Smart Contracts: A Requiem

Eliza Mik*

"Smart contracts" are technologies that facilitate the generation and transfer of blockchain-based crypto-assets. The unfortunate labelling of these technologies as 'contracts' has spawned a plethora of legal theories, which are built on unsubstantiated technical claims, terminological misunderstandings as well as on a general disillusionment with traditional institutions. Concepts such as "validation" or "self-enforcement," both of which constitute permanent fixtures of the "smart contract" narrative, seem to have hijacked common sense with promises of certainty and guaranteed performance to the point where a structured and logical argument is rendered difficult. It is necessary to clarify the meaning of the term and remind those who debate the legal enforceability and validity of "smart contracts" of the basic principles of contract law. In particular, it is necessary to recall that contract law is indifferent to the manner parties express their agreement

Introduction

This paper presents the concept of "smart contracts" and examines how they fit within contract law. On a practical level, "smart contracts" form part of the broader phenomenon of so-called blockchains, or distributed ledgers, which are touted as a technology capable of revolutionizing commercial practices and contract law. On a theoretical level, they exemplify the trend to rely on technology to improve or even replace traditional institutions and to substitute subjective, unpredictable human decision-making with objective, predictable algorithms. In the world of "smart contracts," trust is based on computation, not on reliance on the legal system or prior knowledge of a counterparty. Purportedly, such "digitised trust" ensures that what has been agreed on is actually done.[1] Although the legal controversies surrounding "smart contracts" may derive from a series of terminological and technological misunderstandings, the expression of contractual obligations in computer code and the delegation of performance to automated processes do raise some interesting questions. Existing analyses of this area are, however, rendered difficult by an undisciplined use of domain-specific terminology. While lawyers construct arguments based on ill-understood technical concepts, coders can be blamed for an uninformed use of legal nomenclature. This paper demonstrates that contrary to popular claims, "smart contracts" do not create new legal problems and that questions regarding their enforceability or validity are inherently misplaced. It demonstrates how the mis-labelling of a technological phenomenon has spawned a pseudo-legal framework that captured the imagination of lawyers and regulators. Before putting the "smart contract" debate to rest, however, it is necessary to confront the popular claims concerning their legal significance.

[1] J Sklaroff, 'Smart Contracts and the Cost of Inflexibility' (2017) 166 *U Penn L Rev* 263 at 265.

The paper commences with an overview of the definitions of "smart contracts" and an explanation of their relationship with blockchains. Next, it addresses the misunderstandings that underlie many popular claims and dissects such core concepts as "validation" and "trustlessness." The paper proceeds to discuss the legal status of "smart contracts." Abstracting from situations where the term is used in a strictly technical sense, it explores whether "smart contracts" *can* be contracts. The paper also analyses the relationship between enforceability and so-called "self-enforcement." In this context, it recalls the seemingly obvious difference between performing and enforcing obligations. After addressing the question whether "smart contracts" require enabling legislation, the paper reviews a number of interesting issues that will accompany their deployment – even if they are not contracts in the legal sense. The reader must be warned that this paper is written as a reaction to claims and theories that are often illogical or difficult to follow. As "smart contracts" are contracts in name only, trying to analyze them within the context of contract law resembles "trying to fit a square peg into a round hole."

The Term and the Technology

"Definitions"

The term "smart contract" is used inconsistently. One of the original definitions associates it with the embedding of legal terms in hardware and software to prevent breach or to control assets by digital means,[2] while a recent report published by the US National Institute of Standards and Technology describes a "smart contract" as a "collection of code and data (sometimes referred to as functions and state) that is deployed using cryptographically signed transactions on the blockchain network."[3] The term "smart contract" has gained popularity with the development of the Ethereum blockchain,[4] which was purposefully created to enable their use. Unfortunately, the Ethereum website adds to the confusion. Although it defines "smart contracts" as "applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference," it also uses the term interchangeably with "cryptocurrencies," "tokens," methods of creating or transferring tokens as well as escrow and crowdfunding mechanisms. Other definitions range from the technical, such as "persistent scripts" to the fantastical, such as "decentralized agreements built in computer code and stored on a blockchain."[5]

Absent a standard definition, it is impossible to generalize or make assumptions as to the universal attributes of "smart contracts." It is also illogical to inquire *"are smart contracts enforceable?"* as each instance of *a "*smart contract" must be analysed individually. The legal issues, if any, depend on the definition taken as a point of departure. Unfortunately, despite the absence of a standardized meaning, technical and legal literature abound in claims that "smart contracts" increase legal certainty by technologically guaranteeing performance or that by precluding the possibility of breach, "smart contracts" prevent disputes and eliminate the need for traditional institutions of enforcement. This paper leaves aside such unrealistic claims and proposes that "smart contracts" are technological means for the automation of payment obligations[6] or obligations consisting in the transfer of

[2] Nick Szabo, 'Smart Contracts: Formalizing and Securing Relationships on Public Networks' (1997) 2 (9) *First Monday*; for a general discussion see: E Mik, 'Smart Contracts: Terminology, Technical Limitations and Real-World Complexity' (2017) 9 *LIT* 269.
[3] D J Yaga et al, *Blockchain Technology Overview,* National Institute of Standards and Technology Internal/Interagency Report 8202, 2018, p 54; a similar definition can be found in the tutorial for the most popular "smart contract" programming language, Solidity.
[4] Vitalik Buterin, 'Ethereum White Paper: A Next Generation Smart Contract and Decentralized Application Platform' (2015).
[5] Sklaroff, above, n 1 at 263.
[6] J Cieplak, S Leefatt, 'Smart Contracts: A Smart Way To Automate Performance' (2017) 1 *Georgia L & Tech Rev* 417.

tokens or cryptocurrencies ("crypto-assets").[7] They are not agreements but means of performing obligations deriving from other agreements.

Relationship with blockchains

Although the concept of a "smart contract" dates back to the mid-90's, it only gained in popularity in conjunction with so-called blockchains, i.e. decentralized, peer-validated, cryptographic ledgers providing publicly visible, chronological and permanent record of prior transactions. While it is generally accepted that blockchains are databases, they also lack a universal definition.[8] The original bitcoin blockchain has been designed for one purpose: the prevention of double spending of cryptocurrencies in a system without a centralized entity controlling their issuance or transfer.[9] To this end, it presents a transparent state of accounts and enables transfers of cryptocurrencies between such accounts. The combination of peer-to-peer networking and cryptographic technologies ensures that such transfers are immediately recorded and visible to all. In contrast, Ethereum has been created to equip blockchains with more complex functionality, going beyond the "mere" recording of the generation and transfer of cryptocurrencies. In many contexts, "smart contracts" seem to denote this additional functionality. There are hundreds of blockchains with different attributes and capabilities.[10] Some support only pre-defined, simple "smart contracts," others permit the creation of more complex "smart contracts." In the first instance, the users of a particular blockchain are confined to its existing functionalities. In the second, users can create their own, purpose-specific "smart contracts" and deploy the blockchain for other purposes than the generation or transfer of crypto-assets.[11] As different blockchains support different "smart contracts," each "smart contract" implementation requires separate analysis. To complicate matters, the characteristics of "smart contracts" depend on how they interact with "their" blockchain - a "technicality" that has important practical implications. In principle, if the "smart contract" forms part of the code of the blockchain (it is *in* the blockchain), it shares its characteristics. If, however, the "smart contract" operates *on* the blockchain it does not share its characteristics. Unfortunately, it is often assumed that given the inherently secure, incorruptible and (as explained below) "trustless" character of the original bitcoin blockchain, *all* blockchain-based "smart contracts" are secure, incorruptible and "trustless" – an assumption that may explain the very fascination surrounding their use. It is generally overlooked, however, that "smart contracts" operating "on top of" their blockchains do not leverage its characteristics. Leaving aside those instances where the term is used synonymously with multi-signature scripts in the narrow context of the bitcoin blockchain, "smart contracts" form part of a larger ecosystem of technologies that are built "on" a specific blockchain. Although they are technically related to "their" blockchain, they do not assume its attributes and must be evaluated like any other computer program. The difference between "regular" computer programs and "smart contracts" lies in the fact that the blockchain provides a distributed (and often decentralized) execution environment, where the code of each "smart contract" is executed by many independent

