10-2017

Smart contracts: Terminology, technical limitations and real world complexity

Eliza MIK
Singapore Management University, elizamik@smu.edu.sg

DOI: https://doi.org/10.1080/17579961.2017.1378468

Citation

MIK, Eliza. Smart contracts: Terminology, technical limitations and real world complexity. (2017). *Law, Innovation and Technology*. 9, (2), 269-300. Research Collection School Of Law.
Available at: https://ink.library.smu.edu.sg/sol_research/2341

Smart Contracts: Terminology, Technical Limitations and Real World Complexity

Dr Eliza Mik*

Please note that this document constitutes a production copy of a paper to be published in *Law, Innovation & Technology* (2017) 9.2

ABSTRACT

If one is to believe the popular press and many "technical writings," blockchains create not only a perfect transactional environment but also obviate the need for banks, lawyers and courts. The latter will soon be replaced by smart contracts: unbiased and infallible computer programs that form, perform and enforce agreements. Predictions of future revolutions must, however, be distinguished from the harsh reality of the commercial marketplace *and* the technical limitations of blockchain technologies. The fact that a technological solution is innovative and elegant need not imply that it is commercially useful or legally viable. Apart from attempting a terminological "clean-up" surrounding the term smart contract, this paper presents some technological and legal constraints on their use. It confronts the commonly made claims concerning their ability to automate the transacting process and to ensure perfect performance. It also examines the possibility of reducing contractual relationships into code and the ability to integrate smart contracts with the complexities of the real world. A closer analysis reveals that smart contracts can hardly be regarded as a semi-mythical technology liberating the contracting parties from the shackles of traditional legal and financial institutions. While some of their technical features seem *prima facie* attractive, especially to non-lawyers, a closer analysis reveals their many shortcomings.

1. Introduction

Purportedly, smart contracts are contracts that are represented in code and executed by computers. They are not only formed online but their very performance is enabled and *guaranteed* by a network of decentralized, co-operating computer nodes, known as blockchains. Originally, smart contracts were contemplated within a limited range of transactions, predominantly financial instruments. Progressively, however, the surrounding narrative has become broader, implying that all contracts can be made smart or that many different obligations can be enforced by code. What started as a niche phenomenon in such areas as financial derivatives and prediction markets, is now poised to change the entire legal landscape and "revolutionize" commerce.

Allegedly, smart contracts can streamline the contracting process, reduce transaction costs by eliminating intermediaries and, most importantly, simplify enforcement by obviating the need to seek protection from traditional legal institutions, such as courts. The theories underpinning smart-contracts and blockchains combine multiple, interrelated threads all of which reflect an indiscriminate, if not irrational, fascination with certain technical characteristics of blockchains. They also reflect a surprising lack of trust in humans. As the latter are perceived as inherently biased and unreliable, things should be left to computers. Humans, especially bankers and judges, are fallible and *not* trustworthy.[1] Computers, on the other hand, are objective, infallible and trustworthy.[2] The very idea of smart contracts is thus inextricably tied to the elimination of human judgement, the reduction of dependence on financial intermediaries and, in many instances, a detachment from the legal system.[3] In other, more commercially-oriented contexts, smart contracts can simply be seen as part of the broader trend to use technology to ensure a consistent application of legal rules and agreements.

The legal analysis of smart contracts is rendered difficult by the fact that the phenomenon originated in technical writings, which are characterized by an inconsistent and incorrect use of legal terms. Given the complexity of the technologies underlying smart contracts (distributed networks and asymmetric cryptography, amongst others) it is also difficult to evaluate many claims concerning their actual capabilities and real potential to change (speak: *revolutionize*) the commercial and legal landscape. One is often left with a common-sense estimate of what is (or can be) technologically viable and what is legally permissible or necessary. To complicate matters, the smart contract narrative is often laden with ideologically charged arguments that associate certain technological features of blockchains (e.g. decentralized consensus) with broader social and economic issues, such as the disenchantment with financial institutions or the (perceived) lack of trust in the legal system. Many claims made in technical writings are tainted by the assumption that certain technological features (decentralization, again…) are absolute values and must be preserved at any cost. In the same vein, it is often assumed that because a particular technology is innovative or revolutionary, it is also commercially useful or capable of solving actual legal problems. This, however, is often not the case. Arguably, the entire idea of smart *contracts* may be the result of a series of terminological misunderstandings. At the same time, assuming that in some instances smart contracts *are* or *represent* contracts in the legal sense, their practical deployment may raise some interesting issues that transcend the simple question whether it is technically and legally possible to automate the contracting process.

Roadmap

This paper is structured as a high-level introduction to some basic issues concerning the legal and (to an extent) technical viability of smart contracts. The analysis aims to

* I wish to thank Ernie Teo and Pralhad Deshpande (from IBM research), Jens Krebs, John O'Brien, the coding teams at Attores and Otonomos, especially Gaurang Torvekar and Mano Thalanaban, Jason Teutsch (Truebit), as well as my SMU colleagues Kelvin Low and Simon Schillebeeckx; all mistakes are mine.

[1] For a humorous description see: Schumpeter, Not-so-clever contracts (2016) The Eonomist, at: www.economist.com/news/business/21702758-time-being-least-human-judgment-still-better-bet-cold-hearted

[2] See generally: Gavin Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger (2015), who proclaims the impartiality of autonomous enforcement by code, 1.

[3] Joshua A.T. Fairfield, Smart Contracts, Bitcoin Bots and Consumer Protection (2014) 71 Washington and Lee Law Review Online 35, 37-38.

remain broad and generic, with few references to specific industries or specific instantiations of smart contracts. Its contents and sequence are largely dictated by the popular claims made in technical writings. The discussion commences with an exploration of the very concept of smart contracts, their relationship with blockchains and the core terms surrounding their use, such as “trustlessness,” “self-enforcement” and “validation.” Consequently, the first part of the paper can be regarded as a terminological “clean-up” followed by some technical explanations. This part also inquires whether smart contracts *are* or *can be* legally enforceable. The second part examines two problem areas, which can be reduced to the following questions:

(a) What are the implications of the fact that smart contracts must be expressed in code?

(b) Can the benefits of the blockchain be preserved if most smart contracts necessitate a connection with the real world?

Finally, readers should be aware of the following caveats with regard to the scope and depth of the paper. *First*, bypassing the regulatory aspects of blockchains, this paper does not address the question whether bitcoins or any other crypto currencies (“tokens”) constitute legal tender. It assumes, however, that the use of such tokens is legal and both parties to the smart contract agree to accept such as part of the exchange. After all, each blockchain-based smart contract involves the transfer of crypto-tokens. *Second*, the term “technical writings” denotes various resources, such as white papers or blogs, many of which would not withstand a peer-review process but which are regarded as authoritative by the tech-community. Notably, there is no single source about smart contracts or blockchains comparable to Requests for Comments, which provide reliable descriptions of the core protocols and technologies of the Internet. Consequently, all researchers in this area face the challenges of finding reliable information and reconciling inconsistent terminology. *Third*, *some* technical simplifications are necessary. One could, for example, indulge in detailed explanations of the mining process, delve into comparisons between different blockchains or endlessly dissect the concept of “validation.” While some detail is unavoidable, this paper aims to convey broad ideas without going into technological minutiae. It is acknowledged that there are different blockchains, different smart contracts and different ways in which smart contracts relate to or interact with their underlying blockchains. Some descriptions may fit one configuration, but not another.[4]The difficulties of making generalized statements are particularly acute in light of the division into permissionless and permissioned blockchains. Fourth, the paper does not address the problems of confidentiality, privacy or identification.

2. Terminology & Technicalities

There are multiple definitions of smart contracts. Some of them are purely technical and associate smart contracts with pieces of autonomous code operating on a blockchain[5] or with “systems which automatically move digital assets according to

[5] Wood (n 2), 15.

arbitrary pre-specified rules."[6] Other definitions associate smart contracts with the formalized expression and automated execution of legal contracts, with the use of code to perform contractual agreements,[7] with protocols that "facilitate, verify, execute or embody the terms of a contract"[8] or with the embedding of legal terms in hardware and software to prevent breach or to control assets by digital means.[9] Another group of definitions commences with a technical description only to observe that the given protocol will have serious legal implications.[10] In some instances the term "contract" is used informally, with no claims being made as to its legal significance; in others, technical writings take the "contract" terminology seriously and theorize that smart contracts *in general* will obviate the need for lawyers and judges by automating and guaranteeing contractual performance. Those making such claims, however, often use the term "contract" so liberally that it loses any resemblance to its original definition - that of a *legally enforceable agreement.* One must wonder: once a linguistic clean-up is completed and once it becomes apparent that many smart contracts are not contracts, will there be anything left worth discussing from a legal perspective? After all, if "smart contracts" are nothing but programs that run on a blockchain, there is no need for lengthy academic papers debating their legal implications. The unfortunate terminology is attributable to a paper by Szabo, which describes smart contracts as follows:

> *Smart contracts combine protocols with user interfaces to formalize and secure relationships over computer networks. Objectives and principles for the design of these systems are derived from legal principles, economic theory, and theories of reliable and secure protocols.*[11]

The paper elaborates that smart contracts "utilize protocols and user interfaces to facilitate all steps of the contracting process," including negotiation, performance, and adjudication.[12] Some of the propositions made therein, such as the need to develop new digital institutions, reflect the popular fascination with the "digital revolution," which characterizes *all* early cyberlaw scholarship. They exemplify naïve notions of law, including the perceived inadequacy of "old" (i.e. *pre*-Internet) systems, and the assumption that technological progress can remedy all problems – including those inherent in the operation of the legal system. Given that the "seminal" paper was written in the mid-90s, it does not surprise that it portrays courts and legal principles as inconvenient legacies to be made redundant by suitable technologies. Other propositions made therein, such as the use of technology to secure contractual performance or to ensure adherence to the law, have evolved into the theory that "code is law" and that technology has normative implications.[13] Using technology to enforce the law or private agreement is thus not a novel idea. What is new in the smart contract narrative, however, is the combination of an indiscriminate

