Proceedings of the 53rd Hawaii International Conference on System Sciences | 2020

The Intelligible Contract

Luca Cervone
University of Bologna
luca.cervone@unibo.it

Monica Palmirani
University of Bologna
monica.palmirani@unibo.it

Fabio Vitali
University of Bologna
fabio.vitali@unibo.it

Abstract

This paper introduces a novel model of legal digital contracts automatically executable on Blockchain technologies. Legally enforceable and automatically executable digital contracts are receiving a renewed interest, mostly due to the increasing attention towards Blockchain technologies. However, current implementations of digital contracts are still far from being intelligible to humans, because of their differences with traditional written contracts. The main purpose of this paper is to provide a contribution in bridging the gap between traditional contracts and digital contracts towards the goal of making them intelligible and legal valid. Firstly, after highlighting the shortcomings in current technologies when used for legal digital contracts, the paper introduces a new generic specification for legal digital contracts, namely the Intelligible Contract*. Secondly, we describe our implementation of Intelligible Contracts and, as a proof of feasibility, we model a simple scenario by means of our implementation. The paper concludes with some ideas for future research on Intelligible Contracts.*

1. Introduction

In the last decade, emerging technologies such as Blockchains and Smart Contracts have created new paradigms for implementing and delivering digital applications. In their very first implementations, Blockchains were thought as software infrastructure to enable peer-to-peer transactions of digital currency in decentralized ways, and so without the need of any central verifying authority [1].

Nowadays, after three generations [2], Blockchains are able to serve "*decentralized applications*", or *DApps*, software that rely on atomic transactions performed by means of Blockchains [3], or to handle "*Decentralized Autonomous Organizations*", or DAOs [4], that use Blockchains to hard-code internal rules to make their decision-making processes automatic and more traceable and accountable, or to deliver new businesses [5].

Smart Contracts are small software procedures that can run on Blockchains to ensure the correct execution of these new kinds of applications and organizations [6], or to translate certain legal contractual terms and conditions of services they offer into safe-to-execute code. In this latter situation, a Smart Contract is considered a specific interpretation and translation (a *codification*) of its corresponding legal prose, which is the written expression of a mutual assent about the contractual terms (e.g. the *considerations* of a contract) [7]. However, legal contracts need to be compliant with a complex hierarchy of laws and regulations at the local, national and international level [8], and there is a vast range of civil and penal issues that DApps and DAOs need to take into consideration within every execution context.

As such, we argue that both the intelligibility of the legal prose of contracts and the intelligibility of the whole contractual process must be guaranteed as much as possible, in order to support the enforceability of digital contracts, and to allow legal experts to back-trace what caused an eventual legal dispute. Indeed, the intelligibility of a legal contract is a mandatory requirement[1] in order to have a full awareness of the content and a valid mutual assent, and so, for instance, in order to asses the enforceability of *obligations* contained in contracts. Thus, if unintelligible to humans due to the use of computer code, a legal contract is prone to be a void or a voidable contract (e.g. a *contract mistake* in contract law).

Ricardian Contracts [9] bridge the gap between legal prose and operational code by exposing an open specification for creating "digitally-signed legal contracts" that contain both the legal prose and the

[1]For instance, in EU the *Intelligibility* is a requirement in European eCommerce Law that guarantees transparency and fairness to the consumers, see the Directive 93/13/EEC, Article 5: *"In the case of contracts where all or certain terms offered to the consumer are in writing, these terms must always be drafted in plain, intelligible language. Where there is doubt about the meaning of a term, the interpretation most favourable to the consumer shall prevail."*

URI: https://hdl.handle.net/10125/63959
978-0-9981331-3-3
(CC BY-NC-ND 4.0)

executable code and other information bits that ensure human as well as computer readability. Ricardian Contracts establish therefore a first necessary step forward to intelligible and legally enforceable Smart Contracts, but they still lack in representational power for the complex technicalities that traditional written contracts may contain in their legal prose [10]. Among the many re-interpretations and extensions of Ricardian Contracts, the *Smart Contracts Templates* specification, and its implementation, introduce essential requirements and design options to better address the serialization of the legal prose of contracts and their link to operational code [11]. Unfortunately, Smart Legal Templates still lack in important aspects with regards to intelligibleness of contracts to humans and their readability by machines. Hence, the first research question that we try to address in this paper is the following:

> ***RQ1***: Is it possible to define a model for legal digital contracts that are intelligible to humans and readable by machines?

Our proposal is another type of Ricardian Contract that we ***purposefully*** call "Intelligible Contract"[2]. In our model, not simply the specific clauses of the contract, but the whole legal context of the contract is considered and mapped to the operational code. In addition, an explicit specification of the execution environment of the operational code itself is specified. After having exposed our model we tried to address our second research question, namely:

> ***RQ2***: Is it possible to use standard technologies to implement intelligible legal, digital and automatically executable contracts?

To address our second research question we identified standard technologies already used in legal domain, such as the Akoma Ntoso standard [12] and the LegalRuleML standard [13], and we used them in an implementation of our proposal that is suitable to model a real-world scenario. Eventually, we discussed our implementation and we highlighted possible future researches on Intelligible Contracts.