[7] A cryptocurrency is "a digital asset or unit within the system, which is cryptographically sent from one blockchain network user to another" and a "digital asset" is an asset that is purely digital or is a digital representation of a physical asset, see D J Yaga, above, n 3, p 49.
[8] See generally: Angela Walch, 'The Path of the Blockchain Lexicon (and the Law)' (2016) 36 *Rev Banking & Fin L* 713.
[9] Satoshi Nakamoto, 'Bitcoin: A Peer-to-Peer Electronic Cash System' (2008).
[10] The main two categories are permissionless and permissioned blockchains. The former are open, decentralized and allow anyone to join without disclosing their identity or subscribing to any system rules. The latter are centralized and closed, restricting access to identifiable participants.
[11] J Bacon et al, 'Blockchain Demystified: A Technical and Legal Introduction to Distributed and Centralized Ledgers' (2018) 25 *Rich J Law & Tech* 2 at 47, 48.

nodes. The main benefits of such "distributed execution" lies in its efficiency, security and impartiality. Traditional execution environments, such as operating systems, are generally centralized, i.e. controlled by one entity that can interfere with their operation. They are also inherently prone to security breaches. In contrast, blockchains (at least permissionless ones such as bitcoin or Ethereum) are decentralized and extremely difficult to interfere with. In sum, the difference between "traditional" software applications and "smart contracts" does not lie in the quality of their code but in the manner such code is executed. This important difference is generally ignored leading to the incorrect assumption that "smart contracts" leverage the characteristics of blockchains. This incorrect assumption underlies the common claim that "smart contracts" increase trust between the parties because, once "activated," they execute as coded and thus guarantee performance. As explained below, "executing as coded" and "guaranteeing performance" are two distinct matters.

A Cascade of Misunderstandings

Legal and technical literature abound in claims based on incorrect assumptions and misinterpretations of basic terminology. A vicious circle forms. Technical writings claim that, for example, "smart contracts" reduce the need for courts because they "validate transactions" and "prevent breach." Lawyers believe such statements and, without establishing their technical veracity, build theories around them. Upon closer analysis, it transpires that the technically-minded author of the claim did not understand the relevant legal terms and, surprisingly, the very technology. Unfortunately, the theories built around the original claim are endlessly recycled until they become part of the mainstream view on what "smart contracts" are and what they can do. The source of this conceptual morass may be the original paper on "smart contracts,"[12] which constructs an entire "legal" theory on how they will revolutionize commerce and obliterate the legal profession. At first glance, the paper impresses with its use of legal terminology and bold promises of the technological optimization of contract law. Readers with no legal background will hardly notice that the author conflates public and private law, fails to distinguish between contract and property and, most importantly, misunderstands basic concepts of contract law.[13] Similarly, readers with no technical background may not appreciate that many technical writings grossly misrepresent the actual, technical capabilities of both blockchains and "smart contracts." In sum, many claims and theories derive from an uninformed use of legal and technical terms. Specific examples follow.

First, technical literature frequently mentions "transactions." Legal literature seems to assume that the term refers to an exchange or is virtually synonymous with "contract." Technically, however, a transaction is a modification of a state of a database[14] or a "single execution of a program."[15] In the blockchain context, the term denotes the transfer of a crypto-asset from one account to another. As blockchains are distributed databases, this definition emphasizes their changed state: after the transaction account A has one bitcoin less and account B has one bitcoin more. Pertinently, the technical meaning of "transaction" refers to a *unilateral* act – a one-way transition from

[12] N Szabo, above, n 2.
[13] The "seminal" paper describes vending machines as "contracts with bearer" and claims that "most contractual dispute (sic) involves an unforeseen or unspecified eventuality. A common example of an unspecified eventuality is that the parties behind a pseudonym might change." It defines privity as "the principle that knowledge and control over the contents and performance of a contract should be distributed among parties only as much as is necessary for the performance of that contract."
[14] see generally: M T Özsu, P Valduriez, *Principles of Distributed Database Systems*, 3rd ed, Springer, New York, 2016, p 335.
[15] J D Ullman, *Principles of Database and Knowledge Base Systems*, vol 1, Computer Science Press, New York, 1988, pp 300, 301, 337.

one state to another. It is unrelated to the bilateral, reciprocal nature of transactions in a commercial and legal sense. The practical implementations of "smart contracts" confirm their unilateral character.

Second, the main selling point of blockchains is their "trustlessness." The concept denotes the ability to establish the truth of a state of affairs without recourse to a trusted third party in an environment where no-one can be trusted. [16] Technically, it addresses the problem of imperfect information regarding the failure of individual components in distributed computing systems. In the narrower blockchain context, the state of affairs concerns the contents of a database and its truth is established by means of 'distributed consensus,' i.e. the confirmation by a majority of nodes in a computer network that certain technical requirements have been met. "Trustlessness" is associated with the practical inability to corrupt the contents of the blockchain and with the trustworthy nature of the whole system. The resulting theory is that the ability to trust the blockchain (or, more specifically, the consensus mechanism that generates the individual blocks) obviates the need to trust traditional legal institutions or the counterparty. Trust in people is replaced with trust in technology. It is not, however, immediately apparent how a technical concept from the area of distributed computing could replace legal institutions or how a mechanism that confirms the contents of a database could revolutionize contract law. The following facts are commonly overlooked:

(a) in order to trust the code one must trust those who code – it is humans who created the consensus algorithm; and

(b) the state of affairs that is confirmed by the consensus algorithm only concerns the contents of the ledger, i.e. how many bitcoins (or other crypto-assets) were generated and transferred.

Third, technical writings emphasize that blockchains "validate" transactions or other events. Legal literature may have assumed that the technical meaning of the term overlaps with its legal connotations: establishing compliance with the law or otherwise attesting to the veracity of a fact or statement. If blockchains could validate that, for example, a contract is legally enforceable or that the other party is the rightful owner of certain goods, they would provide an attractive solution, indeed. The technical meaning of validation is, however, very narrow. To understand "validation," we must explore the mechanism underlying the *creation* of individual blocks. This process involves the validation of (a) transactions and (b) blocks. As indicated, a "transaction" denotes the unilateral transfer of crypto-assets from one blockchain account to another.[17] The blockchain itself consists of interconnected blocks, each of which contains a list of transactions. The first step concerns the validation of transactions. Only a valid transaction, one that fulfils *all* technical criteria, can be aggregated into a block. All nodes in the network[18] must confirm that the transaction is technically correct in terms of, amongst others, syntax, size and data structure, number of signature operations and that it uses previously unspent inputs and contains sufficient transaction fees.[19] The nodes must also confirm that the unlocking scripts match the corresponding

[16] Leslie Lampert et al, 'The Byzantine Generals Problem' (1982) 4 *TOPLAS* 382.
[17] A M Antonopoulos, *Mastering Bitcoin,* 2nd ed, O'Reilly, Sevastopol, 2017, p 18.
[18] A node is computer in the blockchain network that runs a full version of the requisite software.
[19] Antonopoulos, above, n 17, pp 24, 25.

locking scripts.[20] The second step concerns the validation of blocks. This process involves so-called mining, i.e. finding a solution to the Proof-of-Work algorithm by repeatedly hashing the block, until the resulting output matches a specified target. In practice, this means that many powerful computers compete to find a particular number. Next, each newly "mined" block is validated by every node against certain criteria, which include establishing that its data structure is syntactically correct, its header hash is less than the target, its timestamp is less than two hours in the future, it is of acceptable size and all transactions contained therein are valid.[21] This somewhat lengthy description demonstrates that "validation" denotes the fulfilment of certain technical requirements. The validating "entities" (nodes) are deterministic computer processes *blindly* following the consensus algorithm. Validation is based exclusively on computing power not on competence or specialized knowledge. More importantly, the validation process is "inward looking" – it can only evaluate the technical parameters pertaining to the generation of blocks or the initiation of transactions. It is not "outward looking" – it cannot detect and hence verify any events in the real world, e.g. whether the parties had legal capacity, whether they were rightful "owners" of the crypto-asset in question, etc. The background of the transaction, including the legality of the contractual subject matter or the trustworthiness of the counter-party, exist *outside of the blockchain* and cannot be validated. If goods are to be paid for with a cryptocurrency, the blockchain cannot determine whether they were delivered or whether they conform with their contractual description. The contracting parties must rely on third parties (so-called "oracles") that confirm delivery and, if possible, conformance. As "oracles" exist outside of the blockchain, they are not "trustless" and there is no technical mechanism preventing them from misrepresenting outside events or providing incorrect information.[22] Consequently, the "trustless" character of the blockchain is of little value if "smart contracts" must rely on sources of information, which are not "trustless." In sum, blockchains work well with information pertaining to their own state. They cannot, however, validate information concerning the state of anything in real world.

