Matricola n. 0000761458

ALMA MATER STUDIORUM
UNIVERSITA' DI BOLOGNA

DIPARTIMENTO DI SCIENZE GIURIDICHE

CORSO DI LAUREA MAGISTRALE IN GIURISPRUDENZA

Molecules of Smart Contracts: Contractual Basics in the *Stipula* Language.

Tesi di laurea in INFORMATICA GIURIDICA

Relatore

Prof. Giovanni Sartor

Correlatore

Prof. Cosimo Laneve

Presentata da

Alessandro Parenti

Sessione 05/10/2021-08/10/2021
Anno Accademico 2020/2021

Index

1. SMART CONTRACTS AND BLOCKCHAIN: BACKGROUND

2. SMART LEGAL CONTRACTS: LEGAL FRAMEWORK

Introduction

Since its birth, the interest around the blockchain technology have increasingly grown. Especially in the last year, the diffusion of many blockchain-based projects caused the exponential increase of the total crypto market capitalisation, thus shedding light on

the strengths of this relatively new technology also in favour of the broader general public. We can mention two sectors above all that gained the most success world-wide, attracting all kinds of investors: the NFTs market, where the new interest for digitalised ownership drew investments in the millions for many projects[1], and the Decentralized Finance (DeFi), in which, as of September 2021, are held more than 82 billion USD[2] throughout the hundreds of decentralised applications (Dapps) currently up and running. The expansion of this new market consequently brought in the picture institutional investors, attracted by the huge economic returns it produced, and by the possibility to employ this technology within their own businesses in order to be the first company to exploit its potentialities. Some of the most important are Visa[3], Intel Corp.[4], JP Morgan[5], many institutional investment funds[6] and also sport clubs[7].

Now, the element that gives blockchain technology the biggest share of its disruptive potential (especially for the 2 sectors mentioned above) is the possibility to implement the so-called *smart contracts*. The possibility of running computer software on a decentralised ledger allows for the employment of many applications that can implement efficiency and security in many sectors of society. As the name suggests, the legal field is, potentially, one of the most influenced by this technology, probably the most further reaching, since every sector is somehow concerned with law. In fact, although the father of the term 'smart contracts' (Nick Szabo) clearly envisioned the application of this technology specifically in the contractual domain, technically speaking, we can't define smart contracts as legal contracts. More precisely, they can assume a legally binding character when they are employed to represent and automate

[1] E. GKRITSI, *NFT Markets Post Record-Breaking Week,* www.coindesk.com, 2021;
[2] https://defipulse.com/
[3] O. AVAN-NOMAYO, *Corporate brands target NFTs and adoption continues to skyrocket*, www.cointelegraph.com , 2021.
[4] How Intel Is Shaping Blockchain Technology, www.intel.com.
[5] D. NELSON, *JPMorgan Launches In-House Bitcoin Fund for Private Bank Clients*, www.coinddesk.com, 2021.
[6] B. NELSON, *Fidelity Bitcoin Fund Attracts $102M in First 9 Months*, www.coindesk.com, 2021.
[7] *Barça and blockchain platform Chiliz join forces for new global alliance to increase interaction with fans*, www.fcbarcelona.com.

the promises of a legal agreement, and they satisfy the conditions laid down by the relevant national law for contracts to be considered as such.

Because of the rising crucial relevance of smart contracts, many academics and experts were pushed to study the subject in order to, on one hand, figure out how to adapt this new technology to the current contractual law and, on the other hand, to foster its diffusion by explaining its usage to professionals. The present work aims at giving an contributing to both these impulses.

One of the main problem encountered for the mass adoption of smart contracts is given by the fact that they, being a computer software, need an IT expert to be created. For the same reason, the knowability of contract's contents by the parties to an agreement is made more difficult, potentially affecting in a negative way the formation of contractual will. Another issue presented by smart contracts that aim at representing legal agreements is their incapability of transposing into code some kinds of elements, terms and concepts that are usually included in the latter.

In the present paper, I analyse the employment of a new domain-specific programming language, *Stipula*[8], for the creation of smart legal contracts on three different contract models. The analysis' goal is to see if and to what extent this new language is capable of capturing the legal concepts that usually shape contracts and, after that, understanding how to deal with the elements that cannot be automated by smart contracts and that are not susceptible of being translated into code. The abstractness and conciseness that *Stipula* features allow for an easier and faster understanding even by non-IT experts and, therefore, help overcoming the difficulties just mentioned above. The research is structured as follows.

The first chapter puts forward a general overview of the background history and technical functioning of smart contracts, explaining, first of all, where this term comes from and then attempting at giving a comprehensive definition. Moving forward, I take into account the technology behind smart contracts, namely, blockchain. I give an

[8] S. CRAFA, C. LANEVE, G. SARTOR, *Pacta Sunt Servanda: Smart Legal Contracts in Stipula,* 2021

overall picture of the history of Satoshi Nakamoto's invention and the concepts and ideas behind it. I briefly touch the technical functioning of blockchain only to the extent necessary to fix the fundamental concepts of this technology: distributed ledger, *peer-to-peer* technology, asymmetric cryptography, *hash* functions, *Proof of Work, Proof of Stake* and consensus mechanism. Knowing these notions helps understanding what the features of blockchain are and where they come from. The chapter then introduce Ethereum, the first blockchain supporting the implementation of smart contracts and explains what innovations it brought to the picture. This is functional to the full understanding of blockchain-based smart contracts' features that are specifically addressed, one by one. In conclusion, I focus on the concepts of *token* and *oracle,* two elements born with the advent of smart contracts-blockchains and that have primary importance in such context.

The second chapter addresses smart contracts from a purely legal standpoint. After drawing the difference between smart contracts and *smart legal contracts,* I try to determine when the latter actually arise, according to law provisions. In order to be legally binding before a state's court, in fact, a contract must respect specific requirements that depend on the different legal systems in which it is created. In this section, I try to adapt contract law requirements to the formation of smart legal contracts and demonstrate that they can potentially satisfy all the requirements. I put forward this analysis from the point of view of international law, EU law, and the Italian national law. The second part of the chapter, instead, focuses on the current legal framework addressing smart contracts and blockchain technology. Although their important impact on society is relatively recent, in fact, legislators throughout the world have started addressing blockchain technology and smart contracts, in an attempt to regulate the phenomenon. The approaches taken by the different institutions towards this task vary depending on several factors: the subject under which the blockchain provision was inserted (fiscal, finance, legal etc.), the degree of strictness employed, the decision of charging an independent authority to legislate in such matter and so on. In this section I describe the legislations adopted by the Italian legislator,

the EU countries that showed more progresses in the field, and the situations in the US' states, that were the first to address this topic.

The third chapter focuses on the *Stipula* language. The approach used for the design of this programming language was to transpose legal agreements into code beginning from the singular legal concepts that form a contract. In other words, breaking complex agreements in simpler units, just like molecules and atoms, find a way to transpose it into code one by one, and then assemble them together to create increasingly complex contracts. According to this approach, the analysis laid out firstly shows the common legal patterns we can find by breaking down contractual agreements into atomic units. After that we look at how these patterns were transposed into code by *Stipula*'s developers, basically outlining the features and primitives constituting the language. This part helps familiarizing with the code, giving the necessary knowledge to recognize its basic elements, and understanding their functioning. After this section, I put forward the implementation of this language using three contract examples: a free rent contract for a locker, a license agreement and a bet contract. The presentation of each example begins with a legal analysis of the contractual type concerned, taking into account where they are placed in the contract law landscape, the legal concepts founding them and the relevant law provisions that apply to it. Then I schematize the contract's lifecycle, to distinguish the various stages it follows. From the latter analysis, I extrapolate the various simple legal patterns shaping it and show which *Stipula* feature can be employed to translate it into code. So, I move forward showing the actual *Stipula* code implementing the contract example, together with a step-by-step description of what each line of code mean. The analysis then concludes with a comparison between the original schematization of the contract's lifecycle: here I check what parts and elements of the contract were effectively implemented through the *Stipula* code and which ones are not implementable and have to be dealt with off-chain. This part is crucial to the study of this language (and smart contracts in general) because it clearly shows to what extent smart contracts can represent legal agreements' contents and which kind of elements are unimplementable.

After having seen how smart legal contracts work in practice and what their limits are, in the fourth chapter I look over the weaknesses of this technology in terms of their capability to effectively represent legal agreements. As it will be outlined, smart legal contracts differ intrinsically from their traditional paper-based counterpart. One can be described as a humans' free creation completely stemming from parties will, that only encounters the limits set by law. The other is, instead, a computer program that, in addition to the rules set by law, also encounters the limits entailed by informatics. This situation necessarily restricts the freedom of expression that parties can carry out, at least if compared to the one allowed by traditional contracts. At the same time though, smart legal contracts present huge benefits which make them worth using over paper contracts, therefore, the point is to find a good trade-off between negative and positive sides. Moreover, methods to overcome or to approximate to nothing these limitations currently exist and are described in this chapter. Special attention is given to *oracles,* as an indispensable element for the creation of an effective smart contract system, and to the dilemma ("*The Oracle Problem*") they bring to the picture, explaining its founding reasons. The solutions existing to face smart contracts' limitations, in fact, may sacrifice, to a certain extent, the benefits given by smart contracts in the first place. The solutions, their applications and the possible consequences they imply are all addressed in this chapter.

In conclusion, I give a summary of the outcomes of my study, focusing on where the *Stipula* language stands in the general discussion on smart contracts and how it contributes to the development and innovation of this technology.

CHAPTER 1

SMART CONTRACTS AND BLOCKCHAIN: BACKGROUND

1.1 SMART CONTRACTS: POSSIBLE DEFINITIONS

The term 'smart contract' was originally coined by computer scientist and cryptographer Nick Szabo in 1994[9]. Szabo is among the key personalities who contributed to the birth of cryptocurrencies and the world of crypto as a whole[10].

Between 1994 and 1996, he introduced the concept of a software protocol that could automatically execute the terms of a contract without the need for intermediaries. He explains his idea by bringing forward the example of a vending machine: the vending machine is programmed to automatically perform (dispense the product) when certain conditions are met (a coin is inserted). This avoids the need for intermediaries by making the breach of contract expensive or non-convenient (the cost of breaching the machine would likely be higher than the amount in the till). Such a mechanism enables contracting between unknown parties who don't need to trust each other but simply trust the machine automated process[11].

Szabo originally defined smart contracts as a "set of promises, specified in digital form, including protocols within which the parties perform on these promises"[12]. The contractual nature was at the centre of his conception and this line of thought was followed by many authors and academics after him[13]. However, technically speaking, a smart contract is nothing more than a self-executing piece of code operated by a computer[14], not much different form a software for smartphones or computers. Only when the code is programmed to execute a set of promises as intended by two parties

[9] N. SZABO, *Smart Contracts, www.fon.hum.uva.nl,* 1994.
[10] Szabo developed BitGold, ancestor of bitcoin.
[11] N. SZABO, *Smart Contracts: Building Blocks for Digital Markets*, *www.fon.hum.uva.nl,* 1996.
[12] Ibidem.
[13] M. RASKIN, *The Law and Legality of Smart Contracts,* in *Georgetown Law Technology Review,* 2017, p. 305; E. MIK, *Smart Contracts: Terminology, Technical Limitations and Real-World Complexity,* in *Law, Innovation and Technology,* 2017, p.
[14] M. DUROVIC, A. JANSENN, *The formation of Blockchain-based smart contracts in the light of contract law,* in *European Review of Private law,* 2018, p. 753.

to an agreement, it may have legal implications where it satisfies the necessary conditions required by the legal system to be considered binding. According to an *if.., then..* logic a smart contract can execute the understated obligations where a set of predefined conditions are met. E.g., *if* the flight gets cancelled, *then* refund the customer.

Even though we do not have a universally accepted definition of smart contract, its main features and functionalities are clear: it is (1) a piece of code that (2) potentially enables contracting with anyone, (3) it performs automatically when specific conditions are met, and (4) presents a security mechanism which make it unattractive to circumvent, due to the potential cost of breaches[15].

Smart contracts remained a theoretical idea for several years after its conception, due to the lack of a technology capable of fully realizing its features. Even though financial institutions were already employing computer code to automate transactions (such as option contracts or bookkeeping) long before Szabo[16], no technology existed that could guarantee the performance as a smart contract should do, completely eliminating the need to rely on an intermediary. For example, software used to automate the execution of a sale contract on e-commerce platforms (e.g., Amazon) are usually managed by the strongest party to the contract, generating a lack of transparency and a potential manipulability[17].

The real breakthrough of Szabo's idea came with the rise of blockchain technology. By running on a blockchain, smart contracts enjoy all the benefits of a distributed ledger and enable the automatic performance of contractual obligations with no possibility of human interferences.

1.2 BLOCKCHAIN: WHY DOES IT HELP?

[15] P. TRILLMICH, M. GOETZ, C. EWING, *Blockchain and Smart Contracts,* in M ARTZT, T. RICHTER (edited by), *Handbook of Blockchain Law,* Croydon, Kluwer Law International, 2020, p. 164.

[16] M. DUROVIC, A. JANSENN, *The formation of Blockchain-based smart contracts in the light of contract law,* in *European Review of Private law,* 2018, p. 753.

[17] Ibidem.

The rise of blockchain technology is linked to the birth of the world's first cryptocurrency, Bitcoin. Its anonymous inventor, Satoshi Nakamoto on Halloween night 2008, posted on the Metzdowd cryptography mailing list, a short paper titled "*Bitcoin: A Peer-To-Peer Electronic Cash System*"[18]. Satoshi outlines an electronic payment system aimed at solving the inherent weaknesses and inefficiencies of online transaction systems by replacing the traditional trust-based model with a structure based on cryptographic proof and decentralised consensus[19]. In traditional payment systems, in fact, transactions are recorded in centralised ledgers usually held by a central authority acting as fiduciary for the parties of the transaction who, most of the times, don't know each other. Such mediation increases transaction costs and, often, cut off the possibility of small transaction, thus increasing inefficiencies[20].

By contrast, Bitcoin's transactions are not recorded in a file held by a central institution; transactions are recorded on a distributed ledger, namely, a blockchain. A distributed ledger technology (DLTs) can be described as a global database shared by all the participants to a peer-to-peer network[21]. A blockchain is a particular type of DLT, which is shaped as a set of data blocks that are chained together in an *append-only*[22] structure.

When new transactions are issued, they are broadcasted to the entire network and the validator nodes provide to verify their 'legitimacy' (whether they are malicious, double-spending attempt etc). Once validated, transactions are grouped together into 'blocks of data' by *miners*[23] who concur with each other to add a new block to the

[18] S. NAKAMOTO, *Bitcoin: A Peer-To-Peer Electronic Cash System,* www.bitcoin.org/bitcoin.pdf, 2008.
[19] Ibidem
[20] ibidem
[21] *Peer-to-peer Networks*, www.academy.binance.com. "In informatics peer-to-peer networks consists of a group of devices that share and store files collectively. Each participant (node) has the same power and performs the same activities. Usually, it doesn't have an administrator or a central server since every node holds a copy of the file and acts both as a client and as a server".
[22] A blockchain structure only allows for the addition of blocks, elimination or modification of blocks is precluded.
[23] Miners are the nodes of the network who participate in the consensus mechanism in order to win the reward.

chain through a *consensus mechanism*[24]. The new block added to the chain is timestamped[25] and inextricably linked to the previous block through the employment of hash function[26]. The main benefits featured by blockchain technology users can be summarized as follows:

- **Public, transparent:** the ledger is accessible by anyone provided with an internet connection with no credentials requirement. Access to it cannot be restricted because the database resides on the network, on each single node, and not within a single institution[27]**;**
- **Privacy:** even though all the information flowing into the ledger is public, identities of users are hidden behind a pseudonymous (the address), thus granting their privacy[28] . "Bitcoin is like making everybody's bank statements public online, but with the identity blacked out"[29];

[24] Since a high number of transactions are issued at the same time, it is crucial for the functioning of a ledger to reach a consensus on the order of transactions. This consensus can be achieved through different systems. The most prevalent used are *Proof of Work* (PoW) and *Proof of Stake* (PoS). With *PoW* each miner creates a block of validated transactions and competes against the other to solve a cryptographic puzzle. The quickest to solve it gets to add its own block to the chain. In *PoS,* instead, each miner must stake its coins for the consensus process. One of all coins staked in the network is randomly chosen by the system and the owner of that coin is the winning miner (the more coins a miner stakes, higher is the chance to be selected). In both systems, the winner is awarded with a reward in coins which serves as an incentive to contribute to the sustainability of the network.

[25] A block's *timestamp* is analogous to a physical timestamp. It proves that certain information—in this case, block's data—existed at a particular time.

[26] A hash function is an algorithm that takes a piece of data of arbitrary length as an input, and produces a string of data of fixed length, named the hash value. Important features of this function are that even the smallest change in the input, results in a completely different output. Moreover, blockchains cryptographic hash functions are one-way functions, meaning that, from a given output, it is computationally infeasible to find the input data.
In a blockchain, the hash of the previous block is included in the next block. If a block were tampered and modified, its hash wouldn't match the hash included in the next block anymore and all the succeeding blocks would be invalidated. Such feature contributes to the security and immutability of the blockchain. See P. FRANCO, *Understanding Bitcoin,* Cornwall, John Wiley & Sons Inc, 2015.

[27] D. TAPSCOTT, A. TAPSCOTT, *Blockchain Revolution,* New York, Penguin Random House, 2016, p.24.

[28] P. FRANCO, *Understanding Bitcoin,* Cornwall, John Wiley & Sons Inc, 2015, p. 9.

[29] A. BACK, *Fungibility*, *Privacy and Identity*, www.youtube.com/watch?v=3dAdI3Gzodo, 2014.

- **Security:** all data on a blockchain is encrypted through asymmetric cryptography[30]. Only *wallet*[31] holders are provided with the cryptographic keys needed to accede to the network and send and receive transactions.
- **Immutable:** When a new block is added to the chain it cannot be tampered with or modified. Such result is achieved by including in the new block the hash of the previous one. However, it shall be specified that blockchains are not *tamper-proof,* but rather *tamper-resistant*: the possibility of corruption of the structure, in fact, exists[32].
- **Trustless?** The very aim of Satoshi's project was to create a "system for electronic transactions without relying on trust"[33]. The employment of a distributed and public ledger, a consensus mechanism and cryptography grants users that a transaction on the market is legitimate, accurate and not duplicated, without relying on a trusted intermediary[34].

  However, as it was correctly noted, the need for trust is not really eliminated in a blockchain but rather shifted form trusted intermediaries to the software itself, the code. The Blockchain protocol, even though open source and potentially reviewable by anyone, is *de facto* shaped and implemented by a

[30] Asymmetric Cryptography (or Public Key Cryptography) is an encryption technology based on a pair of cryptographic keys. A "public key", which is shared in the network (it's usually represented by addresses) and a "private key", which is kept secret by each user. The public key is used by a sender to encrypt the information, while the private key is used by a recipient to decrypt it. By doing so, anyone in the network can encrypt a message (transaction) with any public key and be sure that only the private key holder will be able to decrypt and receive that message.

[31] A blockchain wallet is a tool that allows users to interact with the blockchain network and manage their crypto assets. They don't really store funds, as a physical wallet would do, since they are stored in the distributed ledger. Wallets serve as mean to access to them. Wallets generate the pairs of public-private key and as many addresses.

[32] A. ANDERBERG et al., *Blockchain now and tomorrow,* EU commission publications.jrc.ec.europa.eu, 2019, p.16, 17. Modifying the information, in fact, would be possible where an intruder possessed more than 50% of the overall computational power of the blockchain network. Such operations of reverse engineering already occurred within small cryptocurrencies blockchains like Bitcoin Gold (R. SHARMA, *Bitcoin Gold Hack Shows 51% Attack Is Real, www.investopedia.com, 2018*), or Ethereum Classic (A. SHOME, *Ethereum Classic Suffers Another 51% Attack,* www.financemagnates.com).

[33] S. NAKAMOTO, *Bitcoin: A Peer-To-Peer Electronic Cash System,* www.bitcoin.org/bitcoin.pdf, 2008.

[34] K. WERBACH, N. CORNELL, *Contracts ex Machina, Duke law journal,* 2017

small group of people consisting of core developers and miners[35]. The formers are the only developers who are allowed to make actual changes to the software code (at least in the Bitcoin blockchain). The decision-making power of these two players is exemplified by the hard forks occurred both in the Bitcoin and Ethereum networks where core developers were able to correspond with and persuade particular miners to alter the software they were running[36].

As of June 2021, more than 10 500 cryptocurrencies exist, and the total capitalisation of the crypto market exceeds 1.5 trillion dollars[37]. Since 2009, blockchain technology evolved exponentially throughout the years and found countless fields of application based on the implementation of smart contracts on the ledger.

1.3 ETHEREUM: FIRST TOURING-COMPLETE PLATFORM

An important boost to the evolution of blockchain technology was given by Ethereum, the second most popular and capitalised blockchain project after Bitcoin. It was conceived in 2013 by a nineteen years old programmer Vitalik Buterin when he published the Ethereum white paper "*A Next-Generation Smart Contract and Decentralized Application Platform"*. With the collaboration of Joseph Lubin, the project was crowdfunded in 2014 and finally went live on July 2015.

Compared to Bitcoin, Ethereum uses the same basic approach of distributed ledger, a consensus mechanism for mining new blocks and a native cryptocurrency, Ether (ETH). The real difference of the two ecosystems lays in the purpose of the project: while Bitcoin's goal is to create a decentralised currency, Ethereum's "intent is to create a protocol for building decentralised applications"[38]. The innovative feature of Ethereum is the implementation of a "Build-in Turing-complete programming

[35] A. WALCH, *In Code(rs) we Trust: Software Developers as Fiduciaries in Public Blockchains,* in G. DIMITROPULOS et al. (edited by), *The Blockchain Revolution: Legal and Policy Challenges,* Oxford University Press, 2018

[36] 2013 Bitcoin Fork, www.bitcoin.org. 2016 Ethereum Fork, www.coindesk.com.

[37] www.coinmarketcap.com

[38] V. BUTERIN, Ethereum white paper*, A Next-Generation Smart Contract and Decentralized Application Platform,* www.ethereum.org, 2013.

language[39]" which allows programmers to write and run on the ledger any kind of software. This technology is represented by the Ethereum Virtual Machine (EVM), a runtime system that enables the management and execution of smart contracts separately from the underlying network, so as to secure it from external interferences[40]. This machine is fuelled by the Ether cryptocurrency, which is designed to be used for purchasing computational power to run smart contracts[41], rather than as an alternative currency[42].

