GRCTC
Governance, Risk & Compliance
Technology Centre

A Solution for the Problems of Translation and Transparency in Smart Contracts

Firas Al Khalil
Marcello Ceci
Leona O'Brien &
Tom Butler

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

NUI Galway
OÉ Gaillimh

UCD
DUBLIN

TECHNOLOGY CENTRE
ENTERPRISE IRELAND
IDA IRELAND SUPPORTED

February 2017

Table of Contents

About the Authors

Dr Firas Al Khalil is a post-doctoral researcher in semantic technologies at the GRC Technology Centre

Dr Tom Butler is Professor in Business Information Systems at University College Cork and GRC Technology Centre Principal Investigator.

Dr Marcello Ceci is a post-doctoral researcher in legal informatics and semantic technologies at the GRC Technology Centre

Leona O'Brien is Senior Legal Researcher at the GRC Technology Centre

Abstract

There is increasing interest in Distributed Ledger Technologies (DLT) across industry sectors. This is also true in the financial industry as established firms and FinTech companies seek to leverage the potentially disruptive nature of DLT to automate business processes, thereby reducing costs, and making products and services more information intensive, thereby improving business effectiveness and efficiency. Smart contracts are at the core of this potential innovation of DLT. Several implementations of distributed ledgers have been proposed, and different languages for the development of smart contracts have been suggested. However, as with the dot-com period, technologists are in the driving seat, which increases the likelihood of failed business implementations. We argue that too much attention is being given to the programming aspect of creating smart contracts by computer scientists, as opposed to upstream activities performed by lawyers, business practitioners, and regulators. This position paper argues that more attention should be paid to bridging the yawning gap between a smart contract's *legal semantics, business semantics* and *regulatory semantics* and its *denotational semantics*[1] and ensuring provenance, while guaranteeing the empirical fidelity of the *operational semantics*[2]. Simply put, it is the lawyers and financial professionals and not computer programmers who should be creating smart contracts.

Building on over 3 years of research at the GRC Technology Centre, we propose necessary and sufficient requirements for a standards-based, lawyer-friendly, human- and machine-readable contract authoring language. This can bridge the 'translation' gap and serve as a common, specification language for programmers, counterparties to a contract, and provide a means to ensure regulatory transparency and oversight.

1 Introduction

Distributed Ledger Technology (DLT) emerged due to the development of and relative success of Bitcoin [1]. Accordingly, it has sparked much interest in different communities — from academia to a variety of industry sectors, and from technological and financial spheres to philosophical, legal, and regulatory domains [2,3].

The enthusiasm generated around DLTs is indicative of the potential that exists and awaits to be realized. However, as with previous much-touted disruptive technologies, the business benefits of DLT may be elusive. While it is undeniable that the benefits to the financial industry of cryptocurrencies, such as Bitcoin, may be obvious, there may be other compelling business use cases of DLTs in the financial industry.

Which brings us to smart contracts. This concept was first envisioned by Nick Szabo [4 as far back as 1995, so is claimed. However, the advent of DLTs may make smart contracts a viable business proposition. Smart contracts have been defined in several ways that vary in their faithfulness to the original concept; however, some definitions merely add to

[1] Denotational semantics are concerned with the meaning of a computer program as a function that maps input into output.

[2] Operational semantics provide a formal description of the behaviour a computer program.

the confusion that exists around the concept.

Clack et al. [21] define *"A smart contract [as] an agreement whose execution is both automatable and enforceable. Automatable by computer, although some parts may require human input and control. Enforceable by either legal enforcement of rights and obligations or tamper-proof execution."*

Returning to the Rough Ground and to the seminal concept proposed by Nick Szabo, *"[s]mart contracts […] facilitate all steps of the contracting process";* Szabo correctly argues that search, negotiation, commitment, performance, and adjudication activities are all parts of the contracting process and should be represented [5]. However, we add regulatory transparency and oversight to this list of essential features and activities.