Two clarifications are required. *First*, consensus algorithms must not be associated with agreement in the legal sense. Consensus algorithms involve the automated and deterministic execution of a computer program by nodes in a blockchain network – not an actual agreement between the members of the system, i.e. the persons operating such nodes. "Agreement" in the legal sense carries notions of freedom and choice, of the ability to make decisions within a certain range. Consensus in the technical sense, however, leaves no room for individual decisions beyond what *is permitted* by the underlying program. The decisions pertain only to the state of the blockchain: accept valid transactions and/or blocks, reject invalid ones.[23] To ensure "trustlessness" of the system, the consensus algorithm must be rigid and cannot permit any individual discretion. *Second*, consensus algorithms cannot validate the code of "smart contracts." As indicated, the latter do not leverage the "trustlessness" of their underlying blockchains. Although consensus algorithms effectively guarantee the decentralised execution of "smart contracts," they do not guarantee the quality of their code. When technical literature speaks of the validation of "smart contracts" it generally refers to their formal verification, i.e. the act of proving or disproving the correctness of their underlying algorithms, as evaluated against a formal specification, by means of mathematical techniques.[24]

[20] Antonopoulos, above, n 17, pp 218, 219.
[21] Antonopoulos, above, n 17, pp 238, 239
[22] V Gatteschi et al, 'Blockchain and Smart Contracts for Insurance: is the Technology Mature Enough?' (2018) 10 *Future Internet* 20.
[23] This statement must be qualified as miners can deliberately ignore certain transactions, irrespective of the attached transaction fee.
[24] D Magazzeni, et al, 'Validation and verification of smart contracts: a research agenda,' (2017) 9 *Computer* 50, at 54.

Consensus algorithms, however, cannot perform a formal verification of "smart contract" code and cannot detect coding errors. In effect, "smart contracts" are susceptible to programming errors, both accidental (which are statistically inevitable)[25] and intentional. The resulting security challenges are increasingly recognized.[26] As "smart contracts" are supposed to be immutable, they cannot be amended or stopped.[27] Their ability to guarantee performance is, after all, premised on the impossibility of any interference with their execution and any modification of their code. This means, however, that the code of "smart contracts" cannot be changed - even in the event of a change of circumstances or the discovery of coding errors. Given that an inspection of the code cannot reveal how it will execute the ability to view the code of a "smart contract" is of little value.[28]

The Legal Status of "Smart Contracts"

It is often claimed that "smart contracts" exist independently of the law or "represent a technological alternative to the whole legal system."[29] This reasoning probably derives from the assumption that the technical attributes of blockchains, particularly their decentralised character, place them outside of the purview of traditional legal institutions. It also reflects the recurring theory that technological advancements warrant a detachment from the legal system.[30] While the latter continues to be portrayed as outdated (if not altogether redundant), it is increasingly recognized that for "smart contracts" to gain mainstream commercial acceptance, they must fit within the existing law or be legally enforceable. It has also been suggested that while "smart contracts" eliminate the *need for* legal enforcement, they should not preclude the *possibility of* such enforcement.[31] The reasoning is that "smart contracts" are not intended to be legally enforceable in the sense that the question of their enforceability should not arise in practice because their very nature precludes breach and hence disputes. It is argued, however, that "smart contracts" are still intended to be contracts.[32] In simpler terms: while there is no *need* to enforce "smart contracts" in the courts of law, the parties should retain the *option* to do so. This approach attempts to reconcile claims regarding the technological superiority (and resulting independence) of "smart contracts" with a recognition that the legal system *cannot* or *should not* be excluded.

It is, of course, illogical to debate the validity or enforceability of "persistent scripts" or "stateful objects on the blockchain." The legal status of "smart contracts" can only be contemplated if they create, embody or execute contractual rights and obligations. The discussion must be divided into two parts. First, we must inquire whether "smart contracts" *can* be contracts in the legal sense. Second, after clearing the doctrinal hurdles, we must re-examine the question on a practical level. This peculiar analytical sequence is dictated by the need to address the common doubts expressed in legal and technical writings, which concern the possibility of representing contracts in code and of automating their formation or performance. It is also dictated by the need to distinguish between

[25] The industry average is about 1-25 errors per 1000 lines of code, see: S McConnell, *Code Complete*, 2nd ed, Microsoft Press, Washington, 2004, p 521.
[26] N Atzei, M Bartoletti, T Cimoli, 'A Survey of Attacks on Ethereum Smart Contracts (SoK)' in: M Maffei, M Ryan, eds, *Proceedings of the 6th International Conference on Principles of Security and Trust,* Springer, New York, 2017, p 10.
[27] Christopher D Clack, et al., 'Smart Contract Templates: foundations, design landscape and research directions' (2016) ArXiv e-prints 4.
[28] See generally: K Bhargavan, et al, 'Formal Verification of Smart Contracts: Short Paper' in T Murray and D Stefan, eds, *PLAS,* ACM New York, New York, 2016, at 91.
[29] One article states: "Ostensibly, smart contracts remove the role of courts as enforcement agents. One might say that the contract enforces itself, or that it is enforced by the code. What this means is that parties no longer have the escape hatch of litigation." K Werbach, N Cornell, 'Contracts Ex Machina' (2017) 67 *Duke L J* 313 at 352.
[30] D R Johnson, D Post, 'Law and Borders -The Rise of Law in Cyberspace' (1996) 48 *Stan L Rev* 1367.
[31] Werbach & Cornell, above, n 29, at 352.
[32] Werbach & Cornell, above, n 29, at 353.

the theoretical and the practical aspects of "smart contracts." If, theoretically, an agreement can be expressed in code and automated but if, in practice, existing implementations of "smart contracts" are not agreements, the entire debate seems futile. Questions or validity or enforceability must, however, be preceded by a discussion of "self-enforcement." The latter, purportedly, constitutes a superior alternative to traditional methods of enforcement.

What is "self-enforcement"?

It is claimed that, given the impartial and deterministic character of code, contracts should be enforced by computers not by humans. Judicial enforcement should be replaced with algorithmic enforcement.[33] Purportedly, the value of "self-enforcement" lies in the elimination of *any* human discretion – be it on the side of the contracting parties or on the side of adjudicators. The theory is that as no-one, including the contracting parties, can change the code or interfere with the operation of a "smart contract," its performance is guaranteed. Consequently, "self-enforcement" is associated with the prevention of breach, the reduction of the potential for disputes and with legal as well as commercial certainty. Whereas legal enforcement concerns the protection of performance by means of judicial assistance, technical "*self*-enforcement" ensures actual performance and obviates the need for such assistance. This approach raises numerous objections.

Seemingly, blockchain enthusiasts have derived the concept of "self-enforcement" from the technical term "self-execution." In fact, many technical writings use these terms interchangeably. "Self-execution" refers to computer programs that autonomously (i.e. without the need for ongoing human input) change their state according to pre-defined rules. In this sense, "self-enforcement" denotes that the code operates irrespective of external circumstances and thus guarantees the achievement of the pre-programmed result. Unfortunately, in some contexts the meaning of "self-enforcement" approximates "performance." In fact, it is often claimed that the blockchain, "rather than any appendage of the state," ensures performance.[34] Such claims are non-sensical. Apart from the fact that performance and enforcement concern two distinct stages in the life-cycle of a contract, it bears emphasizing that in common law systems the state does not *ensure* performance as orders for specific performance are granted in exceptional circumstances. The state only *protects* performance. Technical writings fail to differentiate between enforcing and performing contracts, conflating technical and legal terms to a point where they cannot be used in a structured argument. As enforcement always involves third parties or a recourse to an external system, the very idea of a contract enforcing itself is absurd. So is the idea of a contract performing itself. To state the obvious: contracts are performed by the parties to the contract and enforcement relates to the protection of performance by third parties who grant damages or, on rare occasions, require the contract breaker to perform his or her obligations. In fact, the very idea of enforceability is tied to the institutional backing provided by courts and other entities, such as arbitral tribunals or mediators, which are called upon in the event of disputes.[35] Lastly, on a technical level, it is incorrect to claim that blockchains ensure performance. Blockchains can only ensure that the code of

[33] Werbach & Cornell, above, n 29, at 365.
[34] Werbach & Cornell, above, n 29, at 332.
[35] J W Carter, *Carter on Contract*, Lexis Nexis, Sydney, [year?], para 01-140.

the "smart contract" is securely executed by a distributed network of computers. As blockchains cannot ensure that the code correctly represents the underlying obligation and/or does not contain coding-errors, they cannot be said to guarantee performance. To re-emphasize: a blockchain may execute a "smart contract" that does not correspond to the original agreement. This fact alone creates potential for disputes and confirms the need for traditional methods of enforcement – for it cannot be claimed that a distributed network of computers can not only perform contracts but also *resolve* disputes.