[6] Vitalik Buterin Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform (2015).
[7] Josh Stark, How Close Are Smart Contracts to Impacting Real-World Law? (2016) www.coindesk.com/blockchain-smarts-contracts-real-world-law
[8] T. Swanson. Great chain of numbers: A guide to smart contracts, smart property and trustless asset management, 2014. ("*Swanson*") 11, 16
[9] Nick Szabo, Smart Contracts: Formalizing and Securing Relationships on Public Networks (1997) 2 (9) *First Monday* 2.
[10] see e.g. Fan Zhang, et al., Town Crier: An Authenticated Data Feed for Smart Contracts (2016) 1, who define smart contracts as programs that execute autonomously on the blockchain, only to imply that such programs execute the terms of a contract and enforce payments that are due under such contract.
[11] Szabo (n 9) 1.
[12] Szabo(n 9) 2.
[13] Lawrence Lessig, *Code version 2.0* (Basic Books 1999); Roger Brownsword, *Rights, Regulation, and the Technological Revolution* (Oxford University Press, 2013); Frank Pasquale, 'Technology, Competition and Values' (2007) 8 *Minn. J.L. Sci. & Tech.* 607

trust in technology, especially blockchains, with an unparalleled misapprehension of basic legal concepts. The “seminal” paper itself abounds in legal terminology, creating an impression that its propositions are grounded on solid legal principles. Most concepts described therein are, however, misrepresented. What follows is a morass of technological and legal jargon, which is endlessly recycled in subsequent technical writings. The latter are oblivious to the differences between private and public law, between substance and evidence as well as between contract and property law. Given this terminological confusion, it has been suggested to differentiate between smart contracts, which are pure code or computer programs, and smart contracts in the legal sense. [14] While such distinction is *prima facie* attractive, it may be difficult to sustain as even smart contracts in the legal sense may contain smart contract code. Moreover, technical writings rarely acknowledge this division and abound in claims that even smart contracts which are computer programs may have far-reaching legal implications.

Additional misunderstandings derive from the fact that nearly all technical writings regard vending machines as early examples of smart contracts. The resulting theory is that vending machines not only automate the transacting process but also instantiate contractual terms in their hardware. This in turn renders the terms immutable and guarantees that the contract is executed as coded. From a legal perspective, however, a vending machine is not a contract but an offer made to the world at large.[15] The offer is made by the vendor, who uses the machine to display his goods to the public. A contract is formed with whoever selects one of the available options and inserts the required sum. Unquestionably, the vending machine can automate both the formation and the performance of a contract, usually a sale of goods. It ensures that transactions can occur in only one particular manner and, vandalism or malfunction aside, guarantees that the goods are dispensed only to those who provide payment. In this sense, it ensures *perfect* performance. The same could be said of many e-commerce websites, such as Amazon.com or Spotify, which automate contract formation and, whenever the contractual subject matter is digital (e.g. music, eBooks) also the performance of the contract. Embedding business logic in hardware or software does not, however, transform such hardware or software into a contract - just like the automated release of goods by vending machines does not constitute a reification of a contract of sale. Neither vending machines nor websites *are* or *enforce* contracts. They can only dispense goods (or provide access to online content) in response to payment. Moreover, vending machines are technically incapable of embodying (and hence automating) *all* terms of the transaction, such as exclusion clauses, warranties of fitness or suitability for purpose. All terms of the transaction can be displayed *on* the machine – but very few can be instantiated b*y* the machine. In sum, the vending machine example taints most technical writings with the incorrect assumption that the automation of a transaction (or certain stages thereof) transforms the automaton into a smart contract or makes the transaction itself smart.

3. Blockchains

Although the idea of smart contracts predates blockchains, the latter have sparked a new interest in the area. Blockchains are commonly associated with crypto-

[14] See: J. Stark. Making sense of blockchain smart contracts (2016) www.coindesk.com/making-sense-smart-contracts

[15] *Thornton v Shoe Lane Parking Ltd* [1971] 2 QB 163

currencies, such as bitcoins, but it is necessary to distinguish between the two. Blockchains are increasingly, recognized as a generic technology that can be deployed for other purposes, such as a payment network, a platform for asset and supply chain management or a technology facilitating recordkeeping.[16] There are hundreds (if not thousands) of different blockchains that often significantly diverge from the original bitcoin blockchain and share very few of its characteristics.[17] In many instances, it seems more correct to speak of distributed ledger technologies, or "DLTs, which denote a broader category of geographically replicated, synchronized and often decentralized data stores.[18] It is also necessary to distinguish between permissionless and permissioned blockchains as it is only the former that are decentralized, visible to all and devoid of any access restrictions.[19] In principle, many arguments made in the context of permissionless blockchains (such as Bitcoin or Ethereum) lose their validity in the context of permissioned blockchains. In fact, the latter have developed specifically in response to the shortcomings of the former. Such developments have, however, resulted in the "loss" of some of the original characteristics of the blockchain, such as decentralization. Unless indicated otherwise, this paper refers to the original bitcoin blockchain. It is the characteristics of the latter that have spurred revolutionary, if not anarchistic, theories about blockchains "changing the world of commerce" and "the fabric of society".[20] Moreover, the original bitcoin blockchain best illustrates the limitations of this technology.[21] And so, in laymen's terms, the blockchain can be described as a decentralized, peer-validated crypto-ledger that provides a publicly visible, chronological and permanent record of all prior transactions. It resembles a spreadsheet that anybody can add a row to, but cannot otherwise update or delete anything.[22] The original blockchain was devised for a single purpose: the prevention of double spending of cryptocurrencies in a system without a centralized entity controlling the issuance or transfer of such currencies.[23] The problem of double spending was solved by making *all* transactions, both current and past, visible to everybody.[24] The original bitcoin paper did not envisage smart contracts and did not use the blockchain for anything else than the generation and transfer of tokens. In contrast, the Ethereum blockchain that was specifically developed to enable smart contracts.[25] The important point is that there are many different blockchains with many different characteristics.

[16] Trevor I. Kiviat, Beyond Bitcoin: Issues In Regulating Blockchain Transactions (2015) 65 Duke L.J. 569, 575; Jeanne L. Schroeder, Bitcoin and the Uniform Commercial Code (2016) 24 U. Miami Bus. L. Rev. 1 at 65; Joshua A.T. Fairfield, *Bitproperty*, 88 S. CAL. L. REV. 805 (2015).

[17] Melanie Swan, Blockchain: Blueprint for a New Economy (2015) O'Reilly Media, Sebastopol, 9.

[18] *Distributed Ledger Technology: beyond block chain,* UK Government, Office of Science (2016).

[19] See generally: *Tim Swanson, Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems* (2015); for a comparison between the Bitcoin and Ethereum blockchains and a permissioned blockchain, see: Richard Gendal Brown, et al., *Corda: An Introduction* (2016) 13, 14.

[20] Don Tapscott, Alex Tapscott, Blockchain Revolution (2016, Penguin); William Mougayar, The Business Blockchain (2016, John Wiley, Hoboken); WEF

[21] Arguably, these very limitations gave raise to the creation of Ethereum and permissioned blockchains.

[22] Kiviat (n 16) 578.

[23] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System* (2008) 1.

[24] Nakamoto (n 23) 2.

[25] The Ethereum foundation website describes it as an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology. Unlike the Bitcoin protocol, Ethereum was designed to be adaptable and flexible. a next-generation blockchain that had the ambitions to implement a general, fully trustless smart contract platform. Moreover: "The Ethereum platform itself is featureless or value-agnostic. Similar to programming languages, it is up to entrepreneurs and developers to decide what it should be used for. However, it is clear that certain application types benefit more than others from Ethereum's capabilities. Specifically, ethereum is suited for applications that automate direct interaction between peers or facilitate coordinated group action across a network. For instance, applications for coordinating peer-to-peer marketplaces, or the automation of complex financial contracts. Bitcoin allows for individuals to exchange cash without involving any middlemen like financial institutions, banks, or governments. Ethereum's impact may be more far-reaching. In theory, financial interactions or exchanges of any complexity could be carried out automatically and reliably using code running on Ethereum. Beyond

To understand the implications *and limitations* of blockchains in the context of smart contracts, it is necessary to indulge into some technical explanations. As the name implies, the blockchain is made of interconnected blocks. Each block contains a list of all prior transactions. The term "transaction" denotes the transfer of tokens (e.g. bitcoins) from one account to another. In most contexts, it has a narrow and technical meaning, limited to shifts of tokens between accounts. The creation of each block requires a significant amount of computation ("mining"). To create a block and append it to the blockchain, each mining node (i.e. participant in the network) must provide a "proof-of-work:" a piece of data which is computationally difficult to produce but easy for other nodes to verify.[26] As these computations are extremely expensive in terms of electricity costs, it is more economical to produce new valid blocks (i.e. those that follow the rules) than to commit resources to corrupting the blockchain by changing old blocks. Given the cost and difficulty of retrospectively changing the existing blockchain, the possibility of a transaction being altered or reversed is infinitesimal.[27] Consequently, the blocks are mathematically *chained* together and guarantee that a transaction cannot be modified without modifying the block that records it and all following blocks.[28] The blockchain can thus be regarded an incorruptible record of all prior transactions, as one source of truth visible to all. It must be further noted that (a) the mining process relies exclusively on the computing power of the individual nodes and not to any particular skill or knowledge on the side of the node's operator, i.e. the miner; (b) smart contracts concern transactions, not the generation of new tokens. At a basic level then, smart contracts can be regarded as self-executing ledger-modification instructions, e.g. "if X occurs, send Y amount of tokens from public address A to public address B."