The paper is structured as follows: in the next section, we expose the methodology this paper is based on. In section 3, we provide an in-depth background of Smart Contracts, Ricardian Contracts and Smart Legal Templates. Section 4 provides definitions, requirements and implementation of our proposal, the Intelligible Contract. In section 5 we model a sample use case, as a proof of concept for Intelligible Contracts, and in section 6 we discuss our solution and we draw some conclusions and future research directions.

[2]As specified, *any* legal contract must be intelligible to both parties in order to allow them to fully comprehend their *obligations* and their rights

2. Methodology

Our research is the result of a long activity of theoretical studies on the application of modern technological concepts in legal domains, and here on reducing gaps between digital contracts and traditional legal contracts. In order to address our research questions we have: (1) reviewed the existing literature so as to identify technologies and models that have already tried to link legal prose to its operational and legal context, in particular Ricardian Contracts and Smart Contracts Templates; (2) highlighted the pros and cons of these approaches in light of issues we have exposed about the intelligibility and legal validity of contracts; (3) proposed a new model called "Intelligible Contract" that is suitable to fill the gaps we identified; (4) given a formal definition of our model in line with previous existing proposal; (5) verified the feasibility of our model by proposing possible implementations based on standardized and widely-used technologies; (6) discussed the implications of our model and our implementation, and (7) identified desirable developments for the future.

3. Background and Motivations

In this section we describe concepts and technologies that impact on the issues, and motivations behind their adoption in our research for modelling Intelligible Contracts.

3.1. From Blockchains to Smart contracts

Blockchains (hereinafter BCs) are special kinds of digital databases that brought disruptive innovations by means of few concepts.

Firstly, BCs are immutable databases, for they are append-only databases. Each new block of data carries a hash of the previous one so that the integrity of existing data can be verified every time a new block is added [1]. Thus, this mechanism creates a "*chain*" of "*blocks*" of data.

Secondly, BCs are "*distributed*" to the extent that data contained in a BC is potentially hosted on any of the nodes that belong to the network of the BC. Whenever an authority is needed to grant the access to the network then the BC is a "*permissioned*" one. If the access and the interaction with the BC is open and public, so that participants can be anonymous or pseudo-anonymous, then the BC is named "*permissionless*". [14].

Thirdly and lastly, BCs are "*decentralized*" (or at least "*partially-decentralized*"), meaning that participants can reach an agreement on the correct status of the BC without trusting each other [15].

By mixing the above concepts in many different ways, in the last decade BCs have evolved over at least three separate generations [2]. In their first generation, the *Digital Currency* generation, BCs such as BitCoins [1] were able to create digital currency and to handle transactions between users. In their second generation, the *Contracts* generation, BCs features were extended to more complex financial services, such as digital assets exchanges, loans, mortgages, and so on. In the third generation, the *Digital Society* generation, BCs are used to handle many every-day and real-life situations, introducing new paradigms to create applications or organizations, such as Decentralized Applications and Decentralized Autonomous Organizations, that can be applied in both private and institutional contexts [16].

In order to handle the complex applications they are intended to serve, BCs belonging to the second and the third generation, such as the permissionless Ethereum BC [3] and, with some simplifications, the permissioned Hyperledger BC [17], allow developers to publish Smart Contracts that can be executed in a safe and tamper-proof way.

3.2. Smart Contracts

Smart Contracts (hereinafter SCs) are small software procedures run on BCs to ensure their correct execution [6]. SCs are often used to handle the business logic of DAPPs, sometimes introducing novel models of markets and economies [2]. For instance, the CryptoKitties dApp uses SCs for trading non-fungible digital assets, or unique digital items represented as images of cute kittens, whose data and metadata are hashed and stored on BCs [5].

In simple contexts, SCs can execute automatically certain terms of legal contracts between parties, and they can enforce clauses of contracts without lawyers or central authorities [18] but, the term "*contract*" has a precise meaning in the legal world[3], while here a SC is basically a software procedure that is automatically executed whenever specific pre-programmed and pre-agreed conditions are met [8]. However, even considering the limited scope of the definition of "contract" in the BC domain, SCs must be considered as translations of fragments of the *legal prose* of a contract into an executable piece of code [7]. For this reason, given also the current hype on BCs, we argue, several countries are regulating SCs in order to give them a clear legal status[4], and in order to legally enforce the probative value of their executions (the *evidential* value) or of their life-cycle (e.g. the *rescission* from the contract, the *termination* of the contract, and so on).

Therefore, currently, SCs and more generically digital contracts, can be already used with world-wide enforceable value when they are built under some conditions, defined at international level by the UNICITRAL texts [19], following the principles of "non-discrimination" and "technological neutrality". Additionally, if the digital contract is digitally signed, for example by using technologies compliant to the eIDAS Regulation technical specifications applicable in the European jurisdiction [20], the principle of "functional equivalence" to the traditional paper contracts is maintained. In USA the Uniform Electronic Transactions Act (UETA) and Global and National Commerce Act (E-SIGN) are the legal bases for the state-by-state application of rules concerning the enforceability of the execution of SCs [21].

However, we agree with UNICITRAL experts of the eCommerce working group [22], so we argue that the real crucial issue is the *liability in case of torts*, given that, so far, SCs fail to supply the features connected to the complex hierarchy of laws and regulations of real contracts, referenced implicitly or explicitly in the legal prose they use. For instance, at the state of the art, it is not possible to assess the unequivocal and free willingness of the parties to accept the terms of contracts executed by a SCs, and this may lead to several penal and civil implications.