Numerous applications (in the form of smart contracts) can be implemented on top of Ethereum's blockchain. The most widespread today are financial applications that, for example, enable the creation of financial derivatives, hedge contracts, loan contracts, stable-value currencies, all in a decentralised way (Decentralised Finance: DeFi)

1.4 PUBLIC, PRIVATE AND CONSORTIUM BLOCKCHAINS

Blockchains can be distinguished in different categories. Bitcoin and Ethereum, for example, are *public* and *permissionless*. In this kind of blockchains, ledger's contents are visible to everybody and anyone with an internet connection can interact within the network, sending and receiving transactions. At the same time, permissionless means that every participant, keeping anonymity (or pseudonymity), can become a

[39] *Turing Complete,* www.academy.binance.com. A device or programming language is considered to be Turing Complete when it can run and solve any computable algorithm, no matter how complex, long or deep (it can replicate a Turing machine). Ethereum Turing-completeness means that its scripting language can be used by a programmer to construct any smart contract or transaction type that can be mathematically defined.

[40] M. GIULIANO, *La Blockchain E Gli smart Contracts nell'innovazione Del Diritto Nel Terzo Millennio,* in *Il diritto dell'informazione e dell'informatica,* 2018, p.1005.

[41] Transactions fees required to operate on the Ethereum network are referred to as "gas fees". *Gas* is a unit used to measure the computational cost of running a particular transaction. Gas fees' purpose is to avoid possible computational power wastage due to accidental or malicious infinite loops in running the code. Different operations (sending ETH to an address, running a smart contract) require different amounts of gas. The gas cost is calculated by multiplying the amount of gas by the gas price, which is measured in gwei, a fraction of ETH (1 gwei = $1x10^{-9}$ ETH). Using a separate unit to measure the computational cost of running the Ethereum Virtual Machine (EVM) has the purpose of unlinking the transaction cost from the ETH price. However, gas price is still subjected to fluctuations deriving from the congestion of the network. *Gas and Fees,* www.etherum.org, 2021.

[42] K. WERBACH, N. CORNELL, *Contracts ex Machina, Duke law journal,* 2017, p.122.

node of the network and participate to the consensus mechanism to validate transactions.

Private blockchains, in contrast, restrict the possibility to view and interact within the ledger by allowing only identified and authorised users. The conditions required to join the network are usually laid down by an administrator or owner who is hierarchically superior to all other nodes. Such environments loses the decentralised nature of permissionless blockchains but keeps the advantages of the distributed character of the ledger[43].

Private blockchains are always also *permissioned*, in that only known and authorised participants can become nodes of the network and contribute to the consensus mechanism.

A third kind is the *consortium* blockchain. It is a permissioned network where, instead of a single central administrator as in private blockchains, a handful of equally powerful parties manage the nodes and the validation process (partially decentralized environment). The controlling partners can decide whether to keep the ledger public, visible only to authorised users, or completely private.

Permissioned blockchains are deemed to be the best solution for enterprise use-cases[44] since they enjoy numerous technical advantages. They can combine elements of both public and private DLTs. With identified participants, for instance, the trust-less character of the network becomes partially unnecessary, and it can rely on a less computationally intensive consensus mechanisms. Trust is attributed by the administrator authorising participants. This increases transactions speed and lowers the costs. Moreover, the private character of the ledger, provide for a higher degree of privacy and confidentiality of the content[45].

---

[43] They are distributed in that many nodes still maintain a copy of the chain on their machines.

[44] *Private, Public, and Consortium Blockchains*, www.academy.binance.com.

[45] E. MIK, *A Technology for Decentralized Marketplaces,* in L. DIMATTEO, C. PONCIBO', M. CANNARSA (edited by), *The Cambridge Handbook of Smart Contracts, Blockchain Technology and Digital Platforms,* Cambridge, Cambridge University Press, 2019, p.164.

1.5 BLOCKCHAIN-BASED SMART CONTRACTS: FEATURES.

The proliferation of DLTs shed new light on smart contracts and opened a myriad of new opportunities and applications for this technology. That is why, today, definitions of smart contract always entail the link with blockchain.

Contracts are executed in a distributed manner by all the nodes constituting the network, eliminating completely the need of intermediaries. Moreover, because no single party controls a blockchain, the execution of a smart contract, once triggered, cannot be stopped by anyone unless the parties incorporated in the code such possibility[46]. On a blockchain ledger, smart contracts are associated with an address, they display a balance and can send and receive transactions.

Due to the huge inherent potential application of this technology, in the last few years the market of smart contract-supporting platforms has expanded exponentially. Ethereum's adoptions and application has grown a lot in the past years and today the platform is ranked as the 37th most capitalised asset in the world with a total market capitalisation of 270 billion dollars[47]. According to stateofthedapps.com, over two thousand applications are built on Vitalik Buterin's platform[48]. The majority of dApps (decentralised applications), provide services in the financial sector (decentralised finance) such as lending platforms (Aave, Compound) or decentralised exchanges (Uniswap, Bancor). The inherent trust-less character, in fact, makes this environment particularly suited for this sector, in that everyone can manage funds, buy financial products, or get loans avoiding the need and costs of an intermediary.

1.5.1 Automation

This is the most evident benefit of smart contracts. A smart contract is specifically programmed to perform obligation at the fulfilment of predefined conditions according to the logic *if* X happens, *then* Y. In contrast to "offline" contracts where, despite the

[46] P. DE FILIPPI, A. WRIGHT, *Blockchain and the Law – The rule of Code,* Cambridge (MA), Oxford University Press, 2018, p. 75.
[47] www.8marketcap.com.
[48] www.stateofthedapps.com.

very best intentions of the parties, breaches or delays may occur[49], performances are unlinked from the parties' willingness to abide to the terms. The scope for opportunistic breach is diminished, if not completely removed[50]. Such environment facilitates (or makes possible) concluding contracts with unknown individuals since there is no need to trust the counterparty.

Because of its predefined character, a good draft for smart contract will depend on the complete and correct transposition of all the circumstances and conditions relative to the execution[51]. When such conditions depend on "offline" events that are not detectable by the software alone, a crucial role will be played by 'oracles', human or software entities that serve as a bridge between the "off-chain" and "on-chain" world, feeding the necessary input data into the blockchain.

This feature also makes this technology a perfect environment for the implementation of Internet of Things (IoT). The coordinated use of artificial intelligence and smart contracts opens the possibility to machine-to-machine transactions, enabling the automatic conclusion of agreements without human intervention. For example, a vending machine could automatically submit a purchase order when run out of some product, or a self-driving car could autonomously detect the need to refuel and pay for gas[52].

1.5.2 Efficiency, Speed, modularity

If compared to the paper-based counterpart, smart contracts are way more efficient for various reasons. The first is that they increase the speed at which contractual obligations can be executed. As already mentioned, there is no reliance on human will and obligations are executed in real-time as soon as the necessary conditions are

---

[49] P. TRILLMICH et al., *Blockchain and Smart Contracts*, in M. ARTZT, T. RICHTER (edited by), *Handbook of Blockchain Law,* Croydon, Kluwer Law International B.V., 2020, p.167.

[50] P. CATCHLOVE, *Smart Contracts: A New Era of Contract Use,* Available at www.papers.ssrn.com/abstract_id=3090226, 2017, p. 9.

[51] M. GIULIANO, *La Blockchain E Gli smart Contracts nell'innovazione Del Diritto Nel Terzo Millennio*, in *Il diritto dell'informazione e dell'informatica,* 2018, p. 1026.

[52] P. DE FILIPPI, A. WRIGHT, *Blockchain and the Law – The rule of Code,* Cambridge (MA), Oxford University Press, 2018, p.83.

satisfied. When employed in the financial sector, blockchain technology can drastically shorten the time needed for the transaction of many financial products, shortening the settlement time from weeks or days to minutes[53].

Furthermore, as any other code, smart contracts are inherently modular, in that they can be broken down into pieces and used to draft new contracts. Potentially, libraries of smart contract code can be created, allowing anyone to construe his own contract by simply assembling different chunks of code from a library[54]. Such feature speeds up contract drafting on one hand, and on the other hand it could facilitate contract drafting even for non-lawyer parties.

However, it must be pointed out that such arguments can stand to the extent that concerns relatively simple or standardised contracts. As the complexity of a contract increases, the incapability of computer code to entail vague and flexible terms raises the necessity to predict and encode (or at least, try to) all the possible external events that could affect the life of a contract (such as force majeure or hardship events), increasing both the cost and time of drafting[55].

1.5.3 Cost reduction

By spending more time and resources negotiating the terms of the agreement in advance, though, the costs for contract's monitoring, enforcement and settlement drop significantly[56]. The self-enforcing nature of smart contract removes the scope of non-performance for the most part, thus avoiding dispute resolution costs.

---

[53] INSTITUTE OF INTERNATIONAL FINANCE, *Getting Smart Contracts on the Blockchain,* www.iif.com, 2016, p.3.

[54] P. DE FILIPPI, A. WRIGHT, *Blockchain and the Law – The rule of Code,* Cambridge (MA), Oxford University Press, 2018, p.82.

[55] J. SKARLOFF, *Smart Contracts and The Cost of Inflexibility,* in *University of Pennsylvania Law Review,* 2017, p. 280

[56] D. TAPPSCOT, A. TAPPSCOT*, Blockchain Revolution*, New York, Penguin Random House LLC, 2016, p. 118

For the same reason, there is less of a need for parties to repeatedly check and monitor the correct execution of obligations. The smart contract only runs what has been defined in the code without the risk related to opportunistic behaviour[57].

1.5.4 Security and Reliability

Traditional contracts may be lost or, if stored electronically, deleted, or corrupted. By contrast, blockchain inherent features make a smart contract permanently tamper resistant and its contents are shielded by the influences of third parties. Parties gain assurance that the underlying smart contract code has not changed and most likely will not be changed in the future[58].

Terms of contract cannot be overridden by individual malicious attacks. This feature has important implications on reliability. If a smart contract reports a transaction, it means that millions of nodes verified, for example, that Alice paid Bob X on a specific day at a specific time. Therefore, one can assume, with a high degree of certainty, that such event actually occurred[59].

1.6 ORACLES

As mentioned above, a smart contract's execution takes place once the encoded conditions are met. As long as these concern a time laps term or actions occurring on-chian, no issue arises. When the relevant conditions refer to events occurring in the real world or, more generally, to data stored off-chain, inputs from outside the blockchain will be required. For example, a smart contract expressing a flight insurance needs to get information about the flight's time of arrival; financial contracts need data from the stock market. Here is where the so-called "oracle problem"[60] comes

---

[57] P. DE FILIPPI, A. WRIGHT, *Blockchain and the Law – The rule of Code,* Cambridge (MA), Oxford University Press, 2018, p. 80

[58] Ibidem, p.81.

[59] M. RASKIN, *The Law and Legality Of Smart Contracts*, in *Georgetown Law Technology Review*, 2017, p. 319.

[60] DELPHI, *The Oracle Problem,* www.medium.com, 2017; A. EGBERTS, *The Oracle Problem an Analysis Of How Blockchain Oracles Undermine The Advantages Of Decentralized Ledger Systems*, available at www.ssrn.com/abstract=3382343, 2019, p.22.

to the picture. Blockchains cannot pull data in or push data out to external systems: as such, they are isolated networks very similar to computers without an internet connection.

Oracles are individuals or programs that collect information from the outside world and feed them to the blockchain so as to enable blockchain-based smart contract to react to real-world events[61]. We can distinguish different kinds of oracles depending on the complexity of services that they can provide. *Automated oracles* are software or devices that automatically detect the information needed from the outside world or from an internet website (e.g., sensors that measure the outside temperature or a software that checks the flights' time of arrival in the airport website). *Expert oracles*, instead, can deliver even more complex services[62]. The off-chain connection created by oracles, in fact, can be exploited to defer to the outside world all those tasks that the inner features of the blockchain preclude or make harder or less efficient to achieve. For example, oracles can help preserving privacy. You might not want the details of your financial smart contract being on a public blockchain visible to everyone. Moving some parts of the agreement on an off-chain oracle allows to keep confidentiality[63]. Another use case is a smart contract that requires an elaboration of data that would be too hard or too slow to be operated on the blockchain virtual machine (e.g., a smart contract card game that requires some form of randomness, impossible to achieve efficiently on a blockchain)[64]. Oracles can then mitigate the rigidity imposed by smart contracts. An expert oracle could be provided with the authority to assess the promises that cannot be easily encoded into a smart contract, due to their vagueness or open-

[61] P. DE FILIPPI, A. WRIGHT, *Blockchain and the Law – The rule of Code,* Cambridge (MA), Oxford University Press, 2018, p.75.
[62] E. TJONG TJIN TAI, *Challenges of Smart Contracts: Implementing Excuses,* in L. DIMATTEO, C. PONCIBO', M. CANNARSA (edited by), *The Cambridge Handbook of Smart Contracts, Blockchain Technology and Digital Platforms,* Cambridge, Cambridge University Press, 2019, p.83-84.
[63] V. BUTERIN, *Ethereum and Oracles,* blog.ethereum.org, 2014.
[64] J. VAN DER LAAN*, Understanding Blockchain,* in M ARTZT, T. RICHTER (edited by), *Handbook of Blockchain Law,* Croydon, Kluwer Law International, 2020, p.46.

ended character (*force majeure, due diligence, best effort* clauses)[65] and, at the same time, act as a legitimate dispute resolution mechanism[66].

The reliance on an external entity, however, has a downside in that it derogates to the trust-less nature of blockchain, which is probably the main reason for using it in the first place[67]. Consequently, it will be crucial for the parties to a contract to find both a trustworthy oracle, and a reliable data source since the oracle itself does not create or compute the required information but retrieves it from external sources. Moreover, in order to lessen the risks deriving from centralisation (e.g., oracles or data sources corruption) it could be important to create a network of multiple oracles so that information can be collected from different data sources and, only after N-of-M nodes verified the real-world event, the contract performance would be triggered.[68]

In the past years, several projects were carried out attempting to solve the 'oracle problem' in such a way as to minimize the need for trust. Some examples are ChainLink[69] and Provable[70]. The latter is a project founded by Thomas Bertani in 2015 (former Oraclize) and today is the world's most widely adopted data-transport-layer connecting blockchain applications to the external world. It relies on *authenticity proofs*, certificates attached to data feeds demonstrating that the data fetched from the original data-source, and that is genuine and untampered[71].

ChainLink is a project launched by Sergey Nazarov in 2017 which relies upon a decentralised network of oracles. Decentralisation avoids the risk that a malicious node tampering with data feeds could compromise the execution of smart contracts application. Moreover, it employs the tools of *minority reports* and *failover clients* to

---

[65] P. DE FILIPPI, A. WRIGHT, *Blockchain and the Law – The rule of Code,* Cambridge (MA), Oxford University Press, 2018, p.75.

[66] G. O. B. JACCARD, *Smart Contracts and The Role of Law,* available at SSRN: www.ssrn.com/abstract=3099885, 2017, p. 24. On the same point M. DUROVIC, F. LECH, *The Enforceability of Smart Contracts,* in *The Italian Law Journal,* 2019, p. 503.

[67] *Background*, www.docs.provable.xyz.

[68] E. MIK, *Smart Contracts: Terminology, Technical Limitations and Real-World Complexity,* in *Law, Innovation & Technology,* 2017, p. 269-300

[69] www.blog.chain.link.

[70] www.provable.xyz.

[71] *Background*, www.docs.provable.xyz.

ensure the integrity of data and respond to potential malicious attacks minimizing the need for trust[72]. Chainlink also released its cryptocurrency token, LINK, as an ERC20 token on the Ethereum network, which is used to pay oracle nodes for data feeds.

1.7 TOKENS

Tokens are digital tools created through smart contracts that represents units of value on a blockchain. Tokens can stand for money, company shares, art pieces, rights, financial products, and much more. The disruptive potential of this feature lays in the fact that, once tokenized, any kind of value can be partitioned and traded like any other asset, allowing for the creation of a borderless market for digitalised value[73]. The term *smart property* refers to a good or right, digitally represented, that can be enjoyed or exercised through the exclusive disposability of a token[74].

A univocal definition of token has proven difficult to find, since both the value it can represent, and its very function are widely diversified. However, we can identify some common characteristics: all tokens are *valuable*, in that we are usually able to value them; *representative*, because they stand for their holder claims for an asset, good or right; *digital*; *discrete* in that we can always distinguish one from another; and at last, all tokens are *authentic*, meaning that we can always verify their authenticity through the blockchain[75].

A generic distinction can be made between utility tokens and security tokens. This categorisation was firstly stated by the US Securities and Exchange Commission (SEC) as a consequence to The DAO Hack in 2016[76], with the purpose of regulating the tokens market. According to SEC, a token purchase is an investment contract, and, therefore, a security "when there is an investment of money in a common enterprise

---

[72] Chainlink Whitepaper, www.chain.link, 2021.

[73] P. FRENI, E. FERRO, R. MONCADA, *Tokenization and Blockchain Tokens Classification: a morphological framework,* IEEE Symposium on Computers and Communications (ISCC), 2020, p. 2.

[74] D. FAUCEGLIA, *Il Problema dell'Integrazione dello Smart Contract,* in *I Contratti,* 2020, p. 600.

[75] D. TAPSCOTT, *Token Taxonomy,* The blockchain research institute, www.blockchainresearchinstitute.org, 2020, p. 10.

[76] The DAO was a decentralised autonomous organization (a decentralised organisation built through the coordination of several smart contracts) created within the Ethereum network in 2016, acting as venture capital fund.

with a reasonable expectation of profits to be derived from the efforts of others" (Howey Test)[77]. In other words, a security token can be a financial product, company share or other investment that implies an expectation of profit related to the performance of the entrepreneurial initiative that was invested in. By contrast, a utility token is issued by a company to raise funds for a project or to raise the public interest in it, that entitles the holder to enjoy a right or a service provided by the company. Unlike security tokens, its value is not related to the company's performance, but rather to the simple supply-demand ratio of the token itself[78].

Tokens can be fungible or non-fungible. Fungible tokens are used to represent goods that are interchangeable either because they are equivalent in their value (e.g., money, commodities), or because they represent a fraction of a whole good (e.g., company shares). Non-fungible tokens (NFTs), instead, represent goods whose value depends on the particular and unique qualities each unit carries, thus making them non-interchangeable. The Ethereum network introduced some scripting standards for the creation of tokens in order to allow the interoperability between different smart contracts. These dictate a set of basic guidelines and functions that any new token running on the Ethereum network must follow[79]. The most important are the ERC-20 and ERC-721 standards.

The ERC-20 standard (Ethereum Request for Comments - 20) was developed for fungible tokens. These tokens have a property that makes each Token be exactly the same (in type and value) as another Token. They can be used to represent goods like company shares or subcurrencies running on top of the Ethereum blockchain. The ERC-721, instead, introduces a standard for non-fungible items (NFTs: non-fungible tokens). They can represent a wide variety of goods such as artworks, videos, music,

---

[77] US Supreme Court, SEC v. Howey Co., 328 U.S. 293, 1946; US Security and Exchange Commission, *The Dao Report,* Release No. 81207/2017, p. 11; US SEC's Strategic Hub for Innovation and Financial Technology, *Framework for "Investment Contract" Analysis of Digital Assets,* 2019, p. 1-13.
Security laws mainly have two objectives: require that investors receive financial and other significant information concerning securities being offered for public sale; and to prohibit deceit, misrepresentations, and other fraud in the sale of securities.

[78] Often, utility tokens have a max supply cap, necessarily meaning that an increase in demand will have positive effects on the price.

[79] ERC-20 Token Standard, www.ethereum.org.

real estate and digital collectibles. In the last year, the market for NFTs has grown exponentially: many artists released new albums or art pieces as NFTs[80] and major sport leagues such as the NBA or the McLaren F1 motor racing team have issued their own series of crypto collectibles which gained a huge success

[80] B. QUARMBY, *Breaking new ground is never easy' — Kings of Leon's NFT release takes in $2M*, www.cointelegraph.com, 2021; J. CRAWLEY, *Banksy Work Physically Burned and Digitized as NFT in Art-World First, www.coindesk.com,* 2021.

CHAPTER 2:

SMART LEGAL CONTRACTS: LEGAL FRAMEWORK

Before we get into the actual coding of a smart contract through the new programming language *Stipula*, we need to identify in what cases smart contracts gain legal relevance. To do so, we will draw a distinction between the general category of smart contracts as a piece of self-executing computer code, and the category of so-called *smart legal contracts*. The latter refers to those smart contracts that, fulfilling the necessary conditions laid down by national law, become fully enforceable by state power and are suited to produce legal effects within the legal system. This chapter aims at describing what requirements a contract need to satisfy in order to acquire the status of legally binding agreement within the legal system.

2.1 NEITHER SMART NOR CONTRACTS

By now, it is a well-established opinion in the doctrine that the term *smart contract* is a misleading expression[81]. Indeed, the provocative statement "Smart Contract are neither smart, nor contracts"[82] is backed by solid arguments.

On the one hand, "smartness" of blockchain-based contracts is questioned for two kinds of reasons, a technical and a legal one. It was well-exemplified by The DAO hack case[83], that a simple bug in a smart contract's code can result in a multimillion worth loss[84] with no possibility for the parties concerned to do anything to stop it.

---

[81] G.FINOCCHIARO, C. BOMPREZZI, *A legal analysis of the blockchain technology for the formation of smart legal contracts*, in *Medialaws,* 2020, p.115; C. LIM, TJ SAW, C. SARGEANT, *Smart Contracts: Bridging the Gap Between Expectation and Reality, Oxford Business Law Blog,* 2016, www.law.ox.ac.uk/business-law-blog; E. MIK, *Smart Contracts: Terminology, Technical Limitations and Real World Complexity,* in *Law, Innovation and Technology,* 2017, pag.4

[82] O. G. GÜÇLÜTÜRK, *Smart Contracts and Legal Challenges,* www.medium.com, 2018 ; A.MARTIN, *Smart Contracts: neither contracts nor smart,* www.lawahead.ie.edu.

[83] S. FALKON, *The Story of the DAO - Its History and Consequences*, www.medium.com, 2017. The DAO was a decentralised autonomous organization launched in April 2016 on the Ethereum Blockchain. It was meant to operate as a venture capital fund for the crypto and decentralized space. A few months after its launch, , on June 17, 2016, a hacker found a loophole in the coding that allowed him to drain funds from The DAO. In the first few hours of the attack, 3.6 million ETH were stolen.

[84] M. DEL CASTILLO*, The DAO Attacked: Code Issue Leads to $60 Million Ether Theft, www.coindesk.com, 2016.*

Strictly speaking, in fact, the code was not 'hacked' in a malicious way, but rather it was used by the hacker to accomplish something that parties didn't foreseen nor intended as an outcome[85]. In other words, that kind of money flow was in the logic of the (not so) smart contract and "the hacker understood terms and consequences of it, more than its creators"[86].