Can "smart contracts" be contracts in the legal sense?

In essence, assuming the traditional principles of contract law apply, the main pre-requisites of a legally binding contract are intention and consideration. The concept of intention is inextricably linked to the concept of agreement. Naturally, an agreement is the product of two parties *intending* to be bound by their respective promises. The concept of "consideration" emphasizes the mutual and reciprocal nature of contracts as bargains.[36] Technically, consideration does not determine whether an agreement is a contract but whether a promise is enforceable. The "smart contract" narrative commonly reduces problems of their validity and/or enforceability to the questions whether agreements can be expressed in code and whether their formation and performance can be automated. It is worth emphasizing that contract law does not always draw a clear the difference between enforceability and validity. A seemingly valid contract may be unenforceable due to the presence of vitiating factors, such as economic duress or misrepresentation, as well as on grounds of illegality or public policy. A contract will also not be enforceable if one or both parties lack contractual capacity or, in rare circumstances, by reason of the absence of written evidence where such is required by statute.

The expression of intention

Technical and legal writings often inquire whether contracts can be expressed in code or whether "code can be enforceable." There are, however, no doctrinal obstacles for such expression and questions regarding the "enforceability of code" are absurd. Subject to certain statutory exceptions, which dictate that some contracts be made or evidenced in writing, contract law does not prescribe how agreements should be made.[37] Contractual intention can be expressed in any manner.[38] Agreements are put in writing mainly for evidentiary purposes. The ongoing debate concerning the enforceability of "smart contracts" (which are, by definition, written in code) derives from a failure to distinguish between the *substance* of the contract, i.e. the rights and obligations agreed by the parties, and its *expression*, i.e. the document containing the words describing such. To illustrate this misunderstanding, two examples follow. First, the statement that "court enforce contracts" constitutes a linguistic shortcut. Courts do not enforce contracts but *individual obligations*, be it by ordering the contract breaker to pay a sum of money in damages for non-performance (or non-conforming performance) or, as a matter of exception, by ordering its actual performance. The contract *as a whole* is not enforced, if only because many of its clauses do not describe any obligations. Similarly, when we state that "the court enforces a clause or term in a contract" we mean that the court enforces the obligation *described* in such clause or term. As obligations are expressed in words that are grouped into clauses, inquiring whether code is enforceable is tantamount to asking whether *words*

[36] J W Carter, *Carter on Contract*, Lexis Nexis, Sydney, [year?], para 01-140.
[37] UK Law Commission Consultation Paper No 237, *Electronic Execution of Documents* (21 August 2018) at 20.
[38] J W Carter, *Carter on Contract*, Lexis Nexis, Sydney, [year?], para 01-140.

are enforceable. Courts enforce obligations, not words or documents. Leaving aside that the parties are free to use any type of symbol to manifest or memorialize their agreement, it must also be observed that with the exception of low-level machine languages (where code takes the form of numbers) most code takes the form of words. In this sense, creating a contract in code is similar (if not identical) to creating a contract in a foreign language. It is true, however, that code may be difficult to understand by persons with no technical background.

Second, we must then ask: can agreements be expressed in code, if the parties do not understand it? The answer is yes. To illustrate: the parties form a contract and decide to put it in writing. Although both parties are native English speakers (and the contract has been formed in Australia and is governed by the laws of New South Wales), they decide for the written document to be in Chinese. Contract law does not prohibit for the contract to be expressed in a foreign language – even if neither of the parties understands such. The contract written in Chinese ("Chinese contract") is legally enforceable - assuming that there is consideration and intention, its subject matter is not illegal, the parties have contractual capacity and no vitiating factors are present.

Two variations can be considered:

(a) The parties create an English version of the contract and agree to translate it into Chinese. They agree that the Chinese version shall prevail and that the English version should be disregarded. As neither party understands Chinese, they are at the mercy of the translator and can never be certain as to its contents.

(b) The parties instruct a Chinese lawyer to express their agreement in Chinese from the outset. There is no original document written in English - only verbal instructions. It may not make a difference whether the instructions are verbal or in writing. The point is that there is no contract in English that is translated. The only version is in Chinese. Again, the parties must trust whoever created such contract and can never be certain as to its contents.

In both instances, the contract is enforceable, given that *in theory* the language of expression is irrelevant. The parties have agreed for the "Chinese contract" to govern their relationship and have assumed the accompanying risks. The contents of the "Chinese contract" become relevant in the event of a dispute. Admittedly, the usual problems accompanying contractual interpretation become aggravated by having to deduct the English meaning of individual words or clauses from the Chinese version and by potential disputes concerning the correctness of the translation. In principle, however, once the contract is translated from Chinese into English, the usual standards of interpretation apply.[39] The difference between situation (a) and (b) lies in the *evidentiary* difficulty of determining the original substance of the agreement.

When we replace Chinese language with code, the futility of the "smart contract" debate becomes apparent. The source of the parties' respective obligations is the agreement – not words or documents. Contract law is agnostic to the manner of expressing agreement. Of course, the manner of expression affects the difficulty of establishing *what* has been agreed. Somewhat metaphysically, the agreement can be said to exist independently of the

[39] J W Carter, *The Construction of Commercial Contracts*, Hart Publishing, Oxford, 2013, p 413.

document that contains the writing describing the rights and obligations. The physical representation of the agreement, be it words printed on paper or signs displayed on a screen, is not the agreement.[40] The medium used to memorialize the agreement constitutes proof of the parties' obligations but it does not create them. The comparison of code to a foreign language is adequate because in both instances, the parties do not understand the expression of their agreement and must rely on third parties to express their obligations. The difference between the "Chinese contract" and a contract written in code lies in the fact that the latter can be executed by computers.

Consideration and Exchange

Moving on to the second prerequisite, consideration, we must differentiate between two questions: "can crypto-assets constitute consideration?" and "does the "smart contract" involve an exchange?" Regarding the first question, it is trite law that consideration need not be adequate, it only needs to be sufficient in the eyes of the law.[41] The parties can exchange money for goods, peppercorns for cars and bitcoins for crypto-assets. Irrespective of their controversial legal status in some jurisdictions, bitcoins or tokens can constitute valid consideration. Problems concern the second question. While contract law does not require equivalence of value, it requires or *assumes* reciprocity. It is, after all, the very function of contracts (and of contract law) to facilitate bargains: one party does something for or promises something to the other party and, in return, obtains a promise or an act from that party. [42] Most contracts are bilateral in nature: two parties *make and receive* executory promises. In unilateral contracts: one party makes an executory promise *in return* for the doing of an act.[43] If there is no exchange, there is no contract. Consequently, if a "smart contract" does not involve reciprocity, questions of its enforceability do not arise. Unfortunately, this problem has passed unnoticed in both legal and technical writings, which fail to observe that "smart contracts" constitute *unilateral* transfer mechanisms (similar to "standing orders") or tools automating certain types of contractual performance. The existing implementations of "smart contracts" confirm that they are software programs enabling the transfer of crypto-assets or, more specifically, the provision of consideration in the form of crypto-assets. At present, the most popular "smart contracts" operating on the Ethereum blockchain are so-called "token contracts." Their sole purpose is to maintain a consistent state of the blockchain, and to keep track of token transfers. Another popular type of "smart contract" are exchanges, both decentralized and centralized, which facilitate the sale and purchase of tokens or cryptocurrencies in return for other crypto-assets. In this context, the term "exchange" refers to a transacting platform and "smart contracts" constitute components of such platforms. They facilitate exchanges in the technical, not in the legal sense. This approach is confirmed by the US Securities and Exchange Commission, which categorizes "smart contracts" as methods that facilitate electronic trading in digital asset securities[44] and by the Australian National Blockchain consortium, which regards them as protocols facilitating the transfer of digital assets between parties under the agreed-upon terms. In sum, "smart contracts" are computer programs enabling the performance of obligations pertaining to the transfer of crypto-assets. They operate (or *execute*?) *as a result* and *in performance of* an agreement.