So, given the current implementations of SCs, many aspects are on a shaky legal ground, [23]. For these reasons, there is the need to identify technologies that allow to serialize contracts in a way that they are more intelligible and readable both by machines and by humans. We found an initial answer in Ricardian Contracts and Smart Contracts Templates, described in the next section.

[3]A legal contract is the meeting of two or many minds (mutual agreement between parties) concerning an offer that is identically accepted under considerations. For instance, a contract in Italy is valid only if some pre-conditions are respected following art. 1325 of the Italian Civil Code. One of them are the identification of parties, the expressed willingness to agree, the clearness of the scope, the definition of the object, the understandability of the conditions and, as said before, the intelligibility of the clauses and the format.

[4]For instance, the Italian Government has enacted Decree Law n. 135 at 14 December 2018, also known as "Decreto Semplificazioni" (the simplification act) that defines smart contracts as computable programs using distributed ledger technology, whose execution produces legal effects between identified parties.

3.3. Ricardian Contracts and Smart Contract Templates

In order to try to fill the gap between traditional text-based legal contracts and software for the automatic execution of their clauses, companies building BC technologies, like OpenBazaar [24], allow contracts to be described and placed directly on the chain by means of implementations of *Ricardian Contracts* (hereinafter RCs) [9]. The idea behind RCs is that a full automated and legally enforceable contract, including their execution, should be composed of "*parameters*, *code*, and *prose*" [25], linked together and signed by means of some secure cryptographic algorithm. In a nutshell, RCs are tamper-proof and digitally signed $\langle P,C,M \rangle$ triples, where: (P) describes the denotational semantics (the legal prose) of the contract; (C) is the operational semantics, and therefore the code that must be automatically executed to enforce the prose of the contract; and (M) is the mapping, in the form of key-value parameters, of operations expressed in C and the prose expressed in P [10]. Studies have already made on implementation or extensions of RCs.

Smart Contract Templates (hereinafter SCTs) are implementations of RCs whose operational code is standardized and whose behavior is controlled by parameters contained in an electronic representation of the document that also contains the legal prose of the contract [26]. The main purpose of SCTs is to facilitate the management of the whole life-cycle of digital contracts, and examples of implementations have been proposed, mostly in financial contexts [27]. Four key aspects are particularly relevant for the management of SCTs in their life-cycle [11]: (1) tools must be supplied to allow people with legal skills to write contracts in legal prose, facilitating users to link legal prose to operational code even without programming skills; (2) the serialization of contracts must be done by means of standard vocabularies, flexible enough to allow the serialization of complex legal documents; (3) contracts must be serialized by means of standard mark-up that allows linking items in the legal prose to standard ontologies, and so allowing analysis and reasoning, and (4) features must be supplied to link the legal prose to operational code, to pass parameters contained in legal prose to the operational code, and to uniquely identify operational code that executes SLTs.

Although we agree that the four requirements highlighted by STCs are essential requirements for legal digital contracts, they do not take into consideration other important aspects for the intelligibility of contracts. More specifically, we argue that a model for intelligible digital contracts must also supply features to describe: (1) other legal and non legal resources linked to contracts (for instance, *recitals* often refer to Acts that regulate the concepts involved in the contract); (2) the legal context of contracts (for instance, contracts may be challenged differently in a US country or in a EU country); (3) the operational context of contracts (i.e. on which BC an digital contract was executed); and (4) the information related to any automatic execution of contracts or any of their clauses. To fulfill these gaps, we introduce our "*Intelligible Contracts*" exposed in the next section.

4. Intelligible Contracts

Intelligible Contracts are legal contracts written in natural language that can be mapped, entirely or partially, to operational code living on BCs. Like RCs and SCTs, Intelligible Contracts supply an open specification for marking-up the legal prose of contracts by ensuring its readability also to machines, and creating and defining the bridge between partitions of legal prose and the corresponding operational code. Intelligible Contracts extend RCs and SCTs by supplying specifications for the intelligibility of digital contracts, in particular by: (4) linking all resources that compose contracts or define their legal contexts; (5) linking agents that are involved in the life-cycle of contracts; (6) linking the digital resources that describe how to execute the operational code; (7) linking the digital resources that report what happens during the executions of contracts.

The denotational definition of Intelligible Contracts is as follows[5]:

```
Intelligible Contract::=
    UID and
    Document+ and,
    Context+ and,
    Execution Report+

UID::=  URI => HASH

Context::=
    UID and
    Legal Context+ and
    Operational Context+ and

Legal Context::=
    (Legal Document Ref or
    Legal Document)+

Operational Context::=
    Operational Environment Ref+ and
    Operational Agent Ref+ and
    Operational Code Ref+

Operational Environment::= URI
```