Being "trustless"

The fascination with the blockchain derives from the fact that it establishes the truth of an event without recourse to a trusted third party in an adversarial environment where no-one can be trusted.[29] The truth of an event, i.e. the creation and/or transfer of tokens, is established by means of "distributed consensus," i.e. the confirmation by a majority of nodes in a decentralized network that a given block has completed the proof-of-work. Consequently, the blockchain itself is "trustless" because it creates and confirms a certain state of affairs and replaces the need to trust third parties with the ability to trust the technology itself. Trustlessness lies at the core of all theories that associate blockchains with radical disintermediation. From its inception, online commerce has been disadvantaged by the inherently insecure and unreliable character of the Internet.[30] The latter was not designed as a transactional

financial applications, any environments where trust, security, and permanence are important – for instance, asset-registries, voting, governance, and the internet of things – could be massively impacted by the Ethereum platform." http://ethdocs.org/en/latest/introduction/what-is-ethereum.html.

[26] Miners are incentivized to add new blocks by obtaining bitcoins (when *their* block is added to the blockchain) and transaction fees (when they include a particular transaction in their block); it must be noted that proof-of-work-based consensus is only indispensable in permissionless ledgers where the parties cannot trust each other, for an overview of different types of consensus see: *Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus* (2016) Hyperledger Architecture, Volume 1, 3-4.

[27] : Andreas M. Antonopoulos, *Mastering Bitcoin* (O'Reilly, Sebastopol 2015) ("*Antonopoulos*") 162

[28] Böhme, Rainer, Nicolas Christin, Benjamin Edelman, and Tyler Moore, "Bitcoin: Economics, Technology, and Governance," Journal of Economic Perspectives (2015) 29, 213-238.

[29] The problem is commonly referred to as the "Byzantine Generals Problem;" see: Leslie Lampert et al., *The Byzantine Generals Problem*,(1982) 4 ACM TRANSACTIONS ON PROGRAMMING LANGUAGES AND SYSTEMS at 382.

[30]Marjory Blumenthal, David D Clark, 'Rethinking The Design Of The Internet: The End-To-End Arguments Vs. The Brave New World' (2001) 1 ACM transactions on Internet technology 70, 80.

platform and its open nature is particularly unfavorable for transactions between strangers who cannot trust each other because they have limited means to verify each other's identities or ensure payment. These shortcomings of the Internet have given raise of a host of online intermediaries, such as eBay, paypal or Amazon, which provide a secure but closed transacting environment controlled by a single entity. In effect, online transactions are heavily mediated due to the need to compensate for the technical deficiencies of the internet. In contrast, as the blockchain is inherently incorruptible and secure, it is claimed that it has the potential to become a transacting platform with no intermediaries. After all, if the platform itself is trustless, there is no need for any third parties to absorb or reduce the transactional risks that are present on an insecure platform. On a trustless platform, such risks are simply absent.

The above theories are, however, only partially correct. While it is true that the Internet does not constitute a technologically perfect transacting environment, it is incorrect to portray blockchains (especially permissionless ones) as such. The more so, that once the blockchain is represented as a transacting platform, it is frequently implied that *everything* that connects *to*, operates *on* or is embedded *in* the blockchain becomes trustless, incorruptible and secure. Hence, "putting a smart contract on the blockchain" eliminates the need for the parties to trust each other or to rely on intermediaries. The very nature of the blockchain ensures that the smart contract cannot be altered and, as that neither party can influence its execution, its performance is guaranteed. It is generally overlooked, however, that the attributes of the blockchain (especially trustlessness) are not inherited by the computational processes or events that occur outside of its cryptographically secure domain. The fact that "something" connects to the blockchain or writes data into a block does not mean this "something" is or becomes trustless.

The fascination with decentralized consensus (or with decentralization in general) detracts attention from the fact that the original blockchain is "only" a *database* of transactions - not a transaction platform or a transacting environment. Databases, with limited exceptions, do not perform any computations [31] and have limited transactional capabilities. The blockchain is the output of a computationally intensive process but does not perform any complex computations itself. Apart from a limited number of native scripts, no code executes *within* the blockchain. In fact, its trustlessness derives from the fact that it does not perform complex computations and accepts extremely limited external inputs. As a result, the original blockchain cannot be regarded as a transaction platform – unless the term transaction is interpreted very narrowly, as the movement of tokens between accounts.[32] If the blockchain were become a platform for more complex types of transactions, it would be necessary to extend its functionalities. This, in turn, would necessitate the addition of protocol layers or scripts *on top* of it or – the creation of a new blockchain.[33] Such additions may not, however, be the product of distributed consensus and hence be "trustless." Depending on the *type* of blockchain and the *type* of smart contract, smart contracts

[31] It is debateable whether stored procedures constitute computations or not.

[32] At present, there are only five types of standard transactions, based on the limited number of scripts that can be processed by a bitcoin reference client. Most transactions are Pay-to-Public-Key-Hash, Pay-to-public-key and multi-signature (Limited to 15 keys), Pay-to-script-Hash and OP-Return; see: *Antonopoulos* 128, 129, 137.

[33] *Antonopoulos* 218; Such layered design embodies the principle of "separation of concerns," which dictates that the individual functions of a program (e.g. presentation, business logic, data) be independent from each other to ensure their optimized performance, see: Ch. Reade (1989) Elements of Functional Programming. Boston, MA, USA: Addison-Wesley Longman; smart contracts based on the original bitcoin blockchain combine the business logic with the data layers, such separation is more prominent in the second generation blockchain, Ethereum, see: Buterin (n 7) 19.

may operate “in” or “on” the blockchain. The may be embedded within its code or execute outside of it. The important point is that once more functionality is required, the original benefits of the blockchain may be lost.

Validating transactions

Additional misunderstandings concern the concept of “validation.” Technically, a transaction is recorded in the blockchain after it has been validated by the mining process. This, however, only means that the majority of nodes established that a given node completed the proof-of work, that the transaction input exceeded the transaction output and that that the conditions of the locking scripts have been fulfilled.[34] In many contexts, however, the “validation of transactions” is taken to imply that (a) the blockchain can validate smart contracts (a concept generally broader than transactions) and that (b) such validation is legally indispensable or at least highly desirable. In common law systems, however, contracts need not be validated and the concept of “contract validation” does not exist. Whether a contract has been formed and whether it correctly reflects the parties’ agreement are questions of proof that are determined during the process of adjudication. The blockchain provides *evidence* that a transaction occurred: that one or more tokens were transferred from one account to another because the technical conditions of the transfer have been fulfilled. It cannot, however, establish its *validity in the legal sense*. Neither can it establish the validity of the contract the transaction forms part of. The legal validity of a payment or of a contractual relationship always concerns events in the real-world, which cannot be “seen” by the blockchain or validated by the mining process. Transactions do not exist in a vacuum and there is always a *reason* for a transaction. It can be reasonably assumed that every transfer of tokens occurs as a result of some prior agreement, e.g. as payment for goods, or event. The transaction may be recorded in the blockchain *but* the contract underlying such transfer may be legally invalid because, for example, one of the parties lacked legal capacity, acted under duress or the agreement was tainted by illegality. Alternatively, the transaction might have constituted an upfront payment but the counter-performance may have failed or proved inadequate. The mining process cannot determine whether payment was actually due or in any way attest to the reason for the payment. It cannot validate contractual capacity, confirm the absence of vitiating factors or, as explained below, establish events that occur outside of the blockchain – including contractual performance. As in the case of bank transfers, the bank’s records constitute proof of their occurrence but do not reveal *why* money was transferred or prove that a sum was in fact due. “Putting a smart contract onto the blockchain” provides a record of its existence but a record need not always reflect reality – even if the record itself is trustworthy. In sum, the fact that the blockchain “validates” a transaction in a technical sense says nothing about the validity of transaction in the legal sense or about the validity of the smart contract the transaction forms part of.

“Self-enforcement”

Another prominent concept, “self-enforcement,” refers to the automated performance of contractual obligations in the context of blockchain-based smart contracts. Smart

[34] *Antonopoulos* 175, 191

contracts are "self-enforcing" because they automatically transfer tokens upon the occurrence of pre-defined events or automatically block access to cars or flats, in the event of non-payment of a loan or rent.[35] The underlying theory is that humans are biased and generally unreliable. In contrast, code is unbiased and objective. It cannot "change its mind," refuse to perform or deny payment. Consequently, putting the smart contract "on" the blockchain ensures that the contractual obligations are executed without deviations because neither party can influence or interfere with its operation. Legal and commercial certainty are achieved by the fact that performance is *technically guaranteed.* Technical writings associate self-enforcement not only with the automation of transactions and the elimination of human discretion on the side of the *contracting* parties but also, more broadly, with the elimination of the need to seek judicial assistance. The reasoning is that the blockchain shields the smart contract from the vagaries of human discretion and protects the parties from breach. If, then, the likelihood of breach is non-existent (at least theoretically) and performance is guaranteed, traditional enforcement mechanisms are no longer required. The use of the term "enforcement" is, however, somewhat misleading. Contracts are, by definition, *enforceable* agreements.[36] In a legal context, enforcement is associated with the state-sanctioned protection of the parties' economic interest in the performance of the contract. In a colloquial context, it can be said that when parties enforce their rights they seek to obtain what was promised to them, be it by recourse to third parties or by self-help. In a legal context, "enforceability" is thus difficult to disassociate from jurisdiction-specific systems of adjudication. Courts generally enforce contracts by awarding damages for loss resulting from non- or defective performance, seeking to place the aggrieved party in the same position she would have been in had the contract been performed. Courts can also enforce contracts by legally ordering performance, predominantly in the case of actions in debt, where one party claims the payment of a specified sum that has become due. Other types of contractual performance are rarely the subject of such orders. Smart contracts, however, equate enforceability with guaranteed *performance,* effectively collapsing these two concepts. Questions of loss or damages do not even arise, neither does the question whether a particular sum has in fact become due (the simplistic assumption being that it has). This total reliance on technology emphasizes the symbolic independence of smart contracts from the legal system and raises interesting doctrinal questions concerning the very admissibility of such independence, especially in the consumer context. These questions must, however, be relegated to a separate paper.

