A Legal-Relationship Establishment in Smart Contracts: Ontological Semantics for Programming-Language Development

Published in:
Advances in Computing and Data Sciences. ICACDS 2021

Publication Status:
Published (in print/issue): 23/10/2021

DOI:
10.1007/978-3-030-81462-5_58

A Legal-Relationship Establishment in Smart Contracts: Ontological Semantics for Programming-Language Development

Vimal Dwivedi[0000−0001−9177−8341] and Alex Norta[0000−0003−0593−8244]

Tallinn University of Technology, Akadeemia tee 15 a, Tallinn, Estonia
vimal.dwivedi@taltech.ee
alex.norta.phd@ieee.org

Abstract. Machine-readable smart contracts (SC) on blockchains promise drastic enhancements in collaboration efficiency and effectiveness in that cost- and time reductions can be achieved while the quality of services increases. We address existing shortcomings of SCs that are in tendency incomplete for legal recognition especially to smart-contract-enabled funding rounds, not collaborative business-process reflective and are also not aware of their own processing state to justify the claim of smartness. When conflicts occur, tracing the past performance of conventional contract (CC) execution is very slow and expensive while in addition, CCs are challenging to enforce. On the one hand, the legal status of SCs based funding rounds is currently not clarified and the question arises if SCs comprise the necessary legal- concepts and properties. Current SC solutions do not suffice in those regards. To fill this gap, we develop the smart-legal-contract (SCL) ontology to define the legal- and collaborative business concepts and properties in the SCs. Formal methods, such as Colored Petri Nets (CPNs), are suitable to design, develop and analyze processing state of SCs in order to trace the performance of contractual- rights and obligations. In this work, SCL ontology is formalized using Colored Petri Nets resulting in a verifiable CPN model. Furthermore, we conduct a state-space analysis on the resulting CPN model and derive specific model properties. A running case from the automotive supply chain domain demonstrates the utility and validity of our approach.

Keywords: Smart contract · decentralized autonomous organization · legal recognition · blockchain · ontology · business process · B2B .

1 Introduction

Traditionally the concept of the contract covers a spoken or written agreement governed by court procedures. The first prerequisite to becoming a legally valid contract is that the contracting parties are engaged voluntarily to reach a consensus. In a traditional contract, a service is offered for some form of compensation (usually money) and some other provisions (e.g., contract terms and conditions, the service delivery dates, liability and compensation for the breach, and so on). Subsequent transactions are based on trust, and contracting parties generally see contracts as a symbol of an existing business deal. Another drawback in traditional way of establishing and managing contracts is that they are often under-specified. More importantly, traditional contracts do not provide sufficient details about the actual process of the transaction and, as a result, frictions between the contracting parties are very likely to occur, e.g., one party assumes a specific product certificate before delivering a partial compensation, and the other party assumes the contrary. The resulting deadlocks result in costly conflict resolution or even the entire contract transaction collapsing. Traditional contract enforcement is also proving to be either too complicated, time-consuming, or impossible, certainly in international circumstances.

Blockchain has established a new type of decentralized autonomous organization (DAO) whose activities run on a peer-to-peer network, involving governance

as well as decision-making rules. The latter is an organization whose business provisions are written in a programming code, and the necessary business operations are controlled automatically as per the agreeing to the provisions [25]. DAO encourages re-implementing each aspect of traditional organizational governance, replacing voluntary compliance with a business's agreement with actual compliance using pre-agreed smart-contract code. The latter is machine-readable software code that is situated on the protocol layer of a blockchain system to govern transactions between DAOs[2]. Subsequently, Ethereum [1] emerges as the first smart-contract system where the protocol layer is equipped with a Turing-complete programming language. This innovation affects a growing number of application cases, e.g., in logistics [3], e-healthcare [8], cyber-physical systems [28] such as for smart electrical-grid production [10], and so on. In the meantime, various smart-contract systems exist such as Neo[1], Cardano [2], Hyperledger [3], etc. with varying blockchain types, consensus algorithms, and machine-readable languages [29], [15].

By punishing opportunistic behavior, contracts and contract law lead to the enforcement of the intentions initially specified in the contract by the acting parties. Contracts are usually defined as legally-binding agreements that stipulate the rights and obligations of the contracting party towards each other [30]. This study [5] shows when code is law, it refers to the idea that, with the advent of digital technology, code has progressively established itself as the predominant way to regulate the behavior of Internet users. Yet, while computer code can enforce rules more efficiently than legal code, there are also limitations, mostly because of the difficulty to transpose the ambiguity and flexibility of legal rules into a formalized language for interpretation by a machine. A number of studies focus on checking the legality of smart contracts. This article [6] considers the potential issues with legal and practical enforceability that arise from the use of smart contracts within both civil- and common-law jurisdictions.

This paper [13] shows, the technology of smart contracts neglects the fact that people use contracts as social resources to manage their relations. The inflexibility that they introduce, by design, short-circuit a number of social uses to which law is routinely put. Few studies show that smart contracts are a new form of preemptive self-help that should not be discouraged by the legislatures, or courts [23]. A smart contract gives rise to a novel means of legally enforcing obligations and rights. These issues are treated differently from country to country. Smart contracts that underpin transactions in ICOs (Initial Coin Offerings – e.g., KodakCoin) may be illegal in some jurisdictions, while a smart contract that handles intra-institutional banking and other financial transactions is considered as legal, in the same jurisdiction, or elsewhere [22]. Another study show the necessary requirements and design options for the legality of smart-contract forms and proposes future research directions [4]. The future research direction aims to provide straight-through processing of financial contracts, with highly automated smart contract code to entire semantics of smart contract. This future direction opted by the another study [17]. This research shows the significance of highly automated smart contract so called self-aware smart-contract in the real world financial contract. In [22], another initiative investigates the legal enforceability of smart contracts. In this research, the findings render smart contracts legally enforceable by incorporating crypto primitives such as a digital signature.