From a legal standpoint, 'smartness' is criticised because of the rigidity stemming from it. In traditional contracts, flexibility is an important source of efficiency[87]: it frees lawyers from the need to predict every future event that could affect the contract and negotiate the relative outcome. Blockchain-based contracts, instead, rely on a careful prespecification of terms and on automated enforcement of obligations, thus imposing a degree of inflexibility on contractors that fails to take the social complexities of contracting into account[88].

On the other hand, it should be pointed out that smart contracts are not necessarily contracts. They are, in essence, "pieces of computer code that, upon the occurrence of specific conditions are capable of running automatedly"[89]. Only when such software are employed in the contractual domain they are referred to as 'smart legal contracts'[90]. Furthermore, it was claimed that smart contracts "do not create obligations in its legal meaning"[91], which is, instead, a core element of continental contract law. This statement stems from the observation that the concept of obligation necessarily implies a 'will' factor, i. e. the discretion to perform or not to perform.

---

[85] M. RASKIN, *The Law and Legality Of Smart Contracts,* in *Georgetown Law Technology Review,* 2017, p. 337.

[86] I. KAMINSKA, *Legal Exploits and Arbitrage, DAO Edition*, in *Fin. Times,* 2016, www.ft.com.

[87] J. SKARLOFF, *Smart Contracts and the Cost of Inflexibility,* in *Un. of Pennsylvania Law Rev.,* 2017, p.279.

[88] K. E. C. LEVY, *Book-Smart, Not Street-Smart:Blockchain-Based Smart Contracts and The Social Workings of Law,* in *Eng. Science, Technology, and Society,* 2017, p.10.

[89] G.FINOCCHIARO, C. BOMPREZZI, *A legal analysis of the blockchain technology for the formation of smart legal contracts*, in *Medialaws,* 2020, p.115.

[90] *Ibidem.*

[91] A. SAVELYEV, *Contract Law 2.0: «Smart» Contracts As The Beginning Of The End Of Classic Contract Law,* in *Information & Communications Technology Law,* 2016, p.17. On point also K. WERBACH, N.CORNELL, *Contracts Ex Machina, In Duke Law Journal,* 2017, p.129.

A smart contract, on the contrary, doesn't leave this choice and, once initiated, is immutable and it exclusively runs the predefined code[92]. Although interesting, this conclusion can be disproven by acknowledging that, even when concluding a smart contract, parties still express their will to enter in a legal relationship. Moreover, contract law already acknowledges certain types of agreements which are performed instantaneously at the moment of conclusion[93].

Nevertheless, it is true that many of the software processes currently hailed as 'smart contracts' would be more accurately described as performance mechanisms[94].

That's why, usually, researchers distinguish between smart legal contracts 'as means of performance', where the computer program is used to automate the performance of a contract concluded outside the blockchain, and smart legal contracts 'as contracts', where the aim is to express the terms of an agreement in the form of lines of code[95].

The latter case is the focus of the present work and, in order to analyse it properly, we should start by assessing whether they meet the requirements established by contract law.

2.2 FULFILMENT OF CONTRACTUAL REQUIREMENTS

Every legal system lays down the standards that have to be met for an agreement to be legally binding. These partially differ from country to country. Here we will firstly examine the matter from a European standpoint by looking at the general principles of Union law and then we will address the specific Italian situation.

---

[92]M. GIULIANO, *La Blockchain e gli Smart Contracts nell'innovazione del Diritto nel Terzo Millennio* m, in *Il Diritto Dell'informazione E Dell'informatica,* 2018, p. 1026.

[93] A. SAVELYEV, *Contract Law 2.0: «Smart» Contracts As The Beginning Of The End Of Classic Contract Law,* in *Information & Communications Technology Law,* 2016, p.18.

[94] J.G. ALLEN, *Wrapped and Stacked: 'Smart Contracts' and the Interaction of Natural and Formal Language,* in *European Review of Contract Law*, 2018, p.320.

[95] G.FINOCCHIARO, C. BOMPREZZI, *A legal analysis of the blockchain technology for the formation of smart legal contracts*, in *Medialaws,* 2020, p.116.

2.2.1 European Context

Since no unique contract definition exists among EU member states, we will refer to the Draft Common Frame of Reference[96], according to which "A contract is an agreement which is intended to give rise to a binding legal relationship or to have some other legal effect. It is a bilateral or multilateral juridical act"[97]. The only elements required for the conclusion of a valid contract are a "sufficient agreement", and "the intention of the parties to be legally bound, without any further requirement"[98].

The agreement, the so called 'meeting of the minds' is the core of the contract and it's usually expressed through the exchange of a contractual offer and the following acceptance.

As to form, the PECL[99] affirms the principle of informality providing that "A contract need not to be concluded or evidenced in writing nor is it subject to any other requirement as to form. The contract may be proved by any means, including witnesses"[100]. However, in regard to specific contracts, national laws may require some formalities for the validity or to make evidence of the contract. On this point it is questioned whether the electronic form of a smart contract satisfies those requirements.

2.2.1.1 Offer and Acceptance

An offer is an expression to another party or to the community at large, to be bound by the stated terms[101]. The deployment of smart contract code on a distributed ledger is generally deemed to correspond to an offer[102], at least for those individuals who are

---

[96] The DCFR was launched and sponsored by the Commission of the European Union. The project was drafted by two academic working groups and published in 2008. It represents the synthesis of contract law principles common to EU member states and of *acquis communitariae.*
It was meant to be toolbox for future European legislations in the field of contract law (European civil code). In practise it also serves as an "optional instrument", i.e., a set of rules which parties to a transnational contract can agree upon to govern their transactions. See N. JANSEN, R. ZIMMERMANN, *A European Civil Code In All But Name": Discussing The Nature And Purposes Of The Draft Common Frame Of Reference*, in *The Cambridge Law Journal*, 2010, p.98.

[97] Art. II – 1:101 DCFR.

[98] Art. II – 4:101 DCFR.

[99] Principles of European Contract Law document, drafted by a committee of european eminent law academics, coordinated by professor Ole Lando of Copenhagen University. It represents the most direct progenitor of the DCFR.

[100] Art. II – 2:101 (2) PECL.

[101] P. CATCHLOVE, *Smart Contracts: A New Era Of Contract Use,* 2017, p.10.

[102]M. DUROVIC, A. JANSSEN, *The Formation of Blockchain-based Smart Contracts in the Light of Contract Law,* in *European Review of Private Law,* 2019, p. 762.

allowed to interact with the smart contract[103]. When any participant in the network can interact with it and conclude the agreement, then the uploading represents an offer to the public[104]. In order to be valid, an offer must contain all the terms of the agreement (*essentialia negotii*). Otherwise, it will only represent an invitation to treat[105] and the offeror won't be bound by the stated terms.

Once a valid offer has been uploaded in the network it is capable of acceptance by the offeree. An acceptance is a form of statement or conduct that indicates assent to the offer[106]. In contract law, when the declaration made by the offeree doesn't take into account all the terms of the offer it would represents a counteroffer. However, such possibility is excluded by the immutable character of the blockchain[107]. In a smart contract, parties express acceptance by signing a transaction with their cryptographic keys and by sending it to the contract address[108]. As it was noted, the majority of todays' smart contracts represent unilateral contracts, stating, for example, that if X happens I will give you Y[109]. With these types of contracts, acceptance always comes through the act of performance. The transaction expressing consent will likely represent the transfer of control over a digital asset to the smart contracts[110] as, for example, money, cryptocurrency or digital token representing a material good.

2.2.1.2 Contractual Intention

In addition to mutual assent, for a valid contract to be concluded, parties must show the intention to be legally bound. The intention "has to be determined from the party's statements or conduct as they were reasonably understood by the other party"[111]. This

[103] M. B. A. BIJUESCA, *SMART CONTRACTS:Is the Law Ready?*, www.digitalchamber.org, 2018, p. 15

[104] G.FINOCCHIARO, C. BOMPREZZI, *A legal analysis of the blockchain technology for the formation of smart legal contracts*, in *Medialaws,* 2020, p.118.

[105] See n. 95.

[106] Art. II – 2:204 (1) PECL.

[107] See n. 97.

[108] G. O. B. JACCARD, *Smart Contracts and the Role of Law*, in Jusletter, 2017, p.23.

[109] M. DUROVIC, A. JANSSEN, *The Formation of Blockchain-based Smart Contracts in the Light of Contract Law,* in *European Review of Private Law,* 2019, p.763.

[110] P. CATCHLOVE, *Smart Contracts: A New Era Of Contract Use,* 2017, p.11.

[111] Art. II – 2:102 PECL.

means that intention has to be evaluated objectively, assessing whether a reasonable person would see the agreement as binding[112] and not mattering inner intentions or personal understanding[113]. It is generally recognised that the fact that parties submit their cryptographic private keys to commit their resources to the smart contract is proof of such an intent[114].

However, the real issue regarding contractual intent is lies in the fact that computer code is an opaque language to most human being[115]. It is questioned, in fact, whether contracts written in code can provide reliable commitments[116] also because, as mentioned above, smart contracts are mostly unilateral 'take it or leave it' contracts, where non-drafting parties accept the terms in a non-conventional way[117].

In the context of contract entered into online, a comparison can be made with 'clickwrap' or 'browsewrap' agreements, where users agree to the terms by clicking on a box in the website, or by simply continuing the browsing. In such cases, courts have required an "effective notice" of contractual terms[118] so that users can undoubtably be aware of the binding nature of latter. These considerations lead to the conclusion that, as best practises, drafters shall accompany the code with a natural language version of the terms of the agreement[119]

2.2.1.3 Form

The PECL affirms the principle of informality of contracts by stating that "a contract need not be concluded or evidenced in writing nor is it subject to any other requirement as to form"[120]. Therefore, there are no obstacles to expressing legal agreements in electronic form.

---

[112] P. CATCHLOVE, *Smart Contracts: A New Era Of Contract Use,* 2017, p. 11.

[113] G.FINOCCHIARO, C. BOMPREZZI, *A legal analysis of the blockchain technology for the formation of Smart legal contracts*, in *Medialaws,* 2020, p.122.

[114] K. WERBACH, N.CORNELL, *Contracts Ex Machina, In Duke Law Journal,* 2017, p.156.

[115] G. O. B. JACCARD, *Smart Contracts and the Role of Law*, in Jusletter, 2017, p.22.

[116]O.RIKKEN et al, *Smart contracts as a specific application of blockchain technology,* Dutch Blockchain Coalition, 2017, P.22.

[117] See n. 106, p.123.

[118] R. O'SHIELDS, *Smart Contracts: Legal Agreements for the Blockchain*, 2017, p.186.

[119] O.RIKKEN et al, *Smart contracts as a specific application of blockchain technology,* Dutch Blockchain Coalition, 2017, P.22. Even though, "a possible disadvantage of agreements in natural language in addition to code is that these may give rise o differences in interpretation".

[120] Art. II – 2:101 (2) PECL.

On this point the EU Blockchain Observatory and Forum stated that "As fully digital ledgers, blockchains are by definition electronic documents under e-IDAS regulation" and therefore, according to art.46 of e-IDAS regulation, "the data, including smart contracts, contained therein, cannot be denied legal force solely because of their electronic nature"[121]

However, specific national laws may require some formalities for the validity of specific contracts and in such cases parties are usually required to sign contracts in written form. When a contract is in electronic form, it can be signed with electronic signatures[122]. We should then asses whether cryptographic keys used to sign transaction in blockchain systems can meet the legal standards to satisfy the written form.

Within the EU, electronic signatures are regulated by the e-IDAS regulation[123]. This act distinguishes three kinds of signatures: *simple electronic signature*, *advanced electronic signature*s and *qualified electronic signatures*

With respect to the legal value, the regulation affirms a general principle of non-discrimination for all types of signature by stating that "An electronic signature shall not be denied legal effect and admissibility as evidence in legal proceedings solely on the grounds that it is in an electronic form or that it does not meet the requirements for qualified electronic signatures"[124]. Only in relation to qualified signatures it provides that they shall have the same legal value as handwritten signatures[125], while leaving to national legislators the discretion to define the legal effect accorded to the other types[126].

An electronic signature is defined as "data in electronic form which is attached to or logically associated with other data in electronic form and which is used by the signatory to sign"[127]. This definition confirms the EU adoption of the *technological*

[121]REPORT: Blockchain an Digital Identity, *Eu Blockchain Observatory and forum,* www.eublockchainforum.eu, 2019, p.21.

[122] G.FINOCCHIARO, C. BOMPREZZI, *A legal analysis of the blockchain technology for the formation of Smart legal contracts*, in *Medialaws,* 2020, p.129.

[123] Regulation (EU) n. 2014/910. Regulations are sources of European secondary law of general application and are binding in their entirety and directly applicable in all European Union countries. www.eur-lex.europa.eu.

[124] Art. 25 (1) Regulation (EU) n. 2014/910.

[125] Art. 25 (2) Regulation (EU) n. 2014/910.

[126] Recital 49 Regulation (EU) n. 2014/910.

[127] Art 3 (10) Regulation (EU) n. 2014/910.

neutrality and *functional equivalence* principles originated within the UNICITRAL[128]. The ratio of the norm, in fact, is not to identify a specific technology, but rather to determine some criteria to link a technology to some functions attributed to handwritten signatures, in such a way as to comprehend any possible future solution[129]. When signing a transaction on a distributed ledger, parties link some data (the private key) to other data (the transaction) to approve the information included in the latter data (offer and acceptance). So, we can conclude that such signatures can be considered at least simple electronic signatures[130] under e-IDAS.

An advanced electronic signature is defined as "an electronic signature which meets the requirements set out in Article 26"[131]. These requirements are meant to provide a higher degree of certainty and reliability in the connection between data used to sign and the signatory, in particular when it is provided that an advanced electronic signature "shall be capable of identifying the signatory"[132]. This standard doesn't seem to be fulfilled by signatures used to sign transaction in the most common blockchains such as Ethereum or Bitcoin[133].

The same consideration can be made for qualified signatures which are "advanced electronic signatures created by a qualified electronic signature creation device, and which are based on a qualified certificate for electronic signatures"[134]. Blockchain-based signature, in fact, aren't created through the utilisation of a creation device[135],

[128] Art. 3 of Model Law on Electronic Signature. "Nothing in the Law shall be applied so as to exclude, restrict or deprive of legal effect any method of creating an electronic signature that satisfies its requirements or otherwise meets the requirements of applicable law".

[129] M. C. MENGHETTI, *Articolo 3,* in F. DELFINI, G. FINOCCHIARO (edited by), *Identificazione Elettronica E Servizi Fiduciari Per Le Transazioni Elettroniche Nel Mercato Interno,* Torino, Giappichelli Editore, 2017, p. 39

[130] G.FINOCCHIARO, C. BOMPREZZI, *A legal analysis of the blockchain technology for the formation of Smart legal contracts*, in *Medialaws,* 2020, p.130.

[131] Art. 3 (11) Regulation (EU) n. 2014/910 .

[132] Art. 26 b) Regulation (EU) n. 2014/910.

[133] M. NICOTRA, F. SARZANA DI SANT'IPPOLITO, *Diritto della Blockchain, Intelligenza Artificiale e IoT*, Vicenza, Ipsoa, 2018, p. 59.

[134] Art. 3 (12) Regulation (EU) n. 2014/910.

[135] According to art. 3 (22) Regulation (EU) n. 2014/910, is a "configured software or hardware used to create an electronic signature".

nor linked to a qualified certificate[136] issued by a trusted service provider. Thus, we can conclude that cryptographic keys used to sign transaction on open distributed ledgers only meet the threshold of simple electronic signatures under e-IDAS to which, however, cannot be denied legal effect or admissibility on the sole ground of the technological solution adopted. It will be up to legislators and courts to evaluate admissibility of such signatures on a case-by-case basis[137].

However, the situation may change when operating within a permissioned blockchain where participants are allowed only upon satisfaction of certain requirements and/or approval by an administrator[138]. In such context one of the participants in the network could likely assume the role of certifier of the identities of signatories in such a way as to satisfy the requirements for an advanced electronic signature[139].

2.2.2 The Italian context

According to the Italian Civil Code a contract is "l'accordo di due o più parti per costituire, regolare o estinguere tra loro un rapporto giuridico patrimoniale"[140]. Article 1325 lays down the requirements for a valid contract. These are *paries' agreement*, contractual *cause*, *object* of the contract and, only when expressly required by law, *form*. So, even the Italian legal system follows the principle of informality for contracts except for those cases where law specifically provides otherwise.

---

[136] According to art. 3 (15) Regulation (EU) n. 2014/910, it is a "certificate for electronic signatures, that is issued by a qualified trust service provider and meets the requirements laid down in Annex I"

[137] G. FINOCCHIARO, *Articolo 25,* in F. DELFINI, G. FINOCCHIARO (edited by), *Identificazione Elettronica E Servizi Fiduciari Per Le Transazioni Elettroniche Nel Mercato Interno,* Torino, Giappichelli Editore, 2017, p. 214.

[138] E.MIK, *Blockchains: a Technology for decentralised Marketplaces,* in L. A. DIMATTEO - M. CANNARSA – C. PONCIBO', *The Cambridge Handbook of Smart Contracts, Blockchains Technology and Digital Platforms,* Cambridge, Cambridge University Press, 2020, p.164.

[139] M. NICOTRA, F. SARZANA DI SANT'IPPOLITO, *Diritto della Blockchain, Intelligenza Artificiale e IoT*, Vicenza, Ipsoa, 2018, p.64.

[140] Art. 1321 c.c.

2.2.2.1 Parties' agreement

The agreement is reached through the exchange of offer and acceptance[141]. Therefore, we can refer to the arguments described in the above section

2.2.2.2 Cause

The concept of cause identifies with the socio-economic function of the legal transaction[142]. The sole agreement is not enough to create an obligation if it lacks a socio-economic motivation of the deed[143.]

If the smart contract represents a typical contract the cause is evident and can be deduced by the sole computer code. If it's not evident, information describing it can easily be included in the code or in a document attached to the smart contract.[144]. Obviously, the cause has to abide to all the rules provided by art. 1343 and the following, meaning that it has to be legal, not in contrast with public order, morality and not being a means to circumvent law provisions[145]. A contract which breaches such provisions will be considered null and void.

2.2.2.3 Object

The object of the contract is the good or the right that is transferred through the contract or, in general, the performance that the obligation entails[146]. This doesn't create any problem for the implementation of a smart contract as the essence and the main purpose of it is to execute the performance agreed upon by the parties. Thus, it seems totally capable of describing the object of the contract[147] .

---

[141] Art. 1326 c.c.
[142] A. ZACCARIA, *art. 1343,* in C. CIAN, A. TRABUCCHI (edited by), *Commentario Breve al Codice Civile*, Milano, Cedam*,* 2021, p. 1418.
[143] F. GALGANO, *Trattato di Diritto Civile, Cedam, 2014,* p.239.
[144] M. MANENTE, Studio 1_2019 DI L. 12/2019 – *Smart Contract E Tecnologie Basate Su Registri Distribuiti, Consiglio Nazionale del notariato*, www.notariato.it*,* p. 7.
[145] Art. 1344 c.c.
[146] F. GALGANO, *Trattato di Diritto Civile, Cedam, 2014,* p.264.
[147] M. MANENTE, Studio 1_2019 DI L. 12/2019 – *Smart Contract E Tecnologie Basate Su Registri Distribuiti, Consiglio Nazionale del notariato*, www.notariato.it, p. 5.

2.2.2.4 Form

Italian law may expressly require some contracts to be evidenced in a particular form as an exception to the general principle of informality. The legal relevancy of electronic documents and electronic signatures is regulated by the CAD[148] (*Codice dell'Amministrazione digitale*) which also transposes the e-IDAS regulation and follows the same general principles.

The European regulation, after establishing the principle of non-discrimination for electronic form, it leaves to national law the discretion "to define the legal effect of electronic signatures", except for the requirements provided for in the Regulation relatively to qualified electronic signature [149];

The CAD departs from e-IDAS by providing for an additional kind of signature other than the three provided by the regulation, the *digital signature*. This is defined as a specific kind of qualified signature based on a cryptographic system consisting of a public and a private key[150]. With this definition the Italian legislator deviates from the principle of technological neutrality recalled by e-IDAS as it identifies a specific technology[151].

As to the legal effect granted to the different electronic signatures, it is provided that advanced, qualified and digital signatures satisfy the written form requirement and are recognised with the probative value accorded by art. 2702 of Italian Civil Code[152]. Furthermore, electronic documents signed with qualified or digital signatures fulfil the requirement of written form to represents the deeds listed in art 1350 cc[153].

Finally, the probative value of electronic documents signed with other kinds of signatures will be determined on a case-by-case evaluation discretionally made by

---

[148] D. lgs. n. 83/2015.
[149] Recital 49 Regulation (EU) n. 2014/910.
[150] Art. 1 s), d. lgs. n. 83/2015 (CAD).
[151] M. C. MENGHETTI, *Articolo 3,* in F. DELFINI, G. FINOCCHIARO (edited by), *Identificazione Elettronica E Servizi Fiduciari Per Le Transazioni Elettroniche Nel Mercato Interno,* Torino, Giappichelli Editore, 2017, p. 42.
[152] It is the value accorded to the 'private agreement' (scrittura privata), which is considered a legal proof of the identity of the signatory, except where disproven by a fraud compliant.
[153] Art. 21 d. lgs. n. 83/2015 (CAD).

courts, according to the level of security, immutability and integrity of such technologies[154]. The very inner features of the blockchain technology definitely satisfy both the requirement for integrity and for immutability of the data. As for the requirement concerning the security, it was rightly noted that this refers not only to technical security but also to accessibility and therefore, to the identifiability of the signatory[155]. On this point, even though someone deems that computer software in general can never guarantee identification of users as they are simply based on the verification of login credentials [156], it seems fair to admit that, if a court succeeded in finding a *sure* link between the cryptographic key and a specific individual, all the criteria would be satisfied, and the signature would be granted with full legal effect[157]

2.3 THE CONTRACT LIFECYCLE

In order to understand where smart contracts can actually be helpful, it is useful to bear in mind the general lifecycle of a contract. Indeed, most legal issues arise in the interlace between *on-chain* and *off-chain* elements.

2.3.1 Negotiation and Formation

As mentioned above, the 'meeting of the minds' is the core element of a contract, and this is achieved through the first stage, negotiation. In blockchain contexts, the negotiation about the content of a contract is usually held off-chain before deploying the smart contract in the network[158]. In other words, the agreement is negotiated (or unilaterally drafted) off-chain, and legally concluded on-chain.