[5]For the sake of consistency we use an Extended BNF-like form similar to the one in [11] The notation can be summarized as follows: (**::=**) means "is defined as"; (*****) means "zero or more occurrences"; (**+**) means "one or more occurrences"; if neither (*****) or (**+**), then there must be "exactly one occurrence"; (**x and y**) means "both x and y"; (**x or y**) means "x or y or both". Additionally, we use: ($\langle$**x,y, M**$\rangle$) to denote a "triple where M is a mapping among elements belonging to x and element belonging to y", and (**A** $\Rightarrow$ **B**) to denote that "A and B are both mandatory and B is in function of the content of A".

```
Operational Agent::=
    UID
    Document+

Operational Code::=
    UID
    Bit+

Document::=
    UID and
   (Generic Document or
    Generic Document Ref)+ or
   (Legal Document or
    Legal Document Ref)+

Generic Document::= Bit+

Legal Document::=
    Legal Prose+ and
    Metadata+

Legal Prose::= Human Natural Language Statement+

Metadata::=
   (Legal Metadata or
    Operational Metadata)+

Legal Metadata::=
   <Legal Prose,
    Legal Context,
    Description>

Operational Metadata::=
   <Legal Prose,
    Operational Context,
    Description>

Description::=
    Human Description+ or
    Automatic Description+

Execution Report::=
    UID,
    Document+,
```

Informally, an ***Intelligible Contract*** is a unique collection of linked machine-readable resources describing a legal contract, its legal prose, its legal context, and information on which parts of it can be automatically processed and how to do it.

Intelligible Contracts are primarily aimed to describe automatically executable legal digital contracts, but their definition is open to other uses. Implementations must satisfy a set of mandatory components: (1) an identification and referencing component (2) a document component; (3) a context component; and, (4) a process component.

Unique identifiers, or UID, are pairs composed of a Uniform Resource Identifier and a hash value generated by a hashing function on the content of the resource the URI points to. It is worth noting that, since hashes are computed on the content of resources, UIDs expect resources to be immutable. Thus, it is important that URIs reference resource as available at a specific moment in time.

Intelligible Contracts supports both generic and legal documents, as actual resources or references to external locations. Legal documents are documents whose content is needed for the intelligibility and the life-cycle of Intelligible Contracts. Legal documents must be serialized so as to make Intelligible Contracts compliant to RCs and SCTs specifications: human and machine readable, structured so as to preserve the text and the logical and semantic structure of the prose, serialized by means of standard technologies and linkable to other resources. Metadata must be specified for each legal document belonging to Intelligible Contracts. Legal metadata may include: issue date, enforcement date, signature date, amendments to document, info on its versions, etc. Operational metadata describe the prose in function of its operational context. For instance, if an Intelligible Contract is processed to assess its enforcement, then operational metadata must include relevant data about enforcement. These descriptions must be readable by both humans and machines.

Context in Intelligible Contracts is divided into legal context and operational context. The legal context is the legal environment under which the Intelligible Contract is described or executed. It contains or references legal documents, such as acts cited in the introductory text (the *recitals*), or other related contracts. It would be appropriate that all included or referenced documents are serialized using the same data format as the contract. The operational context of Intelligible Contracts is composed by references to operational parts for the execution of the contract. It contains execution parameters, references to operational environment (i.e., the platform that will store and execute the code), operational agents and operational code. Operational agents are references to the entities passively or actively involved in an execution.

Execution Reports are sets of documents identified by a UID summarizing events happened during the execution of the contract, such as a judgment, the logs created by the operational environment during a specific execution, or legal documents related to a subpoena against a failure in an execution. Report documents should be serialized by means of the same markup language of the other documents.

All the requirements described in this section and in the previous ones must be followed by any implementation of Intelligible Contracts.

5. An implementation of Intelligible Contracts

In this sections we expose an implementation of Intelligible Contracts by describing a stack of technologies that, used together, fulfill the list of requirements of Intelligible Contracts discussed in the previous section.

5.1. Akoma Ntoso

Akoma Ntoso (developed by the LegalDocML Technical Committee of OASIS) is an XML OASIS standard for modelling legal resources [12]. Akoma Ntoso (hereinafter AKN) has proved to be effective in several diverse legal and non-legal contexts, such as modelling laws [28] [29], modelling legal changes over time [30], modelling legal documents by inter-governmental institutions [31], modelling legal documents facilities management [32], and modelling interactions among citizens in deliberative systems [33]. The AKN standard is composed of two specifications: (1) an XML vocabulary for structuring legal documents [34], and (2) a FRBR-based naming convention for identification of legal resources [35]. In addition, AKN supplies an informal ontology to describe relevant entities and to connect them to specific parts of the legal text [36]. Akoma Ntoso can fulfill most of the requirements of Intelligible Contracts, and several tools for handling AKN documents already exist (i.e. the LIME editor `http://lime.cirsfid.unibo.it` and the akomando framework `http://akomando.bitnomos.eu`).

5.2. The InterPlanetary Linked Data

The InterPlanetary Linked Data (hereinafter IPLD) defines a set of standards and technologies that can be used to create universally addressable data structures (`https://github.com/ipld/specs`). In a nutshell, IPLD allows to link resources identified by hashes that can refer to diverse resources, like SCs on Ethereum or content on the InterPlanetary File System (hereinafter IPFS). The IPFS is a distributed file system on which it is possible to put versioned content and other data and software that base their functioning on tamper-proof hashing systems (`https://ipfs.io`).

5.3. LegalRuleML

The LegalRuleML (hereinafter LRML) OASIS standard [37] is a interchange language for rules in the legal domain that is both human readable and machine readable [38]. By supplying a rich XML vocabulary, the LRML standard allows Legal Knowledge Engineers to re-express the legal prose contained in legal document, highlighting business rules and connecting them to automatic legal reasoners enriched by Linked Open Data information [13]. Several studies exist that show how to execute automatic legal reasoning by means of LRML [39]. Additionally, engines for automatic legal reasoning using LRML files have been proposed (i.e SPINdle), as well as editors to facilitate the creation and editing of LRML files side-by-side with AKN files [40].