As a cryptocurrency technology platform, Bitcoin is capable of executing smart contracts, but with a lot of restrictions due to its limited scripting language. This limitation, along with the observation that cryptocurrencies can be viewed as *"just another kind of smart contracts"*, led eventually to the development of Ethereum [6]. Ethereum is a decentralized platform where smart contracts are 'first-class citizens'; the DLT in Ethereum is equipped with a Turing complete programming language that enables developers to write 'arbitrary' contracts in code. More recently, platforms built on top of Bitcoin and supporting a Turing-complete smart contracts language were developed (e.g. Rootstock [7]), and maybe more interestingly, platforms for smart contracts with non-Turing-complete languages were also developed, i.e. *τ-Chain* [8].

It is not a surprise that traditional programmers, if one may call them so, are unable to carry out "economical thinking" [9]; indeed, they are also, in our experience, ill-equipped to capture legal or regulatory thinking. The inverse can be said of subject-matter experts, i.e. business analysts and lawyers — they are most unlikely to carry out "computational thinking".

How then are we to effect the development of smart contracts in large financial institutions, where, traditionally, contracts are drafted by legal subject-matter experts? More importantly, how can we reason on the legality of the smart contracts and the accuracy of their *operational semantics*, either manually by a lawyer, or automatically using a tool for compliance checking? A failure to answer these questions inevitably contributes to the skepticism of the financial industry – which has been put in the rack by regulators since 2008 – about the future of smart contracts. To be sure DLTs are attractive for many reasons, but industry and the regulators may be reluctant to adopt this new paradigm.

In this position paper, we argue that smart contracts need to be transparent to stakeholders and have empirical fidelity with the hard copy versions on which they are based. There is need for clear provenance between the *operational semantics* of the smart contract executing on a DLT, its corresponding *denotational semantics, and* the *legal, business and regulatory semantics*[3] when drafted by a lawyer. This is a fundamental example of

[3] A financial instrument is a contract between counter parties. Contracts are drafted using legal prose. However, contracts also include business terms such as shares, bonds, interest rates and so on. In addition, contracts are subject to regulatory supervision. Thus, a smart contract will at base contain *legal* and *business semantics,* but may also include *regulatory semantics*.

the problem of requirements translation for which computer scientists do not address.

The rest of this paper is organised as follows: Section 2 illustrates the diversity of thought on distributed ledger technologies; Section 3 discusses the apparent irreconcilability of the languages and world-views of programmers and subject-matter experts; Section 4 develops our theory on how can we build a bridge that facilitates trust, from an institutional perspective, in smart contracts; we then offer concluding thoughts.

2 On Distributed Ledger Technologies

The introduction of Bitcoin by Satoshi Nakamoto [1] divided the financial industry from the outset. Supporters were extremely enthusiastic about it, to the point where they claimed that Bitcoin is the "next big thing", and detractors were extremely skeptical about the value of cryptocurrency and its underlying technology.

The core innovation of Bitcoin is not the cryptocurrency itself; it is the concept of a shared ledger. This proved to be a very powerful concept that generated the DLT paradigm and saw the emergence of rivals to Bitcoin. The interested reader can refer to Tschorsch and Scheuermann [10] for an excellent technical survey on DLTs. A brief consideration of the currently available DLT platforms inspired by Bitcoin provides good insights into the rising popularity of the technology: for instance, coinmarketcap.com is a site that tracks market capitalisation of different cryptocurrencies and lists 719 platforms.

Programming DLTs

Bitcoin includes a stack-based scripting language that allows computer scientists and software developers to define the conditions for Bitcoin expenditures (e.g. requiring multiple signatures). This revived the hope that smart contracts could find a suitable platform technology. However, Bitcoin's scripting language is purposefully not Turing-complete, which ultimately meant that it is limited in expressivity. In the following, we will take a look at four different platforms that are meant to overcome Bitcoin's scripting limitations, illustrating the different technical choices one can make, regarding the development of smart contracts and their *operational semantics*.