Thus, the state of the art shows that CCs cause high transaction costs due to their multiple shortcomings, SCs lack legal relevance and this yields legal uncertainties for users while furthermore, SCs are inflexible code that are not smart.This paper fills the gap by answering the main research question of how to establish legal relevance for smart contracts that have socio-technical utility? To establish a separation of concerns, we deduce the following sub-questions. What conditions need to be fulfilled for SCs to have legal relevance? What are the properties of an ontology that represent

[1] https://neo.org/Neo blockchain — Home Page
[2] https://cardano.org/Cardano — Home Page
[3] https://www.hyperledger.org/Hyperledger — Home Page

these conditions for legal relevance? What enactment mechanisms ensure the legal enforceability for contracts?

The remainder of this paper is structured as follows. Section 2 presents a running case and additional preliminaries. Section 3 comprises the legal problem factors for smart contracts that require legal relevance. Next, Section 4 translates these elements into an ontology for SCs and Section 5 presents the SLC lifecycle model to monitor contractual rights and obligations. Section 6 demonstrates a feasibility evaluation and discussion that expands the running case of this paper. Finally, Section 7 gives conclusions, limitations, open issues and future work.

2 Motivating Example and Preliminaries

In section 2.1 we present the running contract case that stems from real car production supply chain contracts. In section 2.2 we present related background literature that prepare the reader for subsequent section.

2.1 Running Case

To illustrate the approach of the paper, a generic supply chain running case of car production is shown in Figure **??**. The original equipment manufacturer (OEM) actually assembles the delivered car parts. The other parties of the supply chain involve either the supply side, or demand side. E.g., the Supplier A sources and supplies the raw materials to Supplier B, who manufactures the individual car components. The particular car component are then shipped to the OEM, who assembles the final product.

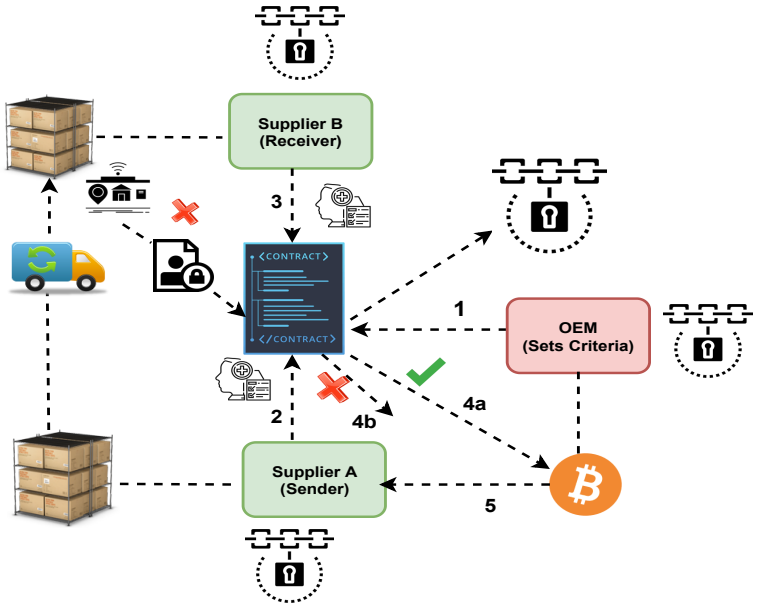

Fig. 1. Supply chain running case.

We deploy the supply chain running case into the blockchain that plays a significant role for checking provenance and tracking of product, which is possible with the integration of smart-contract. The blockchain allows supply-chain partners to access various functions, i.e., partners can access particular function such as checking provenance. For checking the provenance, Supplier A delivers the raw materials to Supplier B and the former publish the records into the blockchain with the integration of sensors. The records include quantity, quality of product and location, time on which

the raw product is shipped. This information is immutably stored into the blockchain and can be accessed by supply-chain partners.

For tracking of the product, when Supplier B receives the raw materials, we assume he confirms that the shipment is in order. If the shipment is received on time and at the correct location as per sensor verification , then the payment in the form of digital transaction is executed automatically by a smart-contract.

We can improve supply chain processes efficient by integration of smart-contract into the blockchain. However we introduce various limitation of this technology by considering supply chain running case. In Figure 1, Step 1 shows the OEM sets the acceptable criteria such as correct location and lead time for each leg of the shipment. The OEM also holds digital currencies while Supplier A is a sender who delivers the raw material and publishes the details on the blockchain in Step 2. When the Supplier B receives the raw material, he publishes the records as well in Step 3. In this step we show some conflict situation that may arise due to immaturity of novel blockchain technology. From supplier B, the data goes to smart contract through sensor for further action. This data can not reliable because it can be altered by external third party. In next phase, when specific criteria meet that are embedded in the smart-contract per Step 4a, then a payment is executed automatically in Step 5. There can be another possibility that the smart contract fails to meet certain conditions. In this situation, an alert is triggered so that partners can rectify any problems in Step 4b. Furthermore, we raise another issue for this paper in asking what happens if a particular obligation is not performed by the partners? For example, assuming the OEM has a payment obligation in Step 5, then in case of late payment, Supplier A has the right to claim late-payment charges. After enabling the corresponding function of rights by Supplier A, the OEM has an obligation to pay. Here, we have seen rights and obligation of parties are not clear. There can be another possibility that the smart contract fails to meet certain conditions. In this situation, an alert is triggered so that partners can rectify any problems in Step 4b. Another challenge of this paper is, if the partner is a non-programmer, he is not capable of understanding what rights and obligation are written in the contract.