[40] Adam J Kolber, 'Not-So-Smart Blochchain Contracts and Artificial Responsibility' (2018) *Stan Tech L Rev* 198, at 219.
[41] *Currie v Misa* (1875) LR 10 Ex 153; *Chappell & Co Ltd v Nestle Co Ltd* [1960] A C 87 (H L).
[42] Ian Macneil, Relational Contract Theory as Sociology (1987) 143 *JITE* 272, at 274.
[43] J W Carter, *Carter on Contract*, Lexis Nexis, Sydney, [year?], para 01-010.
[44] US Securities and Exchange Commission, *Statement on Digital Asset Securities Issuance and Trading*, 2018.

Is Automation a Problem?

The enforceability of "smart contracts" is also being questioned on the ground that they operate, or "self-enforce," without human intervention or supervision. Again, there are no doctrinal obstacles to automating both the formation of contracts and their performance. Computers are often deployed to "enter into contracts" within a range of pre-determined parameters and/or to perform such contracts with no human participation. Typical examples are e-commerce websites, such as Amazon or eBay, which fully automate the transaction from the side of the merchant. The possibility of automating transactions and expressing contractual intention by automated means has been acknowledged by common law[45] and is also recognized by e-commerce regulations, such as the United Nation Convention on the Use of Electronic Communications in International Contracts.[46] Similarly, the Australian Electronic Transactions Act 1999 (Cth) confirms the ability to form contracts by electronic and automated processes. ETA Section 8 states that a transaction is not invalid merely because it took place by means of electronic communications. While these provisions only refer to the electronic *form* of the contract, Section 15C clarifies that the formation process can be automated. It provides that a contract formed by the interaction of an automated message system and a natural person, or the interaction of automated message systems, is not unenforceable purely for the absence of direct human involvement. According to the statutory definition in Section 5, an "automated message system" includes a computer program, "used to initiate an action or respond to data messages in whole or in part, without review or intervention by a natural person each time an action is initiated or a response generated by the system." The key terms are *initiation* and *generation* – the computer is not a mere conduit transmitting legally relevant statements but a "generator" of such statements in the sense that it automatically creates certain content, such as an offer, based on prior human instructions that prescribed the parameters of such content. Those provisions envisage situations where agreements are formed without *any* direct human participation. The underlying assumptions are that (a) a human has created of and/or remains in control of the program that enables such "automated formation," and (b) that the program is never regarded as a legal entity by itself. As none of the above instruments distinguishes between different technologies or degrees of automation, it can be assumed that it captures "smart contracts."

Do Questions of Enforceability Arise?

Although there are no theoretical obstacles for "smart contracts" to be valid or legally enforceable, the entire debate may be futile. Questions of validity or enforceability do not arise if a "smart contract" is the automated performance of an obligation deriving from an underlying agreement or, as illustrated in the discussion on consideration, a mechanism facilitating the generation, transfer and exchange of crypto-assets. Interestingly, the same technical writings that debate the enforceability of "smart contracts" also state that a "smart contract" will perform something, perform itself, execute a contract or that the parties agree what the "smart contract" will do. Contracts do not perform obligations, execute themselves and parties do not agree what *contracts* will do. Parties agree what *they* will do. In similar vein, "smart contracts" are often equated with offers, vending machines or escrows.[47] Needless to say, it is possible to analyse offers or vending machines in terms of their legal effect – but not in terms of enforceability. If something is an offer or a vending machine, it is not a contract.[48] In addition,

[45] *Thornton v Shoe Lane Parking Ltd* [1971] 2 QB 163, [1971] All ER 686.
[46] Convention on the Use of Electronic Communications in International Contracting, Nov 23, 2005, U N Doc A/60/21, Article 12.
[47] Smart Contract Alliance, 'Smart Contracts: Is the Law Ready?' September 2018.
[48] This fact has been overlooked by a group of 12 lawyers who participated in preparing the said whitepaper on Smart Contracts.

contracts involve at least two parties. One cannot contract with himself or herself even if one purports to contract in two different capacities. "Unilateral contracts" do not form an exception to this rule: they are contracts in which the consideration provided by the promisee is executed, usually by doing an act at the request of the promisor. Being a computer program, however, the "smart contract" has no legal capacity and cannot be a party. Unfortunately, technical writings often state that people transact *with* and send value *to* "smart contracts," treating them as discrete entities or quasi participants in the transaction. Lastly, it must be acknowledged that even if "smart contracts" were valid contracts in the legal sense, their enforceability could be affected by circumstances and events that exist in the real world and are unrelated to the fact that they expressed in code or formed by automated means. A valid "smart contract" could be unenforceable due to, for example, the illegality of the subject matter.

Notwithstanding the foregoing, it must be acknowledged that although "smart contracts" are not contracts in the legal sense, they can have legal effects. They can, for example, result in the formation or the discharge of a contract. In certain contexts, a "smart contract" can constitute an offer to purchase crypto-assets that can be accepted by payment in a different crypto-asset or with "traditional" money. A "smart contract" can also transfer crypto-currencies as payment for the performance of another obligation.

Do "Smart Contracts" Require Enabling Regulations?

There have been multiple attempts[49] to regulate "smart contracts." In February 2017, the Arizona state legislature passed a statute[50] providing that "smart contracts may exist in commerce. A contract relating to a transaction may not be denied legal effect, validity or enforceability solely because that contract contains a smart contract term." The statute defines a "smart contract" as an "event-driven program, with state, that runs on a distributed, decentralized, shared and replicated ledger and that can take custody over and instruct transfer of assets on that ledger."[51] Some observations are required. First, it is unclear why an "event-driven program" would require legal recognition and why this term was chosen in the first place given that it is never used in technical writings. It seems unreasonable for a regulator to pre-emptively define a technical concept before it has gained an accepted, standardized meaning or to regulate a technology that is still developing. Second, it is unclear why the statute differentiates between "smart contracts" and "smart contract *terms*" or why "smart contracts" *instruct* transfers of assets. Being a computer program, a "smart contract" can only *execute* instructions, not give instructions. Fourth, the statute excludes "smart contracts" operating on permissioned blockchains and "smart contracts" that are "stateless." One can concur with the first exclusion as permissioned ledgers are usually governed by their own master agreements concluded between the entities that are authorized to access and use such ledgers. It is unclear, however, why the legislator distinguished between *stateful* and *stateless* "smart contracts" without clarifying why this specific technical difference should affect their legal status. Why would one regulate only "smart contracts" that have memory, i.e. that update variables in a database and use sequential business logic? The equivalent statute in Tennessee defines a "smart contract" in a similar fashion but adds that the event-driven program executes on a ledger "that is used to automate transactions, including but not limited to, transactions that (a) take custody over

[49] See: Ariz Rev Stat 44-7061 (2017); Nev S B 398 (2017); Cal A B 2658 (2018); Ill H B 2257 (2018); NY S B 8858 (2018); NY A B A8780 (2017) and Ohio S B 300 (2018).
[50] Ariz Rev Stat 44-7061 (2017).
[51] Ariz Rev Stat 44-7061 (2017) Art 5.

and instruct transfer of assets on that ledger; (b) create and distribute electronic assets; (c) synchronize information; or (c) manage identity and use access to software applications."[52] It also states that "no contract relating to a transaction shall be denied legal effect, validity or enforceability solely because that contract is executed through a smart contract."[53] The provision illustrates the wide range of applications of "smart contracts" and, indirectly (or perhaps *unwittingly?*), confirms that the concept is unrelated to the legal meaning of the term. After all, how could a contract "synchronize information" or "execute" another contract?