5.4. An example scenario: Data Processing Agreements

In this subsection, we show fragment of code (and concepts) related to a Data Processing Agreement (hereinafter DPA) compliant to the General Data Protection Regulation (GDPR) of the EU regulation[6]. There are several motivations to choose this scenario: first, the academical interest in GDPR-compliant BCs applications [41] and, we believe, an intelligible representation of GDPR-compliant legal contracts may help to address the issue. Secondly, DPAs are complex contracts involving several legal and non-legal resources, such as scheduling and privacy policies of services, and several entities and concepts, such as users, processors and so on. Moreover, DPA may be affected by many jurisdictions, because it is often related to world-wide services. Thirdly, a template of the legal prose of GDPR-compliant DPA is publicly available at: `https://gdpr.eu/data-processing-agreement/`.

The next section shows how the naming convention of AKN may be used for an Intelligible Contracts that serializes a DPA template.

5.4.1. Identification and References The AKN naming convention allows to specify human readable URI that are useful to represent the UIDs of contracts and their documents, to specify hierarchical relations between documents, to allow versioning of Intelligible Contracts, to express references to legal documents belonging to the legal context, and to express references within the operational context. The AKN naming convention organizes resources hierarchically in four concepts, the work, the expression, the manifestation, and the item [35] according to the FRBR conceptual model [42]. The work is the most abstract concept of the legal resource. The expression is used to identify specific temporal and linguistic versions of the legal document. The manifestation is a serialization of a version in a specific data format. The item is a physical file where a manifestation is stored.

In our scenario, the AKN identifiers of an Intelligible Contract representing a DPA could be:

Work:

[6]The Regulation (EU) 2016/679 (General Data Protection Regulation) entered into operation at 28 May 2018 with enforcebility in all the EU countries and also to those controllers that operate with European Personal Data.

1. /akn/it/documentCollection/dpa/company/2019-07-12/1

Expressions:

1. /akn/it/documentCollection/dpa/company/2019-07-12/1/ita@2019-09-12
2. /akn/it/documentCollection/dpa/company/2019-07-12/1/eng@2019-11-12/!main
3. /akn/it/documentCollection/dpa/company/2019-07-12/1/eng@2019-11-12/!schedule_1

Manifestations:

1. /akn/it/documentCollection/contract/company/2019-07-12/1/ita@2019-09-12/!main.akn
2. /akn/it/documentCollection/contract/company/2019-07-12/1/eng@2019-11-12/!main.akn
3. /akn/it/documentCollection/contract/company/2019-07-12/1/eng@2019-11-12/!schedule_1.akn

The syntax of the Naming convention allows us to conclude, for instance, that manifestation 3 embodies expression 3 which realizes work 1, etc.

The AKN naming convention can be used to describe the whole tree of resources of an Intelligible Contract, including documents belonging to the legal domain and to the operational domain. The DPA of our scenario explicitly refers to the GDPR in its recitals[7], and the Intelligible Contract can reference to its Work as `/akn/eu/act/regulation/eu/2016-04-05/2016-679/!main` using the AKN Naming Convention.

Entities and concepts belonging to the operational context can also be mentioned in the same way by means of the informal ontology supplied with AKN [36]. For instance, the "company" that acts as the processor (in our specification this is an operational agent) can be identified with a URI such as `/akn/ontology/organizations/eu/companyXY`, and the legal concept of "being a minor" as used in a privacy policy can be identified by a URI such as `/akn/ontology/concept/eu/PrOnto/minor` [43].

[7] **WHEREAS** *omissis* (C) The Parties ...*omissis*... in relation to data processing and with the Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 ...*omissis*... Directive 95/46/EC (General Data Protection Regulation). *omissis*"

5.4.2. Using the InterPlanetary Linked Data Intelligible Contracts need that hashes are calculated on the content of resources, so it is not feasible to put the tamper-proof hash representing itself inside an Intelligible Contract. We must find a way to compute hash of concepts represented by AKN-style URIs that, if needed, can resolve to the hash of the actual content of documents identified by URI.

IPLD and IPFS can be used together to ensure that a logical object always map to the same physical digital object. For instance, when adding to the IPFS a nested folder structure that describes an expression of the DPA, then there a logical object will be created on the IPFS with its own hash. Anyone uploading the same directory structure will obtain the same hash. But if a change happens to the folder, e.g., if the AKN file representing the DPA is inserted with the name dpa.akn, and the folder is uploaded again to the IPFS, we have an hash for the AKN file but the hash of the folder will show the change. Thus we have an immutable addressing system composed of the hash of the content and the URI pointing to the content.

5.4.3. Using the Akoma Ntoso XML Vocabulary All documents or resources referenced in an Intelligible Contract must always use a double reference, both a readable URI and a hash of its content. As such, the markup language must support both. The AKN XML vocabulary, aimed to associate XML markup to legal and parliamentary resources, is highly extensible and customizable [34]. Although it does not supply a document type for contracts, it does supply many elements that are typical of legal documents that can be used to mark up the partitions of the legal prose of a contract. According to their definition in section 3, Intelligible Contracts are "document collections", which are explicitly supported in AKN with its own specific document type. Fragments in legal prose can be connected to legal contexts and operational contexts of Intelligible Contracts by linking texts and concepts to pairs of URIs and hashes, which together compose the UIDs in the Intelligible Contract terminology.