2.2 Related Background Literature

In this section, we describe the computation toolkit to understand the solution of the running-case problem. In Section 2.1, we propose the situation of the dispute among the parties. To overcome this situation, we develop an ontology that comprises the concept and properties of rights and obligations. The ontology is a formal representation of knowledge by a set of concept and relationship among those concepts [14]. The ontology organizes the class hierarchies of relationships and allows the practitioner to understand the relationships of the particular problem domain. We design the ontology in Prot´eg´e tool [16] that is open-source ontology editor and comprises the graphical user interface for visualization of the relationship among classes. We employ the HermiT-tool reasoner [7] that checks the correctness of ontology and identify subsumption relationship among classes.

Later, to automate the concept of rights and obligation in smart-contracts, the Prot´eg´e tool also supports web ontology language[4] (OWL) that express the formal semantics of ontology into machine-readable code. We also employ Coloured Petri Nets (CPN) tools[5] for checking the dynamic behaviour of ontology processing. CPN tool is an software that is useful for simulation and state space analysis of the model. An state space is a directed graph with nodes so-called states and arcs that connect states and transition. The CPN-notation comprises state represented as a circle, transition represented as a rectangle, arcs that connect the states to transitions, and token with color, i.e., attributes with values. Transition fires when all input states hold the tokens and produce condition-adhering tokens into output places.

[4] https://www.w3.org/OWL/
[5] http://cpntools.org/

3 Legal Recognition Factors

In this section, we discuss the legal recognition of the business-to-business (B2B) smart contracts presented in Section 2.1. A core principal in contract law is freedom of contract that has two main elements, a) the right of a legal person to freely decide whether to enter into a contract and b) the right to freely decide together with the other contracting party, or parties about the content of the contract [24]. The prevailing view in law [26] concerning a) is that in principal nothing prevents two B2B parties to voluntarily enter into machine-readable sales contracts that are based on a blockchain[6]. To assess how the rules apply regarding the formation of contracts to smart contracts, the analogy of a vending machine has often been used in the literature [27]. In both cases, a legally binding contract is formed due to the consenting actions of the contracting parties. Just as for a vending machine, a smart contract can be independently designed by an offeror and deployed to a blockchain [6]. The design and deployment thereby indicate the offeror's intention to be legally bound according to the terms stipulated in the smart contract. According to this analogy, the offeree agrees to be bound by the smart contract through conclusive conduct. Equivalent to entering a coin into a vending machine he, or she fulfills the triggering requirements of the smart contract. In regard to our running case, this means that an OEM sends cryptocurrencies to Supplier B's smart contract to order intermediate product. By sending the tokens to the smart-contract wallet address, the OEM consents to the contract terms of Supplier B's smart contract and a legally binding agreement is formed [11]. To conclude, from a legal point of view, the right of a legal person to freely decide whether to enter into a contract extends to the right to enter into smart contracts.

Accordingly, also b) the right to freely decide together with the other contracting party, or parties about the content of the contract, should extend to smart contracts. Still, it is currently unclear how a court, or an arbitrator would interpret smart contacts that do not use legal terminology but are based on programming code [9]. Assuming that for our running case, Supplier B delivers the intermediate product to the OEM who's sensors accurately recognize the delivery of the intermediate product. The cryptocurrency funds are thus released to Supplier B and the goods are stored in the warehouse. Later, the intermediate product is used for production by the OEM and it is discovered that the intermediate products are of inferior quality, most likely due to impurified raw materials delivered by supplier A to Supplier B. The OEM's production process has to stop causing severe consequential damages. The OEM sues supplier A for replacement of the inferior intermediate products and compensation for the lost production. Supplier A claims that the inferior quality of the intermediate products was not caused by impurified raw materials but rather because of inappropriate storage by the OEM. Furthermore, Supplier A argues that the terms and conditions of the smart contract exclude consequential damages and stipulate an immediate notification deadline in case of defective goods that the OEM has missed. The OEM and supplier A also disagree about the responsible dispute resolution forum and the applicable contract law. The current state of smart contracts do not allow a judge to solve these problems. This is because there is currently no complete legal ontology that allows programmers to design smart contracts in the fashion of traditional contracts.

In the next section, we present a comprehensive contract law ontology that allows contracting parties to precisely define the content of a smart contract for machine readability and in case of disputes, also by a human judge, or arbitrator. We argue

[6] Note that this freedom may be limited in the case of business-to-consumer contracts and special kind of contracts with significant consequences for the contracting parties (e.g., the selling and transfer of real estate) where the law may demand a special form requirements. Still, as these contracts are not part of our running case, we will not discuss the legal problems connected to the operationalization of such contracts as smart contracts any further.

that only with the help of our contract-law legally binding smart contracts can be designed.

4 Smart-Legal-Contract Ontology

A smart legal contract (SLC) ontology [7] in Figure 2, 5 comprises essential concepts and properties to support SC legally enforceability. We develop the SLC ontology in protége tool to render the latter machine-readable and explain the concepts and properties for the running case of the car production business case, as illustrated in Section 2.1.