The first platform we are going to look at, which is currently almost synonymous with the term 'smart contract' is Ethereum [6]. Ethereum was proposed as a distributed platform independent of – yet very similar to – Bitcoin. To create distributed trust-less consensus and solve the double-spending problem, Ethereum uses proof-of-work, just like Bitcoin. The Ethereum Virtual Machine (EVM) runs a Turing-complete stack based language, which opens the doors to a hypothetically unlimited number of potential applications. However, developers are not forced to use the EVM's opcode to write smart contracts. Indeed, they can use Solidity or Serpent, which are high-level programming languages, similar to JavaScript or python, which can compile to EVM byte code.

Nxt is a second technology platform currently in use and is one of the earliest smart contract platforms. Unlike Bitcoin and Ethereum, Nxt uses proof-of-stake to achieve consensus and solve the double-spending problem. Moreover, Nxt does not provide a scripting language to smart contract developers; instead, it provides a RESTful API exposing a set of primitive operations (like spending, storing strings, sending messages, etc.) that developers can invoke.

The third platform in use is called Rootstock [7]. Unlike Ethereum and Nxt, Rootstock was developed to complement Bitcoin (as a sidechain [11]). It provides its own Turing-complete virtual machine (the RVM) to enable smart contracts.

The fourth and final platform we will examine is *τ−Chain* [8]. The designers of this technology platform argue that Turing-completeness is not necessary for distributed ledgers, because with Turing-completeness comes undecidability. What this means is that smart contracts can go in an infinite loop and the DLT network will never be able to predict this behaviour.

Ethereum overcomes the problem of undecidability by forcing the caller of the smart contract to provide *'gas'* with the transaction (bought with ether, Ethereum's own cryptocurrency); every instruction on the EVM consumes a predefined amount of *'gas'*, and they are non-refundable, i.e. if the *'gas'* is totally consumed and the smart contract didn't finish execution, the *'gas'* is never returned to the caller.

However, Asor [8] proposes the use of an ontology of rules [12], along with a reasoner, to enable computations on the network. Authors of smart contracts would write them in a totally functional programming language, like Idris [13], which will be ultimately translated into an ontology. This approach will not only make computations decidable, but it also allows the assertion of properties of smart contracts that were impossible with Turing-complete languages. For example, if the contract connects to the Internet or not, or if the contract fulfills some interfaces/requirements/etc.

The interested reader can refer to the survey by Seijas et al. [14] for more information on scripting languages for distributed ledgers. The aforementioned platforms illustrate some of the variations that exist in the distributed ledger technology's ecosystem. These platforms can differ not only in the tooling and the language they expose for smart contract writing, but also in the paradigms that govern them. The development of smart contracts thus requires a deep understanding of the target platform, to say nothing of the semantics of the legal, business, regulatory and denotational aspects of such contracts. In the following section, we will examine what hinders the adoption of such a technology by the financial industry.

3 The Translation Problem

This section deals with extant approaches to enabling software developers to author smart contracts, through specific tools or by creating abstractions. The point being made here is that the gap between the lawyer's semantics and the software programmer's *operational semantics* may bring unacceptable operational and regulatory risks.

Delmolino et al. [9] recently reported on their experiences in teaching smart contract programming, using Ethereum, to undergraduate students at the University of Maryland. The authors concluded that smart contract programming requires an "economic thinking" perspective that traditional programmers may not have acquired. Indeed, students repeatedly made (a) logical errors that ultimately lead to money leaks, (b) failed to use cryptographic primitives to secure the contracts from attackers, (c) failed to account for the incentives of contract callers, and (d) made errors related to Ethereum.

This observation lead to the development of a Master's thesis by Pettersson and Edström [15], and their objective was to help programmers to develop more accurate and risk-free smart contracts, although they would not have put it thus. Nevertheless, Pettersson and Edström aimed to prevent 3 kinds of errors developers make: (1) failure to account for unexpected states; (2) failure to use cryptography; and (3) overflowing the EVM's stack. They proposed the use of a functional programming language called Idris to help remediate the risks. In this scheme of things, Pettersson and Edström developed a code generator that transforms code produced by an Idris compiler to Serpent code, which can be subsequently compiled into EVM bytecode. This process does not, however, solve the translation problem.