An associated concept is that of "tamper-proof" enforcement, which means that the smart contract cannot be stopped or modified.[37] A tamper-proof smart contract continues to operate irrespective of external events until its pre-set expiration date. Tamper-proof self-enforcement concerns not only performance but also other contractual decisions, e.g. whether to terminate the contract in the event of breach or whether to enforce collateral in the event of non-payment. Although the idea of technically-guaranteed perfect performance is appealing, smart contracts that are

[35] Kwesi D. Atta-Krah, 'Preventing A Boom from Turning Bust: Regulators Should Turn Their Attention to Starter Interrupt Devices Before the Subprime Auto Lending Bubble Bursts' (2016) 101 *Iowa Law Review* 1187.
[36] For an overview of definitions, each of which encompasses the concept of enforceability, see E. Peel, Chitty on Contracts, 31st edn (2012) vol 1, para 1-016; see also: The American Law Institute's Restatement of Contracts, 2nd edn, para.1; Brian Coote, The Essence of Contract [14-15]
[37] Christopher D. Clack, et al., 'Smart Contract Templates: foundations, design landscape and research directions' (2016) ArXiv e-prints 4.

both “self-enforcing” and “tamper-proof” create a cascade of problems.

First, if contractual performance is relegated to code, it becomes paramount to ensure that such code contains no errors. It must be remembered that in many contexts, smart contract are not synonymous with simple blockchain transactions but may involve code running *on top* of a blockchain. In such instance, the code of the smart contract would be neither incorruptible nor secure. If, however, self-enforcement is to guarantee performance and if neither subsequent human intervention nor a modification of the smart contract are possible then, logically, its code must be perfect. It is, however, practically impossible to ensure an absence of coding errors (“bugs”) because, statistically, *each* computer program contains such. Perfect performance, implicit in the concept of self-enforcement, may thus be impossible to guarantee. Tamper-proof self-enforcing smart contracts may “shield” the transaction from the vagaries of human discretion but they introduce the risk of performance being affected by coding errors.[38] An interesting result follows: as neither party can interfere with the operation of the smart contract, breach is technically impossible – at least if breach is associated with an event that is somehow related to or within the control of the parties. The smart contract may, however, execute incorrectly due to a coding error. In such instance, it seems more appropriate to speak of a *malfunction* than of breach. Given the practical difficulty of preventing such malfunctions, it may be necessary to allocate the risk of their occurrence by prior agreement. Multiple risk allocation scenarios are possible, depending whether the coding error can be attributed to one of the contracting parties or to a third party. In sum, as the possibility of computer errors affecting the manner the smart contract operates, cannot be eliminated, it is impossible to claim that self-enforcement guarantees perfect performance.

Second, it is necessary to ensure that that the smart contract self-enforces as *intended* or as *promised*, i.e. that the code correctly reflects the parties’ agreement, that implementation matches intention. There may, however, be discrepancies between the original agreement and its implementation, as reflected in the code of smart contract. If the parties used a smart contract created by a third party, there may also be a discrepancy between what the parties were told the smart contract would do and what it actually does. Whoever writes the smart contract may fail to correctly reflect the parties’ original intent, be it due to a misunderstanding of the underlying agreement or specification (see below) or as a result of his incompetence or even malice. Whoever makes the smart contract available for use may incorrectly represent its functionality. If the parties are not programmers themselves, they cannot verify whether the code accurately reflects their intentions or whether it will do what it was promised to do. The point is not that the parties may not be able to determine the actual functionality of the smart contract (as they can hire a specialist to do so) but that it may be difficult to establish what takes precedence in the event the code of the smart contract does not match the agreement it purportedly embodies. The problem is particularly acute given the increasing security concerns surrounding smart contracts.[39] Again, the accompanyin problems must be relegated to future research.

[38] Scholarship increasingly recognizes the dangers of automating legal decisions, including dangers of over-reliance on computers, see: Kenneth A. Bamberger, Technologies of Compliance: Risk and Regulation in a Digital Age (2010) 88 *Texas Law Rev* 669.

[39] See Serpent Compiler Audit, ver. 1.0.0 by Zeppelin Solutions (24 July, 2017), which revealed a multitude of security problems relating to one of the smart contract programming languages in Ethereum, Serpent.

Third, a tamper-proof smart contract requires that all possible events that may occur during its lifetime and affect its operation be anticipated. If a smart contract is to operate over a period of time, no decisions about its performance can be left to humans and every aspect of its operation must be encoded upfront. The benefits of self-enforcement would, after all, be lost if it was possible or *necessary* to revise its code to accommodate future events. It is, however, practically impossible to create an exhaustive list of events that could affect the operation of a smart contract. A tamper-proof self-enforcing smart contract would continue to operate irrespective of any change in circumstances, which could lead to a situation where it became commercially absurd or even illegal. On a broader level, technical writings fail to appreciate that contractual relationships are usually flexible and dynamic - even in those instances when they are based on fixed legal language recorded in formal documents. In traditional contracts, it is common to amend certain provisions to adapt to external circumstances, such changes in the regulatory or commercial landscape. It is also common to tolerate certain deviations from the agreed performance without formally amending the contract. It could thus be argued that smart contracts are rigid and can become easily disconnected from the transactional reality in which they operate because no such adjustments are technically possible. Moreover, in traditional contracts, both parties retain the ability to decide whether to *fulfill* their obligations and whether to *exercise* their rights.[40] Such decisions are often made long after the contract has been formed as there may be a significant time-lag between the creation and the performance of the contract. If, however, deviations from performance are technically impossible and if all rights are exercised automatically, the parties lose the ability to adjust to changed circumstances.[41] To elaborate: in traditional contracts, each party can decide not to perform her obligations. Deliberate non-performance may be frowned upon but is not illegal or prohibited. Quite the opposite. Legal and economic theory expressly recognize the concept of efficient breach: a party is allowed to breach a contract and pay damages, if doing so is more economically efficient than performance.[42] Choosing not to deliver the goods because another buyer is willing to pay a higher price optimizes resource allocation and is legally permissible – as long as the contract breaker compensates the aggrieved party for her loss. Being deprived of the option *not* to perform is thus less commercially attractive and legally "dangerous" than commonly assumed. Self-enforcement may also have drawbacks for the aggrieved party. In the event of breach, the latter can decide whether (and how) to exercise her rights. In principle, she has the right to claim damages for the resulting loss (if any) and, on occasion, the right to terminate the contract. Whether this right is exercised constitutes a fact-specific *commercial* decision. If, however, the entire contractual relationship is locked into immutable code, termination occurs automatically and the aggrieved party is deprived of the option *not* to exercise her rights.[43] It can also be assumed that in practice contractual performance is rarely perfect and that some breaches are deliberately ignored, be it due to their commercial insignificance or to preserve an otherwise beneficial relationship. In tamper-proof self-enforcing transactions, these possibilities disappear: imperfect performance leads to

[40] *Swanson* 28
[41] *Swanson* 29
[42] Charles Goetz, Robert Scott, 'Liquidated Damages, Penalties, and the Just Compensation Principle: A Theory of Efficient Breach' (1977) 77 *Columbia Law Review* 554; Melvin A. Eisenberg, 'Actual and Virtual Specific Performance, the Theory of Efficient Breach, and the Indifference Principle in Contract Law' (2005) 93 *California Law Review* 975.
[43] ISDA & Linklaters, *Whitepaper: Smart Contracts and Distributed Ledger – A Legal Perspective* (August 2017) ("*ISDA*")

termination or other predetermined result. In sum: self-enforcement deprives contractual relationships of their adaptability and preclude the parties from adjusting their legal and commercial positions in response to changed circumstances.[44]

4. Broader problems of enforceability

Technical writings (and some legal scholarship) occasionally claim that blockchains create a parallel transactional universe, or even their own jurisdiction, where the parties can transact outside of the legal system. As a result, blockchain-based smart contracts would, as indicated, obviate the need for legal rules and institutions. These theories may derive from a misapprehension of the practical implications of the decentralized character of the blockchain, especially with regards to its consensus mechanism of validating transactions. Decentralization has, after all, often been associated with the abandonment of traditional legal institutions. Similar claims proliferated in early cyberspace scholarship, which proclaimed the independence of cyberspace from the legal system and advocated radical disintermediation, which seemed to be a natural consequence of the distributed nature of the Internet.[45] With the benefit of hindsight, it can be observed that the decentralization of Internet *infrastructure* in the technical sense is unrelated to and need no result in the decentralization of commerce or a detachment from traditional institutions. It can be cynically observed that once the Internet evolved into a platform for commerce and once parties started using it to exchange real-world goods and services for real-world money, the spirit of cyber-independence subsided considerably. The Internet economy has evolved into a capitalist economy.[46] After a brief period of theoretical turmoil, it has become uncontroversial that contracts formed “in cyberspace” must be treated like all other contracts and are subject to the same legal principles. Each online transaction is governed by the laws of its real-world jurisdiction. Unless the contractual subject matter is illegal, as was the case with most transactions on Silkroad, it is also not immediately apparent why anyone would want to transact outside of the legal system.[47] Once an exchange involves anything of value, the transacting parties must retain the ability to ask the courts for assistance in the event something goes wrong. Technical writings, which imply that smart contracts and blockchains obviate the need for judicial protection, overlook the simple fact that the lack of recourse to established legal institutions would not only incentivize fraudsters and hackers, but also discourage the very use of blockchains and smart contracts in financial transactions. For smart contracts to become a viable commercial tool, they must be not only technically but also legally enforceable. After all, the trustless and incorruptible character of the blockchain is of limited significance if the code of the smart contract executes *outside* of the blockchain and if self-enforcement is incapable of protecting the parties from the risk of computer errors or from the possibility of changed circumstances. It becomes apparent that the parties to a smart contract must retain the ability to rely on traditional legal protections. Such protections are, however, only reserved to those relationships that carry the indicia of a contract.[48]

