2016)

Ricardian Contracts

A Ricardian Contract [9, 10] is a digitlly signed triple $\langle P, C, M \rangle$, where:

1. P is the legal prose, capturing denotational semantics,
2. C is the platform specific code expressing operational semantics, and
3. M is a map (key-value pairs) of parameters used in P

We think that Ricardian Contracts are not enough:

1. P is flat text, with no meaning attached to it.
2. The order of the instructions in C does not reflect the natural order of the contract clauses expressed in natural language
3. The life-cycle of legal prose is independent from the life-cycle of the code.
4. There is not a single smart contract platform, which ultimately means that different parameters (key-value pairs of M) will be needed for different platforms

[9] Grigg, I.: **The ricardian contract** (2004)
[10] Grigg, I.: **On the intersection of Ricardian and Smart Contracts**. `http://iang.org/papers/intersection_ricardian_smart.html` (Feb 2017)

Where Should the Lawyer be Involved?

The involvement of a lawyer, especially in the heavily regulated financial industry, in the authoring of contracts, not only smart contracts, is paramount:

1. Validatation of the its textual version of a smart contract
2. Assessing the compliance of the contract with regulations

Therefore, we think that proper authoring of smart contracts should involve:

1. the lawyer, and
2. the developer

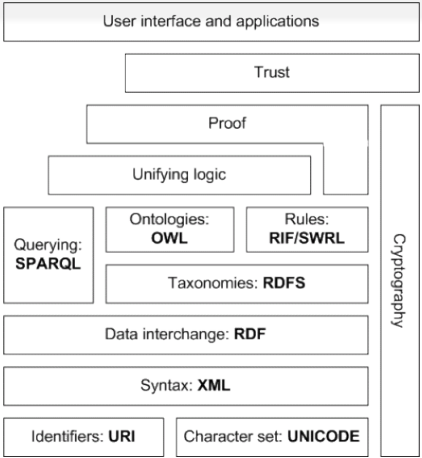
The interaction between both agents should be governed by a common language

Requirements of the Common Language

1. It should not alienate the lawyer
2. It should be expressive enough to allow the authoring of smart and *"not-so-smart"* contracts
3. It should be a Controlled Natural Language with an unambiguous grammar and semantics
   - It should be mappable to a logical formalism, which will enable compliance checking
4. The concepts and actions described in the contract (i.e. the vocabulary) along the clauses of the contract (i.e. the rules) should be shareable across the network
   - Important for both *discoverability* and *negotiation*
5. It should be able to represent the actions coded in the smart contract, i.e. the duties and powers arising from the contract and the meta-rules governing it (e.g. regulation on financial activities, Anti-Money Laundering or consumer protection).

OWL

- Ontologies are explicit descriptions of
  - Concepts in a domain of interest, and
  - Properties and attributes of concepts
- Ontologies are at the core of the W3C's Semantic Web
- *"[…] provid[ing] a common framework that allows data to be shared and reused across application, enterprise, and community boundaries"*



Example Ontology

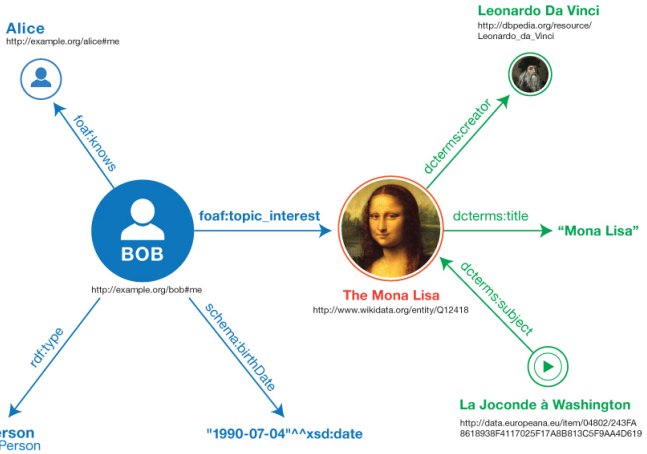
Resources are described in triples of the form ⟨Subject, Predicate, Object⟩

```
<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<the Mona Lisa> <was created by>
<Leonardo da Vinci>.
<the video 'La Joconde à Washington'>
<is about> <the Mona Lisa>.
```