In a different, yet related work, Luu et al. [16] noted that a class of security-related bugs in smart contracts are due to the gaps in the understanding of the distributed semantics of the underlying platform. Noting this problem aside, a solution is required.

Another interesting work is that of Florian et al. [17] who propose the use of logic-based smart contracts. They demonstrated that this approach can complement approaches where smart contracts are drafted in procedural code; where contracts are subject to negotiation, formation, storage, notarization, enforcement, monitoring and where dispute resolution is required.

In articulating a different approach, García-Bañuelos et al. [18] demonstrated how contracts expressed in the OMG's Business Process Modelling Notation (BPMN) can be mapped into executable smart contracts on Ethereum. This development lead Hull et al. [19] to propose a *'Business Collaboration Language'* (BCL) for shared ledgers. Indeed, BCL can be thought of as the equivalent of SQL for relational databases, albeit targeting shared ledgers, regardless of implementation-specific details.

As far as we know, the only works that consider the issue of authoring smart contracts from the subject-matter expert's perspective are those proposed by Frantz and Nowostawski [20] and Clack et al. [21].

Frantz and Nowostawski [20] propose a semi-automated method for the translation of human readable contracts to smart contracts on Ethereum. The authors developed a domain specific language for contract modelling, where rules expressed in plain English, and then translated into the Solidity vocabulary. However, this solution is anchored on Ethereum, and it is not clear how extensible or adaptable it is. In addition, it does not incorporate the semantics of the legal and business language a lawyer would use to draft the *denotational semantics*.

Clack et al. [21] identify two semantic dimensions to smart contracts:

i. *Operational semantics* are concerned with the execution of the contract on a specific platform.
ii. *Denotational semantics* attempt to capture and represent the "legal meaning" of the contract, as understood by a lawyer.

Clack et al. propose the use of smart contract templates, based on the idea of Ricardian Contracts [22, 23]. A Ricardian Contract is a digitally signed triple ⟨P,M, C⟩, where P is the legal prose (i.e. the *legal, business and regulatory semantics*) from which the *denotational semantics* may be captured and represented, M is a map (key-value pairs) of parameters used in P and C, and where C is the platform specific code that expresses *operational semantics*. The translation problem arises as computer scientists view the representation of legal and business prose and their semantics as being unproblematic.

While the use of smart contract templates, based on Ricardian Contracts, appears like a move in the right direction, we argue that prose should not be directly tied to code for the following reasons:

- While the semantics of legal and business language can be expressed as a set of deontic defeasible rules, the code is rather procedural. The order of the instructions in the procedure does not reflect the natural order of the contract clauses expressed in natural language [17].
- The life-cycle of legal and business prose is independent from the life-cycle of computer code. Take, for example, a lawyer might describe the terms of a contract in prose and never return to it, while a developer will, most likely, iterate through different implementations and variations (e.g. bug fixes).
- There does not exist a single smart contract platform. This ultimately means that different parameters (key-value pairs of M) will be needed for different platforms. For example, several works (e.g. [24, 25, 26]) describe data feed systems that enable smart contracts to consume data feeds from outside the distributed ledger (e.g. a stock market index). Thus, while the notion of an external feed might be familiar to a lawyer, its technical details, and thus the choices related to the adoption of one method over another, are beyond his interest or control.

In the following section, we will identify the key issues regarding the adoption and use of smart contracts. We then propose a solution to this problem.

4 The Problem of Trust and Transparency in Smart Contracts

Section 2 demonstrated, through a non-exhaustive list of examples, how distributed ledgers can differ technologically. This simple fact requires a software developer to possess a high degree of technological knowledge and skills in order to draft smart contracts. We also argued above that the current focus is on developing technical tools and infrastructure aimed at facilitating the machine implementation of smart contracts. However, there is a major lacuna in all this: that is the upstream translation or mapping of *legal* and *business semantics* to *denotational semantics*.