---

[154] Art. 20 d. lgs. n. 83/2015 (CAD).
[155] M. NICOTRA, F. SARZANA DI SANT'IPPOLITO, *Diritto della Blockchain, Intelligenza Artificiale e IoT*, Vicenza, Ipsoa, 2018, p. 61.
[156] U. BECHINI – M. C. CIGNARELLA, Quesito Antiriciclaggio n. 3-2018/B, Consiglio nazionale del Notariato, www.notariato.it, 2018, p. 3.
[157] M. NICOTRA, F. SARZANA DI SANT'IPPOLITO, *Diritto della Blockchain, Intelligenza Artificiale e IoT*, Vicenza, Ipsoa, 2018, p. 62.
[158] G. SARTOR, G. GOVERNATORI, F. IDELBERGER, R. RIVERET, *Evaluation of Logic-Based Smart Contracts for Blockchain Systems, 2016.*

However, sometimes algorithms can be employed to find a contracting party and/or to negotiate the terms of the agreement[159]. In such cases the smart contract acts as an "artificial agent". This kind of smart contracts raises a whole new series of legal issues. When a software acts as a negotiator and thus, takes decisions on his own, it is questioned whether or not it has the legal capacity to do so and whether it acts as a contracting party or just as a representative[160]. Furthermore, some kinds of algorithmic contracts clearly "introduce a gap between the objectively manifested intent of the party using the algorithm and what the artificial agent does"[161]. However, such matters are not the object of the present work and we will focus instead on the first type of smart contracts.

2.3.2 Storage/Notarizing

As a general principle of informality applies, contracts can be concluded in many ways, such as oral agreement, handshake, or written agreement. Nevertheless, problems may arise when there's a dispute between the parties, on the very content of the agreement. To avoid such problems, it is useful to store a written record of what was agreed and to have it certified by a trusted third party both as to the time of conclusion and as to the meaning of the content.

At this stage blockchain can prove particularly helpful. Given the immutable character of the ledger, data can be stored with no risk of corruption and with relatively accurate timestamp that certifies the time of conclusion[162].

[159] M. DUROVIC, A. JANSSEN, *The Formation of Blockchain-based Smart Contracts in the Light of Contract Law,* in *European Review of Private Law,* 2019, p. 760.

[160] The EU Blockchain Observatory and Forum, *REPORT: Legal and Regulatory Framework Of Blockchains And Smart Contracts*, www.eublockchainforum.eu, 2019, p. 31.

[161] L.H. SCHOLZ, Algorithmic Contracts, in *Stanford Technology Law Review,* 2017, p. 128. Particularly in the case of Black box algorithmic contracts where the logic of the algorithm is functionally opaque and, as a consequence, the algorithm's behaviour cannot be anticipated by the principal.

[162] G. SARTOR, G. GOVERNATORI, F. IDELBERGER, R. RIVERET, *Evaluation of Logic-Based Smart Contracts for Blockchain Systems, 2016.*

2.3.3 Enforcement and Monitoring

The enforcement refers to the encouraged or forced execution of the required performance of a contract[163]. This stage is where smart contracts' show their most valuable functionality, self-enforceability. This feature can be understood in the sense that, once a smart contract is stored in the blockchain and its condition are satisfied, the terms will be executed even if the original parties to the contract
no longer wish them to. Indeed, "discretion whether or not to abide by the terms of the contract is taken away from the parties, and once they have concluded a smart contract they will abide by the terms"[164].

As to monitoring, this is the activity of checking whether the appropriate actions are taken. Issues related to monitoring frequently involve the high costs of the procedure[165]. In smart contracts, control methods can be integrated in the code of the software[166] and this would result in a cost reduction.

2.3.4 Modification

If all parties perform their duties as agreed, then no modification is required. In the lifecycle of a contract, though, exogenous events may change the surrounding circumstances in such a way as to induce the parties to modify the terms of their agreement. If all the parties agree to the change, the contract can be amended. In current blockchain systems, data cannot be modified. However, the employment of particular techniques in the design of a smart contract, can enable flexible solutions and allow the smart contract to be updated[167].

---

[163] *Ibidem*.
[164] M. DUROVIC, F. LECH, *The Enforceability of Smart Contracts,* in *The Italian Law Journal,* 2019, p. 438.
[165] J. REHFUSS, *Contracting Out and Accountability in State and Local Governments-The Importance of Contract Monitoring,* in *State & Local Government Review,* 1990, p. 95
[166] G. SARTOR, G. GOVERNATORI, F. IDELBERGER, R. RIVERET, *Evaluation of Logic-Based Smart Contracts for Blockchain Systems, 2016*.
[167] *Ibidem*. For example, the 'hub-and-spoke' model "where one main smart contract holds addresses/pointers to all other necessary contracts that contain the specific clauses and functionality".

2.3.5 Dispute Resolution

The very nature of smart contract doesn't square easily with the possibility of conflicts relating to it. In fact, any event that was not foreseen and included in the code will rule out self-governing feature of smart contracts and will have to be dealt off-chain. One possible solution could be to integrate in the smart contract an alternative dispute resolution (ADR) mechanism specifically designed for such contract and that could operate on the blockchain network (e.g., the use of distributed juries). However, such a solution would lack of efficiency as it would require a lot of work to define a specific set of rules for ADR during the drafting of each contract.[168] If not ADR is employed, parties will have to use traditional dispute resolution channels: court litigation, arbitration and consensual resolution (such as mediation or negotiation. In relation to the former method, many problems were highlighted on the actual capacity for a court to address disputes relating to smart contracts. Issues often concerns the difficulty courts would have in identifying contractors (given the potential anonymity of smart contracts' users) and in enforcing judgments towards parties set across multiple jurisdictions [169]. A possible answer to these issues could be using court's judgments as "oracles": territorial courts could resolve disputes and provide inputs to the smart contract enforcing judgments through code[170].

As to consensual resolution, parties could specify in a smart contract that a committee of humans or computational arbitrators should be consulted first when a dispute arises[171].

---

[168] M. CLEMENT, *Smart Contracts and the courts,* in L. A. DIMATTEO et al. (edited by), *The Cambridge Handbook of Smart Contracts, Blockchains Technology and Digital Platforms,* Cambridge, Cambridge University Press, 2020, p. 285.

[169] Ibidem.

[170] D. W. E. ALLEN, A. M. LANE, M. POBLET, *The Governance of Blockchain Dispute Resolution,* in *Harvard Negotiation Law Review,* 2019, p. 87. "However, an ongoing practical challenge is that court orders would require specific standards to be machine-readable and machine-computable so that the blockchain platform can read the judgement and execute its orders".

[171] G. SARTOR, G. GOVERNATORI, F. IDELBERGER, R. RIVERET, *Evaluation of Logic-Based Smart Contracts for Blockchain Systems, 2016*.

Of course, each one of these solutions has pros and cons in terms of efficiency and it will be up to parties to make a choice based on their interests and the nature of the agreement.

2.4 BLOCKCHAIN AND SMART CONTRACTS IN NATIONAL LAWS.

Blockchain technology is a little more than ten years old but the great expansion it experienced in the last few years shed light on some important legal issues it raises, especially relating to smart contracts. That is why several countries around the world already enacted legislations aimed at addressing and regulating the phenomenon. This section lays down an overview on the most significant normative actions taken by national governments on the issue, both from a European and extra-EU standpoint.

2.4.1 Italy

Italy was one of the first European countries to adopt a legislation dealing with the blockchain phenomenon, in a clear attempt to stimulate innovation and to present the country as a "blockchain-friendly"[172] land for potential investors. However, this attempt was not received without critiques by commentators, both from a technical point of view and for its implementation modalities[173].

The dispositions were introduced in the d.l. n. 135/2018 with the l. 12/2019.

Art. 8-*ter,* 1st par.:

> "Si definiscono 'tecnologie basate su registri distribuiti' le tecnologie e i protocolli informatici che usano un registro condiviso, distribuito, replicabile, accessibile simultaneamente,

[172] C. BOMPREZZI, *Commento in materia di Blockchain e Smart contract alla luce del nuovo Decreto Semplificazioni*, in *Diritto, Mercato e Tecnologia,* www.dimt.it, 2019

[173] R. DE CARIA, *Blockchain e smart contract: questioni giuridiche e risposte regolatorie tra diritto pubblico e privato dell'economia,* in M. T. GIORDANO, R. BATTAGLINI (a cura di), *Blockchain e Smart contracts*, Giuffrè Francis Levbre. 2019, p.205; R. BATTAGLINI, *La Normativa Italiana sugli Smart Contracts,* in M. T. GIORDANO, R. BATTAGLINI (a cura di), *Blockchain e Smart contracts*, Giuffrè Francis Levbre. 2019, p.376. The author expresses doubts on the legislator's choice of addressing such a complex phenomenon in just one article of an (theoretically) emergency piece of legislation.

architetturalmente decentralizzato su basi crittografiche, tali da consentire la registrazione, la convalida, l'aggiornamento e l'archiviazione di dati sia in chiaro che ulteriormente protetti da crittografia verificabili da ciascun partecipante, non alterabili e non modificabili."

The first paragraph gives a definition of DLTs (Distributed Ledger Technologies), enlisting several features that such technology must satisfy. The first critical point concerns the two requirement of "immutability" and "tamperproof" ("immodificabilità" and "inalterabilità"). Such characteristics are achievable only through a blockchain (actually, not even so[174]) that, as already mentioned, is not a synonym for DLT: the two concepts have a genre-species relation in that a blockchain is a DLT but not every DLT is a blockchain. Therefore, the choice of words seems to be reducing the scope of DLTs only to those that implement strong cryptographic and security features[175].

The 2nd par. lays the definition of smart contract:

"Si definisce 'smart contract' un programma per elaboratore che opera su tecnologie basate su registri distribuiti e la cui esecuzione vincola automaticamente due o più parti sulla base di effetti predefiniti dalle stesse. Gli smart contract soddisfano il requisito della forma scritta previa identificazione informatica delle parti interessate, attraverso un processo avente i requisiti fissati dall'Agenzia per l'Italia digitale con linee guida da adottare entro novanta giorni dalla data di entrata in vigore della legge di conversione del presente decreto."

[174] As mentioned above, the blockchains are not one hundred percent tamper-proof, but rather *tamper-resistant,* in that there exist ways to take control of the majority of the nodes of the network, unwind the chain and modify the data.
[175] G. RINALDI, *Smart Contract: Meccanizzazione Del Contratto Nel Paradigma Della Blockchain*, www.academia.edu, 2019, p.27.

This paragraph also addresses the probative force of smart contracts stating that the written form is satisfied when the contract complies with the guidelines set by the Agenzia per l'Italia Digitale adopted within 90 days from law's the entry into force. Beside the fact that, more than two years after the law was published, still no guidelines have been adopted by the agency, it is hard to comprehend why the legislator didn't make reference to already existing laws like the CAD (Codice per l'italia Digitale), or the e-IDAS regulation that address the legal value of electronic documents (under whose scope smart contract definitely fall[176]) and related electronic signatures, but rather created a separated scheme only for those smart contracts built upon a DLT. Form this prospective, the disposition could turn out to be redundant or even threaten to limit the application of this technology[177].

3rd paragraph:

> "La memorizzazione di un documento informatico attraverso l'uso di tecnologie basate su registri distribuiti produce gli effetti giuridici della validazione temporale elettronica di cui all'articolo 41 del regolamento (UE) n. 910/2014 del Parlamento europeo e del Consiglio, del 23 luglio 2014"

The third paragraph provides documents recorded on a DLT with the legal force accorded to electronic timestamps by e-IDAS regulation in art. 41. According to the regulation, an electronic timestamp is "data in electronic form which binds other data in electronic form to a particular time establishing evidence that the latter data existed at that time"[178]. The issue with this disposition is that it does not specify whether it refers to a 'simple electronic timestamp' or to a 'qualified electronic timestamp', both

[176] EU BLOCKCHAIN OBSERVATORY & FORUM, *Blockchain and Digital Identity,* 2019, p.20.
[177] C. BOMPREZZI, *Commento in materia di Blockchain e Smart contract alla luce del nuovo Decreto Semplificazioni*, in *Diritto, Mercato e Tecnologia,* www.dimt.it, 2019.
[178] Art.3 (33), EU Regulation n. 910/2014.

addressed to in art. 41 of the regulation. Such uncertainty could raise conflicts of interpretation since the different electronic timestamps are provided with different legal force[179]: the qualified shall enjoy the presumption of accuracy of date and time, and the integrity of the data throughout all EU countries[180], while the determination of the legal value of the simple one is left to courts' evaluation.

In conclusion, despite the valuable initiative by the Italian legislator to address the blockchain phenomenon as a forerunner, the evident flaws of the disposition cannot be ignored, especially because they could turn out to impose restraints to innovation.

2.4.2 EU Countries

In the European Union, Malta is the member state that made more steps further in the regulation of blockchain and smart contracts. In 2018, the Republic of Malta passed three laws[181] to encourage the employment of blockchain technology by creating a robust legislative framework. Through the MDIA Act, it was instituted the Malta Digital and Innovation Authority, an independent authority whose task is to develop services that rely on new technologies and supervise on the compliance of such services to the standards provided by the relevant law. In art. 2, a definition of DLT is provided, as a "database system in which information is recorded, consensually shared, and synchronised across a network of multiple nodes"[182]. Then, by contrast to Italian law, another disposition specifies that a DLT may (but not necessarily) also present some other features (cryptographic protection, public or private character, permissionless or permissioned character) that are specific of a blockchain[183]. The same art.2 includes a definition of smart contract which correctly makes a distinction

[179] C. BOMPREZZI, *Commento in materia di Blockchain e Smart contract alla luce del nuovo Decreto Semplificazioni*, in *Diritto, Mercato e Tecnologia,* www.dimt.it, 2019.
[180] Art.41, EU Regulation n. 910/2014.
[181] REPUBLIC OF MALTA, Act No. XXXI of 2018 (MDIA Act); REPUBLIC OF MALTA, Act XXXIII of 2018 (ITAS Act); REPUBLIC OF MALTA Legal Notice 355 of 2018.
[182] REPUBLIC OF MALTA, *Malta Digital and Innovation Authority Act (MDIA Act), 2018,* Art.2, par.8.
[183] REPUBLIC OF MALTA, *Innovative Technology Arrangements and Services Act (ITAS Act),* 2018, First Schedule, Art.1.

between smart contract as a computer protocol, and smart contract as a legal agreement[184].

France also addressed the blockchain technology in two decrees[185] where it recognizes the legal value of blockchains used for the record and transfer of *minibons*[186] and other unlisted securities.

In the UK, despite no national binding law was adopted, the issue is currently under study by the Law Commission, an independent authority whose task is to supervise and analyse UK's laws complex, ensure effectiveness of the legal system, and make reform recommendations to the parliament. In its last report[187] on the subject, the commission addresses all the main points relating to smart contracts: it provides for a definition of both smart contracts and DLTs, describes how smart contracts can satisfy the relevant requirement to be legally enforceable, and proposes some options for dispute resolution.

2.4.3 United States

The US generally established the non-discrimination principle for electronic form at federal level with the Electronic Signatures in Global and National Commerce Act (ESIGN). It provides the records, contracts, or signatures in electronic form with the same legal value of the paper equivalents[188].

Moreover, several states enacted bills specifically regulating smart contracts and Distributed Ledger Technologies. The first in the line was Arizona that, with House Bill n. 2417, recognizes that "smart contracts may exist in commerce" and that "a

[184] REPUBLIC OF MALTA, *Malta Digital and Innovation Authority Act (MDIA Act), 2018,* Art.2, par.17.
[185] Ordonnance n. 2016-520 du 28 Avril 2016 relative aux bons de caisse e Ordonnance n° 2017-1674 du 8 décembre 2017 relative à l'utilisation d'un dispositif d'enregistrement électronique partagé pour la représentation et la transmission de titres financiers.
[186] *Minibons* are a kind of minibond usually issued by unlisted companies for crowdfunding.
[187] UK LAW COMMISSION, *Smart Contract Call for Evidence,* www.lawcom.gov.uk*,* 2020.
[188] Section 101, a), US PUBLIC LAW 106–229 (ESIGN Act), 2000.

contract relating to a transaction cannot be denied legal effect, validity or enforceability solely because that contract contains a smart contract term"[189]. According to the bill a smart contract is "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". Arizona's legislation inspired other states like Tennessee[190], Arkansas[191], North Dakota[192] who adopted similar legislations in the following years.

A different approach was taken by Wyoming Senate Bill n. 38 of 2021, which links the concept of smart contract to that of 'automated transaction' as defined in W.S. 40-21-102(a)(ii)[193]. According to this disposition, the smart contract's author (and owner of the relevant private key) is entitled to dispose of the assets which are the object of the transaction[194].

After this brief overview on DLTs and smart contracts, both from a technical and a legal perspective, we can observe that the scope of application for this technology is extremely broad and that, undoubtably, it is still under development. On the legal side, the consequence is the tendency by legislators to provide for an open legal framework that avoids strict definitions or requirements and that can adapt to future evolutions.

[189] STATE OF ARIZONA, House Bill. No. 2417 of 2017, art. 5 E, 2.
[190] STATE OF TENNESSEE, Senate Bill no. 1662 of 2018.
[191] STATE OF ARKANSAS, House Bill no. 1944 of 2019.
[192] STATE OF NORTH DAKOTA, House Bill no. 1045 of 2019.
[193] According to the STATE OF WYOMING Senate Bill no. 38 of 2021, art. 17-31-102 a) (ix), the definition of 'automated transaction' is extended to "any substantially similar analogue, which is comprised of code, script or programming language that executes the terms of an agreement and which may include taking custody of and transferring an asset, administrating membership interest votes with respect to a decentralized autonomous organization or issuing executable instructions for these actions, based on the occurrence or non-occurrence of specified conditions".
[194] N. TRAVIA, *Profili Internazionali del Diritto degli Smart Contracts,* in M. T. GIORDANO, R. BATTAGLINI (a cura di), *Blockchain e Smart contracts*, Giuffrè Francis Levbre. 2019, p. 393.

CHAPTER 3
STIPULA

The core element of smart legal contract technology is the transposition in computer code of legal agreements' terms and clauses. The interlace between formal and natural language raises essentially two types of problems. A technical or objective problem, given by the difficulty formal languages have in addressing *pragmatics*[195](a): machines are not good at solving ambiguities concerning the meaning of a word by drawing inferences from the context in which words are uttered[196] (in other words, inflexibility). A subjective problem (b): computer code is a language that only highly qualified people can comprehend, thus making knowability of contracts' contents hard or impossible to achieve for the average man. Such problem may also have a negative impact on the correct formation of contractual will[197].

This chapter brings forward *Stipula,* a new domain-specific language for the creation of smart contracts that may help in solving part of the aforementioned issues. *Stipula* is pivoted on a small number of abstractions that captures the main abstract elements of contract law such as *obligations, permissions, prohibitions* or *aleatory events*. Functions are expressed in a relatively straightforward language, allowing for the comprehension and the drafting of smart legal contract even by non-ICT experts (b)[198]. In the following sections, I will first present the primitives featured by *Stipula* and the legal pattern they aim at representing; after that, I will show some examples of smart legal contracts implemented through *Stipula* and explain how to read the code.

---

[195] *Pragmatics*, www.dictionary.cambridge.org: "The study of how language is affected by the situation in which it is used, of how language is used to get things or perform actions, and of how words can express things that are different from what they appear to mean".
[196] J.G. ALLEN, *Wrapped and Stacked: 'Smart Contracts' and the Interaction of Natural and Formal Language*, in *European Review of Contract Law,* 2018, p.323.
[197] Citazione sul problema della formazione della volontà tramite il codice.
[198] S. CRAFA, C. LANEVE, G. SARTOR, *Pacta Sunt Servanda: Smart Legal Contracts in Stipula,* 2021, p. 2.

3.1 DISTILLING SOME COMMON PATTERNS OF LEGAL CONTRACTS

The strength of *Stipula* as a programming language derives from the "chemical" approach it is based on. Any contract can be broken down in multiple elements, each carrying its legal scope. If we were to deconstruct different contracts, we would be able to identify patterns common to every agreement (e.g., obligations): just like every organic body can be broken down in carbon atoms. Similarly, *Stipula* identifies the simple legal particles of a contract and translate them into code one piece at a time, unit by unit. This *modus operandi* makes the drafting of a smart legal contract much easier for lawyers and for non-IT professionists. Writing a contract would consist in identifying the legal elements of an agreement, finding the respective *Stipula* primitives, and assemble them in the correct order. Starting from the simplest contract consisting of two or three elements (e.g., a bet: "If it's tails, I'm giving you 10 euros"), different patterns can be added and assembled in order to create increasingly complex contracts.

From the analysis of quite common contracts like the bet, the free rent, and a license contract, we were able to identify some basic patterns that we can sum up as follows:

- Obligations: for the purpose of this work, this has not to be intended as the whole legal relationship deriving from the contract (*obligatio*), but rather as the specific legal position of one party who committed to perform a certain action[199];
- Permissions and prohibitions: when a certain behaviour by a party is allowed or forbidden by the contract;
- Constitutive effect: the contract creates a new right assigned to one of the parties;
- Translative effect: a right is transferred by one party to the other;
- Aleatory event: a risk consciously undertaken by the parties, which is connected to the verification of an uncertain event outside of the parties' control. It usually is a future event, but it may also concern an already occurred

[199] F. ROMANO, *Obbligo [XXIX,1979],* in *Enciclopedia del Diritto,* Giuffrè Francis Lefebvre, 1979.

event and only the acknowledgment of its outcome is projected into the future. Different outcomes can produce favourable or unfavourable effects on the parties[200];

- Negotiation: the moment preceding the contract where parties set and agree on its contents;
- Meeting of the minds: the moment when both parties express consent on the terms of the agreement and the contract produces its legal effects;
- Real-world data retrieval: contracts' execution is often subordinated to the occurrence of real-world events that make performances become due. In such cases parties must agree on a data source to retrieve information from.

3.2 TRANSALTING PATTERNS INTO CODE

This section puts forward the programming solutions adopted in *Stipula* to encode the single legal patterns identified through the above analysis.

3.2.1 States

"*Stipula* is committed to a state-aware programming style"[201]. The *State machine Design pattern* is a traditional programming model in computer science. This technique allows for the breakdown of an entire program's complex task into multiple smaller tasks. Dividing a software into multiple units helps managing the order in which software tasks have to be performed[202]. By doing so, we are able to enforce the intended behaviour by allowing or precluding the call of specific functions depending on the current state of the program. To exemplify, when in state Q of a program, only functions *f* and *f'* can be called, while to call function *f'''* the software must transit to state Q'' first.

In the code, states' names are indicated by the "@" before the word (e.g., "@Inactive")

[200] N. ROSARIO, *Alea [I,1958]*, in *Enciclopedia del Diritto,* Giuffrè Francis Lefebvre, 1958.
[201] S. CRAFA, C. LANEVE, G. SARTOR, *Pacta Sunt Servanda: Smart Legal Contracts in Stipula,* 2021, p. 1
[202] K. NATH, *State Machine Design pattern - Part 1: When, Why & How*, www.medium.com, 2019.

In smart legal contracts, *states* represent the stages of contract's life. A contract, for instance, can be inactive, in the offer, acceptance, execution or terminated stages.