* example taken from the W3C RDF primer @ `https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/`

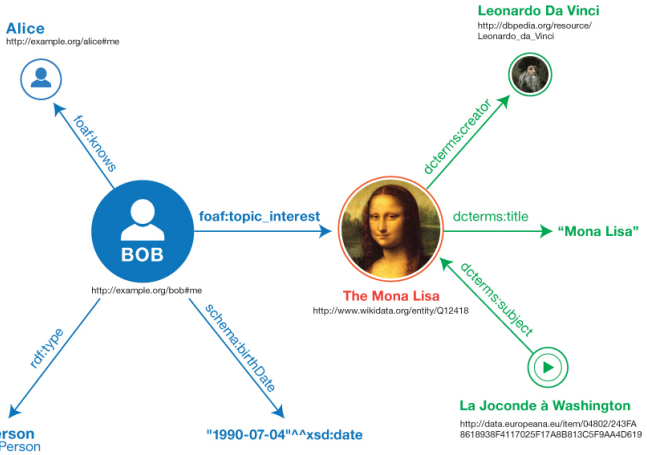
Example Ontology

Every resource is identified by an IRI



Example Ontology

foaf:<http://xmlns.com/foaf/0.1/> **dcterms:**<http://purl.org/dc/terms/>



DLT $\longleftrightarrow$ Ontologies?

How DLTs and the Semantic Web can affect one another [1]?

1. DLTs can provide IRIs (viz. Namecoin)
2. Ontologies can provide a shared understanding of key DLT concepts (implementation agnostic)

[11] English, M., Auer, S., Domingue, J.: **Block chain technologies & the semantic web: A framework for symbiotic development** (2016)

Working with Ontologies

- Design:
  - Define concepts
  - Arrange concepts in a taxonomic (subclass-superclass) hierarchy
  - Define properties and describe allowed values for these properties
  - Populate with individuals

Working with Ontologies

- Design:
  - Define concepts
  - Arrange concepts in a taxonomic (subclass-superclass) hierarchy
  - Define properties and describe allowed values for these properties
  - Populate with individuals
- Reasoning:
  - Satisfiability of a concept: Is a description of a concept C contradictory? Can C have any instances?
  - Subsumption of concepts: is concept C *included* in concept D?
  - Equivalence: Are concepts C and D equivalent?
  - Disjointness: Can concepts C and D have common members?
  - Consistency: determine whether individuals violate descriptions of concepts and roles

Working with Ontologies

- Design:
  - Define concepts
  - Arrange concepts in a taxonomic (subclass-superclass) hierarchy
  - Define properties and describe allowed values for these properties
  - Populate with individuals
- Reasoning:
  - Satisfiability of a concept: Is a description of a concept C contradictory? Can C have any instances?
  - Subsumption of concepts: is concept C *included* in concept D?
  - Equivalence: Are concepts C and D equivalent?
  - Disjointness: Can concepts C and D have common members?
  - Consistency: determine whether individuals violate descriptions of concepts and roles
- Querying: check if individual is an instance of a concept, find all individuals that are instances of a concept, find all concepts which the individual belongs to, ...