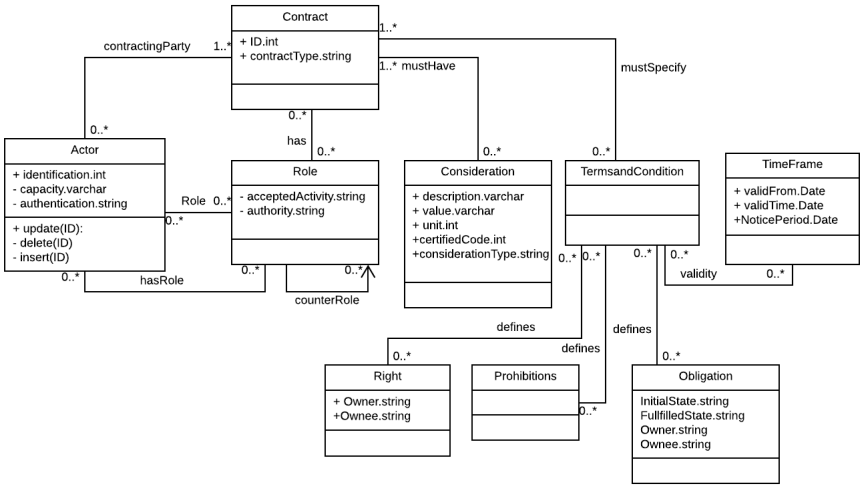

Fig. 2. Outline for Upper Level Smart Contract Ontology.

The SC ontology defines fundamental concepts such as roles, considerations, rights, and obligations, etc., as presented in Figure 2. In our running case, Supplier A promises to deliver the raw materials to Supplier B in return a specific amount of money. Supplier B manufactures the car component and promises to deliver to the OEM. To prevent a conflict among the parties the role is defined in the SC. A promise is a statement of commitment to fulfil some act or perform certain deeds and when a promise is given with legal intent of enforcement in court, later becomes legal obligation.

The valid SC must hold two or more contracting parties, such as offeror, offeree, and acceptor etc., [9]. An offeror sends an offer to an offeree for delivering services, and the offeree accepts an offer, offeree becomes acceptor. In our running case, Supplier B acts as an offeror for car components to an OEM that is an offeree. When an offer is accepted by an OEM, then the later transforms into an acceptor. Furthermore, contracting parties must have the necessary competence and capability to perform certain deeds. Supplier B must own the car components as he has promised to deliver the car components, and the OEM must have the capital to pay for it. Further, a valid SC can be a complete-, or incomplete contract. For example, the car manufacturer writes the SC of shipment of cars without giving the deadline, and it considers a valid contract while it is incomplete because there is nothing obligations are specified for the parties.

Consideration is defined as exchanged value for the trade of which the contracting parties agree to enter into a SC. Car components and raw materials are the asset,

[7] Full ontology: https://bit.ly/3c5eYO5

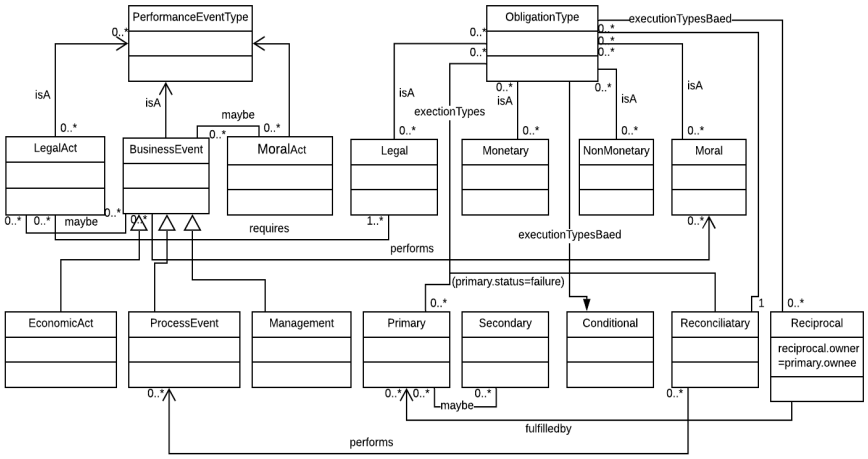

Fig. 3. Rights and Obligations.

or consideration of the contract. The selling of raw material, or car components constitutes the performance that fulfills the promise of the contract. Selling of the car component to an OEM is an obligation of supplier B that is realized when an actual business act of given car component is performed in return of compensation. An obligation must have obligee who is obliged to perform a particular action and obliger beneficiary who receives the consideration for whom a promise is established. In this case, Supplier B is an obligee who is obliged to deliver car components, and OEM is an obliger who receives the car component.

Also, the SC contains the deadline, or time frame under which the promised performances shall occur. If certain promises are not performed under the deadline, or in unsatisfactory manner, then the state of obligation will change as to unfulfilled. In our running case, if supplier B does not execute the promises as planned and agreed, then the obligation is unfulfilled, resulting in the OEM may seek the pre-agreed rights as compensation. In this case, the OEM may have the right to seek a remedy in the form of a penalty, or can terminate the contract. Also, the OEM can choose not to do anything and settlement occurs by mutual consent.

We explain the simple running case of ontology, where we see an obligation may activate another obligation and rights. Similarly, the rights too may enable new obligations being formed. Therefore, we present the types of rights and obligations in Figure 5. Also, we describe the change of obligation state under which rights may activate.