Unsurprisingly, the statutes have attracted critique. Of particular interest is the response formulated by the Chamber of Digital Commerce ("The Chamber"), a US advocacy group focused on promoting blockchain technology. The Chamber prepared an explanatory document, which contains its own definition of "smart contracts:" "computer code programmed to execute transactions based on pre-defined conditions." [54] The Chamber believes that existing U.S. law already supports the enforceability of smart contracts, that existing law applies to "smart contracts" and that there is no need to treat them differently. Apart from acknowledging the confusion surrounding the term, the Chamber has criticized the temptation to "clarify" the legal status or to "enable" "smart contracts" by legislative means. "Smart contracts" need not be contracts in the legal sense and may only represent or give effect to some obligations. They can be regarded as "the programmatic means by which some or all of the terms of the legal contract are performed" or executed.[55] Logically, "programmatic means" need not be debated in terms of validity or enforceability. The Chamber has, however, expressed concerns regarding situations where legal contracts "overlap" with "smart contracts," a situation that seems to relate to the co-existence of code and natural language in one contractual document. In such instance, questions could *purportedly* arise as to the "validity or enforceability of components to a contract that exist in a digital format." Such concerns seem, however, unjustified. Enforceability can only concern obligations or promises – not other "contractual components." Moreover, as explained above, the manner of expressing agreement or individual obligations is generally irrelevant and the Chamber emphasizes in a different portion of its response that the digital format of "smart contracts" is unproblematic. According to the Chamber, questions pertaining to such format have already been addressed by legislation, which gives legal efficacy to electronic signatures, records, and contracts. The Chamber has also stated that the Electronic Signatures in Global and National Commerce Act ("ESIGN Act") and the Uniform Electronic Transactions Act ("UETA") provide the legal basis for "smart contracts embedded with the terms of a legal contract to be granted efficacy once it is electronically signed."[56] To recall, UETA and ESIGN ensure that (a) if a law requires a signature, an electronic signature satisfies the law, and (b) if a law requires a record to be in writing, an electronic record satisfies the law. They also state that a contract, signature, or related record may not be denied legal effect or enforceability solely because it is in electronic form. As existing laws already provide a foundation for "smart contracts," additional legislation would be redundant and create confusion. These "clarifications" are, unfortunately, inconsistent with the Chamber's recognition that "smart contracts" need not be contracts or that they only constitute the electronic expression of some contractual obligations. It seems unnecessary to find a legal foundation for terms expressed in code or for *signing* a "smart

[52] Tennessee Senate Bill 1662 Section 1, 47-10-201 (2).
[53] Tennessee Senate Bill 1662 Section 1, 47-10-202 (c).
[54] Digital Chamber of Commerce, 'Smart Contracts Legal Primer, Why Smart Contracts Are Valid Under Existing Law and Do Not Require Additional Authorization to Be Enforceable,' 2018.
[55] Digital Chamber of Commerce, above, n 54 at X.
[56] UETA § 5(b), 15 U.S.C. §§ 7001 et seq

contract" *embedded* with the terms of a legal contract. Moreover, the Chamber's observations regarding "signatures" misrepresent the role of signatures in contract law and hence the applicability of these statutes.

First, contracts generally do not require signatures. Legislation enabling electronic signatures is only relevant if a statute requires a signature as part of a formal requirement and where such requirement cannot be fulfilled by means of statutory interpretation. For example, the Australian Electronic Transactions Act 1999 (Cth) governs the way in which formal requirements imposed by federal laws can be satisfied electronically.[57] If, however, no statutory laws apply to "smart contracts," the statute becomes irrelevant. *Second*, in common law jurisdictions the concept of a "signature" is broad and the use of electronic signatures is uncontroversial. The latter can take the form of scanned manuscript signatures, typed names, PINs, clicks on "I agree" buttons, biodynamic versions of handwritten signatures or even email addresses.[58] To state that "smart contracts" can be electronically *signed* is confusing because it implies that (a) signatures are indispensable and (b) the use of electronic signatures is controversial and requires legislative affirmation. The statement also misrepresents the role of signatures in the context of blockchain-based "smart contracts." In such instance, signatures concern the use of a private key within a digital signature scheme. While an "electronic signature" relates to any electronic representation of a signature, a "digital signature" denotes a type of an electronic signature produced by asymmetric key cryptography. The primary purpose of using a private key is to access or transfer crypto-currencies. It is unrelated to the expression of consent or to the authenticating function of signatures.

What are the interesting legal issues?

The broader idea of expressing contracts, or obligations, in code raises interesting legal and technical issues that are increasingly discussed outside of the "smart contract" context. While there are no doctrinal objections to expressing agreements in code, there are multiple challenges accompanying such expression. The ability to guarantee that contracts are performed *exactly as agreed* is predicated on the ability to express contracts in code or, to be more specific, on the ability of code to express obligations exactly as agreed. Computer programs cannot directly "execute" contracts. While they can parse contractual documents, they cannot follow the instructions contained therein as they cannot *understand* natural language. The key challenge lies in translating legal prose into code or expressing contracts in a manner that can be *executed* by computers. Unsurprisingly, technical scholarship focuses on the creation of domain-specific programming languages for "smart contracts" that could capture the semantic richness of legal prose and/or the complexity of the underlying transaction.[59] It is often unclear, however, whether the assumption is that the legal prose of a pre-existing legal agreement will be translated into code or whether the agreement is created in code from the outset. As described below, in the first instance, the programming approach will be aimed at replicating legal prose; in the second, the parties will form contracts with the constraints of the available programming languages in mind. The paragraphs below present a simplified overview of the problems.

[57] The Electronic Transactions Act 2000 (NSW) ("ETA 2000") does the same for the laws of NSW and largely replicates its federal counterpart.
[58] See generally: Law Commission Consultation Paper No 237, p. 12; *Stuart v Hishon* [2013] NSWSC 766 at [34].
[59] F Al Khalil, et al, 'A Solution for the Problems of Translation and Transparency in Smart Contracts,' Government Risk and Compliance Technology Centre, Report, 2017.

Achieving "Equivalence"

Proceeding on the assumption that the "smart contract" constitutes a translation of an existing contract, most difficulties concern the mapping of a contract's denotational semantics, which capture its legal meaning, onto its operational semantics, which concern the execution of the respective obligations.[60] Such mapping requires the translation of abstract concepts embodied in legal prose into specific computer instructions. It is claimed that "smart contracts" can only gain mainstream acceptance if their denotational semantics ("legal language") can be faithfully represented by operational semantics ("technical language" or "code" in its precompiled, human-readable version). The technical language should be as close to the legal language as possible, that is: *equivalent* in terms of content and hence legal effect. The preceding statement hides the complexity of the accompanying problems. While from a legal perspective, it is permissible to express contracts in code, it is technically difficult to ensure a equivalence between legal prose and code[61] - if only due to the fact that lawyers and programmers use different languages and have different goals when preparing their documents or programmes. While lawyers will not shy away from complexity and ambiguity if they serve their client's interests, programmers generally aim for conciseness and simplicity.[62] Although the arguments made in technical literature are often difficult to decipher due to their uninformed use of legal terms, we can assume that the said equivalence refers to a situation where code adequately represents the substance of the contract and where the conversion of contractual provisions into lines of computer instructions does not result in a simplification of the agreed obligations. A parallel debate concerns the selection of the programming approach that is most suitable for "smart contracts." The main candidates are imperative and declarative languages.[63] According to the prevailing imperative approach, the "smart contract" must state the computational operations to be performed to implement the agreement. The problem with this approach is that programmers might find it difficult to extract the order of instructions from contracts expressed in natural language. In contrast, the declarative approach states the agreed legal arrangements but disregards the computations needed to implement them.[64] Programmers need not write the sequence of steps specifying *how* something must be done, only *what* must be done. Purportedly, this allows for a more compact representation of the relevant obligations, one closer to natural language and aligned with many "traditional" contracts where parties agree on the desired result without specifying the manner of its achievement. It has also been observed that declarative languages ensure a closer correspondence between the legal agreement and its executable implementation[65] while imperative languages may introduce a gap between the legal prose and its expression in code. On a broader level, attempts to achieve equivalence between denotational and operational semantics expose a number of misconceptions concerning the manner contracts are drafted. Logically, natural language – including legal prose - is not like mathematical language and contractual relationships are not reducible to mathematical formulae. It is difficult to conceive of a programming language that resembles natural language while, at the same time, constituting a vocabulary and a collection of grammatical rules instructing computers. Moreover, in their quest for precision, many programmers seem to underappreciate that the complexity of contracts (and the resulting intricacy of legal prose) reflects the complexity of the underlying transaction. In most

[60] Khalil above, n 59, at 5.
[61] Khalil above, n 59, at 6.
[62] This is dictated by the need to create "clean code," one that is easy to debug and can be subjected to formal verification.
[63] Guido Governatori et al, 'On Legal Contracts, Imperative and Declarative Smart Contracts, and Blockchain Systems' (2018) 26 *Artif Intell L* 377, at 378; Florian Idelberger, et al, 'Evaluation of Logic-Based Smart Contracts for Blockchain Systems,' in: J Alferes et al eds, *Rule Technologies. Research, Tools, and Applications. RuleML 2016,* Springer, Cham, 2016.
[64] Governatori above, n 63, at 386, 389.
[65] Governatori above, n 63, at 392.

instances, the legal prose cannot be simplified because the transaction it expresses cannot be simplified. In a seminal paper on the computational representation of financial agreements, a very basic loan had to be further reduced in complexity so that its logical structure could be mapped onto the formal terms of a deterministic finite automaton.[66] In other words, in order to be expressed in code, the transaction had to be simplified to a degree that made it commercially unrealistic. The search for programming approaches that achieve equivalence between legal and technical language might be thus doomed from the outset. This leads to the next issue.