[44] It is also increasingly recognized that perfect enforcement may not be desirable, see: Lisa A. Shay et al., Confronting Automated Law Enforcement, in: Ryan Calo, Michael Froomkin, Ian Kerr eds., *Robot Law* (Edward Elgar 2016) 258; for an explanation of the dangers of perfect enforcement see: Christina M. Mulligan, ‘Perfect Enforcement of the Law: When to Limit and When to Use Technology’ (2008) 14 *Richmond Journal of Law & Technology* 13.
[45] David R Johnson and David Post, ‘Law and Borders – The Raise of Law in Cyberspace’ (1996) 48 Stanford Law Review 1367
[46] Manuel Castells, *The Rise of the Network Society* (Wiley & Sons, 2nd ed, 2010) 160.
[47] For example, blockchain-based assassination contracts.
[48] Jack Goldsmith and Tim Wu, *Who Controls the Internet? Illusions of a borderless world* (Oxford University Press, 2006) 138.

The prerequisites of enforceability

The prerequisites of a legally enforceable contract can be reduced to two elements: intention and consideration. The presence of intention is always evaluated objectively, on the basis of the parties' words and actions. It is commonplace to establish intention by means of the offer and acceptance model, which often facilitates the determination of the precise time and place a contract is formed. What is more pertinent to the present discussion is the fact that intention can be expressed in any manner as contract law is inherently form free.[49] The existence of a contract is rarely tied to any formal requirements, such as writing, signatures or being "certified by notaries." A contract can be formed orally or by conduct,[50] it can be expressed in words (either spoken or written), Morse code or in computer instructions, including self-executing code. With very few exceptions introduced by statute, contract law detracts from form and focuses on substance. If the parties want to express their agreement in code or to relegate the performance of their obligations to a set of automated processes running on a blockchain – there are no *legal* obstacles preventing them from doing so. The second prerequisite of enforceability, consideration, is often described as something promised or given in exchange. In practice, some right, interest, profit or benefit must accrue to one party, or some forbearance, detriment, loss or responsibility must be given, suffered or undertaken by the other.[51] Consideration need not be adequate, it only needs to be sufficient in the eyes of the law.[52] The focus is on reciprocity, not on equivalence of value. The parties can exchange money in return for goods or services, or bitcoins in return for digital assets.[53] As a result, only those smart contracts that involve exchanges can fulfill the requirement of consideration. If no exchange is involved, it must be assumed that the term "smart contract" is used in a purely technical sense. After all, *legal* enforceability is hardly required for smart "contracts" which are stock tickers or weather apps. At the same time, while there are no theoretical obstacles for *some* smart contracts to be legally enforceable, it is impossible to make a general statement that *all* smart contracts can be legally binding. Each case must be evaluated separately to establish whether the parties had the requisite intention and whether each party provided something to the other. In practice, an *exchange* of tokens for goods, services or for another type of tokens renders superfluous any additional proof of intention and consideration.[54] In most instances problems of enforceability do not arise as the smart contract relies on (and reflects) an existing legally binding agreement and is only a tool facilitating its performance.

The fact that smart contracts automate the performance of contractual obligations is legally uncontroversial.[55] The debates surrounding automated contracting have been largely resolved by the scholarship that evolved around so-called "electronic agents,"

[49] Andrew Phang, ed. *The Law of Contract in Singapore* (Singapore: Academy Publishing, 2012) at p 418: "Unless otherwise provided-for, generally by way of statute, the common law does not impose any requirements as to formalities or the manner of execution of a contract for such agreement to be legally binding."
[50] *Harvey v Johnston* (1848) 6 CB 295; *Brogden v Metropolitan Rly Co* (1877) 2 App. Cas. 666, HL.
[51] *Currie v Misa* (1875) LR 10 Ex 153
[52] *Chappel & Co. Ltd. v. Nestle Co. Ltd.* [1960] A.C. 87 (H.L.).
[53] I bypass the question whether bitcoin is legal tender, sale or barter?
[54] B.A. Hepple, "Intention to Create Legal Relations" (1970) 28 Cambridge L. J. 122 at 127-128; *The Aramis* [1989] 1 Lloyd's Rep 213 at 225 per Lord Bingham; Warren Swain, "Contract as Promise: on the Role of Promising in the Law of Contract. A Historical Account." (2013) 17 Edin. L.R. 1 at 20.
[55] R Nimmer, *Electronic Contracting: Legal Issues* (1996) 14 J Marshall J Computer & Info L 211; +EDI

computer systems that assist and act on behalf of their human operators.[56] The possibility of automating transactions or expressing contractual intention by means of computer processes has also been expressly recognized in e-commerce regulations, such as the United Nation Convention on the Use of Electronic Communications in International Contracts [57] or the Uniform Electronic Transactions Act.[58] To deny the possibility of "delegating" the formation and performance of contracts to computers is to deny the legal viability of automated securities trading, vending machines and practically all e-commerce websites.[59] In each of these examples, the computer executes a prior human decision to engage in a transaction within pre-set parameters with whoever complies with these parameters. The intention to be legally bound is manifested *by means* of a computer. In this sense, smart contracts represent the intentions of the transacting parties, who chose to express their obligations in code and automate certain aspects of contractual performance.

Lastly, putting a smart contract "on" the blockchain does not place it outside of the legal system or otherwise insulate it from the laws of a given jurisdiction. Apart from meeting the prerequisites of enforceability, smart contracts must also remain compatible with their jurisdiction-specific legal framework. If a transaction is prohibited on grounds of illegality, such as when the parties agree to sell illicit drugs or infringe import restrictions, it will not be permitted just because it is "on" the blockchain. Similarly, if a smart contract executes an agreement that was formed under duress or as a result of an actionable misrepresentation, the smart contract will self-enforce but the underlying agreement may be set aside. If a particular type of exchange is prohibited or if a contractual provision is illegal or unenforceable under the rules of a specific jurisdiction - it remains prohibited, illegal or unenforceable if embodied in a smart contract. Blockchains *do not* create decentralized marketplaces operating "free from the reach of regulation."[60] Some blockchain enthusiasts may have misinterpreted the statement "code is law" as implying that code can supersede the law or that decentralized networks create their own legal regimes. Code may guarantee contractual performance or facilitate the transacting process but the instructions it executes must remain within the confines of the law.

5. Natural Language and Code

The very viability of smart contracts hinges on the ability to express contractual obligations in code. Logically, as natural language cannot be directly executed by a computer, self-enforcement requires that the terms of the smart contract be computer-readable. There are multiple options: the smart contract can be a translation of an existing agreement, it can be created in code from the outset or,

[56] T Allen, R Widdison, *Can Computers Make Contracts*? (1996) 9 Harv J Law & Tech 25; J H Sommer, *Against Cyberlaw* (2000) 15 Berkeley Tech L J 1145.
[57] see: Convention on the Use of Electronic Communications in International Contracting, Nov. 23, 2005, U.N. Doc. A/60/21, Article 12: A contract formed by the interaction of an automated message system and a natural person, or by the interaction of automated message systems, shall not be denied validity or enforceability solely on the ground that no natural person reviewed or intervened in each of the individual actions carried out by the automated message systems or the resulting contract.
[58] See UETA comment 1 to Section 14, which confirms that contracts can be formed by machines functioning as electronic agents for parties to a transaction. It negates any claim that lack of human intent, at the time of contract formation, prevents contract formation. When machines are involved, the requisite intention flows from the programing and use of the machine.
[59] interesting issues arise in situations where one or both parties believe that SC are not legally binding and engage in automated transactions on this assumption, mistake of law or absence of legal intention?
[60] Aaron Wright, Primavera Filippi, 'Decentralized Blockchain Technology and the Rise of Lex Cryptographia' (2015), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2580664, 104.

lastly, a contract can be drafted in natural language with subsequent encoding in mind. The accompanying challenges concern the conversion of natural language into code and, more broadly, the expression of contractual obligations in code (i.e. the "encoding" of obligations). Two points must be made before proceeding. *First*, it must be observed that the present discussion would be redundant if technical writings confined smart contracts to simple payment obligations occurring within narrowly defined relationships, where each party's performance is reducible to fixed formulae, as is the case of options or interest rate swaps.[61] While it is increasingly acknowledged that only some contracts can or should be smart,[62] most technical writings continue to extoll the ability of smart contracts to transform *all* types of contracts, including employment contracts, leases and mortgages. *Second*, technical writings provide little guidance as to how smart contracts are formed or entered into. It is generally assumed that that parties create their own smart contract or agree use an existing smart contract created by somebody else. While there are no obstacles to the creation of one-off, customized smart contracts, economies of scale dictate that smart contracts take the form of generic programs that can be used on a mass-scale. They could, for example, embody popular standard form agreements, such as car loans, mortgages or interest rate swaps. In such instance, only certain values would be customized for individual transactions. Again, different configurations seem possible. The important point is that in many circumstances the smart contract is not coded by the parties themselves or that at least one of the parties does not participate in its creation. Consequently, either both parties or one of them are unable to verify whether the code accurately reflects their agreement or to determine how the smart contract will operate in practice.