We refine an obligation type by adding sub-classes such as monetary-, non-monetary-, moral-, and legal obligation to express a particular remedy. The monetary obligation may create a remedy such as late-payment-charges, penalty, etc. For instance, if the OEM receives a car component from Supplier B and does not pay the money within a deadline, then Supplier B may provide a remedy as to a late-payment-charge. Similarly, if a supplier delivers defective car parts then the OEM may have legal rights to cancel the SC. The OEM may have non-monetary obligations to arrange a carrier to provide the car components but does not have monetary obligation between supplier and OEM. There may be another moral obligation type that is not severely binding but is morally, or ethically an expected obligations. For instance, the OEM has an obligation to pick up the car components from the supplier premises. Still, the OEM may urge for help to arrange the transportation. The supplier may not legally bound to help the OEM, but morally he is bound to assist the OEM.

We identify several obligation states to track the SC fulfillment process more systematically, and an individual obligation may exist in more than one category. The proposed obligation states in Figure 5 are adopted from similar work proposed

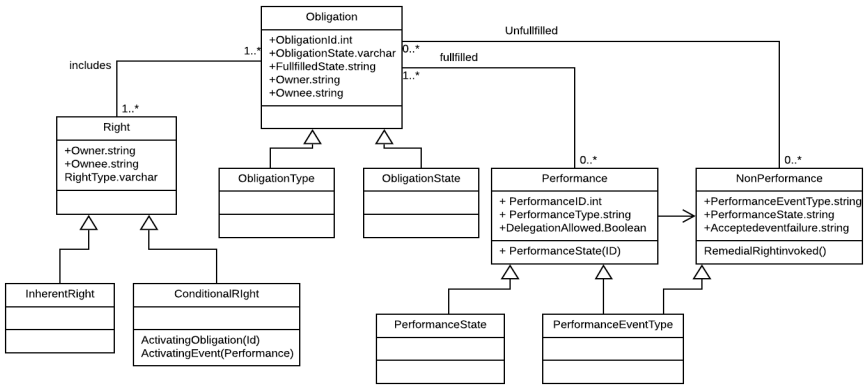

Fig. 4. Common Obligation Types.

by LEE [12]. Initially, the SC has an inactive state when the supplier and OEM have signed the SC, but the SC execution does not commence. The SC is said to be active when the performance event is performed. For instance, the OEM demands to deliver the car components to Seller B who receives an order; the supplier obligation to deliver the car components is triggered. The SC obligation state is pending when the supplier has dispatched the car components from the warehouse and is waiting for a carrier for transportation. The obligation state will be pending until the entire essential fulfillment process is completed. When the OEM receives the car components that satisfy the stipulated performance condition, then the obligation state is fulfilled and when the obligation is terminated by the obligee with mutual intent, the obligation state changes to termination.

5 Rights- and Obligations Monitoring

We develop an SLC lifecycle model[8] to monitor contractual rights and obligations defined with the ontology of Section 4. We adopt the existing formalized smart-contracting lifecycle in [21], [18], [20], where the startup phase commences with the configuration of a business network model (BNM). The latter is a cross-enterprise collaboration blueprint and contains the legally valid template contract that inserts the service type with organization roles. The latter enables fast and semi-automatic identification of contracting parties for knowing their identity, services, and reputation. We include the rights and obligations throughout the existing smart-contracting lifecycle to monitor the related contractual fulfillment process, and the updated lifecycle is called the *SLC lifecycle model*.

The SLC lifecycle is divided into two modules, set up- and perform phase. In the *setup module*, the proposal of SC is finalized and negotiated among the contracting parties. Further, the performances of smart- contracting are accomplished in the *perform module*. The rights and obligations are stipulated throughout the entire lifecycle. Still, we present the transaction of rights and obligations in the smart-contracting setup phase and especially in the BNM selection that is an ecosystem for breeding service types with rights, obligations, and roles to become a part of BNM. The latter is divided into several sub-modules namely, *repository accessing*, *manage service offer*, and *manage service type*, as presented in Figure 7. A *repository accessing* exists in *BNM selection*, where we assume a user inserts the service type with rights and obligations over the time and the same assumptions hold for the repository of service offer in the *manage service offer* module. Finally, the *conformance validation* module is developed to conform to the validation of service offer against the chosen service type in the BNM draft specification.

[8] Full download CPN model: shorturl.at/cxBE9

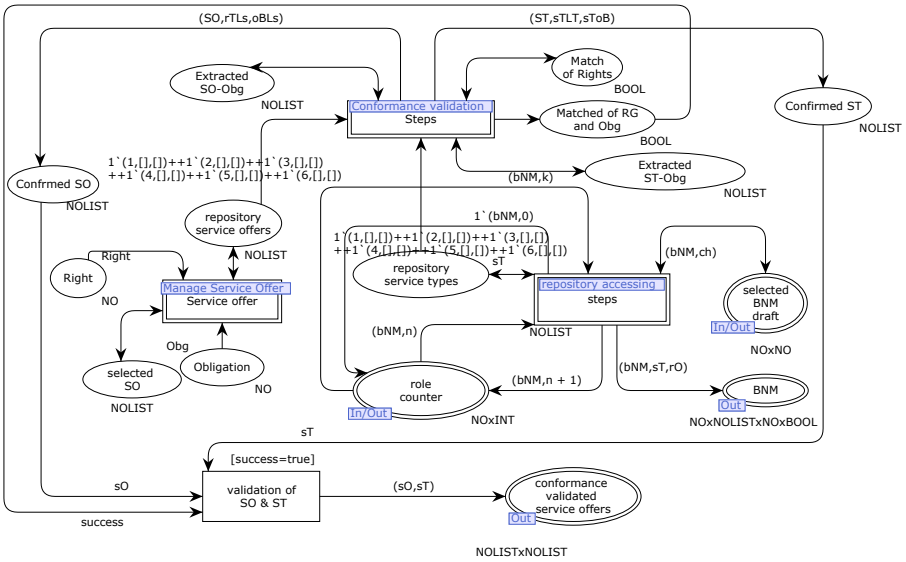

Fig. 5. Rights and obligations selection in BNM.