τ—Chain

- Turing completeness is not necessary for distributed ledgers

[12] Asor, O.: **About Tau-Chain**. ArXiv e-prints (Feb 2015)

τ—Chain

- Turing completeness is not necessary for distributed ledgers
- Turing completeness $\mapsto$ undecidability

[12] Asor, O.: **About Tau-Chain**. ArXiv e-prints (Feb 2015)

τ—Chain

- Turing completeness is not necessary for distributed ledgers
- Turing completeness $\mapsto$ undecidability
- Etherum solves undecidability with (non-refundable) *gas*

[12] Asor, O.: **About Tau-Chain**. ArXiv e-prints (Feb 2015)

τ—Chain

- Turing completeness is not necessary for distributed ledgers
- Turing completeness $\mapsto$ undecidability
- Etherum solves undecidability with (non-refundable) *gas*
- τ — *Chain* [12] proposes:
  - To Author smart contracts with a totally functional programming language, e.g. Idris
  - To *"compile"* smart contract to an ontology
  - To use a reasoner to perform the actual computations

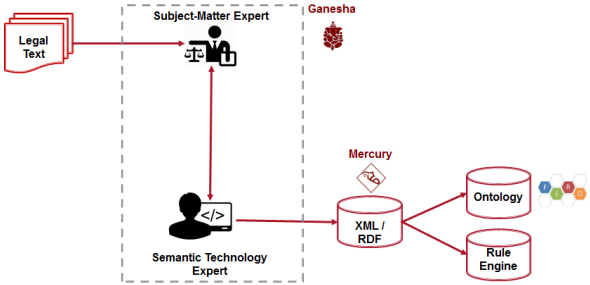
[12] Asor, O.: **About Tau-Chain**. ArXiv e-prints (Feb 2015)

τ—Chain

- Turing completeness is not necessary for distributed ledgers
- Turing completeness $\mapsto$ undecidability
- Etherum solves undecidability with (non-refundable) *gas*
- τ — *Chain* [12] proposes:
  - To Author smart contracts with a totally functional programming language, e.g. Idris
  - To *"compile"* smart contract to an ontology
  - To use a reasoner to perform the actual computations
- Benefits:
  1. Decideability
  2. Assertion of some properties of the contract

[12] Asor, O.: **About Tau-Chain**. ArXiv e-prints (Feb 2015)

Regulatory Compliance @ GRCTC



Trust in Smart Contracts

- Is there a mechanism $\mathcal{G}$ that automatically generates code from prose?

Trust in Smart Contracts

- Is there a mechanism $\mathcal{G}$ that automatically generates code from prose?
- Is there a mechanism $\mathcal{C}$, maybe $\mathcal{C} = \mathcal{G}^{-1}$, that proves the correspondence of the code to the prose?

Trust in Smart Contracts

- Is there a mechanism $\mathcal{G}$ that automatically generates code from prose?
- Is there a mechanism $\mathcal{C}$, maybe $\mathcal{C} = \mathcal{G}^{-1}$, that proves the correspondence of the code to the prose?

Is the existence of $\mathcal{G}$ and $\mathcal{C}$ is essential for trust?

Trust in Smart Contracts

- Is there a mechanism $\mathcal{G}$ that automatically generates code from prose?
- Is there a mechanism $\mathcal{C}$, maybe $\mathcal{C} = \mathcal{G}^{-1}$, that proves the correspondence of the code to the prose?

Is the existence of $\mathcal{G}$ and $\mathcal{C}$ is essential for trust?

1. If τ—Chain is really feasible, then these mechanisms may exist, but we are tying ourselves to 1 specific platform.

Trust in Smart Contracts

- Is there a mechanism $\mathcal{G}$ that automatically generates code from prose?
- Is there a mechanism $\mathcal{C}$, maybe $\mathcal{C} = \mathcal{G}^{-1}$, that proves the correspondence of the code to the prose?

Is the existence of $\mathcal{G}$ and $\mathcal{C}$ is essential for trust?

1. If τ—Chain is really feasible, then these mechanisms may exist, but we are tying ourselves to 1 specific platform.
2. For platform with stack-based languages we have 2 possibilities:

Trust in Smart Contracts

- Is there a mechanism $\mathcal{G}$ that automatically generates code from prose?
- Is there a mechanism $\mathcal{C}$, maybe $\mathcal{C} = \mathcal{G}^{-1}$, that proves the correspondence of the code to the prose?