Contract Translation

Given that the coders who create the smart contract cannot (or *should* not) decide on its commercial and legal aspects,[63] it is reasonable to assume that there must be an document describing the substance of the agreement. Consequently, many smart contracts will originate as documents written in natural language that require subsequent translation into code. The difficulty of this process is generally underestimated. The reasons are twofold.

First, technical writings extol the consistent progress in the areas of machine learning and natural language processing and assume that the translation of natural language into code can be automated or at least significantly facilitated by technological means. At the switch of a button, contracts could be converted into executable code in the same manner source code is compiled into object code. Despite consistent progress in the said areas, it is presently impossible to automate the conversion of natural language into code – at least not without a significant compromise in the quality of the output of such conversion.[64] Machine learning enthusiasts might have been misled by the relative success of google translate or the sensationalistic reports of AI-based systems beating their human opponents at complex games.

[61] Kiviat (n 16) 607.

[62] Karen E.C. Levy, 'Book-Smart, Not Street-Smart: Blockchain-Based Smart Contracts and The Social Workings of Law' (2017) 3 *Engaging Science, Technology, and Society* 10, 11; see also: I. Grigg, 'On the Intersection of Ricardian and Smart Contracts' (2015).

[63] C. K. Frantz and M. Nowostawski, 'From institutions to code: Towards automated generation of smart contracts' (2016) *iEEE 1st international workshops on foundations and applications of self-systems (fASW)* 210.

[64] Wright and Filippi (n 60) 104; for a non-technical explanation of machine learning see: Harry Surden, 'Machine Learning and Law' (2014) 89 *Washington Law Review* 87, 91-95.

Approximations seem permissible in automated translations between natural languages, where the overall meaning of a sentence can be gleaned from the context. They are intolerable, however, when it comes to legal provisions, which are drafted with meticulous precision and where one single word may give rise to unintended commercial consequences and prolonged disputes.[65] Developers fail to appreciate the low tolerance for mistakes in legal documents.[66] Moreover, precision seems paramount when the smart contract is to self-execute and cannot be stopped or amended. If a smart contract is to embody an existing agreement, its translation into code will involve a tedious manual process.

Second, developers seem to view contracts as sets of conditional statements, abundant in standardized clauses that can be endlessly re-used for different transactions. While it is true that lawyers often rely on contractual templates and (sometimes too eagerly) copy-and-paste individual provisions, it must not be forgotten that the standardization of legal language does not imply that such language is capable of a reduction into an algorithm. Despite its formalistic nature, legal text is still natural language – and natural language is inherently imprecise as the meaning of words always depends on the context. With its lengthy sentences, subordinate clauses, nested expressions and references to abstract concepts, legal language may be more difficult to translate into code than "normal" natural language. It is often suggested that smart contracts necessitate the creation of a custom-built, domain-specific programming language that could capture the nuances of legal text.[67] The main problem, however, is that the translation of natural language into code does not constitute a straightforward process of converting legal prose into computer-readable instructions but requires the prior *interpretation* of the legal prose. Interpretation is not an academic exercise but serves to establish the exact scope of the parties' obligations, the result to be achieved under the contract or the level of effort to be expounded in performing a particular obligation. As indicated, the successful performance of a contract may hinge on the meaning of a single word and a dispute over a single word can lead to protracted litigation. There is hardly a contract that would not require *some* interpretation and thus the presence of some legal and commercial knowledge on the side of the "interpreter." Contractual interpretation is usually performed by courts, *after* a dispute has arisen. In the smart contract scenario, it would have to be performed before or in parallel with the process of translating the legal text into code. Developers, however, can hardly be expected to perform this task. The latter requires an in-depth knowledge of the principles governing contractual interpretation - principles that are surrounded by multiple controversies relating to the question *how* to determine objective meaning of words and expressions, or: the meaning that must be *deemed to have been intended by both parties*. Such meaning can depend on other words used in the given contractual document or, more broadly, the context in which they are used.[68] Whoever interprets the contract must be able to decide between the literal and the purposive approach and, in the event of competing interpretations, select the one that is more consistent with business common sense.[69] The interpreting developer would

[65] On the limits of Natural Language Processing *see* Robert Dale, *Classical Approaches to Natural Language Processing, in* HANDBOOK OF NATURAL LANGUAGE PROCESSING 1 (Nitin Indurkhya & Frederick J. Damerau eds., 2d ed. 2010)
[66] Daniel Martin Katz, 'Quantitative Legal Prediction—or—How I Learned to Stop Worrying and Start Preparing for the Data-Driven Future of the Legal Services Industry' (2013) 62 *Emory Law Journal* 909, 936.
[67] See e.g. Stephen Wolfram, Computational Law, Symbolic Discourse and the AI Constitution—Stephen Wolfram Blog (2016) http://blog.stephenwolfram.com/2016/10/computational-law-symbolic-discourse-and-the-ai-constitution/#comments
[68] Peel, 229 [6-009]
[69] *Rainy Sky SA v Kookmin Bank* [2011] UKSC 50; *Re Sigma Finance Corp* [2009] UKSC 2.

have to ascertain the meaning that the contract "would convey to a reasonable person having all the knowledge which would reasonably have been available to the parties in this situation in which the where at the time of the contract."[70] Moreover, it must not be forgotten that the process of interpretation is not limited to situations when words are ambiguous but may sometimes reveal the very presence of ambiguity.[71] It may not be immediately apparent that a particular word or expression is capable of multiple interpretations. In the context of smart contracts, the problem would not lie in the parties disagreeing over the meaning of words but in the possibility of incorrect interpretations by those who decide how to convert a particular obligation into code. To further aggravate matters, most contracts contain gaps and require that terms be implied to make the agreement workable in practice. Again, the implication of terms is traditionally performed by courts, not by the contracting parties. Logically, had they discovered the gap, they would have addressed it. Given that that the implication of terms requires an understanding of the legal rules *and* the commercial context of a particular transaction, coders may not be able to identify and to fill contractual gaps themselves. Ultimately, while it could be argued that the aforementioned problems of interpreting or supplementing contractual language may be solved by lawyers and coders co-operating in the translation of legal documents into executable code, it must be acknowledged that despite such co-operation neither the parties, nor their lawyers are able to ascertain whether the code of the smart contract correctly reflects the originating legal document. Assuming that the smart contract should mirror this document in all its nuances, there is potential for discrepancies between what was agreed and what was implemented. Such discrepancies are particularly unsettling given that once the smart contract commences self-enforcement, it cannot be stopped or amended.

Direct coding

To avoid the difficulties of translating legal language into code, it is sometimes suggested that smart contracts be written in code from the outset. Such "direct coding" would not only facilitate the execution of the smart contract by a computer but also reduce, or even eliminate, the ambiguity of natural language.[72] The contract would be smart from its inception. To bypass the stage of drafting in legal prose, lawyers would, of course, have to learn how to program. Alternatively, programmers would have to learn the basic principles of contract law. Technical writings suggest that such direct coding of smart contracts would force lawyers to be more precise and structured in describing the rights and obligations of the parties. In this sense, smart contracts *could* in fact reduce ambiguity because must be capable of a single interpretation. Given that most lawyers are unlikely to become programmers (just as most programmers are unlikely to become skilled drafters), it is suggested that even if smart contracts cannot be coded directly, contracts could be drafted with encoding in mind. To this end, the contracting parties (and their lawyers) could reorient the manner in which they express their agreement to facilitate its subsequent translation into code,[73] e.g. describe their obligations in a formalized, structured manner and provide objective, measurable criteria that must be met for such obligation to be considered performed. In this context, some commentators acknowledged that the

[70] *Hombourg Houtimport BV v Agrosin Private Ltd (The Starsin)* [2003] UKHL 12.
[71] *Napier Park European credit opportunities fund ltd v Harbourmaster Pro-Rata CLO BV* [2014] EWCA Civ 984
[72] Stephen Wolfram, Computational Law, Symbolic Discourse and the AI Constitution—Stephen Wolfram Blog (2016) http://blog.stephenwolfram.com/2016/10/computational-law-symbolic-discourse-and-the-ai-constitution/#comments
[73] Harry Surden, Computable Contracts (2012) 46 *UC Davis Law Review* 635, 674-675.

ability to trust the blockchain is unrelated to the ability to trust the smart contract. As the latter must be transparent to stakeholders and correctly reflect the agreement on which it is based, it must be created by lawyers *and* programmers together. [74] This in turn requires that those two groups be able to communicate via a common language. Smart contracts should thus be written by lawyers in a "controlled legal natural language" that is logical, clear, unambiguous, and comprehensible to programmers.[75] Lawyers would be using this language to write *de facto* specifications guiding the actual coding of the smart contract. The above approaches, particularly the latter, appear *prima facie* attractive as they could reduce or even eliminates ambiguity and minimize the potential for disputes. They mistakenly assume, however, that (a) the elimination of ambiguity from contractual language is not only possible but also desirable and that (b) all contractual obligations can be described in a manner that enables their expression in code. Each of these assumptions raises a multitude of technical and legal problems the detailed description of which exceeds the scope of this paper. Some general observations are, however, possible – if only to illustrate the complexity of the issues involved.

(a) Eliminating Ambiguity?