Words or Obligations?

We can also criticize the focus on translating legal prose or *whole contracts* into code. Parties do not perform clauses and courts do not enforce words. As performance and enforcement concern *obligations*, the focus should be placed on representing obligations in code (or "*encoding*" obligations) – not on translating clauses or words. Very few parts of a contract can be automated (i.e. are candidates for self-enforcement) and not all contractual clauses describe obligations. To elaborate: the main selling point of "smart contracts" is that they guarantee performance. Technically, the types of performance that can be guaranteed are limited. As blockchain-based "smart contracts" cannot control objects in the physical world, their current capabilities are confined to guaranteeing the transfer of crypto-assets under pre-defined conditions. They cannot guarantee the delivery of goods or the provision of services in the real world. At present, few contractual obligations can be automated or executed "by" a "smart contract."[67] It is hence pointless to translate into code those terms that describe obligations that cannot be automated. Similarly, there is little point in encoding terms that contain recitals, indemnities or clauses limiting the liability of the parties. "Smart contracts" need not represent entire agreements because they are always confined to obligations that can be executed by a machine. Consequently, the focus should be placed on ensuring that the relevant *obligations* are correctly encoded. This necessitates a shift to from creating programming languages that represent natural language to the creation of programming languages that enable the representation of specific transactions or obligations.[68] It also necessitates a recognition that not all parts of the agreement can and need to be expressed in code. The only reason to express an *entire* contract in a computer-readable form is to facilitate document management or for the purposes of computer-assisted "document assembly,"[69] areas that are unrelated to "smart contracts." It can be suspected that the ongoing focus on translating legal prose into code may derive from a fascination with natural language processing, a subfield of computer science, information engineering and artificial intelligence that examines the interactions between computers and natural languages. Certain research areas falling within natural language processing, such as machine translation or natural language generation, may have trickled into the "smart contract" narrative and created unnecessary confusion.

The Need for Co-operation

[66] Mark D Flood, Oliver R Goodenough, 'Contract as Automaton: the Computational Representation of Financial Agreements' (2015) Office of Financial Research Working Paper, 2015.
[67] J Cieplak above, n 6, at 420.
[68] An example is the Digital Asset Modelling Language, DAML, which is an expressive language for modelling financial agreements.
[69] See generally: Richard E Susskind, *The End of Lawyers?Rethinking the Nature of Legal Services*, Oxford University Press, Oxford, 2010.

Irrespective of whether we focus on translating legal prose into code or on expressing obligations in a computer-executable manner, it must be assumed that programmers cannot perform either operation by themselves. Programmers are unable to translate legal language into code because (just like computers) they cannot *interpret* legal language. Even a simple contract, one written in clear and seemingly unambiguous language, cannot serve as a "technical specification" for a "smart contract" because the individual obligations may be difficult to establish when, as is often the case, one obligation is described in multiple clauses that co-define its scope or when the meaning of individual words depends on the context of the transaction. The process of converting prose into code creates opportunities for errors that effectively annihilate any benefits of "smart contracts." It seems pointless to guarantee performance if such performance does not represent the original agreed obligation. It is also pointless to develop complex programming languages that resemble legal prose if programmers are likely to misunderstand the source document. When the "smart contract" constitutes a translation of a written agreement, the primary problem concerns the necessity of its interpretation – not the adequacy of the programming language. It is thus increasingly recognized that "smart contracts" require that lawyers and programmers work together. As programmers are incapable of "legal thinking" and lawyers are incapable of "computational thinking,"[70] it is necessary to create a common language enabling their co-operation. Lawyers would write contracts in such language, while programmers could use it as a specification guiding their coding. Such common language would approximate legal prose, be transparent and rooted in formal logic. It would also achieve a high degree of equivalence between the legal prose and the derivative, encoded version. While this approach seems *prima facie* commendable, it seems to underestimate the aforementioned difficulties of creating a programming language that straddles legal prose and computational expression as well as the significant educational investment for both lawyers and programmers. At the same time, while the assumption that lawyers would use such common programming language to draft agreements may be over-optimistic, the "common language"-approach implicitly recognizes the need to create contracts with subsequent encoding in mind. As explained below, this may affect the very substance of such contracts.

The Limits of Code

It has been suggested that the continued use of legal prose, which lacks precise semantics, is not sustainable as "it is often not clear what a contract is intended to say." [71] In this context, "smart contracts" are supposed to create legal certainty: "written in programming language, smart contracts *remove* the ambiguity inherent to natural language."[72] While the assumption that contractual ambiguity is always undesirable is incorrect, it must be conceded that for an obligation to be expressed in code, the obligation itself must be described in a manner that leaves no room for interpretation. If code, in order to be executed,[73] cannot be ambiguous, then the obligation it expresses cannot be ambiguous. Any ambiguity would result in errors in interpretation on the side of the programmer and, inevitably, in disputes regarding the correctness of the "smart contract." Consequently, it seems more accurate to state that "smart contracts" *preclude* the possibility of using ambiguous language or *force* an

[70] See generally: F Al Khalil et al., 'Trust in Smart Contracts is a Process, As Well' in: M Brenner et al, eds, *Financial Cryptography and Data Security*, Springer, New York, 2017, p 2.
[71] William M Farmer, Qian Hu, 'FCL: A Formal Language for Writing Contracts,' in: S H Rubin, T Bouabana-Tebibel, eds, *Quality Software Through Reuse and Integration, Advances in Intelligent Systems and Computing*, Springer, New York, 2018, p 2.
[72] Michelle Finck, *Blockchain Regulation and Governance in Europe*, Cambridge University Press, Cambridge, 2019, pp 26, 27.
[73] In computer science the term "interpreter" denotes a program that executes, i.e. *follows* the instructions written in a programming or scripting language, without its prior compilation into a machine language program.

unprecedented exactness of drafting. To claim otherwise would mean that every obligation expressed in code constitutes a modified (*simplified*?) derivative of the agreed original. Two sets of problems arise. One concerns the very character of legal obligations, the other concerns the inherent limitations of programming languages. Both illustrate the challenge of ensuring the equivalence of legal and technical language.

First, not all obligations can be translated or expressed in code. Many obligations cannot be described in a precise manner without resorting to open-ended terms,[74] such as "reasonable efforts" or "good faith." While the inclusion of such terms could simply be avoided, such avoidance would deprive the contractual relationship of any flexibility and limit the subject matter of "smart contracts" to those obligations that can be expressed in exact terms.[75] Alternatively, instead of using concepts like "best efforts" or "reasonable endeavours" we could list the specific actions or prohibitions that would describe the desired behaviour or result. Such operation would, however, result in more lines of code and thus increase the statistical possibility of coding errors. Moreover, as demonstrated by dozens of cases on the construction of contracts, which often concern seemingly precise contractual clauses, human language - including its more formalized subcategory of legal prose - is inherently ambiguous and there is nearly always room for disputes concerning the meaning of specific words or sentences. Leaving aside concepts like "good faith" or "reasonable," even words like "delivery" or "warehouse" may be difficult to encode without complex technical descriptions. Again, lengthy descriptions increase transaction costs as well as the potential for coding errors. In sum, the difficulty of drafting in a precise manner is largely underestimated.