Stipula uses states to implement prohibition or permission patterns. In a bike rental, once the borrower has legitimately initiated the contract, the lender is prohibited form preventing the usage of the bike until the agreed time comes up. The contract state will not allow the call of the function that sends the bike token back to the lender. Moreover, *Stipula* states are abstract enough to be easily identifiable by lawyers and to allow an effective management when building a contract.

3.2.2 Agreement Constructor

This is the initial stage of every *Stipula* contract and basically represents the negotiation stage. This is a crucial element for any smart contract because it lays down the terms according to which the contract will have to operate.

The *agreement* indicates, in the first place, who are the subjects participating in it. After it, one party propose the terms she agrees on (=SET=) and the other is able to express her consent on those terms (=OK=). Contracts can present two or multiple parties, but no specific order is required for setting and accepting its elements. When the contract also requires the presence of an oracle, such an entity may be called to express its consent on certain terms and, therefore, included in the agreement. The subjects to the agreement only set or agree on terms by which they are affected, not all of them. Note that only field values are set here and not assets (see *infra*-2.3.4).

When this stage is concluded, the smart contract is set to the initial *state.* This moment represents the conclusion of contract's drafting where the whole contract body is complete. However, parties' still need to express their consent on the whole body of contractual obligations in order for the legal bond to be established (signing the contract). Such "meeting of the minds" is implemented into the smart contract by calling specific functions that represents the moments of "offer" and "acceptance" of contract's formation and the consent will be expressed *facta concludentia*.

3.2.3 Events

Events are sequences of timed continuations that can be included in the body of a function. They are employed to schedule the execution of future operations if specific preconditions are met. Usually, contract states are used as preconditions. Events consist of a timeout, which is initiated when the function is called, a precondition (*state*) and the consequential operation that may be triggered (a *statement*). According to the formula *E >> @Q {S} → @Q'* if the timer expires with the contract state still being @Q, then the statement S is executed, and the contract's state is switched to @Q'.

Events can be employed to automatically enforce obligations. The operations included in the statement can represent either the automated performance or the penalty for the non-performance which is automatically issued by the smart contract. The timeout expiry coincides with the moment where the obligation become due. For example, in the free bike rental, when the timer expires and the borrower has not returned the bike yet, an event is triggered that automatically sends back to the lender the token that allows the bike usage.

3.2.4 Assets and Fields

A peculiar feature of *Stipula* is the dichotomy between *asset* and *field* values. These language components shall be thought at as locations where contract's data are stored. The difference between them is given by the kind of data held. Fields contain simple data values that are updated at will by the parties in the `agreement` phase. They can concern, for instance, time limits, costs, or a good identification number. These values are updated in the *agreement* constructor and agreed upon by the parties.

On the other hand, assets values indicate linear resources. They represent the physical or virtual goods managed by the smart contract like money, a personal right or a token representing a good. The key difference compared to field values is that assets must respect a total supply, they cannot be duplicated or leaked: when they are sent to a new

address, the location previously holding the value is emptied[203]. The same does not apply to fields (when a field value is updated to the contract, no previous location exists, or needs to be emptied). This design choice promotes a safer programming discipline that reduces the risk of double-spending, accidental loss, or the locked-in assets[204]. Assets generally contain goods in token form: either a divisible token representing a sum of money or an indivisible token representing a non-fungible good.

3.2.5 Lollypop and arrow operators

The lollypop (⊸) and arrow (→) symbols are two operators used to express assignments. They are employed to differentiate updates of assets and of fields, respectively. The lollypop expresses an asset's update. The point of having a different operator for assets is to clearly spot the operations where the moving of a resource entails the emptying of the previous location. Assets, as mentioned above, must preserve a total supply. When a value "E" of a linear resource (e.g., a sum of money or a non-fungible token) is moved from inside the smart contract (asset "h") to party B's address, the lollypop expresses the operation as follows: "E ⊸ h, B". Such formulation indicates that the asset "E" is subtracted from field "h" and sent to party "B". In the following smart contracts examples this formulation will be abbreviated when the value sent and the previous location are indicated with the same letter. So, for example, the operation "h ⊸ h, A", representing the transfer of value "h" from location "h" to location "A", would be abbreviated in "h ⊸ A".

The arrow (→), instead, is used to indicate a simple data sending. For example, the operation *"time's up → B"* means that the message "time's up" was sent to location B, which represents one party's address. As it is evident, there's no need to indicate the previous location since there is no actual transfer of value but rather the creation and upload of mere data.

---

[203] S. CRAFA, C. LANEVE, G. SARTOR, *Pacta Sunt Servanda: Smart Legal Contracts in Stipula,* 2021, p. 8

[204] Ibidem.

On the legal side, when these operators are used to manage goods and rights object to the contract, they have different legal meanings.

When a token is transferred through a lollypop it means that one party of the agreement gives in the exclusive control of the token either to another party or to the smart contract itself. Tokens generally represent goods or rights and therefore, the lollypop here shows a right or good transfer. For this reason, we assert that the lollypop operator indicates a translative effect. The right (or good) moved pre-existed in another location and is now held by a new entity. At the same time the lollypop transfer ensures that no one will tamper with the enjoyment of that right, since it was completely removed from the previous location. This also gives a "hamper-proof" feature to the rights transferred through such operator. Consequently, the lollypop fits perfectly for representing the transfer of ownership rights. In fact, ownership always entails a precedent holder and needs to be exclusive to be fully effective.

By contrast, the arrow indicates a constitutive effect, i.e., it establishes a new right not existing previously. The arrow has been employed to represent the creation of a usage right on a certain good through the creation of a code (password) that allows the access to the good. This operation is represented in *Stipula* with the formula "`uses(t) → A`" where a usage code for token "t" (`uses(t)`) is sent to party `A`'s address.

The validity of such code, however, is controlled by the original token, meaning that the token holder will have influence on the right exercise. So, by contrast to the lollypop, the arrow does not provide any exclusiveness feature since the token was not moved to another location and still remains in custody of its holder. A solution is to place the token inside the smart contract so that one party cannot interfere with the other's right (though this solution cannot be permanent).

3.2.6 Authority

Stipula contracts may be drafted as to include a trusted third party in the agreement which is conventionally called *Authority*. The authority plays an oracle's role, providing real-world data into the smart contract. Its tasks may vary depending on the

contract's necessities. It can act as an automated oracle[205] (e.g., a data provider), whose role could simply be to collect data and feed it into the smart contract . In this case, the choice of a data source from which information have to be collected by the provider, is crucial for the parties. Once agreed upon, they will be bound by the information released by that data source since the smart contract is automatically enforced. An authority, for example, is always necessary in aleatory contracts like a bet or an insurance contract, whose execution depends on the verification of off-chain events. A trusted entity is needed to collect the necessary information on the aleatory event and feed them into the contract.

On the other hand, the Authority can also act as an expert oracle and be assigned with more complex tasks that are not achievable on-chain. Either because they involve an evaluative process, or because they consist of making assessment on real-world situations. For example, an authority can be charged to check on the conformity of a purchased good to what has been agreed on, or to make assessments on vague terms of the contract such as "*force majeure*" or "due diligence" clauses. Moreover, the authority is a good solution to integrate in the contract a trusted dispute resolution mechanism.

At the code level, the authority is initially included in the agreement constructor where it can be required to set or agree to some contract's terms, such as its fee or the time limit within which it must perform. The authority is enabled to interact with the smart contract by the provision of *ad hoc* functions that can be called to provide data.

3.3 *STIPULA* CONTRACTS IN PRACTICE

This section lays down a demonstrative implementation of three simple contracts using the *Stipula* language: a free rent contract (free loan for use), a digital license contract, and a bet contract. The analysis will be brought forward by initially identifying the simple legal patterns composing the contract; we will then proceed by putting forward

[205] Cfr. Sec. 1.6.

and describing step by step the *Stipula* code implementing the contract and conclude the analysis making some technical and legal considerations.

3.3.1 The Free Rent Contract

The free rent contract can be assimilated to the "comodato d'uso gratuito" under the Italian legal system, regulated by artt. 1803-1812 of Italian civil code. The contract involves two parties, a lender and a borrower. The former party hands over to the latter a movable or immovable good for it to be used for an agreed purpose. The borrower commits to return the good after he used it according to the contract, or at the end of an agreed time laps[206]. The rent is free of charge[207]. The borrower also commits to a few side duties: the obligations of diligent storage and care of the good, not granting the usage to a third party without the lender's consent and the obligation of using the good only as intended in the agreement[208].

The civil code puts forward some dispositions concerning the borrower's liability in case of destruction of the good. Article 1805 establishes the borrower's liability even when the destruction is due to causes not attributable to him, where he used the good not as intended in the agreement, or for a longer time than the one agreed upon[209]. The liability stands also in cases of unforeseeable events, if the borrower could have saved the good from destruction by replacing it with its own[210]. Moreover, according to article 1806, where the parties estimated the value of the good including it in the agreement, the borrower is always liable for the destruction.

Article 1812 c.c. provides for the lender's liability for damages caused to the borrower by defects of the good, in case he was aware of those and didn't warn the borrower.

Article 1808 c.c. provides that the lender must refund the costs incurred in by the borrower for the preservation of the good, if those were urgent and indispensable.

---

[206] Art. 1809 c.c.
[207] Art. 1803 c.c.: "..Il comodato è essenzialmente gratuito".
[208] Art. 1804 c.c.
[209] Art. 1805, c.2 c.c.
[210] Art. 1805, c.1 c.c.

The contract is terminated either when the good is returned, when the time expires, or if the borrower violates one of his side duties, as the lender can demand the immediate return of the good plus a compensation for damage. However, article 1809 provides that the lender may demand the return of the good even before the time or the moment agreed upon by the parties, should he or she have an urgent and unforeseeable need[211].
Under the Italian legal system, the free rent contract is considered a "real" contract ("contratto reale") meaning that the handover of the good (lat. *Res:* thing) is required to trigger the legal bond. Before that, no obligation arises. Moreover, it only produces binding effects, i.e., it only creates obligations upon the parties without any transfer of real rights (such as property). The category of "real" contracts stands against the one of consensual contract ("contratti consensuali") where, by contrast, obligations become effective (thus, the legal bond arises) with the simple meeting of the minds. This formal distinction has its roots in roman contract law and stems from the choice, by the legal system, of protecting one party's interest in the performance more than the other[212]. So, for example, in the free rent contract, in the moment of consent, the borrower is not empowered to enforce the lender's performance (good's delivery); by contrast, after the delivery, the lender has legal force to demand the return of the good. The opposition between real and consensual contracts become worthless when implementing smart contracts. As we will see, both the moment of the meeting of the minds and the moment of good delivery basically occur at the same time, within the execution of the same function. Consent, in fact, is expressed *facta concludentia* which means already performing relevant actions.
Although civil code's provisions do not show any real consideration between parties' obligations (e.g., lender's side obligations provided by artt. 1808-1812 do not stand at the same level of the borrower's obligation to return the good), we can at least assert

[211] Art. 1809, c.2 c.c.
[212] N. ZORZI GALGANO, *L'Effetto Traslativo del Contratto,* in F. GALGANO (edited by), *Trattato di Diritto Civile,* Padova, CEDAM, 2014, p. 337.

that a negative obligation to abstain from hindering the usage of the good exists on the lender's side[213].

According to art. 1803, the borrower must return the *same* thing (*idem res*). This means that the good must be non-fungible and non-consumable[214]. This can be easily represented in the contract by a non-fungible token associated with the good.

3.3.1.1 Patterns

The general lifecycle of the free rent contract can be outlined as follows:

1. The contract is arranged, and the following terms are set: the good and the intended use (i), the time limit or other return instances (ii); possible esteem of the value (iii);
2. Lender agrees;
3. Borrower agrees;
4. The good is handed to the borrower: the legal bond is effective, the obligations become due:
    a. The borrower is allowed to use the good;
    b. The lender must refrain from precluding the usage;
    c. The borrower is obliged to return the good at the end of the time limit or after he used it according to the contract (1803);
    d. The borrower is committed to the fulfilment of the side duties (art. 1804);
5. Termination:
    a. The borrower returns the good before the deadline;
    b. The borrower completes the agreed usage, or the time expires and he must return the good;

[213] L. PELLEGRINI, *Capo XIV: del Comodato*, in D. VALENTINO (edited by), *Commentario del Codice civile – Dei Singoli Contratti – Volume III,* Torino, UTET Giuridica, p. 53.
[214] L. PELLEGRINI, *Capo XIV: del Comodato*, in D. VALENTINO (edited by), *Commentario del Codice civile – Dei Singoli Contratti – Volume III,* Torino, UTET Giuridica, p.39.

c. The lender demands the immediate return of the good for an urgent and unforeseeable need (1809);

6. Violations or problems:
    a. The borrower violates the side duties: the contract is terminated, and the lender can ask for compensation (1804);
    b. Defects known by the lender cause damage to the borrower: the lender must compensate the damages (1812);
    c. The borrower incurs in urgent and indispensable costs for the preservation of the good: the lender must refund him (1808).
7. Destruction:
    a. The good was esteemed in the agreement: borrower is always liable (1806).
    b. The borrower had used the good not as intended (i) or for a longer time than the intended one (ii): he is always liable unless he proves that the good would have perished even if he had abided to the terms of the contract (1805);
    c. The good could have been saved if the borrower substituted it with its own: he is liable even in unforeseeable circumstances (1805).

Through the analysis of this contract, we can identify the following legal patterns:

- **Meeting of the minds**: parties agree to the contract terms (point 1);
- **Negotiation**: parties define the content of contract's terms;
- **Creation of a new right** on the good, namely, usage right(point 4);
- Borrower is **permitted** to use the good (point 4.a);
- Lender's **prohibition** on precluding the usage by the borrower (point 4.b);
- Borrower's **obligation** to return the same good (point 4.c);
- Borrower's side **obligations** of diligent care, not granting the usage to a third party without the lender's consent and intended usage (point 4.d);
- **Liabilities** both upon the lender and the borrower for several situations (point 7.a, 7.b, 6.a, 6.b). Liability always gives rise to an obligation to compensate.

In some situation the liability can be assessed objectively (point 7.a) and in others it requires an evaluative activity (points 7.b, 6.a, 6.b).

3.3.1.2 Code

Here we propose an implementation of a free rent contract using the *Stipula* language. The purpose is to show how contract clauses can be implemented in code starting from the simplest contract models. For this reason, the code sticks to a basic form of free rent without considering the whole body of provisions provided by the civil code. However, the possible implementation of such provisions will be addressed in the following subparagraph.

This specific contract concerns a free rent contract for a locker for a predefined amount of time, at the end of which the good (locker's "key") is returned to the lender. The borrower may also return those before the time expiry, thus terminating the contract earlier.

```
1  legal_contract Free_Rent {
2    assets token
3    fields numBox, t_start, t_limit
4
5    agreement (Lender, Borrower) {
6         Lender =SET=> t_start, t_limit
7         Borrower =OK=> t_start, t_limit
8    } ⇒ @Inactive
9
10   @Inactive Lender :  boxProposal (n)[t] {
11        t —o token
12        n → numBox
13        now + t_start ≫ @Proposal {
14              token —o Lender } ⇒ @End
15   } ⇒ @Proposal
16
17   @Proposal Borrower :  boxUse {
18        (uses(token), numBox) → Borrower
19        now + t_limit  ≫ @Using {
20              "Time_Limit_Reached" → Borrower
21                token —o Lender
22         } ⇒ @End
23    } ⇒ @Using
24
25    @Using Borrower : returnBox {
26          token —o Lender
27    } ⇒ @End
28 }
```

Lines 1 to 3: we have the general heading of the contract. Line 1 shows that the contract is a "free rent" and lines 2 and 3 clearly enlists the components of the contract indicating which ones are assets and which fields. The only asset of this contract is "token", which virtually represents the locker and that provides full control on it. The fields are the locker's identification number ("numBox"), the time within which the borrower must accept the rent proposal by the lender before the contract is terminated ("t_start"), and the duration of the rent ("t_limit")

Lines 5 to 8: the agreement constructor. Here, parties agree on the terms of the contract. In this case the only terms that have to be agreed-upon are the time start and the time limit (we assume that the choice of the specific locker is indifferent to the borrower). The lender sets the fields (=SET=), and the borrower agrees on those (=OK=). Once the agreement is reached, the contract drafting is completed, but parties still express their consent for the agreement to be binding. That is why, after this operation, contract's state is set to "@Inactive" (line 8).

Lines 10 to 15: "boxProposal" function. The first words shown in line 10 before the function's name indicate that this function can be only called when the contract's state is "@Inactive" and only by the lender. This function involves two elements: the token that represents the locker "[t]" and its identification number "(n)". The different type of brackets used indicates the different kind of value they contain: square brackets contain asset values (i.e., linear resources) and round brackets simple field values (simple data). By calling this function the lender sends both elements to the smart contract's address: "t" is sent in the contract's asset "token" (line 11), and "n" to field "numBox" (line 12). Note that the token is transferred through the lollypop operator, meaning that it was subtracted from the previous location (lender's address). In particular, line 11 is an abbreviation for "t ⊸ t, token" (*cfr*. 2.3.5). Lines 13-14 express an event: a timer is triggered and, if the limit expires ("t_start") with the contract's state still being "@Proposal", then the token is sent back to the lender and the contract

is terminated (@End state). After the three operations are executed (transfer of [t], (n) and the timeout triggered) the contract's state is switched to "@Proposal".

Lines 17 to 23: "boxUse" function. In line 17 we can see that this can only be called with the contract being in the "@Proposal" state and only by the borrower. By calling this function the borrower expresses the acceptance to lender's proposal. He receives the locker's identification number and a temporary code that allows the access to it (line 18). In lines 19-23 we have another timeout that triggers an event. If "t_limit" is reached with the contract state still being in "@Using" (line 19), the borrower receives a message saying "time limit reached" (line 20) and the token representing the locker is sent back from the smart contract to the lender's address. By doing so, the lender gets back the full control on the good (line 21), and the contract is terminated by switching to "@End" state (line 22).
Calling this function switches the contract state to "@Using".

Lines 25 to 27: "returnBox" function. This allows the borrower to return the object even before the time limit is reached. It can only be called by him and in contract's state "@Using" (line 25). The function simply sends the token back to the lender (line 26) and terminates the smart contract changing the state to "@end" (line 27).

3.3.3.3 Legal Considerations

By comparing the code to the contract scheme outlined in section 2.3.1.3 we can better evaluate *Stipula* effectiveness in implementing the legal contract in a way that automatically enforces its premises.
Point 1 represents the negotiation stage. It could occur off-chain between two equal parties, or as an adhesion contract, where the stronger party unilaterally sets all the clauses and the weaker one can only decide to accept or refuse it. In both cases, once terms have been set, one party would upload them into the smart contract through the agreement constructor. Hypothetically, the code could also integrate the possibility to

of a negotiation phase: where parties are not stuck with the roles of proponent and acceptor of a particular term, but rather with the possibility to counter propose a different term in case the previous one didn't match one's interests. This possibility is not provided in the current version of *Stipula* but it could be easily implemented. Once the agreement operations are executed, contract's drafting is concluded.

Points 2 and **3** express the meeting of the minds and this is effectively implemented in the code by the functions "`boxProposal`" and "`boxUse`". The two functions represent, respectively, the offer and acceptance stages of contract formation. The calling of "`boxUse`" function unleashes contract's effect and parties become bound to each other. The borrower acquires a right to use the object for a determined amount of time, at the end of which he has committed to return it. Employing an arrow to represent the right's acquisition (line 18) indicates that a new right (not pre-existing) has been created and conferred to the borrower. Once the code has been received, the borrower is in the position to enjoy the right as described in **point 4.a**.

Point 4.b is enforced through the "`boxProposal`" function through which the lender sends the token to the smart contract's asset (line 11). By using the lollypop operator, the token is fully removed from lender's possession and only exists inside the smart contract. Lender loses the control on the object and cannot preclude the borrower from enjoying his right.

The obligation to return the object at the end of the agreed time outlined by **point 4.c** is enforced by the event in lines 19-23. If the timeout expires and the borrower has not returned the good yet (i.e., the contract's state is still "`@Using`" and not "`@End`"), line 21 shows that the token currently held by the smart contract is sent back to the lender, thus making him capable of invalidating the usage code.

Point 4.d involves the side duties of diligent care, using the good only as intended and not granting the use to a third parties without the lender's consent. Such obligations require both a monitoring activity on the object in the real world and an evaluative activity (assessment on the diligent care). Evidently, such issues need to be dealt with off-chain and, for simplicity reasons, were not implemented in the above code.

However, a solution could be found by including a trusted third party (in *Stipula*, "Authority") in the smart contract. This could be an individual charged to monitor the fulfilment of the obligations in the real world and be empowered to call specific contract's functions triggering, for example, a compensation, or the contract termination.

Termination instances **5.a** and **5.b** are effectively represented in the code. On one hand, 5.a is implemented through the "`returnBox`" function in lines 25-27 that the borrower can freely chose to call. The token is sent back to the lender and contract's state is switched to "`@End`". On the other hand, **5.b** simply represents the conclusion of the event triggered by the "`useBox`" function where the conditions are met (lines 13-14). The termination cause provided by article 1809 c.c. (**point 5.c**), instead, is not taken into account by this smart contract. It would require an off-chain assessment on the urgent and unforeseeable character of the need by the lender, in order for it to be exercised legitimately. Such instance could only be implemented through an authority able to verify that need.

Points 6.a, 6.b and **6.c** follow the same principle. All situations require the monitoring of the real world and an evaluative activity impossible to achieve on-chain. The only possible solution is an oracle-Authority.

The liabilities for the destruction of the good follow separated paths. **Point 7.a** is not considered in the present code but is potentially implementable through the help of IoT devices like sensors. A deposit as a grant for the integrity of the good should be included in the agreement and held in custody by the smart contract. Once the object has been esteemed the borrower bears an absolute liability that can be assessed objectively. Placing sensors around the object can enable the detection of the good's destruction and in that case the deposit held in the smart contract would be automatically sent to the lender. For **point 7.a.i** and **7.b** we follow the same pattern as point 6. **7.a.ii**, instead, is precluded by the very nature of smart contracts. It is a techno-forbidden action. At the end of the timeout the token will automatically be returned by the smart contract with no chance of violating the time limit term.

3.3.2 The License Agreement

The second example we put forward is a license agreement for the use of a digital service, like a software. This contract is not directly regulated by law and shall fall under the umbrella of atypical contracts[215]. In a license agreement, the holder of an author's right on an intellectual work (licensor) grants to the licensee the right of usage of such work (or a copy of it) for a limited period, upon the payment of a sum of money. It is hard to identify a clear legal classification for this contract, as academics split in two different lines of thought. Some academics bring the license closer to a rent contract highlighting the usage rights (and not full ownership rights) that both contracts confer[216]. Others assimilate the license to a purchase contract leveraging on the fact that the consideration corresponds to the value of the object and not correlated to the usage *quantum*[217].