Most programmers - or those who propound the idea that smart contracts constitute a cure to bad legal drafting - assume that it is *always possible* to draft complete and unambiguous agreements. For an developer, the failure to do so evinces the limited skill or incompetence of the lawyer. By virtue of their training developers perceive ambiguity as inherently bad. It is important to understand, however, that ambiguity has both advantages and disadvantages. While it may increase the potential for disputes over the exact scope of the parties' obligations, it also creates flexibility as it allows the parties to retain a measure of flexibility in performing their side of the bargain and in evaluating each other's performances. It also enables both parties to adapt to changing circumstances without having to redraft the agreement. Developers fail to recognize that in contract law, ambiguity is a feature not a bug. Apart from the natural ambiguity accompanying all human languages and the ambiguity that results from sloppy drafting, many contractual provisions are deliberately written in a broad, slightly imprecise manner to ensure a certain degree of leeway. The ambiguity of certain provisions may also reflect the stronger bargaining position of one party, who drafts the contract in a manner enabling it to deliver the absolute minimum without being accused of breach. Terms may also be left vague because of an unwillingness to invest resources in extended negotiations or drafting, or due to the widespread approach that the contract is only a formality while the "real" agreement is reflected in the ongoing commercial relationship.[76] Although it is trite law that contracts must be "certain and complete" to be considered binding and enforceable, it is not necessary for all contractual obligations be described with algorithmic precision. Traditional contracts generally work well in the context in which they were made without providing for all eventualities or describing all obligations in detail.[77] It has

[74] . K. Frantz and M. Nowostawski (n X), Clack (n 37).

[75] F. Al Khalil, et al., 'A Solution for the Problems of Translation and Transparency in Smart Contracts' (2017) 3, 8, 9, stated that "more attention should be paid to bridging the gap between a smart contract's legal semantics, business semantics and regulatory semantics and its denotational semantics and ensuring provenance, while guaranteeing the empirical fidelity of the operational semantics. Simply put, it is the lawyers and financial professionals and not computer programmers who should be creating smart contracts."

[76] Levy (n 62) 7.

[77] Peel (n 39) para 6-013.

been observed that gains in precision may result in rigid determinism that will not necessarily breed certainty but destroy adaptability, especially in long-term contractual relationships where the potential for a change in circumstances is particularly high.[78]

Assumedly, technical writings regard smart contracts as a technological "cure" to commercial and legal uncertainty. It is, however, impossible to create certainty by technological means. Moreover, in every contractual relationship or commercial transaction *some* uncertainty is always anticipated and tolerated – be it with regards to unforeseen external events (such as force majeure) or with regards to the manner obligations are performed. Automation-oriented "precision drafting" may deprive the contract of its natural adjustability and isolate it from the surrounding context. As observed by two commentators, programming requires actors to "quantify the qualitative, discretize the continuous, or formalize the non-formal."[79] For a contract to be devoid of ambiguity, it would have to anticipate *all* possible events that might affect its operation and describe endless combinations of external variables. After all, traditional contracts "operate" in a complex and uncertain environment: the real world. Coding smart contracts, for complex, uncertain environments is extremely difficult due to the number of things that can go wrong and the inability to predict how the evolving commercial context will affect the contractual relationship.[80] It must also be assumed that an unambiguous contract (assuming that its creation is possible) would, inevitably, be extremely long because many obligations would have to be described with a large number of eventualities in mind. In comparison to their traditional counterparts, contracts written in code – or contracts written with subsequent encoding in mind - would increase in volume and complexity. As the number of errors increases in proportion to the number of lines of code, longer smart contracts would contain more errors and display a higher potential for erroneous self-enforcement. Some technical writings specifically acknowledge that certain errors may arise due to the unique nature of smart contract programming.[81] In sum, precision comes at a cost: the contractual relationship becomes rigid and deterministic, while the increase in the length of the contract leads to a higher likelihood of malfunction.

(b) Encoding obligations

Additional problems concern the assumption made by many technical writings that *all* obligations can be expressed in code.[82] To recall, smart contracts were initially envisaged in a limited number of contexts, such as financial instruments. As the smart contract narrative became more ambitious, other contractual obligations are potential "candidates" for self-enforcement. Such claims must, of course, be viewed with skepticism. It is reasonable to assume that only those contractual obligations can

[78] Surden (n 73) 674-675; see also: R. W. Gordon, 'Macaulay, Macneil, and the Discovery of Solidarity and Power in Contract Law' (1983) 3 *Wisconsin Law Review* 565, 569, who emphasizes that contracts are not set in stone at the moment of formation but change as circumstances change.

[79] Batya Friedman & Helen Nissenbaum, 'Bias in Computer Systems' (1996) 14 *ACM Transactions On Information Systems* 330, 333 (1996) as quoted by Bamberger (n 41) 708.

[80] https://medium.com/@coriacetic/in-bits-we-trust-4e464b418f0b

[81] Kevin Delmolino, et al., 'Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab' (2016) *Financial cryptography and data security - FC 2016 international workshops, BITCOIN, VOTING, and WAHC, Christ Church,* 79.

[82] Logically, this correlates with the question *which obligations can be automated*. It must be assumed that some obligations can be automated, such as those involving payment, while others are inherently unsuitable for automation, such as those involving personal performance or uniquely human skills.

be expressed in code that can be represented as an algorithm or are otherwise capable of an exact, formulaic description. The problem is not one of ambiguity or errors in interpretation but concerns the very nature of certain obligations and legal concepts. For example, some types of contractual performance rely on abstract concepts such as “good faith” or “reasonableness” and may be impossible to represent as a closed catalogue of actions or in the form of an objectively measurable result. Developers may view every contract as a collection conditional statements and assume that each contractual provision can be reduced to an algorithm or a finite result. This misunderstanding of contracts (and contract law!) may underpin the grandiose theories of smart contracts disrupting the legal landscape. Admittedly, many contractual provisions are operational in nature and prescribe sequences of actions e.g. “deliver [object] to [place] on [date]” or the achievement of specific results.[83] Many contractual obligations are, however, based on reasonable care, where the parties must undertake, or refrain from, certain actions without having to produce a measurable outcome. It may be difficult to reduce them to sequences of steps *and* to provide objective benchmarks against which they can be evaluated. Obligations based on care are, after all, frequently qualified by concepts such as “reasonableness” or “best efforts.” It also seems impossible to catalogue all component obligations falling under the concept of “reasonable endeavors” or to describe how to perform an obligation in “good faith.” Additional problems would concern contractual provisions that are non-operational and concern administrative issues, such as the choice of jurisdiction, or co-define the operational provisions by limiting or excluding contractual liability in the event of breach. It should thus be conceded that not all obligations can be expressed in code and that not all contracts can “be smart.”

6. Integrating with the Real World

A discrete set of technical challenges concerns the event(s) triggering the transfer of tokens. For smart contracts to function, parties must not only formulate agreements in a manner facilitating their expression in code and define objective criteria to determine contractual performance *but also* provide the smart contract with data that enables such determination.[84] The ability to encode contractual obligations must be distinguished from the ability to automatically determine the fulfillment of such obligations. The distinction can be explained as follows: the obligation to deliver a box to a certain place by a certain time can be precisely described and expressed in code. It is also unproblematic to encode the obligation to pay once the box is delivered. The smart contract to pay upon delivery must, inevitably, contain references to *physical* objects (box) and events in the real-world (delivery). If, then, payment is conditioned on delivery and if the entire process is to be automated, as implicit in the concept of *self*-enforcement, it must be possible to establish delivery *without* human involvement. To recall: eliminating the latter guarantees objectivity and prevents disputes. Moreover, the event triggering payment (i.e. contractual performance) must be *computationally verifiable*.[85] As a result, the self-enforcement of the payment obligation is premised on (a) the availability of relevant data concerning such performance, (b) the ability to *feed* such data to the smart contract and (c) the performance itself being of a type that can be objectively and

[83] See *ISDA* 18-20, for list of examples.
[84] Surden (n 73) 677.
[85] Levy (n 62) 10, 11.

automatically established. [86] For the sake of simplicity, it is assumed that only the payment obligation self-enforces and that the other obligation is not automated, i.e. the box is delivered by a human not by a robot.[87] At the same time, while the delivery itself is not automated - the *determination* of delivery is.

To establish that the payment condition has been fulfilled, smart contracts must communicate with the physical world. The resulting problems are particularly noticeable in blockchain-based smart contracts. The original blockchain was designed as an insulated environment, which cannot accept input from the "real world."[88] In this context, technical writings distinguish between on-chain and off-chain events. If a particular process, asset or event concerns or occurs in the blockchain, it is referred to as "on-chain." Only on-chain events are natively visible to the blockchain. Such events are few: the passage of time, the addition of blocks (which includes the generation of tokens and the validation of transactions) and the transfer of tokens, which occurs in response to the presentation of private keys (see below). All processes, objects or events in the physical world are "off-chain." The blockchain cannot "see" or accept direct input about or from off-chain events.

To understand the limitations of smart contracts, it is necessary to explain how transactions are initiated. Transactions rely on public key cryptography, which involves the use of a public and a private key. The keys are generated together by a complex algorithm, which guarantees that it is impossible to derive the private key from the public key. The public key, which resembles a bank account number and remains publicly visible, is required to receive tokens. The private key, which resembles a PIN or a password, is required to spend them. When tokens are transferred to the account represented by the public key, only the person with the correct ("corresponding") private key can access them. No tokens can be moved from the public key unless the correct private key is used. In practice, each transaction contains a script that states that the token(s) are payable to whoever presents the private key corresponding to the account (public key) associated with the payee. The assumption is, of course, that only the intended payee can present the private key.[89] On a technical level, scripts "lock" tokens to a specific public address and to unlock (i.e. spend) them it is necessary to provide the private key.[90] Scripts may require the presentation of one or multiple private keys.[91] Although technical writings state that locking scripts can include "complex conditions," which must be satisfied by the *un*locking script to release payment,[92] it must be emphasized that those "complex conditions" only refer to various configurations of private keys, as prescribed by the locking script. They cannot, for example, make direct references to *any* off-chain events. In sum, the unlocking scripts can only respond to on-chain events, which involve the presentation of one or more private keys. Smart contracts can thus be compared to hermetically sealed boxes, which can only be opened with one or more keys but are otherwise insulated from the outside world. As a result, if a smart contract conditions payment on the occurrence of an off-chain event, it is necessary to involve a third party that will sign the unlocking script after verifying

[86] Surden (n 73) 664.
[87] The movement of physical assets by means of smart contracts presents even more technical challenges, see: Swanson 48.
[88] see generally: *Antonopoulos* 125, 132.
[89] *Antonopoulos*, 23
[90] *Antonopoulos*, 117.
[91] *Antonopoulos,* 98, 99, 100.
[92] *Antonopoulos* 123, 124.

that the off-chain event has taken place. Entities that furnish the technical infrastructure to communicate information about off-chain events to smart contracts are commonly referred to a "oracles." Oracles do not feed such information into the blockchain directly (as this would compromise its trustlessness) but "only" sign the script unlocking the tokens with their private key when an off-chain event is established as true. Needless to say, this approach leaves no room for a nuanced evaluation of contractual performance, as the oracle can only sign or *not* sign the script. Despite its apparent simplicity, this model is plagued by a cascade of technical interdependencies.