Second, the available programming language will dictate what obligations can be agreed on. In other words, irrespective of the fact that the performance many obligations cannot be automated (hence their expression in code is pointless) and that many concepts in contract law cannot be encoded (hence their use is precluded), the programming language will determine what types of transactions are candidates for "smart contracts." The practical problems depend on whether the parties decide to express their obligations in code from the outset or whether they translate an existing obligation into code. In the first instance, the substance of the agreement will be tailored to the method of its expression. The parties will agree on obligations that can be encoded with one of the available programming languages. The code will dictate what can be agreed on. In the second instance, the parties may agree on an obligation without realizing that it cannot be adequately expressed in any of the existing programming languages. In such situation, they will either abandon the idea of encoding this particular obligation or re-define it so that it can be encoded. Again, the final substance of the obligation will be dictated by the "expressiveness" of the available programming language. In the context of blockchain-based "smart contracts" the range of such programming languages may be further limited by the very nature of their execution environment. Unsurprisingly, one author stated that "smart contracts are by their nature limited to those contractual terms that can be specified in computer-readable code, and further limited by any constraints imposed by the blockchain system in which the contract operates. As a result, they are unable to capture the real-world

[74] M P Gergen, 'The Use of Open Terms in Contract' (1992) 92 *Col L Rev* 997.
[75] Karen E C Levy, 'Book-Smart, Not Street-Smart: Blockchain-Based Smart Contracts and The Social Workings of Law' (2017) 3 *Engaging Science, Tech & Soc* 10 at 11.

complexity of all but the simplest transactions."[76] In sum, the type of obligations that the parties can agree on will change once we move from natural language to code.

Hybrid Approaches and "Code-as-Contract'

Additional difficulties concern situations where (a) a contract is expressed by a combination of code and natural language and (b) where the entire contract is expressed in code, i.e. where "code is the contract," (assuming, for the sake of argument, that this is technically possible). Regarding (a), we can imagine a situation where the parties agree that whenever A delivers the goods to a B, B will transfer a certain amount of crypto-currencies to A. To this end, the payment clause is translated into code, i.e. it becomes a "smart clause." Some additional differentiations are necessary. If the "smart clause" constitutes a translation of a clause written in natural langue ("natural clause"), we must inquire whether both the natural clause and the smart clause remain in the contract or whether after translation the natural clause is deleted. If both remain, we must establish which clause prevails in case of conflict. A conflict would exist if, for example, the smart clause did not correctly represent *or* execute the obligation described in the natural clause. An incorrect representation would be the result of errors in translation or the limitations of a programming language, while an incorrect execution would be the result of a coding error. In both instances, the smart clause diverges from its natural language equivalent creating a situation where the contract contains two clauses, one "smart" and one "natural," that *theoretically* describe the same obligation. Consequently, contracts containing natural language and code require additional provisions stating the hierarchy between the code and the natural language as well as provisions allocating the risks resulting from the operation of the code. Where the natural clause is deleted after translation, i.e. where the payment obligation is only expressed in code, potential problems may concern the question what was originally agreed on. The party seeking to establish that the smart clause does not correctly represent *or* execute the original payment obligation would encounter evidentiary difficulties as to what was agreed on. It is debateable, however, whether this problem can be described as one of *interpreting* the smart clause, as is often implied, because the very dispute concerns the fact that the smart clause is incorrect. While both of the above situations assume that code and natural language co-exist in one document, a different approach assumes that the "smart contract" is accompanied by an additional natural language contract that governs its operation, a so-called "legal wrapper."[77] Such "legal wrapper" would contain terms that are unsuitable for automation and/or that cannot be converted into code. It would also allocate any risks resulting from the execution of the "smart contract." The "smart contract" would incorporate the "legal wrapper" by reference or the other way around. The practical difference between combing code and natural language in one document or having two separate documents that cross-refer to each other is negligible, if any. Both approaches implicitly recognize that "smart contracts" cannot function without an accompanying legal agreement and that it is impossible to express an entire contract by means of code.

In (b), the situation falls within the scope of theories popularly labelled "code is law" or "code is the contract." The theories propose that if an obligation or a contract are expressed in code, the parties are bound by the code and must accept all consequences of its operation.[78] More importantly, if "code is the contract" then any bugs in the code form part of the contract – even if they were deliberately created to enable one of the parties or a third

[76] J Bacon, above, n 11 at 99.
[77] Cheng Lim, T J Saw, 'Smart Contracts: Bridging the Gap Between Expectation and Reality,' Oxford Business Law Blog, 2016.
[78] Adam J Kolber, 'Not-So-Smart Blochchain Contracts and Artificial Responsibility' (2018) *Stan Tech L Rev* 198.

party to obtain undue benefits.[79] Allegedly, if the code enables a certain action then it cannot be said such action constitutes a breach or is otherwise unlawful. As the code is available for inspection, it is theorized that the parties can inform themselves as to "what it says" and "what it will do." It is also suggested that the code overrides any other descriptions, terms or representation made in relation to it, that the transaction is "governed by the code" or that the "code is binding." The precise reasoning underlying these theories is often unclear as the arguments are inconsistent and, again, plagued by an uninformed use of legal terminology as well as a failure to distinguish between the substance of an agreement and its expression. The fact that, from a legal perspective, an agreement can be expressed in code does not mean that the code is binding or that the parties must accept whatever it produces. The difference with the aforementioned Chinese contract lies in the fact that while the latter constitutes an expression of agreement that neither party can understand, it does not *in itself* produce any direct effects as it does not instruct computer systems to perform any operations. In contrast, a contract expressed in code produces such effects. The question remains: are the parties bound to accept whatever output is produced by the operation of the "smart contract," assuming they had the opportunity to inspect its code? Even outside of a consumer context, it can be doubted whether the mere disclosure of the code would constitute sufficient notice of the terms of the transaction, assuming that the code itself is treated as synonymous with the terms. The parties could only be "bound by the code" if they had agreed to be so bound. The agreement to be bound by the code would not, however, be part of the code. In addition, "code is law" theories falsely assume that it is technically feasible to encode every aspect of a transaction and that it is legally permissible to exclude the operation the legal system, i.e. that contracts "governed by code" are not subject to any real-world jurisdiction. They also fail to appreciate that, from a commercial perspective, having a contract that cannot be stopped and that will be performed irrespective of any surrounding circumstances, is not appealing. The more so, that in all of the above instances, "code-savvy parties" could take advantage of the "code-naive"[80] by, for example, using "smart contracts" that that contain a vulnerability that will benefit them. In sum, while interesting from an academic perspective, "code is law" theories rest on so many false assumptions that their practical significance is limited.

Concluding Observations

The discussion of "smart contracts" would gain in clarity (or maybe become redundant altogether?) if the term *contract* was abandoned or if "smart contracts" were given a definition that would expressly acknowledge their limited legal relevance in the area of contract law. Unfortunately, the term is deeply ingrained in the blockchain narrative so that a reversal is unlikely. At present, the undisciplined and uninformed use of legal terminology has produced incohesive arguments and unjustified claims. Going forward, "smart contracts" must be regarded as technologies automating the performance of obligations deriving from traditional contracts or, more broadly, as technologies enabling the generation and transfer of crypto-assets.[81] "Smart contracts" can bring about the formation or performance of contracts – but they are not contracts. Questions regarding their enforceability or validity are inherently misconceived. Enforceability concerns obligations, not their manifestations or mechanisms facilitating their performance. Technical and legal writings seem oblivious to the difference between the substance

[79] This was famously the case with the Digital Autonomous Organization, or "DAO", where an unknown third party was able to exploit a coding error to extract funds from the common investment pool.
[80] B Marino, A Juels, 'Setting Standards for Altering and Undoing Smart Contracts,' in: J Alferes, et al, eds, *Rule Technologies. Research, Tools, and Applications. RuleML 2016,* Springer, Cham, 2016, at 157.
[81] J Bacon, above, n 11, at 46.

and the expression of an agreement and to the seemingly simple principle that contractual intention can be expressed in any manner. The question is not whether it is legally permissible to express contracts in code but whether a specific *obligation* is capable of being automated and/or whether it can be adequately expressed by a particular programming language. It is necessary to abandon grand theories of what "smart contracts" can do and recognize the inherent limitations of computer code. After examining the existing, practical implementations of "smart contracts," it is also necessary to acknowledge their limited, potential impact on contract law or the legal system in general.