This contract is peculiar to intellectual property because it allows the right holder to make profit out of his or her own work, without exhausting his distribution right. The exhaustion principle was originally established at European level by Directive 91/250 on the legal protection of computer programs and implemented by Italian law in article 64-bis l. n. 63/1991. It provides that "The first sale in the Community of a copy of a program by the right holder or with his consent, shall exhaust the distribution right within the Community of that copy, with the exception of the right to control further rental of the program or a copy thereof.[218]. We can now understand why it is so important for software producers to clearly specify when they are selling a product or just granting a license.

A landmark decision on the issue was taken by the EU Court of Justice in the Usedsoft Gmbh vs Oracle International corp. In the judgement, the court established that the

[215] P. E. SAMMARCO, *I Contratti dell'Informatica,* in N. LIPARI, P. RESCIGNO (edited by), *Diritto Civile, Volume III – Obbligazioni, III - I Contratti,* Milano, Giuffrè Editore, 2009, p.781.
[216] Ibidem, p.779
[217] M. CARTELLA, *La Licenza d'Uso,* in V. ROPPO (edited by), *Trattato dei Contratti – III – Cessione e Uso dei Beni,* Milano, Giuffrè Editore, 2014, p.815.
[218] Art. 4 (c), CEE Council Directive n. 91/250.

distribution right for a copy of a computer program is exhausted if the copyright holder granted a right to use that copy for an "unlimited period" and in return for payment of a fee intended as a remuneration corresponding to the value of the product itself (and not commensurate with the time of usage)[219]. This principle stands regardless of the label used to name an agreement: if the conditions laid down by the court are satisfied, it will be considered as a sale for the purposes of the distribution right[220].

3.3.2.1 Elements

For the purposes of the present work, we considered a basic license agreement for a digital product review, that provides for a free evaluation period granted to the licensee, at the end of which the license would be withdrawn if it hasn't been bought yet. In addition, both parties commit to keep a specific behaviour by contract. A drafted version of this contract would take the following form:

> **Article 1**. Licensor grants Licensee with a licence to evaluate the Product and fixes (i) the evaluation period and (ii) the cost of the Product if Licensee will buy it.
> **Article 2**. Licensee will pay the Product in advance; he will be reimbursed if the Product will not be bought with an explicit communication within the evaluation period. The refund will be the 90% of the cost because the 10% is paid to the Authority (see Article 3).
> **Article 3**. Licensee must not publish the results of the evaluation during the evaluation period and Licensor must reply within 10 hours to the queries of Licensee related to the Product; this is supervised by an Authority that may interrupt the licence and reimburse either Licensor or Licensee

[219] CJEU, case C-128/11, 07/03/2012.
[220] M. CARTELLA, *Contratti su Programmi per Elaboratori,* in V. ROPPO (edited by), *Trattato dei Contratti – III – Cessione e Uso dei Beni,* Milano, Giuffrè Editore, 2014, p.809.

according to whom breaches this agreement.

Article 4. This license will terminate automatically at the end of the evaluation period if the licensee does not buy the product.

The contract lifecycle can be schematised as follows:

1. The contract is arranged, and the following terms are set: the evaluation period duration (i); permanent license cost (ii); the supervisory authority (iii); authority's fee (iv);
2. The licensee offers;
3. Licensee accepts. The legal bond arises:
    a. Licensee has to deposit the escrow;
    b. Licensor must send the temporary license;
    c. Licensee has a limited time (i) to use it freely;
    d. Licensee must not publish the review before the evaluation period ends;
    e. Licensor is obliged to answer the queries within 10 hours;
4. Licensee buys the license within the evaluation period:
    a. Licensor must send the permanent license to licensee
    b. The escrow is sent to the licensor
5. Termination:
    a. The proposal is not accepted before the deadline
    b. Licensee doesn't buy within the evaluation period
    c. Licensee buys the license
6. Violations:
    a. Licensee publishes before the evaluation period: the escrow is sent to licensor, the temporary license is withdrawn and the contract is terminated;
    b. Licensor doesn't answer within the 10 hours: the escrow is sent back to Licensee, temporary license is withdrawn and the contract is terminated.

Like in the previous example, we can identify some legal patterns:

- Contract's **negotiation** and **drafting;**
- A **meeting of the minds:** parties agree on the terms of the agreement;
- **Creation of a new right**: the temporary license right;
- **Transfer of a pre-existing right:** the permanent license right;
- **Escrow;**
- **Obligation** to grant the temporary license on licensor;
- **Obligation** to send the permanent license on licensor;
- **Permission** to use the product;
- **Prohibition** to use the product at the end of the evaluation period if not bought;
- **Obligation** to answer within 10 hours on licensor;
- **Obligation** not to publish during the evaluation period on licensee.

3.3.2.2 Code

Compared to the free rent example, this contract presents a few differences. It holds two assets, an indivisible token that represents the license, and another one, a balance, which corresponds to the amount of currency escrowed kept in custody inside the smart contract. Another difference is that the agreement also includes an authority put in charge of monitoring and sanction the correct performance of off-chain obligations.

```
1  legal_contract Licence {
2    assets token, balance
3    fields cost, t_start, t_limit
4
5    agreement (Licensor,Licensee,Authority) {
6      Licensor =SET=> cost, t_start, t_limit
7      Licensee =OK=> cost, t_start, t_limit
8      Authority =OK=> _
9    } ⇒ @Inactive
10
11   @Inactive Licensor : offerLicence [t] {
12     t —o token
13     now + t_start ≫ @Proposal {
14                 token —o Licensor } ⇒ @End
15   } ⇒ @Proposal
16
17   @Proposal Licensee : activateLicence [b]
18     (b == cost){
19        b —o balance
20        balance*0,1 —o balance, Authority
21        uses(token,Licensee) ⊸ Licensee
22        now + t_limit ≫ @Trial {
23                        balance —o Licensee
24                        token —o Licensor
25                      } ⇒ @End
26   } ⇒ @Trial
27
28   @Trial Licensee : buy {
29     balance —o Licensor
30     token —o Licensee
31   } ⇒ @End
32
33   @Trial Authority : compensateLicensor {
34     balance —o Licensor
35     token —o Licensor
36   } ⇒@End
37
38   @Trial Authority : compensateLicensee {
39     balance —o Licensee
40     token —o Licensor;
41   } ⇒ @End
42 }
```

Lines 1 to 3: the smart contract indicates what are the fields and the assets. As mentioned already, we have two assets, a token and a balance. The fields outline the license cost, the deadline for the acceptance of the license contract proposal (“t_start”) and the evaluation period duration (“t_limit”).

Lines 5 to 9: the agreement constructor. Licensor sets fields values and licensee express his consent. This agreement also includes a third party, “Authority”, which

does not specifically agree on anything (empty statement "_") but it is important to include it, since it plays an important role in contract's life by calling the functions to compensate parties. Moreover, it would be plausible to call the Authority to set at least the fee that parties owe.
After the agreement, the contract state is set to "@Inactive" (line 9).

Lines 11 to 15: "offerLicense" function which can be called by the licensor when the contract is "@Inactive". Licensor sends the token representing the license [t] to smart contract's asset "token" (line 12). At the same time, an event is triggered. If at the expiry of "t_start" the contract is still in the "@Proposal" state (line 13), it will be terminated by sending the token back to licensor (line 14).
The contract state switches to "@Proposal" (line 15).

Lines 17 to 26: "activateLicense" function. The licensee decides to activate the contract by sending the amount of money agreed upon in the agreement (b), to the contract's asset "balance" in escrow (line 19). The contract then sends him back a use code generated by the token that would grant the temporary access to the software (line 21). Note that, in line 20, a 10 % of the total sum escrowed is sent from the asset "balance" to the Authority as a fee.
The event showed in lines 22 to 25 enforces the prohibition to use the product after the evaluation period, where not bought yet. If after "t_limit" expires the contract is still in "@Trial" state (line 22), then it will be terminated by sending the balance left in the smart contract back to the licensee (line 23), and the token back to the licensor (line 24).
State is switched to "@Trial".

Lines 28 to 31: "buy" function. Licensee chooses to buy the license. The amount of money escrowed in the "balance" is sent to the licensor (line 29) and the licensee

receives the token representing the license (line 30). With the purchase, the contract finds one of its physiological termination causes and its state is set to "@End".

Lines 33 to 36: "compensate licensor" function. As showed in line 33 this can only be called by the authority and when the contract is in "@Trial" state. With this function licensor receives a compensation in case the licensee violated the obligation not to publish the results of the review during the evaluation period, which the authority is charged to monitor. If the licensee breaches the contract, the money left in the balance are sent to the licensor as compensation together with the token representing the license.
The contract is terminated switching to "@End" state (line 36)

Lines 38 to 41: "compensate licensee" function. By contrast, this represents the penalty for the breach of licensor's obligation. If he does not answer to the queries within ten hours, the authority may decide to terminate the contract (line 41) by sending the balance back to licensee (line 39) and the token back to the licensor (line 40).
The contract is terminated switching to "@End" state (line 41).

3.3.2.3 Legal Considerations

As to **points 1**, **2** and **3** of the above scheme, the considerations laid out for the previous contract also stand for this example. Specifically, here the offer is represented by the "offerLicense" function, and "activateLicense" stands for an acceptance by conduct. The licensee, by depositing an escrow to the smart contract, expresses his consent to the license contract and unleashes contract's legal effects (**point 3)**. The token representing the license is placed in the smart contract so that, in case of acceptance by the licensee, the license will be provided automatically without the need for licensor's performance. In this moment, the obligations enlisted at letters 3.a, 3.b and 3.c become due, and the smart contract can enforce them automatically. **Point 3.a**

basically represents a prerequisite for calling "activateLicense" and is implemented in line 19 where a sum of money [b], whose value must be equal to the "cost" field laid down in the agreement (line 18), is sent by licensee to the smart contract's asset "balance" . The obligation of **point 3.b** is automatically enforced in line 21. This is possible thanks to "offerLicense" function, where licensor sends the token representing the license to the smart contract (line 12). By doing so, licensee doesn't need to rely on licensor's will to receive the use code and this is, instead, created and sent automatically by the smart contract. At this stage, only a temporary usage code is transferred and not the token itself. The temporary nature of the right is transposed in code by the event scheduled in lines 22, 23, 24. When the time limit expires without the licensee having bought the license (i.e., the contract is still in "@Trial" state), the token is sent back to the licensor, empowering him to change and control the temporary use code.

In **points 3.d** and **3.e** parties commit to keep a specific behaviour off-chain. Since the enforcement of such obligations cannot be achieved automatically on-chain, a trusted third party, the Authority, is included in the agreement, charged to monitor the correctness of performance, and issue, where due, the penalties provided by the contract. The smart contract provides two functions that the Authority can use to compensate either licensor or licensee ("compensateLicensor", "compensateLicensee"). Since the obligations that need to be monitored are clearly and objectively verifiable, the Authority could also be represented by an automated oracle[221]. For example, a software connected to the mailbox, could check how long it takes to the licensor to answer licensee's queries and be enabled to call the compensation when the time reaches the limit. The same goes for licensee's obligation: in this case the software should be connected to the major review platforms or websites and be able to detect the reviews representing a breach of contract terms. The complexity and extent to which software can perform this kind of monitoring activity

[221] See sec. 1.6.

depends, of course, from the level of technology employed and, more generally, form the currently available technology.

Point 4 is implemented with the function "buy" (lines 28-31). The Licensor receives the money (**4.a**) escrowed by the smart contract through the "activateLicense" function, and the token representing the license is transferred to the licensee in line 30 (**4.b**). These operations occur automatically because both the token and the money are held by the smart contract.

The lollypop operator indicates a translative effect, i.e., that a pre-existing right is transferred to a different holder who is now in full control of his or her right. Because of the particular nature of license as a legal construct, though, it might be difficult to identify a real translative effect, in that the licensor can basically create as many licenses as he wants and doesn't seem like really losing any kind of control on his product (the software). However, as mentioned in the above section, when a license is granted for an indeterminate time, and in return of a fee corresponding to the product's value as a whole, the licensor exhausts the distribution right on that specific copy, thus conferring in turn to the licensee the same distribution right. The licensee gets the power to subsequently alienate the license. That is the object of the translative effect represented by the lollypop, the distribution right. Moreover, once the token is in licensee's custody, he has the guarantee that no one will hamper with the exercise of his right.

Point 5 shows the causes for termination of the contract. In this example, the contract is terminated when enters in state "@End". The @End state is automatically entered into by the contract if the deadline for activating the license expires (line 14); if the time limit expires before licensee has bought the license (line 25); after the licensee buys the product through "buy" function (line 31); and in case of breach of contract, after the Authority has issued the compensation (lines 36, 41).

As already outlined, the violations described by points **6.a** and **6.b** are implemented through the Authority. With "compensateLicensor" and "compensateLicensee" functions, the authority can compensate the licensor with the money escrowed in the

contract in case of breach of the licensee, or terminate the contract returning the money back to the licensee in case of breach by the licensor. Therefore, also the whole point 6 is fully implemented by the smart contract.

3.3.3 The bet

The third example we put forward is the bet, an aleatory contract. The concept of *Alea* refers to a risk that parties agree to take on by contract, that may affect the original balance in contract's consideration. The risk is usually linked to the verification of a future uncertain event or, if the event already occurred, to the acknowledgment of that verification[222]. By contrast, in normal contracts the relation between parties' performances is known and defined *ab origine*.

When looked closely, every agreement entails a certain degree of risk that the contract's outcome won't resemble what originally envisioned by the parties[223]. However, not every contract can be considered an aleatory contract. Think for example, to a delayed contract for the purchase of a good: the risks of currency depreciation or good's devaluation are beard respectively by the seller and the buyer[224]. This kind of risk which is common to almost every agreement is defined in the Italian legal system as a normal *Alea,* namely, the risk deriving from the uncertainty on costs and benefits deriving from the contract that is related to possible variations of performances' outcome[225]. Only those variations, however, that do not exceed the threshold set by specific law provision like, for example, hardship or breach of contract, is comprehended in the normal alea.

By contrast, in an aleatory contract, the *Alea* is an essential and inherent element of the contract itself which was specifically wanted and accepted by the parties[226]. This

---

[222] R. NICOLO', voce *Alea*, in F. CALASSO (edited by), *Enciclopedia del Diritto,* Varese, Giuffrè editore, 1958, p. 1024.
[223] L. BUTTARO, *Del Giuoco e della Scommessa,* in A. SCIALOJA, G. BRANCA (edited by), *Commentario del Codice Civile,* Zanichelli, 1959, p. 69.
[224] G. DI GIANDOMENICO, *I Contratti Aleatori,* Torino, Giappicchelli, 2005, p. 102.
[225] Ibidem, p. 101
[226] R. NICOLO', voce *Alea*, in F. CALASSO (edited by), *Enciclopedia del Diritto,* Varese, Giuffrè editore, 1958, p. 1029.

risk is necessary for the contract in order to fulfil its legal function and it may affect the size or even the very existence of parties' performances[227].

The bet contract is an aleatory contract where the legal risk is artificially created by the parties. This means that no risk exists upon the parties before the conclusion of the agreement (unlike the insurance agreement, where the risk beard by the policy holder exists anyway outside the contract) but it is created specifically for it, and it represents the contract's very cause[228]. Therefore, the *Alea* will necessarily be part of the terms that shape the meeting of the minds, and a vice affecting it would invalidate the whole contract.

The bet example's purpose is to show how to implement in a smart contract, an external element, totally outside the parties' control, that affects the outcome of the legal relationship using the *Stipula* language.

The digital encoding of such contract requires parties, first of all, to agree on a few new specific terms: the *alea fact* (the uncertain event to which contract effects are subordinated); the respective forecasts on the alea fact result (*value1*, *value2*); a specific *source of data* from which the relevant information will be retrieved. Parties accept to be bound by the data coming from that specific data source, and not another one: it is crucial to choose a trusted data source. These terms will be included as fields in the agreement constructor. Secondly, it is necessary to include in the contract a trusted third party charged to feed the data into the smart contract by calling specific functions. This entity (i.e., *Data Provider*) will be included in the smart contract as a subject of the agreement constructor, just like the authority in the previous example. Moreover, it is pivotal to the correct functioning of the smart contract to provide it with precise time limits for the performance of the relevant actions like, for instance, placing the bets, providing the actual value of the aleatory event and so on. Indeed, a number of issues blocking the execution of the contract may occur (e.g., the football

[227] G. DI GIANDOMENICO, *I Contratti Aleatori,* Torino, Giappichelli, 2005, p. 74.
[228] G. DI GIANDOMENICO, D. RICCIO, *Del Giuoco e della Scommessa,* in D. VALENTINO, *Commentario del Codice civile – Dei Singoli Contratti art. 1861-1986,* Torino, UTET Giuridica, p. 341.

match is cancelled or the online service providing data is down) and time limits help terminating the contract in such cases.

3.3.3.1 Elements

The bet contract we put forward is a basic bet agreement. Each party places her bet by sending the agreed amount of money to the smart contract together with her own forecast. If the value provided by the dataprovider matches one of the parties forecast, the smart contract will send the balance kept in custody to the winner. If the value doesn't match any forecast, parties take back their stakes.

The lifecycle of our bet contract unfolds through the following stages:

1. The contract is arranged and its terms are set by the parties: the alea event (i); the data source (ii); the amount of money at stake (iii); time limit for placing the bet (iv); time limit for feeding the data into the smart contract (v); data provider's fee (vi).
2. One party accepts;
3. The other party accepts: the meeting of the minds is effective, and the contract produces its effects:
    a. At the occurrence of the event, the loser party will be obliged to pay the winner;
    b. Data provider must feed the data into the smart contract within the time limit;
4. Data provider provides the information:
    a. Better1 is the winner
        i. Better2 pays Better1;
        ii. Better2 pays provider's fee;
        iii. Contract is termianted
    b. Better2 is the winner:
        i. Better1 pays Better2;
        ii. Better1 pays provider's fee;
        iii. Contract is terminated

c. Neither party wins:
    i. Both parties pay the data provider
    ii. Contract is terminated

5. Violations:
    a. One party doesn't place the bet within the time limit: the other party get her stake back and the contract is terminated;
    b. The data provider fails to provide the required value: parties take back their stakes and the contract is terminated;
    c. One party already knew the result of the event → the contract is void for lack of alea.

The relevant legal patterns we can extrapolate from this contract are:

- The **meeting of the minds;**
- An **aleatory value;**
- **Obligation** to pay: parties commit to pay depending on the aleatory value;
- A **trusted third party** charged to provide the relevant data.

3.3.3.2 Code

```
1  legal_contract Alea {
2    assets bet1, bet2
3    fields alea_fact, val1, val2, data_source,
4           fee, amount, t_before, t_after
5
6    agreement(Better1,Better2,DataProvider){
7      DataProvider =SET=> fee, data_source
8      DataProvider =OK=> t_after, alea_fact
9      Better1 =SET=> alea_fact, amount, t_before, t_after
10     Better1 =OK=> fee, data_source
11     Better2 =OK=> alea_fact, amount, fee, t_before, t_after
12                   data_source
13   } ⇒ @Init
14
15   @Init Better1 : place_bet(x)[y]
16     (y == amount){
17         y ⊸ bet1
18         x → val1
19         t_before ≫ @First { bet1 ⊸ Better1 } ⇒ @Fail
20     } ⇒ @First
21
22   @First Better2: place_bet(x)[y]
23     (y == amount){
24         y ⊸ bet2
25         x → val2
26         t_after ≫ @Run {
27             bet1 ⊸ Better1
28             bet2 ⊸ Better2 } ⇒ @Fail
29   } ⇒ @Run
30
31   @Run DataProvider : data(x,z)[]
32     (x==alea_fact){
33         (z==val1){              // The winner is Better1
34             fee ⊸ bet2,DataProvider
35             bet2 ⊸ Better1
36
37         }
38         (z==val2){              // The winner is Better2
39             fee ⊸ bet1,DataProvider
40             bet1 ⊸ Better2
41             bet2 ⊸ Better2
42         }
43         (z != val1 && z != val2){       //No winner
44             bet1 ⊸
45             bet2 ⊸      Better1
                           Better2
46         }
47     } ⇒ @End
48 }
```

Lines 1 to 4. The assets of this smart contract “bet1” and “bet2” will hold the amount of money staked by the parties. The other smart contract’s fields are the aleatory event, the data source, time limits (“t_before” represents the time parties have for placing

the bet, while "t_after" represents the limit within which the data provider has to feed the data) and the amount of money necessary to place the bet ("amount"). "val1" and "val2" are the values that represent parties' forecasts on the aleatory event and they will be filled by parties when placing the bet.

Lines 6 to 13: the agreement. As in the license example, we have a third party included in the agreement, with the difference, though, that in this case it is required to both set and agree on some terms. Note that parties to the agreement only have to agree ("OK") on terms set by others (e.g., better 1 only agrees to the terms set by the data provider) and only to those that are relevant to the party itself (e.g., the data provider does not agree on the "amount" because it is not affected by it).
At the end of the agreement the contract enters in the initial state "@Init" (line 13).

Lines 15 to 20: the "place_bet" function. Better 1 can place the bet calling this function. Letters (x) and [y] in line 15 indicate the necessary elements involved in the bet. The prediction value "x" and the sum of money to be staked "y", are inserted in in different types of brackets to highlight the different nature between of the two elements: an asset [y] and a normal field (x). The asset staked must be equal to the agreed amount as indicated in line 16. The first is sent to the smart contract asset location bet1 (line 17) and the prediction value is uploaded in the field "val1" (line 18). Line 19 schedules an event necessary to terminate the smart contract in case the other party doesn't place the bet within the time limit. If the contract state's will still be "@First" when "t_before" expires, then the bet already staked by Better1 will be sent back to his address and the contract will be terminated switching its state to "@Fail".
Contract state switches to "@First" (line 20).

Lines 22 to 29: better2 "placebet" function. This function can only be called if the other party already placed her bet. The only difference from the previous function is

represented by the event scheduled in lines 26-28. According to this event, parties will receive back their bets where the data provider fails to provide the relevant data within the agreed "t_after" (i.e., if the contract's state is still "@Run", as showed in line 26) and the contract will be terminated switching to "@Fail" state.
After this call, contract's state switches to "@Run" (line 29).

Lines 31 to 47: "data" function. This is the function used by the data provider to feed the data into the smart contract. The data consist of two values. "x" represents the aleatory event and it must match with the "alea fact" value included in the agreement (line 32). "z" stands for the result of such event and it may match either better1 or better2 prediction ("val1" or "val2"). In the first case (line 33), both assets "bet1" and "bet2" are sent to better1 (lines 35,36). Otherwise, if better 2 is the winner (line 38), he receives the assets. As shown in lines 34 and 39, the fee due to the data provider is paid using a fraction of the bet staked by the loser party. Lines 43-46 describes the situation where the value provided by the provider doesn't match any of the parties' predictions (line 43). In this case the smart contract provides both bets to be sent to the data provider (lines 44, 45).
After the data have been provided and the contract produced its effects, the smart contract physiologically terminates and its state switches to "@End".