We agree with Clack et al. [21] on the need to address both *operational* and *denotational semantics* in smart contracts. However, we argue that trust, by all stakeholders, including regulators, in smart contracts can only stem from the ability of lawyers in financial institutions to understand, express, and ultimately validate the *denotational semantics* of a contract. However, we disagree with the assumptions of Clack et al. on the suitability of languages proposed to express the *legal* and *business semantics*. Here we refer to the assumption on the correspondence between a "legal language" and a "technical language". We argue currently no correspondence can be achieved and that a lawyer can neither understand, nor predict the behaviour of, the smart contract code, as there is no intermediate language that bridges the gap.

What is missing from the extant research literature, to say nothing of practitioner commentary, is the realisation that the involvement of a lawyer, especially in the heavily-regulated financial industry, in the authoring of contracts, not only smart contracts, is paramount. Why? A lawyer's knowledge of the explicit and implicit rights and obligations, counterparties, stakeholders, schedules and penalties, and regulations governing a financial contract has to be represented in the *denotational* and *operational semantics* of a smart contract

There are two scenarios where the lawyer's involvement in the process of smart contract creation and execution is unavoidable:

i. When the contract is partly fulfilled through code, because the lawyer can only validate its textual version [17], i.e. the prose.

ii. When assessing the compliance of the contract with regulations, from the point of view of both the legal requirements introduced by the regulation (e.g. on financial activities, anti-money laundering, or consumer protection), and of the effects that these regulations automatically bind to the contract (naturalia negotii [27]).

Therefore, we argue that smart contracts should be authored by both the lawyer and the developer. It is also clear that the interaction and communication between both actors should be governed by a *common language*. This should not be the *controlled natural language* of the computer scientist. Rather, the lawyer should author contracts in a controlled *legal natural language* (LNL) that is logical, clear, unambiguous, and comprehensible by a computer programmer, while being as close as possible to representing the *denotational semantics*. It could then be employed by the computer programmer as a specification guiding the technical implementation. This common controlled *legal natural language* should have the following properties:

- It should not alienate the lawyer; i.e. it should be as close as possible to the language of contracts s/he is used to.
- It should be expressive enough to allow the authoring of smart and "not-so-smart" contracts.
- It should possess an unambiguous grammar- the LNL should be mappable to a logical formalism, which will facilitate compliance checking with existing regulations.
- The concepts and actions described in the contract, i.e. the vocabulary, along with the clauses of the contract, i.e. the rules, should be shareable across the network, which is important for both discoverability and negotiation – two defining aspects of smart contracts – by human and autonomous agents.
- It should be able to represent the actions coded in the smart contract [21], the duties and powers arising from the contract [20], and the meta-rules governing it (e.g. regulation on financial activities, anti-money laundering or consumer protection).

In a previous work [28], we describe Mercury, a language whose purpose is to capture and represent regulations for compliance checking, among other use cases. We also developed a methodology [29] to capture the legal knowledge thus expressed and translate it to OWL [30]. Mercury is based on the

Semantics of Business Vocabulary and Business Rules [31] (SBVR) specification, but the language of smart contracts will require, we believe, a further extension of SBVR e.g. to capture the powers arising from the contract. The SBVR-based LNL should be mapped to a logical formalism, e.g. OWL, where reasoning on compliance is feasible.

In a recently published technical report, English et al. [32] investigated how distributed ledger technologies and the Semantic Web can interact with one another. Indeed, a blockchain can provide secure resource identifiers (by ensuring authenticity, human-readability, and decentralisation), and ontologies based on OWL can provide a unified way to understand blockchain concepts between humans, and exposing blockchain data according to an ontology enables the interlinking with other linked data and to perform reasoning.

Our proposal improves transparency, which is one of the major qualities of distributed ledgers, and which is also a determining factor of the trust-less trust in a blockchain network. But problems arise when it comes to trust in the fact that the contract, as written by the lawyer, was correctly translated into code: that is, the trust whether the *operational semantics* faithfully represent the *denotational semantics* and whether these in turn capture the meaning of the *legal, business* and *regulatory semantics*.