In the next subsection, we provide the details of each module in the subsections below.

5.1 Repository accessing

We assume a contracting party inserts the rights, roles, and obligation for service types in the repository in Figure 6.

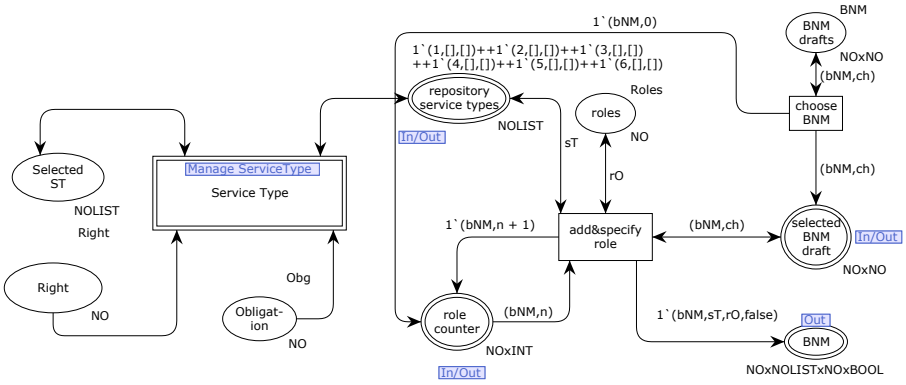

Fig. 6. Repository of service type with rights, roles, and obligations.

As a first step, the contracting party chooses a BNM from stored BNM drafts. Thereafter, the latter inserts the rights and obligations in the *manage service type* module that is stored in the state labeled *repository service types*. Further, the actual BNM selection involves choosing a BNM draft for validating service offers and roles to be filled subsequently with rights and obligations. The rights and obligations to be filled in the *manage service type* module are presented below.

Manage service type The actual repository of service types commences with choosing the rights and obligations in the *Manage service type* module, as presented in Figure 7. Initially, the list of rights and obligations of service type is empty in the

repository. At a first step, the contracting party chooses the service type Id from *choose ST* transition. After that, the latter inserts the rights and obligations simultaneously in *selected ST* by firing the *insert right* and *insert obligation* transitions.

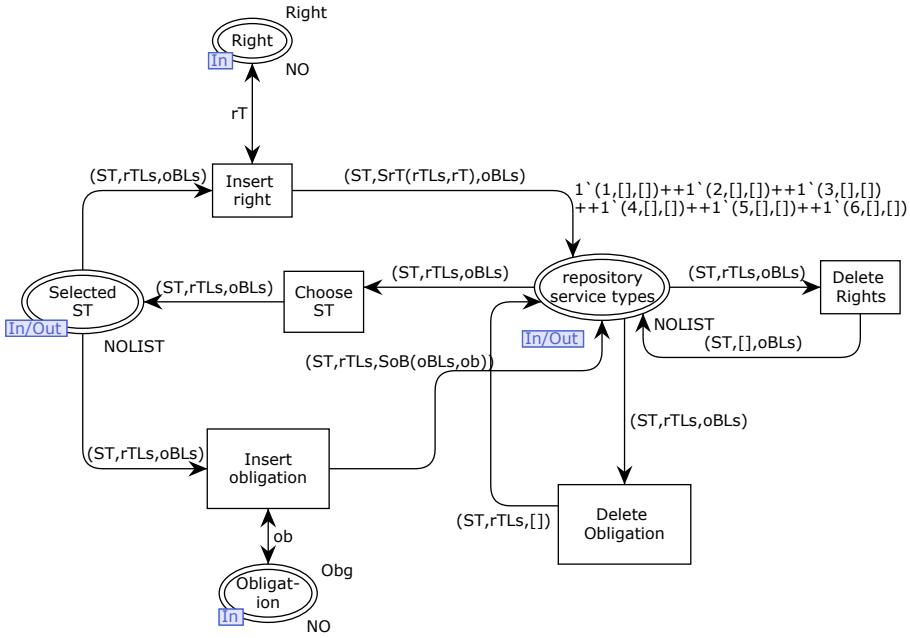

Fig. 7. Insertion, deletion of rights and obligations in service types.

Additionally, the inserted rights and obligations are deleted by firing the transitions *delete right* and *delete obligation*. The same assumption holds for choosing the rights and obligations for service offer in *manage service offer* module.

5.2 Conformance validation

To be considered as a service offer for finalizing the proto-SC, beforehand, a conformance validation is necessary, as depicted in Figure 8. The selected service offer and service type from the BNM repository are extracted for the validation. The chosen right and obligation properties inherit the properties of the service type. Thus, we extract the rights and obligations from the state labeled *repository service type* and *repository service offer*. Further, a service offer matched with service types is stored in the state labeled *confirmed SO* and *confirmed ST*.