The following XML code contains a fragment of the recitals of DPA:

```
<recital>
 <num>(C)</num>
 <p>
  The Parties ...omissis... in relation to data processing
  and with the
<ref refersTo="#gdpr-expression"
href="/akn/eu/act/regulation/eu/2016-04-05/679@2018-05-25">
    Regulation (EU) 2016/679 of the European Parliament and
    of the Council of 27 April 2016
  </ref>
  ...omissis...
  of the Directive 95/46/EC (General Data Protection
  Regulation).
 </p>
</recital>
```

The ref elements that wraps the GDPR regulation points to the work URI of the GDPR as well as to an internal id (in its refersTo attribute). The refersTo attribute points to another fragment of XML that contains the hash of the document expressed as yet another URI:

```
<references source="#editor">
 <TLCReference
   GUID="gdpr-expression"
    name="GDPR"
    showAs="General Data Protection Rule"
    href="/akn/references/expression/eu/gdpr/ipfs/QmU...A3Nn">
 </TLCReference>
</references>
```

The reference to the GDPR is thus bound to both a human readable URI and to a tamper-prof hash, in accordance with the Intelligible Contracts requirements.

5.4.4. Using LegalRuleML

We now explain how to link the legal prose and other metadata to actual operational code, such as smart contracts residing on the block chain, using LRML, that is our last brick for bridging the legal prose to operational code.

For a very basic example, we consider a clause of the terms and conditions of a service of the information company that collects personal data from the data subject:

> "**Users of the Service** - This Service is provided exclusively to individuals who are not minors and live in a EU country or in the US."

This clause involves non-trivial legal concepts that may be difficult to resolve. Firstly the concept of "*minor*", secondly, the concept of "*living in a country*", and thirdly, the concept of minor must be evaluated with regard to the country in which the person is living. For instance, if the person is a US citizen, the definition of minor is provided by the "*Children's Online Privacy Protection Rule*" (COPPA), whereas, if the user is an EU citizen, the concept of minor will be regulated by the GDPR itself.

```
...omissis...
<meta>
...omissis...
<references source="#editor">
 <TLCReference eId="gdpr-expression" name="GDPR"
showAs="General Data Protection Rule"
href="/akn/ontology/references/expressions/eu/gdpr/ipfs/
QmU...A3Nn" />
 <TLCLocation eId="europe" showAs="European Union"
   href="/akn/ontology/locations/eu" />
 <TLCLocation eId="us" showAs="United States of America"
   href="/akn/ontology/locations/us" />
 <TLCConcept eId="livingInACountry" showAs="lives in"
    href="/akn/ontology/concept/livingInACountry" />
 <TLCConcept eId="minor" showAs="minor"
    href="/akn/ontology/concept/minor" />
</references>
...omissis...
<meta>
...omissis...
<clause eId="cls_6">
 <heading>
  Users of the Service
 </heading>
 <content>
  <p>
   This Service is provided exclusively to individuals who
   are not <concept referstTo="#minor">minors</concept> and
   <concept refersTo="#livingInACountry">live in
   </concept>  a  <location refersTo="#eu">
   EU country </location>  or in the
   <location refersTo="#us"> US </location>.
   </p>
  </content>
</clause>
...omissis...
```

```
...omissis...
 <lrml:Comment> GDPR - minor consent </lrml:Comment>
 <lrml:LegalReferences
 refType="http://example.org/lrml#LegalSource">
  <lrml:LegalReference
  refersTo="ref1"
  refID="/akn/eu/act/regulation/2016-04-27/2016-679/eng@2018-
  05-25/!main#art_8__para_1"
  refIDSystemName="AkomaNtoso3.0-2017-06"  />
 </lrml:LegalReferences>
 ... omissis ...
 <lrml:Statements >
 <lrml:PrescriptiveStatement key="ps1">
  <ruleml:Rule
   key=":ruletemplate2"
   closure="universal">
   ...omissis...
   <ruleml:if>
    <ruleml:And key=":and1">
     <ruleml:Atom key=":atom1">
      <ruleml:Rel iri=":child" />
      <ruleml:Var >X</ruleml:Var>
     </ruleml:Atom>
     <ruleml:Atom key=":atom2">
      <ruleml:Rel iri=":atLeast16years" />
      <ruleml:Var >X</ruleml:Var>
     </ruleml:Atom>
     <ruleml:Atom key=":atom4">
      <ruleml:Rel iri=":informationSocietyService" />
      <ruleml:Var >D</ruleml:Var>
      <ruleml:Var >S</ruleml:Var>
     </ruleml:Atom>
     <ruleml:Atom key=":atom5">
      <ruleml:Rel iri=":Controller" />
      <ruleml:Var >Y</ruleml:Var>
      <ruleml:Var >S</ruleml:Var>
     </ruleml:Atom>
    </ruleml:And>
   </ruleml:if>
   <ruleml:then>
    <lrml:Obligation iri=":obligation">
     <ruleml:Atom key=":atom6">
      <ruleml:Rel iri=":ObtainConsent" />
      <ruleml:Var >X</ruleml:Var>
      <ruleml:Var >Y</ruleml:Var>
      <ruleml:Var >S</ruleml:Var>
     </ruleml:Atom>
    </lrml:Obligation>
   </ruleml:then>
  </ruleml:Rule>
 </lrml:PrescriptiveStatement>
</lrml:Statements>
...omissis...
```

The above listings show, respectively, how to mark up in AKN the clauses of the terms and conditions for minors, and the mapping from the condition to concepts expressed in the GDPR marked up by means of the LRML. The second fragment can be seen as "an interpretation of the article 8 of the GDPR in function of the terms and conditions of the Intelligible Contract". This demonstrate how the LRML standard supplies the last necessary technology to Intelligible Contracts to be at the same time human and machine readable. In the next section we discuss implications of introducing implementations of Intelligible Contracts in BCs.