One may argue that trust can only be guaranteed if there is a mechanism $\mathcal{G}$ that enables code to be generated from prose expressed in a LNL and/or a mechanism $\mathcal{C}$, potentially the inverse of $\mathcal{G}$, which proves the correspondence of the code to the LNL and the contract prose. However, a closer inspection of the literature illustrates that:

a. There is evidence that $\mathcal{G}$ and $\mathcal{C}$ can exist, especially from [17] and τ-Chain [8]. Indeed, if the vision of τ-Chain is possible, then there is an opportunity to go directly from *legal* and *business semantics* to *denotational semantics* and then to *operational semantics* using our approach. However, this may imply the restriction of said trust to one specific distributed ledger technology.
b. It is not really clear, at least for us, if $\mathcal{G}$ and $\mathcal{C}$ exist for shared ledgers that use stack-based languages. This is an open question that deserves closer attention, and can have one of two clear answers:
    i. It is possible or practically feasible, which is great news for everyone, or
    ii. It is impossible or practically infeasible. Then it is only reasonable to ask: is the existence of $\mathcal{G}$ and $\mathcal{C}$ a prerequisite for the establishment of said trust? We conjecture that it is not, for two reasons:
c. The implementation processes of existing financial contracts in the form of software is already opaque, especially to the consumer, and our proposed approach would only facilitate transparency.
d. Trust can be assured through the establishment of reputation: the better you are in effectively transforming your specification to code, the more reputable you are; the more reputable you are, the more trustworthy you are perceived to be.

The next and final section provides concluding elaborations of our thesis.

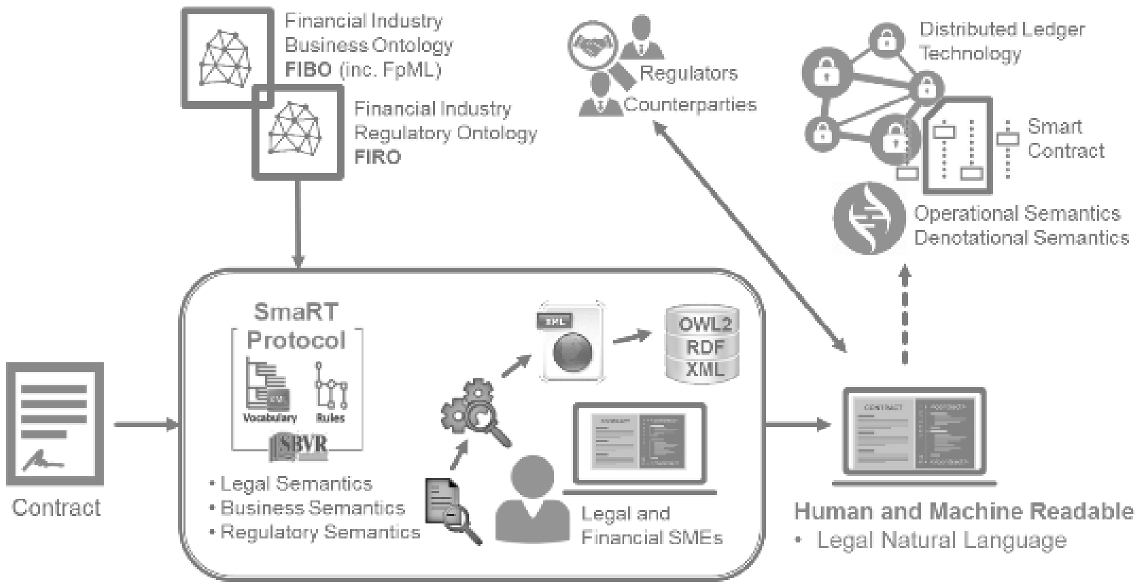

Figure 1 A Solution for the Translation and Transparency Problems with Smart Contracts

5 Conclusions

In this position paper, we reasoned that regulatory, legal and counterparty trust in smart contracts can be achieved. It is true that cryptographic guarantees are enablers of, and integral to, trust in distributed ledger technology (DLT), but we argue that another kind of trust is needed — one that is established by a process that involves lawyers expressing the *legal* and *business semantics* of contracts using the precepts of ISO *common logic*. We have in our Mercury platform demonstrated practically how this can be achieved when it comes to expressing *regulatory semantics* in a controlled regulatory natural language (RNL) using *deontic* and *alethic logic*.