Is the existence of $\mathcal{G}$ and $\mathcal{C}$ is essential for trust?

1. If τ—Chain is really feasible, then these mechanisms may exist, but we are tying ourselves to 1 specific platform.
2. For platform with stack-based languages we have 2 possibilities:
   2.1 It is possible, or practically feasible $\rightarrow$ 

Trust in Smart Contracts

- Is there a mechanism $\mathcal{G}$ that automatically generates code from prose?
- Is there a mechanism $\mathcal{C}$, maybe $\mathcal{C} = \mathcal{G}^{-1}$, that proves the correspondence of the code to the prose?

Is the existence of $\mathcal{G}$ and $\mathcal{C}$ is essential for trust?

1. If τ—Chain is really feasible, then these mechanisms may exist, but we are tying ourselves to 1 specific platform.
2. For platform with stack-based languages we have 2 possibilities:
   - 2.1 It is possible, or practically feasible $\rightarrow$ 
   - 2.2 It is impossible $\rightarrow$?

Trust in Smart Contracts

- Is there a mechanism $\mathcal{G}$ that automatically generates code from prose?
- Is there a mechanism $\mathcal{C}$, maybe $\mathcal{C} = \mathcal{G}^{-1}$, that proves the correspondence of the code to the prose?

We conjecture that the existence of $\mathcal{G}$ and $\mathcal{C}$ is not essential for trust:

1. The implementation processes of existing financial contracts in the form of software is already opaque
2. Trust can be gained through the establishment of reputation
3. Our proposal improves transparency, which is one of the major luring qualities of distributed ledgers, and a determining factor of the trust-less trust in the network.

References

[1] Delmolino, K., Arnett, M., Kosba, A.E., Miller, A., Shi, E.: **Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab**. In: Financial Cryptography and Data Security - FC 2016 International Work- shops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers. pp. 79–94 (2016).
[2] Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A.: **Making smart contracts smarter**. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 254–269. CCS '16, ACM, New York, NY, USA (2016).
[3] Pettersson, J., Edström, R.: **Safer smart contracts through type-driven development**. Master's thesis, Dept. of CS&E, Chalmers University of Technology & University of Gothenburg, Sweden (2015)
[4] García-Bañuelos, L., Ponomarev, A., Dumas, M., Weber, I.: **Optimized Execution of Business Processes on Blockchain**. ArXiv e-prints (Dec 2016)
[5] Idelberger, F., Governatori, G., Riveret, R., Sartor, G.: Evaluation of Logic-Based Smart Contracts for Blockchain Systems, pp. 167–183. Springer International Pub- lishing, Cham (2016)
[6] Hull, R., Batra, V.S., Chen, Y.M., Deutsch, A., Heath III, F.F.T., Vianu, V.: **Towards a Shared Ledger Business Collaboration Language Based on Data-Aware Processes**, pp. 18–36. Springer International Publishing, Cham (2016)
[7] Frantz, C.K., Nowostawski, M.: **From institutions to code: Towards automated gen- eration of smart contracts**. In: 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W). pp. 210–215 (Sept 2016)
[8] Clack, C.D., Bakshi, V.A., Braine, L.: **Smart Contract Templates: essential re- quirements and design options**. ArXiv e-prints (Dec 2016)
[9] Grigg, I.: **The ricardian contract**. In: Proceedings. First IEEE International Work- shop on Electronic Contracting, 2004. pp. 25–31 (July 2004)
[10] Grigg, I.: **On the intersection of Ricardian and Smart Contracts**. `http://iang.org/papers/intersection_ricardian_smart.html` (Feb 2017)
[11] English, M., Auer, S., Domingue, J.: **Block chain technologies & the semantic web: A framework for symbiotic development**. Tech. rep., Technical report, University of Bonn, Germany (2016)
[12] Asor, O.: **About Tau-Chain**. ArXiv e-prints (Feb 2015)