6. Discussions and Conclusions

In this paper we have exposed the "*Intelligible Contract*", a novel approach to serialize automatically executable legal digital contracts that can be understood both by humans and by machines.

We argue that our model and definition of Intelligible Contracts, exposed in section 4, replies to our first research question (***RQ1***). Firstly, indeed, Intelligible Contracts are intelligible to humans because they can be written in legal prose without omitting important fragments of the legal text (i.e. recitals, citations and so on). Secondly, Intelligible Contracts are understandable by machines because their legal prose and legal context must be serialized in machine-readable formats.

In section 5 we answered to our second research question (***RQ2***) by identifying standard technologies to implement Intelligible Contracts. Indeed, with our implementation we shown that: (1) AKN can be used to serialize the legal prose of contracts and to serialize and link other legal and non legal documents; (2) the AKN naming convention and IPLD technologies can be used together to uniquely identify contracts and their related resources; (3) LRML can be used to map the legal prose of contracts to its legal and operational contexts.

It is worth noting that our implementation is environment agnostic, meaning that it does not supply any specification on how to store and handle LRML and AKN documents. As a sample, the architecture depicted in Figure 1 is suitable to handle a simplification of the scenario described in section 5. Figure 1, is also intended to supply a more clear sight on how components of Intelligible Contracts can be pieced together in a (partially) distributed and (partially) decentralized system.

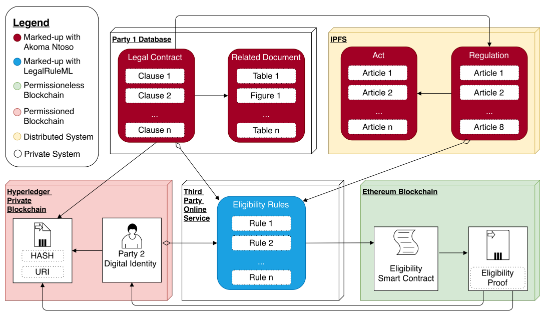

Figure 1. A sample architecture for handling Intelligible Contracts.

In the architecture depicted in Figure 1, the legal contract and its related resources are marked-up in AKN and stored in a private database. The legal contract is then referenced, by means of its hash and its URIs, in a centralized BC (based on Hyperledger), that also contains info of involved parties. An external online service uses a reasoner (such as SPINdle) and a LRML document to verify if the party two is a minor. Rules for this check are retrieved from the legal contract and from regulations and laws that can be stored on a distributed database (i.e. IPFS). Eventually, the service calls a SC that creates a immutable "*eligibility token*" on the BC whose ownership means that the party is eligible to sign that contract in that legal and operational context. The eligibility SC is a simple and standardizable piece of code, and because the proof of eligibility must be publicly available (as a requirement of this fictitious scenario), the SC can be stored on a public BC (i.e Ethereum).

We argue that Intelligible Contracts could be a first brick to fill several technical and legal gaps related to using digital contracts in BCs. For instance, Intelligible Contracts, and our implementation of them, can help to analyze lack o willness of parties, to analyze liability in case of torts, and to overcame current limitation derived by the immutability of BCs (by supplying a versioning system inherited by the adoption of AKN and its naming convention).

Our work is yet to be completed. In the future we plan: (1) to model and implement full real-world scenario in order analyze pros and contra of using Intelligible Contracts; (2) to investigate benefits or limitations of Intelligible Contracts in relation to specific BCs environments (i.e. permissioned vs. permissionles); (3) to further customize AKN for better modelling the concept of contracts according to private law theory, and (4) to address the standardization of operational code that executes Intelligible Contracts on BCs.

References

[1] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] M. Swan, *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[3] V. Buterin *et al.*, "Ethereum white paper: a next generation smart contract & decentralized application platform," *First version*, 2014.

[4] S. Voshmgir, "Tokenized networks: What is a dao?," 2019.

[5] K. Turner, "The cryptokitties genome project," December 2017.

[6] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts: platforms, applications, and design

patterns," in *International Conference on Financial Cryptography and Data Security*, pp. 494–509, Springer, 2017.

[7] S. Murphy and C. Cooper, "Can smart contracts be legally binding contracts?," *white paper, R3cev and Norton Rose Fulbright*, 2016.

[8] M. Raskin, "The law and legality of smart contracts," 2016.

[9] I. Grigg, "The ricardian contract," in *Proceedings. First IEEE International Workshop on Electronic Contracting, 2004.*, pp. 25–31, IEEE, 2004.

[10] F. Al Khalil, T. Butler, L. O'Brien, and M. Ceci, "Trust in smart contracts is a process, as well," in *International Conference on Financial Cryptography and Data Security*, pp. 510–519, Springer, 2017.