6 Evaluation

We use the CPN tool to evaluate the model concerning the correctness and performance checking, especially considering aspects that are required for system development. Due to page limitations, we do not present the entire steps that are taken to produce the evaluation results. Several properties, such as reachability, loops, etc., as depicted in Figure 9, are essential to evaluate in the model. Due to the size of models, computation of states verification through automatic simulation of token games is challenging. Thus we are focused on the detection of loops to prevent the desired termination reachability. Furthermore, specific attention is required to exit loop-conditions effectively, such as elements of the business-policy control. Performance peaks are calculated during runtime either in designing for sufficient resources or in restricting the load with the business policy control. The utilization property is used to ensure the

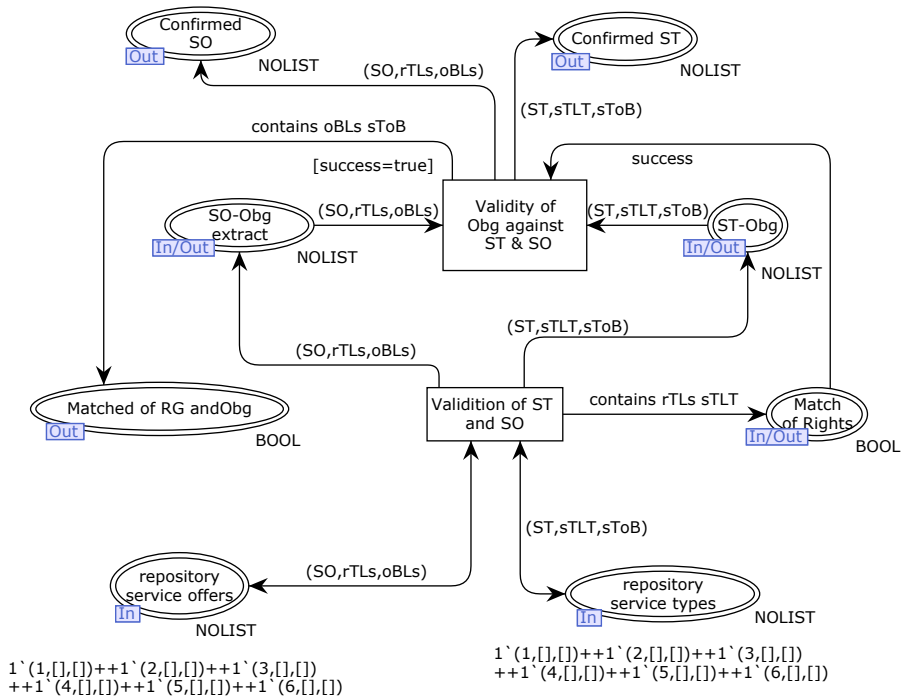

Fig. 8. Conformance validation of service offers and service types.

effectiveness of each model in a specific scenario. Finally, home marking is needed for consistent termination to ensure simple testing of a real system.

We generate the state space on CPN modules where the computation is feasible and present the results in Table 9. Loops exist in *manage service type* module. For the Manage service type, a party inserts the rights and obligations.

	Module property					
Module	**Loops**	**Performance peaks**	**Liveness**	**Utilization**	**Home marking**	**Dead marking**
BNM Selection	No	Evenly balanced	ND/NL	Yes	No	Multiple
Repository accessing	No	Manage service type	D*/NL	Yes	No	Multiple
Manage service type	Yes	Insertion of rights and obligations	D*/NL	Yes	No	No
Conformance validation	No	validating service type and service offer	ND/NL	Yes	No	Yes

Fig. 9. Model checking.

Loops exist in *manage service type* module. Managing the service type loop is self-restricting as it only processes the rights and obligations of parties, respectively. The results show for the remaining modules in Table 9; they do not contain the loops.

Performance peaks exist in Table 9 to represent the places of the SLC lifecycle that are the performance bottlenecks. Peaks exist in each module but not in *BNM selection*. For the repository accessing, the peaks exist for choosing a BNM draft and party's roles and also for inserting the rights and obligations.

There is no home marking, as presented in Table 9, and the result for dead marking differ. Multiple dead marking and home marking show test cases are more demanding for practitioners to validate the implementation. D* means a dead marking result that shows the intentional disabling of marking path for the purpose of focusing on a particular module under investigation. Finally, the utilization test in Table 9 shows there is no unused sub-module exist.

Due to page limitations, we refer to the reader [19] for more details of evaluation.

7 Conclusion

Smart contracts are machine-readable software code that is situated on the protocol layer of a blockchain system to govern transactions. The later gives rise to a novel means of legally enforcing rights and obligations. State of the art shows that CCs cause high transaction costs due to their multiple shortcomings, SCs lack legal relevance, and this yields legal uncertainties for users while furthermore, SCs are inflexible code that is not smart.

In this paper, we identify a gap between business process management and SC execution, and their fulfillment. Thus, we introduces the ontological concepts of rights and obligations for SC's that are defined in business contracts. For developing the ontology we opt the protege tool that is open source ontology editor and employ the HermiT-tool reasoner that checks the correctness of ontology. Further, we propose the obligation states through which each obligation is passed. For ensuring the legal enforceability of SC's, we present an SLC lifecycle model to monitor contractual rights and obligations. For exploring the SLC lifecycle in a dependable way, we choose CPN Tools that has a modeling notation backed with formal semantics.