First, it is necessary to find a trustworthy and reliable oracle. Oracles exist outside of the blockchain and are neither trustless nor decentralized. Despite the naming convention, they do not constitute infallible sources of truth and cannot guarantee that an off-chain event actually occurred. Moreover, oracles do not create or compute the required information about off-chain events themselves but obtain if from external data sources, such as websites, commercial providers (e.g. Bloomberg), prediction markets (e.g. Augur), answer engines (e.g. WolframAlpha) or other blockchains. Apart from finding a trustworthy oracle it is equally important to find a trustworthy source of information.[93] As a result, the parties to a smart contract (or its creators) must select an oracle *and* a data source to be used thereby. To prevent disputes, they must also agree beforehand that they will accept the information provided by their chosen oracle (and its data source) as true. Smart contract evangelists rarely mention the *practical inability* of ensuring that both the oracle and its data source are as trustless as the blockchain.[94] As both oracles and data sources can be compromised,[95] it may be necessary to create a network of oracles, which obtain the information from *multiple independent* data sources. In such instance, the off-chain event triggering payment must be confirmed by multiple oracles, i.e. the payment will be released once N-of-M oracles sign the unlocking script.[96] In effect, the benefits of the trustless and incorruptible nature of the blockchain are easily lost once the smart contract requires information about off-chain events.

The second set of challenges concerns the very act of obtaining information about off-chain events. The latter can be divided into those that are public and universally visible, such as stock prices or weather conditions, and those that do not possess those attributes. Most types of contractual performance, including the aforementioned delivery of a box, fall into the latter category. Some off-chain events generate public data that can be obtained from multiple authoritative sources. For example, it is easily established that at a particular time the NASDAQ reached a particular value. This value will be identical even if reported by multiple sources, e.g. Bloomberg and Reuters. Difficulties arise when different sources provide divergent information about the same event, e.g. the temperature at a particular location is reported differently depending on the placement and sensitivity of sensors. In such instance, it may be necessary for the parties to agree on *one* specific source to avoid later disputes. The real complications start, however, when information about an off-chain event is not publicly available or otherwise difficult to obtain. Returning to the

[93] examples of oracles are: reality keys at https://www.realitykeys.com; oraclize at http://www.oraclize.it.
[94] the majority of oracles must provide their signatures for the event to be established, it remains vital to ensure that the oracles are not in collusion; see Fan Zhan (n 10) 1.
[95] http://docs.oraclize.it/#overview-problem
[96] https://blog.oraclize.it/understanding-oracles-99055c9c9f7b#.ulbbmk519; in Ethereum the oracle is not putting a signature once some conditions are verified, but providing the data directly.

box example, its delivery may be publicly visible and verifiable, but information about this event may be difficult to obtain – at least not in an automated fashion or from a single, unbiased authoritative source. If, then, there are no readily available sources of information about certain events, the parties must create such sources themselves. This may require the establishment of an elaborate physical infrastructure, such as equipping the gates of warehouses with sensors and embedding all boxes or containers with QR codes, NFC tags etc. Alternatively, the parties must rely on third party providers who already have such infrastructure in place. In either instance, it becomes apparent that smart contracts conditioning payment on off-chain events, require an environment filled with sensors, physical objects embedded with tracking technologies and computers that continually monitor their surroundings.[97] It is often argued that many off-chain events will become easier to determine automatically with advances in the Internet-of-Things, which involves the proliferation of Internet-connected sensors providing real-time information about the state of different physical objects. Unfortunately, such claims underestimate the relative infancy of Internet-of-things technologies, particularly with regards to their insecurity and absence of standardization. As in the case of oracles and their data sources, the original benefits of putting the smart contract on the blockchain are lost as neither the infrastructure creating the data source nor its provider can be guaranteed to be trustless or secure.

Lastly, it must be acknowledged that self-enforcement may be limited to transactions where off-chain events are computationally verifiable. Just like not all contractual obligations can be represented in code, not all off-chain events can be captured as computer-readable data or measured with objective criteria. Many types of contractual performance may require human involvement. For example, while the delivery of a box constitutes an objectively ascertainable event, establishing whether its contents conform with the contractual description may involve both quantitative and qualitative elements the evaluation of which may necessitate human inspection. It may, be difficult to *automatically* and *correctly* determine that the box contains grade "A" Granny Smith apples or genuine Louis Vuitton bags. While establishing the quality of apples is, at least in theory, achievable by means of sophisticated image recognition technologies, the authenticity of a Louis Vuitton bag requires a manual inspection by a trained human specialist.[98] Self-enforcement is also impossible in transactions that require flexibility in assessing performance or when contractual performance must be evaluated holistically.[99] These technical limitations confirm the very narrow range of contractual obligations that are candidates for self-enforcement.

7. Concluding Observations

Any serious debate of smart contracts must be based on a solid understanding of what is *legally* and *technologically* possible. Quite surprisingly, there are relatively few (if any) legal obstacles to the use of smart contracts in commercial transactions. The main challenges to their use seem to be technical more than legal. On a broader level, one can question the very benefits of putting smart contracts onto blockchains. While the original bitcoin blockchain is trustless and secure, it is also extremely

[97] Shay (n 47) 237.
[98] The determination whether a Louis Vuitton bag is authentic necessitates a detailed examination of its stitches.
[99] Surden (n 73) 683, 685.

limited in its processing capabilities and permits only very simple transactions. Effectively, smart contracts become synonymous with the automated execution of simple payment instructions. More complex transactions require more advanced blockchains or protocols running on top of them. Such blockchains or protocols may not, however, be decentralized, secure or trustless. In effect, smart contracts must simply be seen as programs operating in distributed computing environments and not as a semi-mythical technology liberating the contracting parties from the shackles of traditional legal and financial institutions. As a result, the revolutionary claims concerning disintermediation and self-enforcement made in the context of the original blockchain can no longer be made. Moreover, once smart contracts involve the performance of obligations in the real world, their operation involves multiple dependencies on external, centralized and (possibly) insecure entities. As smart contracts can only be as trustless as the oracle and the data source that provide them with information about off-chain events, the fact that the blockchain itself is trustless and secure, becomes largely irrelevant. In other words, once smart contracts involve more complex transactions, i.e. exchanges that go beyond the movement of tokens in response to the presentation of one or more private keys, the benefits of putting them on the blockchain seem easily lost.

Another set of problems, unrelated to blockchains *per se*, concerns the subject matter of smart contracts. Apart from the theoretical question whether full automation (i.e. self-enforcement) of certain obligations is advisable, it must be acknowledged that from a practical perspective, only *some* contractual obligations can be expressed in code and only *some* obligations can be regarded as computationally ascertainable. Consequently, smart contracts may be suitable for a rather narrow range of transactions. They are difficult to envisage for contracts involving obligations relying on such concepts as "reasonableness," "best efforts" or for obligations the fulfillment of which requires an overall evaluation of contractual performance. Given their inability to fully reflect and allow for the complexities of the real world, the practical use of smart contract may thus be limited. At the same time, on a theoretical level, the provide fertile ground for research in the area of computational law and force practitioners to be more aware of ongoing developments in programming languages. In light of the frequent suggestions concerning the development of a language that could serve as a common ground for lawyers and developers or even permit the direct coding of contractual obligations, it is also necessary to inquire about the *desirability* of reducing (or even eliminating) ambiguity in contracts and to investigate the trade-offs between precision and certainty on one hand and ambiguity and flexibility on the other. The reduction of ambiguity, praised by technical writings as one of the main benefits of smart contracts, may be less attractive than originally assumed. Self-enforcing smart contracts are rigid, deterministic and insulated from their commercial context. The elimination of human judgment and the automation of choice can easily evolve into a situation where the contracting parties effectively lose the ability to choose whether and how to exercise their rights. Furthermore, it must not be forgotten that smart contracts can ensure perfect performance and lower transaction costs resulting from the elimination of intermediaries *and* traditional enforcement mechanisms only on the assumption that their code is perfect: it correctly reflects the parties' commercial agreement, it contains no coding errors and no security loopholes. Needless to say, such assumption cannot be made. Interestingly, the technical shortcomings of smart contracts seem to increase the need for prior agreement and/or complex regimes allocating liability for coding errors and/or irregularities in the functioning of oracles and data sources. Smart contracts may eliminate human bias and the risk of non-performance, but they introduce the risks of programming errors, security breaches and discrepancies between original intent and actual implementation. Given that the aforesaid events may give raise to complex disputes, it is unlikely that smart contracts will reduce the need for lawyers and courts or otherwise diminish the significance of the legal profession. For the time being, the opposite may be expected.