[11] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates: essential requirements and design options," *arXiv preprint arXiv:1612.04496*, 2016.

[12] M. Palmirani and F. Vitali, "Akoma-ntoso for legal documents," in *Legislative XML for the semantic Web*, pp. 75–100, Springer, 2011.

[13] M. Palmirani, G. Governatori, A. Rotolo, S. Tabet, H. Boley, and A. Paschke, "Legalruleml: Xml-based rules and norms," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pp. 298–312, Springer, 2011.

[14] M. Vukolić, "Rethinking permissioned blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pp. 3–7, ACM, 2017.

[15] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *2017 IEEE International Conference on Software Architecture (ICSA)*, pp. 243–252, IEEE, 2017.

[16] D. Efanov and P. Roschin, "The all-pervasiveness of the blockchain technology," *Procedia Computer Science*, vol. 123, pp. 116–121, 2018.

[17] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310, 2016.

[18] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman, *et al.*, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, vol. 2, no. 6-10, p. 71, 2016.

[19] U. N. C. on International Trade Law, *United Nations Convention on the Use of Electronic Communications in International Contracts*. United Nations Publications, 2007.

[20] "Regulation (eu) no 910/2014 of the european parliament and of the council of 23 july 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec," July 2014.

[21] J. A. Beckham, M. Sendra, and T. Greenberg, "Smart contracts lead the way to blockchain implementation," *THOMSON REUTERS*, 2019.

[22] A. Mukherjee, "Smart contracts – another feather in uncitral's cap," February 2018.

[23] R. O'Shields, "Smart contracts: Legal agreements for the blockchain.," *NC Banking Inst.*, p. 177, 2017.

[24] H. Subramanian, "Decentralized blockchain-based electronic marketplaces.," *Commun. ACM*, vol. 61, no. 1, pp. 78–84, 2018.

[25] J. Hazard and H. Haapio, "Wise contracts: smart contracts that work for people and machines," 2017.

[26] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates: foundations, design landscape and research directions," *arXiv preprint arXiv:1608.00771*, 2016.

[27] L. Braine, "Barclays," *Smart Contract Templates', Barclays London Accelerator, available at: https://vimeo. com/168844103/and www. ibtimes. co. uk/barclays-smart-contract-templates-heralds-first-everpublic-demo-r3s-corda-platform-1555329/*, 2016.

[28] K. Gen, N. Akira, M. Makoto, O. Yasuhiro, O. Tomohiro, and T. Katsuhiko, "Applying the akoma ntoso xml schema to japanese legislation," *JL Inf. & Sci.*, vol. 24, p. 49, 2015.

[29] M. Palmirani, F. Vitali, A. Bernasconi, and L. Gambazzi, "Swiss federal publication workflow with akoma ntoso.," in *JURIX*, pp. 179–184, 2014.

[30] M. Palmirani, "Legislative change management with akoma-ntoso," in *Legislative XML for the semantic Web*, pp. 101–130, Springer, 2011.

[31] S. Peroni, M. Palmirani, and F. Vitali, "Undo: the united nations system document ontology," in *International Semantic Web Conference*, pp. 175–183, Springer, 2017.

[32] J. Dimyadi, G. Governatori, and R. Amor, "Evaluating legaldocml and legalruleml as a standard for sharing normative information in the aec/fm domain," in *Proceedings of the Lean and Computing in Construction Congress (LC3)(to appear, 2017)*, 2017.

[33] L. Cervone, *Digital Technologies for Deliberative Democracies: Models and Applications for Continuous Civic Engagement*. PhD thesis, alma, 2017.

[34] M. Palmirani, R. Sperberg, G. Vergottini, and F. Vitali, "Akoma ntoso version 1.0 part 1: Xml vocabulary," OASIS standard, August 2018.

[35] F. Vitali, M. Palmirani, and V. Parisse, "Akoma ntoso naming convention version 1.0," OASIS standard, February 2019.

[36] F. Vitali, M. Palmirani, R. Sperberg, and V. Parisse, "Akoma ntoso version 1.0. part 2: Specifications," OASIS standard, August 2018.

[37] M. Palmirani, G. Governatori, T. Athan, H. Boley, A. Paschke, and A. Wyner, "Legalruleml core specification version 1.0.," OASIS standard, May 2017.

[38] T. Athan, H. Boley, G. Governatori, M. Palmirani, A. Paschke, and A. Z. Wyner, "Oasis legalruleml.," in *ICAIL*, vol. 13, pp. 3–12, 2013.

[39] T. Athan, G. Governatori, M. Palmirani, A. Paschke, A. Z. Wyner, *et al.*, "Legal interpretations in legalruleml.," in *SW4LAW+ DC@ JURIX*, 2014.

[40] M. Palmirani, L. Cervone, O. Bujor, and M. Chiappetta, "Rawe: An editor for rule markup of legal texts.," in *RuleML (2)*, 2013.

[41] M. Finck, *Blockchain regulation and governance in Europe*. Cambridge University Press, 2018.

[42] B. Tillett, "What is frbr? a conceptual model for the bibliographic universe," *The Australian Library Journal*, vol. 54, no. 1, pp. 24–30, 2005.

[43] M. Palmirani, M. Martoni, A. Rossi, C. Bartolini, and L. Robaldo, "Legal ontology for modelling gdpr concepts and norms.," in *JURIX*, pp. 91–100, 2018.