3.3.3.3 Legal Considerations

This bet contract is a very basic example consisting of few elements and, for this reason, is amenable to be fully automated by a smart contract in its entirety. As to the arrangements of contract's terms and the meeting of the minds (**points 1, 2, 3**), these stages are fully represented in the smart contract respectively by the agreement constructor and the two "place bet" functions. The consent is expressed *facta concludentia* by sending the money to the smart contract. Escrowing the money at this stage serves the purpose of enforcing the obligation laid out in **point 3.a**. Once in custody of the smart contract, at the occurrence of the relevant event, the money will

be automatically sent to the winner party without relying on parties' behaviour (**points 4.a.i, 4.b.i**). The same consideration is valid for the data provider's fee (**points 4.a.ii, 4.b.ii**). The two payment obligations are activated by the function "`data`" called by the provider.

The data provider's obligation to feed the required values within the time limit outlined in **point 3.b** is implemented by the event scheduled in Better2's "`place_bet`" function. Such provision keeps the contract from running indefinitely in case some issues prevent the provider to perform the required actions. In this case, the legal situation standing prior to contract conclusion is restored, and the contract is automatically terminated (**point 5.b**).

The event scheduled in better1's "`place_bet`" function addresses the same need and terminates the contract if the second party doesn't place the bet within the time limit (**point 5.a**).

The violation of **point 5.c** describes the case where one party, either by fraud or error, already knew the result of the aleatory event. This circumstance affects and, probably, completely undermines one of the contract's essential elements, namely the alea. Alea, in fact, is a constituent part of the bet's contractual cause and the very reason why parties decided to make an agreement (in the sense that, if the event was known, there would have been no reason to conclude such contract). According to the Italian legal system, the lack of an essential element in a contract renders it null and void[229]. Automating such an outcome in a smart contract, however, is not possible. Smart contracts can only react to external events, and not to situations inherent to its underlying code because. Just like a software cannot react to an inherent bug, but simply keeps executing the code affected by it, a smart contract cannot react to a vice in its essential principles. Therefore, the process to claim and enforce the invalidity of the contract will have to be dealt with completely off-chain, just like in paper contracts. The only difference lays in the fact that, with paper contracts, should one party get aware of the contract's invalidity, she can just abstain from performing her obligation.

[229] Art. 1418 c. 2 c.c.

By contrast, in smart contract execution is automated, and even an agreement contrary to the legal system would, once deployed, will be executed. Same issues arise when the contract has an illegal object. This matter will be better addressed in the next chapter.

3.4 CONCLUDING REMARKS

The analysis of contract examples implemented through *Stipula* allows to highlight the language's overall strengths and weaknesses.

A clear strength lays in *Stipula* code's clarity. The language elements, primitives and operators are expressed in a manner that is closely comparable to natural languages, requiring only a little bit of practise to be able to understand and handle it.

Another advantage is its legal-based nature. Every language's feature was specifically designed to represent a legal concept, just like obligations, permissions, or the constitution of a right *ex novo*. Starting by these simple elements, one can build increasingly complex contracts by assembling different pieces together, just like when drafting normal contracts.

As to *Stipula* capability of fully implementing and automating contractual terms we can sum up the main points as follows:

- Consent to contract's formation is effectively given by conduct (*facta concludentia*) through a function call. This usually entails sending a sum of money or the virtual representation of a good (token) to the smart contract.
- Monetary obligations or the commitment to transfer a good are implemented by the coordinated use of two features. The good or money is escrowed by the smart contract in the initial stages so that they are taken away from parties' control. When the necessary preconditions are fulfilled by the parties, the smart contract automatically sends the goods the one concerned (e.g., the "buy" function in the license agreement). In order to avoid malicious non-performance, *Stipula* employs the *event* primitive to set a deadline for the performance. Should the deadline expire, the smart contract could either

enforce the obligation automatically (since it controls the relevant assets. E.g., event in the "useBox" function in the free rent contract), or issue a penalty from the escrowed money.

- Obligations that entail an off-chain performance are dealt with using an *Authority* (an oracle). Since it's not possible for these to be enforced on-chain, a third party is included in the agreement to act as a supervisor for the correct off-chain performance.

  Depending on the content of the obligation, different types of oracles can be employed, in order to rely the least possible on third parties' trust. When the performance of an obligation is objectively verifiable (e.g., obligation to answer to licensee's queries within ten hours in the license agreement), the trusted third party can consist of an automated oracle, i.e., a software or a device programmed to detect the occurrence or non-occurrence of performance in the real word and call the necessary functions in the smart contract (e.g., a sensor that detects that scans the products to be delivered).

  On the other hand, when an interpretation activity is needed to assess the correct outcome of the performance (e.g., the obligation to diligent care of the good in the free rent contract), then an expert oracle will be required. At the current level of technology an expert oracle activity can only be carried out by a human party that is able to determine whether a performance meet the threshold required by a vague or not clearly defined law provision (e.g., diligent care, the "intended use" in the free rent contract etc.)
- Similar considerations can be made for parties' liability for contract breaches. When the conditions required for liability are verifiable deterministically (strict liability where the good was esteemed in the free rent contract), penalties issuing can be automated in smart contract with the help of an automated oracle. When, by contrast, liability assessments require the occurrence of event open to interpretation (e.g., unforseeability, force majeure, and especially

subjective liability), an expert oracle (acting as an arbitror or judge) will have to be employed.

- As to contract invalidity, detection and reaction to such situation cannot be implemented inside the smart contract because it would be impossible for it to question its own foundations (the underlying code). Just like in paper-based contract, the invalidity shall be claimed off-chain and decreed by a court. Hypothetically, parties could build a smart contract providing it with a termination function to be called by an authority in case of invalidity of the contract. However, should the contract have already executed a transfer of rights or goods, the restoration of the situation previous to the contract formation will have to be dealt with through traditional channels.

CHAPTER 4

SMART CONTRACT LIMITATIONS

As it arises from the analysis carried out in the previous chapter, smart contracts feature some strengths potentially capable of bringing enormous innovations in the current system of legal relationships and, more broadly, in the whole social system. At the same time though, we were able to clearly identify elements, concepts, and mechanisms that this technology fails to incorporate.

For the most part, these limitations stem from the same features that give smart contracts and blockchain their innovative power. It could be the very fact that smart contracts are essentially a computer software, the automatic enforcement that they provide, or the immutable character of the blockchain itself. As we will see, all these characteristics both have positive and negative sides. The weaknesses stemming from here are therefore inherent to the very nature of this technology, meaning that a perfect solution overcoming those while preserving all the strengths at the same time, is logically infeasible. Technology (today's and future's) can help approximating to zero the drawbacks, eventually sacrificing some of the positive sides. The crucial job when employing smart contracts, consist then in optimizing the trade-off between negative and positive sides, taking into account the sector of use, what strengths are necessary and which ones can be sacrificed.

The present chapter gives an overview of such limitations and describe the main means currently employed to solve them.

4.1 INHERENT LIMITATIONS

Smart legal contracts' aim is basically to codify legal agreements in order for them to be automatable and to avoid reliance on parties' trust and will. The achievement of such task centers around the interplay of legal language and programming languages. Many authors suggest that this interplay is facilitated by the fact that both languages

feature formality[230]. Formal languages can be defined as languages "designed for use in situations in which natural language is unsuitable, as for example in mathematics, logic, or computer programming. The symbols and formulas of such languages stand in precisely specified syntactic and semantic relations to one another"[231]. In other words, unlike natural languages where its rules (i.e., grammar) are inferred through the empirical analysis of something already existing (grammar created *ex post*), in formal languages, grammar is not "discovered" but rather stipulated *a priori* (*ex ante*) and used to build the language from scratch. This characteristic allows for the creation of languages where each symbol or formula is provided with a univocal meaning so as to remove ambiguities as much as possible.

On one hand, it's undeniable that formality is a fundamental feature of any legal system[232]. This is exemplified by the rule application decision process: the applier only limits himself to identify the relevant elements of the situation, which, *per se,* trigger the response and doesn't question whether that particular response, considering the whole context, is the best one[233]. From this perspective we can say that both programming and legal languages are mechanical in their functioning. This is the formal character they have in common.

Yet it must be said that this approach has different intensities in different countries. Legal formality significantly affects the interpretation of both law and contract provisions and indeed we can observe different approaches between legal systems on this matter. When interpreting contracts' content, English courts, for example, are less inclined to use contextual elements outside the contract (such as parties' true intent, reasonableness or fairness concepts) to define its meaning, as would likely happen in civil law countries, but rather prefer to stick with its literal content[234]. Similarly, EU Court of Justice takes a more teleological approach in statutory interpretation, taking

---

[230] J. M. LIPSHAW, *The persistence of "Dumb" contracts,* in *Stanford Journal of Blockchain Law & Policy,* 2019, p. 31.
[231] www.collinsdictionary.com
[232] D. KENNEDY, *Legal formality*, in *The Journal of Legal Studies,* 1973, p. 359.
[233] Ibidem, p. 360.
[234] M. FONTAINE, F. DE LY, *Drafting International Contracts,* Leiden, Brill Publishers, 2009, p. 113.

into account the original *stimuli* giving birth to the law provision, while English courts favour the technical or ordinary meaning of words as their primal interpretation criterium[235].

So, although legal language can be said to have a formalistic character in that it aims at being applied mechanically, its texts are however expressed in natural language which necessarily entails fuzziness[236].

Fuzziness is often given by what we can call *continuous* predicates, that are usually included in contracts. Continuous predicates can be defined as values that need to be measured to be determined and that are amenable to take any value on a given scale[237]. An example can be height or temperature: on these scales any interval can be divided into infinite subvalues. By contrast, *discrete* predicates are datasets that can only assume a limited number of precise values like for example the number of pages in a book, or the current date[238]. Now the problem is that computers can only operate on discrete data, meaning that every continuous predicate in an agreement has to be broken up into discrete elements to be implemented in a smart contract[239]. So, for example, predicates like "fairness" or "serious damages" are continuous in their nature because they need to be measured on a potentially infinite interval of values. The implementation of such terms requires a decomposition of the dataset in different and precise subvalues that are amenable to mechanical detection through Boolean Logic (TRUE/FALSE). This translation necessarily cuts off a good share of all the infinite values possible, in other words creating approximations. Approximations, however, can be very dangerous when in the legal ambit. Contracts have to be written as precisely as possible since just one mistaken word could give rise to highly expensive disputes and litigations[240].

---

[235] R. S. SUMMER, *The Formal Character of Law,* in *The Cambridge Law Journal,* 1992, p. 243.
[236] J. WRÓBLEWSKI, *Legal Language and Legal Interpretation,* in *Law and Philosophy*, 1985, p. 241.
[237] P. NEWBOLD, W. L. CARLSON, B. THORNE, *Statistica – seconda edizione,* Pearson, 2010, p. 24.
[238] Ibidem, p. 24.
[239] M. LIPSHAW, *The persistence of "Dumb" contracts,* in *Stanford Journal of Blockchain Law & Policy,* 2019, p. 34
[240] E. MIK, Smart Contracts: Terminology, Technical Limitations and Real World

Removing approximations (for what is possible) when coding smart contracts may require a great amount of effort by developers which increases costs and computational complexity. The latter doesn't overlap with legal complexity. Indeed, while this one arises from the sheer number of rules employed for the infinite number of possible circumstances to which they apply[241], the former is related to the "ability to produce quantifiable measures of how long a program will take to process input of various sizes"[242]. In other words, problems that wouldn't be complex in normal contracts, become such when implemented in a smart contract because of the high number of variables as to be intractable[243].

The limitations just described derive from the inner nature of smart contracts, that is, being a computer software that necessarily expresses everything in binary values, 0 and 1. Such characteristic cannot be avoided or overcome and that's the reason why these problems will always exist for smart contract to a certain extent.

4.2 INFLEXIBILITY

A direct consequence to the issue outlined above is that smart contracts must be designed to execute provisions suited to algorithmic determination[244]. This, in turn, forces parties to sacrifice flexibility, which, even though it is sometimes seen as inherently bad, causing disputes and slowness in justice, is indeed an important source of efficiency in legal relationships.

Parties, in fact, often need to include in their agreements terms that are either inherently open-ended, like the term 'good faith' that is used to refer to a precise concept in natural language and that cannot be decomposed in other predicates suitable to objective determination; or terms that are purposefully left vague.

Complexity, in *Law, Innovation & Technology*, 2017, p. 17.

[241] E. KADES, *The Laws of Complexity & the Complexity of Laws: The Implications of Computational Complexity Theory for the Law,* in *Rutgers Law Review,* 1997, p. 407.

[242] Ibidem, p. 427.

[243] M. LIPSHAW, *The persistence of "Dumb" contracts,* in *Stanford Journal of Blockchain Law & Policy,* 2019, p. 31.

[244] C. LIM, TJ SAW, C. SARGEANT, *Smart Contracts: Bridging the Gap Between Expectation and Reality*, in *Business Law Blog,* 2016, p.5.

The latter situation is common to contract drafting because parties can never completely predict all the possible future events capable of affecting their legal relationship and reduce the whole agreement to fully defined terms ex ante[245]. This stands both for terms that determine the correctness of performance and for those that provide for excuses, modification, or exemption to performance depending on changing of circumstances. As to the first case, in fact, it's been historically understood that legal criteria provided to assess the correctness of performance are often not aimed at identifying objectively right performance outcomes, but rather they imply an ex-post policy balancing evaluation to be carried out by legal authorities to find the best (or fairest) solution among conflicting interests of societal actors[246]. In computable contracts, instead, parties choose to evaluate performance themselves ex-ante, at the time of drafting[247].

The same considerations can be made for excuses to performance, modification or termination causes. Predicting all the possible events that could complicate performance is clearly unfeasible. A more realistic model of agreement is that parties use standards (e.g., *force majeure,* hardship clauses or impossibility of performance) as a form of semantic flexibility that, instead of leaving a term undefined or possibly wrongly defined, call a court to evaluate the situation later in time[248]. Such flexible terms though, cannot be executed by a computer software for the reasons mentioned above, thus giving rise to the need of trusted third parties capable of evaluating the relevant situation.

4.3 SELF-ENFORCEMENT AND IMMUTABILITY AS A DRAWBACK

Automatic self-enforcement and immutability are two of the main features representing the strength and innovative power of smart contracts. At the same time

[245] J. SKARLOFF, *Smart Contract and the Cost of Inflexibility*, in *Un. of Pennsylvania Law Review,* 2017, p. 280.
[246] H. SURDEN, *Computable contracts*, in *UC Davis Law Review,* 2012, p. 685.
[247] J. SKARLOFF, *Smart Contract and the Cost of Inflexibility*, in *Un. of Pennsylvania Law Review,* 2017, p. 281.
[248] Ibidem, p. 281.

though, they act as a limitation when it comes to the applicability of the general legal framework. Self-enforcement, for instance, prevents the application of many contract law's fundamental provisions and doctrines addressing validity and illegality of contracts.

Think, for instance, to a contract entered into by a minor lacking the legal capacity to bargain. Or again, a contract to transfer illegal drugs, arms, or to set a price-fixing cartel. While in a traditional context, enforcement institutions would take action to prevent the execution of such contracts, with a smart contract the performance would be certain and automatic.

This is because smart contracts fail to take into account whether the essential elements of the agreement conform to the relevant law provisions, and simply enforce what they have been programmed to. They operate within their own closed technological framework creating a conflict among what is legally and technically binding[249].

Self-enforcement also precludes parties to abstain from performing when it becomes more efficient to do so (*efficient breach* doctrine). Especially in common law countries, breaching the contract promise when the cost of performance turns out to outweigh the contract's value is deemed to be acceptable (provided that the breaching party pays a compensation to the other)[250]. Smart contracts, by contrast, make such practise particularly difficult, potentially leading to overall efficiency losses[251]. Such an obstacle can, however, be overcome by escrowing a sum of money by way of compensation for the non-performance at the moment of contract's conclusion. Many smart contracts already provide for such a solution, included the license and free rent agreements examples described in chapter 3. The initial escrow is used to compensate the party that suffered a breach of contract terms. In such a way, parties are enabled to

---

[249] A. WRIGHT, P. DE FILIPPI, *Decentralized Blockchain Technology And The Rise of Lex Cryptographia*, available at SSRN www.ssrn.com/abstract=2580664, 2015, p. 26.

[250] J. COLEMAN, J. KRAUS, *Rethinking the Theory of Legal Rights*, in *The Yale Law Journal 95,* 1986, p. 1335.

[251] H. EENMAA-DIMITRIEVA, M. J. SCHMIDT-KESSEN, *Creating markets in no-trust environments: The law and economics of smart contracts*, in *Computer Law and Security Review, 2019,* p. 85.

decide whether to perform or to opt for the money compensation in case the latter is a better trade-off.

As of today, we have already several mechanisms that can be employed to mitigate the drawbacks deriving from the automatic execution of unwanted performance, either because one party changes her mind or because she gets aware of the contract's illegality. One of these is the multisignature verification technology that, when used, requires multiple parties to sign a transaction (needed to trigger the execution of a smart contract) before it can actually take place. So, for example, if Alice and Bob had to create a smart contract for the purchase of a good, they could implement multisig verification requiring two out of three parties (Alice, Bob and a trusted third party, an arbitror) to sign the transaction triggering the execution (i.e., the good delivered, and the sending of the payment). If both seller and buyer agree to the execution their signature will be enough to trigger the contract; where either party refused to sign, claiming a breach or an illegality, the arbitror would have the last world on the execution[252].

Another way to prevent the execution of a smart contract is to implement a so called "kill switch" or "self-destruct" operation in the solidity code. Once called, this function sends the contract's balance to a predefined address and then deletes the contract's code from the blockchain going on. This means that from that point on, contract's functions cannot be called anymore, thus halting auto-performance[253]. Note that using the self-destruct function is not like deleting data from a hard disk; the contract code is still in the history of the blockchain, but the operation avoids the possibility to interact with the smart contract in the following blocks[254]. Basically, the self-destruct function acts as a way to terminate the contract where a cause of early termination occurred. The grounds for termination may derive either from law provisions or doctrines (termination by right), from one clause of the agreement (termination by

[252] K. WERBACH, N. CORNELL, *Contracts ex Machina, Duke law journal,* 2017, p. 133
[253] B. MARINO, A. JUELS, *Setting Standards for Altering and Undoing Smart Contracts*, in *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, 2016, p. 162.
[254] https://docs.soliditylang.org, *Deactivate and Self-destruct.*

agreement), or from a court's decision. In any of these cases, a smart contract should ensure that the outcome of the termination process respects three fundamental standards. First, that the smart contract's auto-performance is definitively disabled; second, that the termination is operated only if the party entitled of such right decides to exercise it; and third, the termination should restore the situation existing before the conclusion of the agreement, thus compensating for the part of performance already operated[255]. The first and second standards are relatively easy to achieve with a smart contract's self-destruct function: as just mentioned, the function deletes the code from the subsequent blocks halting auto-performance. Moreover, it is easy to encode that a function can only be called by a specific address. The real issue arises when the right to terminate the contract arises from an event occurring off-chain like, for example, a change of circumstances or a malicious behaviour by one party. Given the natural incapability of smart contract to communicate with the outside world, the only possibility is to defer to a trusted third party (i.e., *oracle*) the task of monitoring the potential arising of termination causes, and, in such case, of empowering the relevant parties to take the necessary actions.

Similar issues can be faced when doctrines like objective impossibility of performance, hardship or force majeure come to the picture. These situations may also represent grounds for the termination or reform of the contract, but, unlike the object's illegality or parties' incapacity to bargain, they are external to the contract and don't affect it *ex tunc* (since its inception), but only *ex nunc* (since the occurrence of the event). Here is where the blockchain's immutable character comes to play a drawback role. As it is evident, such feature complicates the modification of the code embedded in the distributed ledger. All of these situations originate a cause of termination or at least modification of contract provisions, clearly in contrast with the immutability and self-enforcement of smart contracts. Ways to modify the contract's code when already deployed on the blockchain exists, even though they often imply that of such

[255] B. MARINO, A. JUELS, *Setting Standards for Altering and Undoing Smart Contracts*, in *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, 2016, p. 156.

possibility needs to be provided in the original contract's code. When the term that needs to be changed is captured by a simple variable in the code (e.g., costs, time limits), the modification is relatively easy if a specific set-function for that variable was included in the smart contract in sight of potential changes[256]. When, instead, the term that has to be modified is captured by a function, the situation gets more complicated. While variables can be changed freely, functions in an Ethereum contract code are immutable, once issued. Some authors suggest that functions could be deleted added or swapped by using Solidity's enums and modifiers[257].

Another way to modify smart contracts' terms is to create a new smart contract that will act as an intermediary between the old one and the user. Through the function "delegatecall" the calls and transactions directed to the first contract will be redirected to the new one, thus executing the code of the "upgraded" smart contract[258]. In practise, users still refer to the original contract as to the address and balance, and only the code is taken form the calling address[259].

Now, if these technical methods can somehow help overcoming the problem deriving by the immutable character of the blockchain, the other crucial issue arising in this context concerns the capability of smart contracts of detecting the events and the changes of circumstances that entitle the parties with a modification right. Here, the inclusion of an oracle is always required for two kinds of reasons: first, to connect the smart contract to the off-chain world and then communicate the occurrence of the relevant events; and second, because often the events that give rise to modification or termination causes are defined using open-ended or purposefully vague terms. In other

---

[256] B. MARINO, A. JUELS, *Setting Standards for Altering and Undoing Smart Contracts*, in *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, 2016, p. 15.
[257] Ibidem, p. 16. For a definition and explanation of the employment of *enums* and *modifiers* see https://docs.soliditylang.org: "Functions *modifiers* can be used to change the behaviour of functions in a declarative way. For example, you can use a modifier to automatically check a condition prior to executing the function". "Enumerated types (*enums*) is a user-defined data type used in computer programming to map a set of names to numeric values. Enumerated data type variables can only have values that are previously declared. In other words, they work with a finite list of values" www.techopedia.com.
[258] M. GRINCALAITIS, *Can a Smart Contract be Upgraded/Modified?*, www.medium.com, 2018
[259] https://docs.soliditylang.org, *Delegatecall / Callcode and Libraries.*

words, the assessment on what can be considered a force majeure event or when a performance has become impossible cannot be made objectively by a software for the reasons outlined in par. 4.1. Such evaluative task often needs to be deferred to a trusted third party included in the agreement.

4.4 OFF-CHAIN CONNECTION: THE ROLE OF ORACLES

One last clear weak point of this technology is the fact that the blockchain is a closed ecosystem. As already highlighted multiple times in this work, the blockchain cannot, by itself, communicate (i.e., sending and receiving information, data) with the off-chain world. This is no small matter when it comes to implementing actual legal contracts: those are, most of the times, linked to the real-world either because they represent a tangible asset, because the performance has to occur off-chain or because their execution might be dependent on a real-world event.