Above we demonstrated how DLTs can vary significantly at a technical level. This has led to the development of tools and abstractions to help developers to program smart contracts. All this is essential for a functioning technological ecosystem. However, we also illustrated that extant research does not take into account the issue of *legal, business* or *regulatory semantics* that govern compliance with existing and ever increasing regulations.

To that end, we proposed the criteria required for a natural language to express smart contracts, which has empirical fidelity with the *legal* and *business* prose or *semantics*. These criteria have transparency at their core. We also point to a practical and achievable solution to the problem of translating the prose of a contract and expressing *legal, business* and *regulatory semantics* using an approach based on the Object Management Group's SBVR specification, which is rooted in *common logic.* The output of this solution sees the *legal semantics,* and related *business* and *regulatory semantics* expressed in a language—the controlled Legal Natural Language—that is both human and machine readable.

Figure 1 presents a model of our schema. It is technologically feasible. This is, we believe, the missing piece, the key to solving the problems of transparency, trust and translation in smart contracts. Our Mercury-based SmaRT Protocol is, in essence, the Rosetta Stone that will ensure complete *trust* in autonomous or semi-autonomous smart contracts.

References

[1]S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[2]W. Reijers, F. O’Brolcháin, and P. Haynes, “Governance in blockchain technologies & social contract theories,” Ledger, vol. 1, no. 0, pp. 134–151, 2016.

[3]M. Swan, “Blockchain temporality: Smart contract time specifiability with blocktime,” in Rule technologies. research, tools, and applications: 10th international symposium, ruleML 2016, stony brook, nY, uSA, july 6-9, 2016. proceedings, J. J. Alferes, L. Bertossi, G. Governatori, P. Fodor, and D. Roman, Eds. Cham: Springer International Publishing, 2016, pp. 184–196.

[4]N. Szabo, “The Idea of Smart Contracts.” https://web.archive.org/web/20160831070942/http://szabo.best.vwh.net/smart_contracts_idea.html, 1997.

[5]N. Szabo, “Formalizing and Securing Relationships on Public Networks.” https://web.archive.org/web/20050217172626/http://www.firstmonday.dk/ISSUES/issue2_9/szabo/index.html, 1997.

[6] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” Ethereum Project Yellow Paper, vol. 151, 2014.

[7] S. Demian Lerner, “Rootstock. bitcoin powered smart contracts. white paper.” Nov. 2015.

[8] O. Asor, “About Tau-Chain,” ArXiv e-prints, Feb. 2015.

[9] K. Delmolino, M. Arnett, A. E. Kosba, A. Miller, and E. Shi, “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 workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, revised selected papers, 2016, pp. 79–94.

[10]F. Tschorsch and B. Scheuermann, “Bitcoin and beyond: A technical survey on decentralized digital currencies,” IEEE Communications Surveys and Tutorials, vol. 18, no. 3, pp. 2084–2123, 2016.

[11]S. Demian Lerner, “Drivechains, sidechains, and 2-way hybrid peg designs,” Jan. 2016.

[12]“OWL 2 Web Ontology Language Document Overview (Second Edition).” https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[13]E. BRADY, “Idris, a general-purpose dependently typed programming language: Design and implementation,” Journal of Functional Programming, vol. 23, no. 5, pp. 552–593, Sep 2013.

[14]P. L. Seijas, S. Thompson, and D. McAdams, “Scripting smart contracts for distributed ledger technology.” Cryptology ePrint Archive, Report 2016/1156, 2016.