The limitation of the paper is that we are only focused on presenting the transaction of rights and obligations in the smart-contracting setup phase. Future work in SC's domain includes analysis and modeling of other types of business legal SC's. Further, resetting of human to the machine for the self-aware SC's are a possible extension to ongoing work.

8 Acknowledgments

This article is based on research from the Erasmus+ Strategic Partnerships Project - 2018-1-RO01-KA203-049510 "Blockchain for Entrepreneurs - a non-traditional Industry 4.0 curriculum for Higher Education".

References

1. Buterin, V., et al.: Ethereum white paper. GitHub repository pp. 22–23 (2013)
2. Butterin, V.: A next-generation smart contract and decentralized application platform (2014)
3. Casado-Vara, R., González-Briones, A., Prieto, J., Corchado, J.: Smart contract for monitoring and control of logistics activities: Pharmaceutical utilities case study. In: The 13th International Conference on Soft Computing Models in Industrial and Environmental Applications. pp. 509–517. Springer (2018)
4. Clack, C.D., Bakshi, V.A., Braine, L.: Smart contract templates: foundations, design landscape and research directions. arXiv preprint arXiv:1608.00771 (2016)
5. De Filippi, P., Hassan, S.: Blockchain technology as a regulatory technology: From code is law to law is code. arXiv preprint arXiv:1801.02507 (2018)
6. Giancaspro, M.: Is a 'smart contract'really a smart idea? insights from a legal perspective. Computer law & security review **33**(6), 825–835 (2017)
7. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: an owl 2 reasoner. Journal of Automated Reasoning **53**(3), 245–269 (2014)
8. Griggs, K., Ossipova, O., Kohlios, C.P., Baccarini, A., Howson, E., Hayajneh, T.: Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. Journal of medical systems **42**(7), 130 (2018)
9. Idelberger, F., Governatori, G., Riveret, R., Sartor, G.: Evaluation of logic-based smart contracts for blockchain systems. In: International Symposium on Rules and Rule Markup Languages for the Semantic Web. pp. 167–183. Springer (2016)
10. Imbault, F., Swiatek, M., De Beaufort, R., Plana, R.: The green blockchain: Managing decentralized energy production and consumption. In: Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), 2017 IEEE International Conference on. pp. 1–5. IEEE (2017)
11. Lauslahti, K., Mattila, J., Seppala, T.: Smart contracts–how will blockchain technology affect contractual practices? Etla Reports (68) (2017)

12. Lee, R.M., Dewitz, S.D.: Facilitating international contracting: Al extensions to edi. International Information Systems **1**(1), 94–123 (1992)
13. Levy, K.E.: Book-smart, not street-smart: blockchain-based smart contracts and the social workings of law. Engaging Science, Technology, and Society **3**, 1–15 (2017)
14. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent systems **16**(2), 72–79 (2001)
15. Mohanta, B., Panda, S., Jena, D.: An overview of smart contract and use cases in blockchain technology. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–4. IEEE (2018)
16. Musen, M.A., et al.: The protégé project: a look back and a look forward. AI matters **1**(4), 4 (2015)
17. Norta: Self-aware smart contracts with legal relevance. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2018)
18. Norta, A.: Establishing Distributed Governance Infrastructures for Enacting Cross-Organization Collaborations . https://doi.org/10.1007/978-3-662-50539-7
19. Norta, A., CINCO, C., Computing, I.: Safeguarding trusted ebusiness transactions of lifecycles for cross-enterprise collaboration. University of Helsinki, Department of Computer Science, Helsinki, Finland, Tech. Rep. C-2012-1 (2012)
20. Norta, A., Othman, A.B., Taveter, K.: Conflict-Resolution Lifecycles for Governed Decentralized Autonomous Organization Collaboration (2015). https://doi.org/10.1145/2846012.2846052, http://dx.doi.org/10.1145/2846012.2846052
21. Norta, A.: Creation of Smart-Contracting Collaborations for Decentralized Autonomous Organizations, https://sci-hub.tw/10.1007/978-3-319-21915-8_1
22. Patel, D., Shah, K., Shanbhag, S., Mistry, V.: Towards legally enforceable smart contracts. In: International Conference on Blockchain. pp. 153–165. Springer (2018)
23. Raskin, M.: The law and legality of smart contracts (2016)
24. Schafer, I.: Ott, lehrbuch der okonomischen analyse des zi-vilrechts, 4 (2005)
25. Singh, M., Kim, S.: Chapter four - blockchain technology for decentralized autonomous organizations. In: Kim, S., Deka, G.C., Zhang, P. (eds.) Role of Blockchain Technology in IoT Applications, Advances in Computers, vol. 115, pp. 115–140. Elsevier (2019). https://doi.org/https://doi.org/10.1016/bs.adcom.2019.06.001, https://www.sciencedirect.com/science/article/pii/S0065245819300257
26. Smits, J.M.: Contract law: a comparative introduction
27. Szabo, N.: Formalizing and securing relationships on public networks. First Monday **2**(9) (1997)
28. Teslya, N.: Industrial socio-cyberphysical system's consumables tokenization for smart contracts in blockchain. In: International Conference on Business Information Systems. pp. 344–355. Springer (2018)
29. Wang, S., Yuan, Y., Wang, X., Li, J., Qin, R., Wang, F.: An overview of smart contract: architecture, applications, and future trends. In: 2018 IEEE Intelligent Vehicles Symposium (IV). pp. 108–113. IEEE (2018)
30. Wulf, a.J.: Institutional competition of optional codes in european contract law. European journal of law and economics **38**(1), 139–162 (2014)