A partial solution to this and the other limitations outlined before, is the employment of Oracles. Oracles, as already described in par. 1.6, work as a communication channel between the on-chain and the off-chain worlds. These connections to the outside world can be employed in different ways to provide a solution to the various smart contract's limitations. The most direct use is certainly to provide to the smart contracts the needed information collected off-chain. They can digest extrinsic and non-deterministic data into a format that smart contracts can understand, and feed them to the contract by calling a specific function[260] (e.g., the *Stipula* Bet contract). But on the other hand, a connection to the external world can be useful to defer outside the blockchain issues that cannot be dealt with by the smart contract like, for example, providing contractual flexibility. An oracle could be charged with the task of making an assessment on ambiguous contractual terms that smart contracts' code is unable to process. However, in both cases, relying on an external entity to improve smart contracts' performance, contrasts with the trust-less feature of this technology. "This dependence implies

[260] B. CURRAN, *What are Oracles? Smart Contracts, Chainlink & "The Oracle Problem",* available at www.blockonomi.com, 2018.

trusting something in addition to the smart contracts" which is not code-controlled nor necessarily decentralised[261]. Indeed, oracles are usually centralised and third entities that can be malicious, not impartial, and thus representing a potential threat to the integrity and truthfulness of the data. More specifically, when using an oracle, a potential trust failure arises at two different levels. At the data source level, where parties decide they want to retrieve the information from: parties bind themselves to the data provided by that specific source, thus bearing the risk of it being malicious, mistaken or having a technical problem keeping it from providing the data. And at the oracle level, which is the entity charged by the parties to send the information from the agreed data source to the smart contract: similarly, the oracle could be corrupted or having a malfunction that hampers the correct contract's execution. This situation gives rise to what has been named "The Oracle Problem".

The oracle problem can be defined as "the security, authenticity and trust conflict between third party oracles and the trust-less execution of smart contracts"[262]. Such problematic led many to conclude that the best use for smart contracts could only be in the field of digital goods' transactions, where all the stages of the contractual lifecycle can occur on-chain[263], with no need for oracles and maximizing decentralisation. When smart contracts are linked to tangible assets, in fact, performance cannot be ensured (e.g., delivery of a package) without a trusted third party supervising it[264]. Moreover, if on one hand smart contracts reduce transaction costs avoiding intermediaries, on the other hand, ensuring oracles' trustworthiness increases costs, thus partially eliminating another of the main benefits of this

[261] G. CALDARELLI, *Real-world blockchain applications under the lens of the oracle problem. A systematic literature review,* in *2020 IEEE International Conference on Technology Management, Operations and Decisions*, 2020, p. 2.
[262] B. CURRAN, *What are Oracles? Smart Contracts, Chainlink & "The Oracle Problem",* available at www.blockonomi.com, 2018
[263] A. GUADAMUZ, *All Watched Over by Machines of Loving Grace: A Critical Look at Smart Contracts,* in *Computer law and Security Review,* 2019, p. 15.
[264] CALDARELLI, *Real-world blockchain applications under the lens of the oracle problem. A systematic literature review,* in *2020 IEEE International Conference on Technology Management, Operations and Decisions*, 2020, p. 4

technology[265]. Although the validity of these arguments cannot be contested, it must be also pointed out how smart contracts can be effective even when employed in conjunction with oracles.

In finance, for example, contracts usually require stock market's prices and data, and the only way to for a smart contract to retrieve those is through an oracle. Here, however, oracles don't really create a relevant trust problem since stock market's data are publicly available. It is relatively easy for both users and authorities to spot unreliable oracles.

Another important field where smart contracts can provide many benefits is the traceability for the supply-chain. In this regard, it was held that, compared to the improvement that this technology provides, the risk of imprecise information provided by oracles and the cost to ensure their reliability can be considered negligible[266]. Especially when traceability is required to ensure sustainability or a hallmark, in fact, reliability of product's background data becomes crucial. An in-depth case study conducted on an Italian company producing a certified product (D.O.P.) shows how the oracle problem doesn't play a big role. Italian laws on high quality products require companies to track all production phases to ensure the provenance of all raw materials. In these contexts, oracles are required to upload only information that has been strictly verified by the relevant authority, making it difficult or, meaningless for oracle companies to upload counterfeited data: identifying the altered information would be easy for the certifying authority[267].

Anyway, methods to get around the trust issue created by oracles exist and have already been employed. One example is the use of multisignature system (see par 4.3), where instead of relying on just one oracle, multiple independent oracles are called to agree on the same information of the outside world. If an enough number of oracles

---

[265] J. PEREIRA, M. MAHDI TAVALAEIB, H. OZALP, *Blockchain-based platforms: Decentralized infrastructures and its boundary conditions*, in *Technological Forecasting & Social Change,* 2019, p.98.

[266] A. KUMAR, R. LIU, Z. SHAN, *Is Blockchain a Silver Bullet for Supply Chain Management? Technical Challenges and Research Opportunities,* in *Decision Science,* 2019, p. 21.

[267] G. CALDARELLI, C. ROSSIGNOLI, A. ZARDINI, *Overcoming the Blockchain Oracle Problem in the Traceability of Non-Fungible Products*, in *Sustainability,* 2020, p. 11.

sign the information, that will be fed into the smart contract[268]. The same goal can be achieved through the use of a decentralised oracle system like the ones outlined in section 1.6. The basic idea is that if multiple oracles are called to provide the same information, it would be less likely to all be malicious or to make mistakes. However, the risk that the whole system may be corrupted or simply mistaken persists[269].

The oracle problem cannot be completely eliminated, and it always needs to be addressed when working with smart contracts. In order to maximize the benefits of smart contracts and minimize oracles' negative sides, a case-by-case approach is necessary, taking into accounts, the employment sector, the task which the oracle is charged with, the costs that such employment would entail and the possibility to use a multisignature system or a decentralised oracle. Obviously, the wider and more important the oracle's role (e.g., an oracle feeding to the smart contract the final score of a football match, compared to one charged to assess on the correct performance of a behavioural obligation), the less decentralisation and trust-less characters the smart contract will preserve, getting closer and closer to traditional paper-based contracts.

[268] H. EENMAA-DIMITRIEVA, M. J. SCHMIDT-KESSEN, *Creating markets in no-trust environments: The law and economics of smart contracts*, in *Computer Law and Security Review,* 2019, p. 84.
[269] J. THEVENARD, *Decentralised Oracles: a comprehensive overview,* www.medium.com, 2019.

Conclusions

Through the present research we were able to achieve relevant findings, related both to the applicability of the *Stipula* language and to the general capability of smart legal contracts to effectively represent legal agreements. These achievements can be summarized as follows.

For what it concerns *Stipula*, we can affirm that:

- The language, being specifically designed for smart legal contracts, is fully capable of expressing with its code the core concepts of contract law. Firs of all, it can effectively reproduce all the stages of contract formation necessary to express parties' contractual will. Consequently, a *Stipula* contract can match the requirements set by law for it to be considered legally binding and thus enforceable before a national court.

  In addition, contractual situations like prohibition, permission, obligation, escrow, alea, rights transfer and constitution (and so on), are clearly captured by the elements and primitives of *Stipula,* without any relevant ambiguity issue on their correlation.

  The main strength of this language is that its primitives and functions are abstract enough to be easily understandable even by non-IT professionals. Its code is not as cryptic as other programming languages, but rather has a pretty straightforward meaning, only requiring a bit of practice with the main features to achieve manageability. This work is also aimed to help achieving such task.
- We were also able to verify *Stipula'*s capability of retrieving information from off-chain sources. To this end, a trusted third party (*oracle*), named "Authority" or "DataProvider" in the examples, is included in the smart contract beginning from the agreement constructor, and then it is assigned with a number of functions that it can call, in order to feed into the smart contract the necessary data.

- As explained in the body of the research, the ones presented here are only basic examples useful to explain the functioning of this language. Further studies could be carried out to find out how to transpose legal concepts that were not taken into consideration here. However, *Stipula*'s strength lays exactly here: once learnt the basic elements constituting it, more complex contracts can be drafted simply assembling more and more simple units together. Moreover, steps ahead in the development of this language can be made by implementing new functionalities not provided in the version put forward here. An example is the possibility to introduce a negotiation stage for the determination of contract terms. This could be implemented directly in the agreement constructor by simply allowing all parties to both agree ("=OK=") to terms proposed by others, and propose ("=SET=") new ones where the previous term didn't match the party's interest, instead of being stuck to the predefined roles of proponent and acceptor.
- The study was also able to determine what are the limits encountered by this language, which approximately overlap those entailed by smart contract technology in general. When the content of an obligation entails a behaviour that has to be kept in the real world, in fact, a smart contract is not capable of automating its execution. To do so, an oracle monitoring the execution off-chain must be employed.

  The same situation is faced when open-ended terms that require a subjective assessment to be applied are included in the agreement.
- A necessary step towards the effective adoptability of *Stipula* for the creation of smart contracts is its encoding in the lower-level programming languages underlying the most important blockchains supporting smart contracts. An example is the Ethereum's *Solidity* language, at the base of the Ethereum Virtual Machine (EVM). A first step in this direction was already made by

Stipula developers in the paper "*Pacta sunt servanda: smart legal contracts in Stipula*" [270].

For what it concerns the general study of smart contracts, the present work demonstrates, first of all, how these can potentially meet the threshold requirements set by national jurisdictions (specifically, under Italian national jurisdiction, European jurisdiction and the international law context) in order to produce the effects of a legally binding contract. In particular, the achievement of an agreement (offer and acceptance) can be effectively represented using a smart legal contract, and the expenditure of an electronic signature is sufficient to validly assume a legal obligation.
After an overview of the main legislative actions taken by national states to regulate the blockchain phenomenon, we can affirm that the best approach to this end would be addressing the blockchain technology without providing strict definitions. Given the high and fast changeability of technology, in fact, strict definitions would fail at adapting to new versions or evolutions. A restrictive approach would be, on one hand, a source of inefficiency, requiring a new law provision for every slight development; on the other hand, it could act as a restraint to innovation requiring people to comply with a provision made for, for example, an outdated technology. The criteria of *technological neutrality* and *functional equivalence* introduced by UNICITRAL seem to be the most appropriate for the legislation in this field.
In conclusion, we acknowledge the limitations that this technology still entails. Some of them are inherent to smart contracts and will likely never completely go away. Others, instead, can be partially overcome thanks to present and possibly future's technology. The crucial issue is to acknowledge the presence of these weaknesses in order to be able to optimize the effectiveness of smart contracts.
Smart contracts, just like blockchain, do not have to be considered as a universal panacea for all the flaws of the current legal system (and other sectors as well), even

[270] S. CRAFA, C. LANEVE, G. SARTOR, *Pacta Sunt Servanda: Smart Legal Contracts in Stipula,* 2021.

though it is sometimes spoken of as such. Just like any other new technology, it can bring innovations, carrying weaknesses at the same time. It’s up to operators to comprehensively know the subject to optimize the result of its application.

Bibliography

AA. VV., *Blockchain and Smart Contracts,* a cura di ARTZT M. e RICHTER T., *Handbook of Blockchain Law,* Croydon, Kluwer Law International, 2020.

AA. VV. *Blockchain e Smart contracts*, a cura di BATTAGLINI R. e GIORDANO M. T., Giuffrè Francis Levbre. 2019.

AA. VV., *Commentario del Codice Civile,* a cura di BRANCA G. e SCIALOJA A., Zanichelli, 1959. Relativamente al Capo XXI- Del Giuoco e della Scommessa.

AA. VV., *Commentario Breve al Codice Civile,* a cura di CIAN C. e TRABUCCHI A., Milano, Cedam*,* 2021.

AA. VV. *Identificazione Elettronica E Servizi Fiduciari Per Le Transazioni Elettroniche Nel Mercato Interno,* a cura di DELFINI F. e FINOCCHIARO G., Torino, Giappichelli Editore, 2017.

AA. VV., *The Cambridge Handbook of Smart Contracts, Blockchain Technology and Digital Platforms,* a cura di DIMATTEO L. et al., Cambridge, Cambridge University Press, 2019.

AA. VV., *The Blockchain Revolution: Legal and Policy Challenges,* a cura di G. DIMITROPULOS et al., *,* Oxford University Press, 2018.

AA. VV. *Blockchain and Digital Identity,* in Eu Blockchain Observatory & Forum, www.eublockchainforum.eu, 2019.

AA.VV. *Trattato dei Contratti – III – Cessione e Uso dei Beni,* a cura di ROPPO V., Milano, Giuffrè Editore, 2014.

AA. VV. *Commentario del Codice civile – Dei Singoli Contratti art. 1861-1986*, a cura di VALENTINO D.*,* Torino, UTET Giuridica, 2011

AA. VV., *Commentario del Codice civile – Dei Singoli Contratti – Volume III,* a cura di VALENTINO D.*,* Torino, UTET Giuridica, 2011. Relativamente al Capo XIV del Comodato.

AA. VV. *Trattato di Diritto Civile,* a cura di ZORZI GALGANO N., Padova*, Cedam,* 2014.

ALLEN D. W. E. et al., *The Governance of Blockchain Dispute Resolution,* in *Harvard Negotiation Law Review,* 2019.

ALLEN J.G., *Wrapped and Stacked: 'Smart Contracts' and the Interaction of Natural and Formal Language,* in *European Review of Contract Law*, 2018.

ANDERBERG A. et al., *Blockchain now and tomorrow,* EU commission publications.jrc.ec.europa.eu, 2019.

BACK A., *Fungibility*, *Privacy and Identity*, www.youtube.com/watch?v=3dAdI3Gzodo, 2014.

BECHINI U. CIGNARELLA M. C., Quesito Antiriciclaggio n. 3-2018/B, Consiglio nazionale del Notariato, www.notariato.it, 2018.

BIJUESCA M. B. A., *SMART CONTRACTS: Is the Law Ready?*, www.digitalchamber.org, 2018.

BOMPREZZI C., FINOCCHIARO G, *A legal analysis of the blockchain technology for the formation of smart legal contracts*, in *Medialaws,* 2020.

BOMPREZZI C., *Commento in materia di Blockchain e Smart contract alla luce del nuovo Decreto Semplificazioni*, in *Diritto, Mercato e Tecnologia,* www.dimt.it, 2019

BUTERIN V., *A Next-Generation Smart Contract and Decentralized Application Platform,* www.ethereum.org, 2013

CALDARELLI G., *Real-world blockchain applications under the lens of the oracle problem. A systematic literature review,* in *2020 IEEE International Conference on Technology Management, Operations and Decisions*, 2020.

CALDARELLI G. et al, *Overcoming the Blockchain Oracle Problem in the Traceability of Non-Fungible Products*, in *Sustainability,* 2020.

CARLSON W. L. et al., *Statistica – seconda edizione,* Pearson, 2010.

CATCHLOVE P., *Smart Contracts: A New Era of Contract Use,* Available at www.papers.ssrn.com/abstract_id=3090226, 2017.

COLEMAN J., KRAUS J., *Rethinking the Theory of Legal Rights*, in *The Yale Law Journal 95,* 1986.

CRAFA S., LANEVE C., SARTOR G., *Pacta Sunt Servanda: Smart Legal Contracts in Stipula,* 2021.

CURRAN B., *What are Oracles? Smart Contracts, Chainlink & "The Oracle Problem",* available at www.blockonomi.com, 2018.

DE FILIPPI P., WRIGHT A., *Decentralized Blockchain Technology And The Rise of Lex Cryptographia*, available at SSRN www.ssrn.com/abstract=2580664, 2015.

DE FILIPPI P., WRIGHT A., *Blockchain and the Law – The rule of Code,* Cambridge (MA), Oxford University Press, 2018.

DE LY F., FONTAINE M., *Drafting International Contracts,* Leiden, Brill Publishers, 2009.

DI GIANDOMENICO G., *I Contratti Aleatori,* Torino, Giappichelli, 2005.

DUROVIC M., JANSENN A., *The formation of Blockchain-based smart contracts in the light of contract law,* in *European Review of Private law,* 2018.

DUROVIC M., LECH F., *The Enforceability of Smart Contracts,* in *The Italian Law Journal,* 2019.

EENMAA-DIMITRIEVA H., SCHMIDT-KESSEN M. J., *Creating markets in no-trust environments: The law and economics of smart contracts*, in *Computer Law and Security Review*, 2019.

EGBERTS A., *The Oracle Problem an Analysis of How Blockchain Oracles Undermine the Advantages Of Decentralized Ledger Systems*, available at www.ssrn.com/abstract=3382343, 2019.

FALKON S., *The Story of the DAO - Its History and Consequences*, www.medium.com, 2017.

FAUCEGLIA D., *Il Problema dell'Integrazione dello Smart Contract,* in *I Contratti,* 2020.

FERRO E. et al., *Tokenization and Blockchain Tokens Classification: a morphological framework,* IEEE Symposium on Computers and Communications (ISCC), 2020.

FRANCO P., *Understanding Bitcoin,* Cornwall, John Wiley & Sons Inc, 2015.

GIULIANO M., *La Blockchain E Gli smart Contracts nell'innovazione Del Diritto Nel Terzo Millennio,* in *Il diritto dell'informazione e dell'informatica,* 2018.

GOVERNATORI G. et al., *Evaluation of Logic-Based Smart Contracts for Blockchain Systems, RuleML,* 2016.

GRINCALAITIS M., *Can a Smart Contract be Upgraded/Modified?*, www.medium.com, 2018.

GUADAMUZ A., *All Watched Over by Machines of Loving Grace: A Critical Look at Smart*
Contracts, in *Computer law and Security Review,* 2019.

GÜÇLÜTÜRK O. G., *Smart Contracts and Legal Challenges,* www.medium.com, 2018.

JACCARD G. O. B., *Smart Contracts and The Role of Law,* available at SSRN: www.ssrn.com/abstract=3099885, 2017.

JANSEN N., ZIMMERMANN R., *A European Civil Code In All But Name": Discussing The Nature And Purposes Of The Draft Common Frame Of Reference*, in *The Cambridge Law Journal*, 2010.

JUELS A., MARINO B., *Setting Standards for Altering and Undoing Smart Contracts*, in *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, 2016.

KADES E., *The Laws of Complexity & the Complexity of Laws: The Implications of Computational Complexity Theory for the Law,* in *Rutgers Law Review,* 1997.

KENNEDY D., *Legal formality*, in *The Journal of Legal Studies,* 1973.

KUMAR A. et al., *Is Blockchain a Silver Bullet for Supply Chain Management? Technical Challenges and Research Opportunities,* in *Decision Science,* 2019.

LEVY K. E. C., *Book-Smart, Not Street-Smart:Blockchain-Based Smart Contracts and The Social Workings of Law,* in *Eng. Science, Technology, and Society,* 2017.

LIM C. et al., *Smart Contracts: Bridging the Gap Between Expectation and Reality, Oxford Business Law Blog* (www.law.ox.ac.uk/business-law-blog), 2016.

LIPSHAW J. M., *The persistence of "Dumb" contracts,* in *Stanford Journal of Blockchain Law & Policy,* 2019.

MANENTE M., Studio 1_2019 DI L. 12/2019 – *Smart Contract E Tecnologie Basate Su Registri Distribuiti, Consiglio Nazionale del notariato*, www.notariato.it, 2019.

MIK E., *Smart Contracts: Terminology, Technical Limitations and Real-World Complexity,* in *Law, Innovation and Technology,* 2017.

NAKAMOTO S., *Bitcoin: A Peer-To-Peer Electronic Cash System,* www.bitcoin.org/bitcoin.pdf*,* 2008.

NATH K., *State Machine Design pattern - Part 1: When, Why & How*, www.medium.com, 2019.

NICOLO' R., voce *Alea*, a cura di CALASSO F., *Enciclopedia del Diritto,* Varese, Giuffrè editore, 1958.

NICOTRA M., SARZANA DI SANT'IPPOLITO F., *Diritto della Blockchain, Intelligenza Artificiale e IoT*, Vicenza, Ipsoa, 2018.

O'SHIELDS R., *Smart Contracts: Legal Agreements for the Blockchain*, 21 North Carolina Banking Inst. 177, 2017.

PEREIRA J. et al, *Blockchain-based platforms: Decentralized infrastructures and its boundary conditions*, in *Technological Forecasting & Social Change,* 2019.

RASKIN M., *The Law and Legality of Smart Contracts,* in *Georgetown Law Technology Review,* 2017.

REHFUSS J., *Contracting Out and Accountability in State and Local Governments-The Importance of Contract Monitoring,* in *State & Local Government Review,* 1990.

RIKKEN O.et al, *Smart contracts as a specific application of blockchain technology,* Dutch Blockchain Coalition, 2017.

RINALDI G., *Smart Contract: Meccanizzazione Del Contratto Nel Paradigma Della Blockchain*, www.academia.edu, 2019.

ROMANO F., *Obbligo [XXIX,1979],* in *Enciclopedia del Diritto,* Giuffrè Francis Lefebvre, 1979.

ROSARIO N., *Alea [I,1958]*, in *Enciclopedia del Diritto,* Giuffrè Francis Lefebvre, 1958.

SAMMARCO P. E., *I Contratti dell'Informatica,* a cura di LIPARI N., RESCIGNO P., *Diritto Civile, Volume III – Obbligazioni, III - I Contratti,* Milano, Giuffrè Editore, 2009.

SAVELYEV A., *Contract Law 2.0: «Smart» Contracts as The Beginning Of The End Of Classic Contract Law,* in *Information & Communications Technology Law,* 2016.

SCHOLZ L.H., *Algorithmic Contracts*, in *Stanford Technology Law Review,* 2017.

SKARLOFF J., *Smart Contracts and The Cost of Inflexibility,* in *University of Pennsylvania Law Review,* 2017.

SUMMER R. S., *The Formal Character of Law,* in *The Cambridge Law Journal,* 1992.

SURDEN H., *Computable contracts*, in *UC Davis Law Review,* 2012.

SZABO N., *Smart Contracts, www.fon.hum.uva.nl,* 1994.

SZABO N., *Smart Contracts: Building Blocks for Digital Markets*, *www.fon.hum.uva.nl,* 1996.

TAPSCOTT D., *Token Taxonomy,* The blockchain research institute, www.blockchainresearchinstitute.org, 2020,

TAPSCOTT D., TAPSCOTT A., *Blockchain Revolution,* New York, Penguin Random House, 2016.

THEVENARD J., *Decentralised Oracles: a comprehensive overview,* www.medium.com, 2019.

WERBACH K., CORNELL N., *Contracts ex Machina, Duke law journal,* 2017.

WRÓBLEWSKI J., *Legal Language and Legal Interpretation,* in *Law and Philosophy*, 1985.