[15]J. Pettersson and R. Edström, “Safer smart contracts through type-driven development,” PhD thesis, Master’s thesis, Dept. of CS&E, Chalmers University of Technology & University of Gothenburg, Sweden, 2015.

[16]L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, “Making smart contracts smarter,” in Proceedings of the 2016 aCM sIGSAC conference on computer and communications security, 2016, pp. 254–269.

[17]F. Idelberger, G. Governatori, R. Riveret, and G. Sartor, “Evaluation of logic-based smart contracts for blockchain systems,” in Rule technologies. research, tools, and applications: 10th international symposium, ruleML 2016, stony brook, nY, uSA, july 6-9, 2016. proceedings, J. J. Alferes, L. Bertossi, G. Governatori, P. Fodor, and D. Roman, Eds. Cham: Springer International Publishing, 2016, pp. 167–183.

[18]L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber, “Optimized Execution of Business Processes on Blockchain,” ArXiv e-prints, Dec. 2016.

[19]R. Hull, V. S. Batra, Y.-M. Chen, A. Deutsch, F. F. T. Heath III, and V. Vianu, “Towards a shared ledger business

collaboration language based on data-aware processes," in Service-oriented computing: 14th international conference, iCSOC 2016, banff, aB, canada, october 10-13, 2016, proceedings, Q. Z. Sheng, E. Stroulia, S. Tata, and S. Bhiri, Eds. Cham: Springer International Publishing, 2016, pp. 18–36.

[20]C. K. Frantz and M. Nowostawski, "From institutions to code: Towards automated generation of smart contracts," in 2016 iEEE 1st international workshops on foundations and applications of self* systems (fAS*W), 2016, pp. 210–215.

[21]C. D. Clack, V. A. Bakshi, and L. Braine, "Smart Contract Templates: essential requirements and design options," ArXiv e-prints, Dec. 2016.

[22]I. Grigg, "The ricardian contract," in Proceedings. first iEEE international workshop on electronic contracting, 2004., 2004, pp. 25–31.

[23]I. Grigg, "On the intersection of Ricardian and Smart Contracts." http://iang.org/papers/intersection_ricardian_smart.html, Feb-2017.

[24]F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in Proceedings of the 2016 aCM sIGSAC conference on computer and communications security, 2016, pp. 270–282.

[25]"PriceFeed smart contract." http://feed.ether.camp/.

[26]"Oraclize: 'The provably honest oracle service'." http://www.oraclize.it/.

[27]A. Frignani, "Some Basic Differences between the Common Law and the Civil Law Approach." http://www.jus.unitn.it/CARDOZO/Review/Comparative/Frignani-1997/Washingt.htm, 1996.

[28]M. Ceci, F. Al Khalil, and L. O'Brien, "Making Sense of Regulations with SBVR," in RuleML 2016 challenge, doctoral consortium and industry track hosted by the 10th international web rule symposium (ruleML 2016), 2016.

[29]E. Abi-Lahoud, L. O'Brien, and T. Butler, "On the road to regulatory ontologies," in AI approaches to the complexity of legal systems: AICOL 2013 international workshops, aICOL-iV@IVR, belo horizonte, brazil, july 21-27, 2013 and aICOL-v@SINTELNET-jURIX, bologna, italy, december 11, 2013, revised selected papers, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 188–201.

[30]F. Al Khalil, M. Ceci, K. Yapa, and L. O'Brien, "SBVR to oWL 2 mapping in the domain of legal rules," in Rule technologies. research, tools, and applications: 10th international symposium, ruleML 2016, stony brook, nY, uSA, july 6-9, 2016. proceedings, Springer International Publishing, 2016, pp. 258–266.

[31]"Semantics of Business Vocabulary and Business Rules (SBVR) Version 1.3." http://www.omg.org/spec/SBVR/1.3/PDF; Object Management Group, May-2015.

[32]M. English, S. Auer, and J. Domingue, "Block chain technologies & the semantic web: A framework for symbiotic development," Technical report, University of Bonn, Germany, 2016.