Symboleo: Towards a Specification Language for Legal Contracts

Sepehr Sharifi, Alireza Parvizimosaed, Daniel Amyot, Luigi Logrippo, John Mylopoulos
School of EECS, *University of Ottawa*, Ottawa, Canada
{sshar190, aparv007, damyot, logrippo, jmylopou}@uottawa.ca

***Abstract*—Legal contracts specify the terms and conditions (in essence, requirements) that apply to business transactions. Smart contracts are software systems that monitor and control the execution of contracts to ensure compliance. This paper proposes a formal specification language for contracts, called Symboleo, where contracts consist of collections of obligations and powers that define the legal contract's compliant executions. The formal semantics of Symboleo is based on an extension of an ontology for Law and is described in terms of logical axioms on statecharts that describe the lifetimes of contracts, obligations and powers. Our proposal includes a preliminary evaluation through the specification of a real life-inspired Sale-of-Goods contract, with a prototype execution engine. We envision this language to enable formally verifying contracts to detect requirements-level issues and to generate executable smart contracts (e.g., on blockchain technology).**

***Index Terms*—Legal contracts, smart contracts, software requirements specifications, formal specification languages**

I. Introduction

Legal contracts specify the terms and conditions that apply to business transactions. Legal contracts are commonly expressed in natural language and contain many legal requirements that are often ambiguous, incomplete, and possibly inconsistent. *Smart contracts* are programs intended to partially automate the execution of legal contracts and also monitor them for compliance with relevant terms and conditions. We are interested in formal specifications of such contracts that can enable automated analysis as well as the generation of smart contract programs.

For example, a smart contract may monitor the execution of a Sale-of-Goods contract between an Argentinian meat producer, call it A, and a Canadian supermarket chain, call it C, by receiving and recording events on a blockchain ledger capturing the execution flow of the contract. Events monitored may be pickup of the meat from A, delivery to Buenos Aires port, loading on a cargo vessel, delivery to the Vancouver port, pickup and delivery to C. The smart contract may also carry out some of the actions called for in the contract, such as payment for the transaction by transferring funds held in an escrow account. There is tremendous interest in the food supply chain industry for such systems, but also in other sectors, including energy, insurance, and government affairs [1].

The research reported herein is partially funded by an NSERC Strategic Partnership Grant titled "Middleware Framework and Programming Infrastructure for IoT Services" and by SSHRC's Partnership Grant "Autonomy through Cyberjustice Technologies (ACT)"

The idea of smart contracts has been around for more than 20 years [2], going back to seminal work by Nick Szabo. However, interest in them has surged in the last ten years, thanks to increased availability and reduced cost for IoT technologies (sensors, actuators, robotic devices, etc.[1]), as well as the rise of distributed ledger or blockchain technologies. Blockchain provides only one of several possible monitoring methods for smart contracts, but it can be essential when integrity and security warranties for execution logs are required. It should be noted that in this work, we subscribe to Szabo's original definition of smart contracts, from which more recent views have deviated by referring to any application software running on a blockchain platform, although there are common elements in the two definitions [2].

In this paper we propose a formal specification language for smart contracts called *Symboleo*[2] that is sufficiently expressive to represent many types of real-life contracts, while still capable of supporting their analysis, as we plan to demonstrate in future work.

A contract can be viewed as an outcomes-oriented process that specifies its compliant executions. However, contracts specify legal processes (as compared to business processes) where there are provisions for penalties and compensations whenever any party violates its obligations. Looking at them from this perspective, contracts are very interesting processes because they provide alternative compliant executions if terms and conditions are violated, including the imposition of new obligations on non-compliant parties through *powers* (special kinds of *rights*). They can also specify the possibility of subcontracting, as well as the delegation of obligations to third parties at execution time.

Our research methodology is based on Design Science. The main artefacts are a language (with preliminary results reported here) with tools for verification and code generation (future work). The problem relevance was assessed by interacting with engineering and lawyers from industry and government in a large, six-year long international project (anonymized for this submission), by exploring the literature and by analyzing dozens of contracts on supply chains, construction, and energy. The contributions of this paper include:

- A formal specification language (Symboleo) for smart contracts that accounts for obligations and powers, using

[1] Statista 2016, see http://bit.ly/2M3QT0U

[2] From the Greek word $\Sigma\upsilon\mu\beta o\lambda\alpha\iota o$, which means contract.

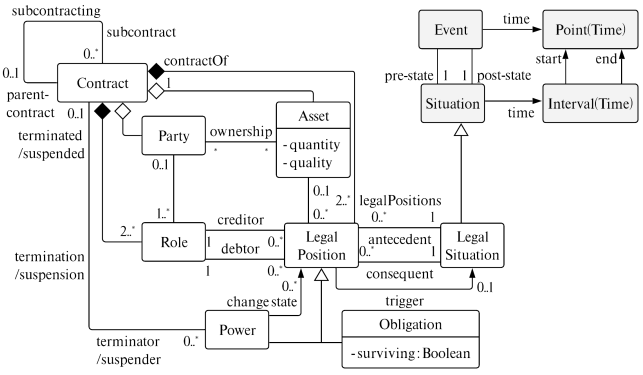

Fig. 1. Proposed Contract Ontology

domain concepts and axioms. Symboleo specifications provide requirements for smart contract executions that can be monitored at runtime.

- A formal semantics based on statecharts that define the lifecycle of contracts, obligation and power instances, following earlier work on process monitoring [3].
- An illustrative example (international meat sales contract) inspired from real-life contracts is used to demonstrate the language, and offer a preliminary evaluation.

The rest of this paper is structured as follows. Section II presents our research baseline, including the nature of legal contracts and the ontology we adopted for specifying them. Section III presents our specification language through an example, while Section IV presents the syntax and semantics for Symboleo. Section V discusses related work, whereas Section VI concludes and highlights future work.

II. Research Baseline

A contract is a collection of obligations and powers between participating parties. As a legal artifact, a contract has its own lifecycle that begins with *proposals* and *negotiations*, during which three necessary conditions apply: *offer*, *acceptance* and *consideration*. The execution (*performance*) of a contract is initiated after signing (*formation*). Execution may be *suspended*, *successfully* or *unsuccessfully terminated*, *renegotiated* or *renewed*.

Contracts can be understood as prescriptions of allowable execution processes [4], [5]. Relative to business processes, contracts are outcome-oriented processes focusing on 'what' the obligations of different parties are, and leaving the 'how' to the parties responsible. In addition, contracts fundamentally differ from business processes in that they can change during their execution through the exertion of powers. For the meat sales example, the contract specifies that the seller needs to deliver the meat to a freight company, who delivers it to a shipping company; however this obligation can be violated, which may give the buyer the power to terminate the contract.

Ontologies capture the primitive concepts of domains at various levels of abstractions [6]. We propose a *contract* domain ontology, depicted in Fig. 1, which refines a Core Legal Ontology, namely UFO-L, which in turn is an extension of the Unified Foundational Ontology (UFO) [6]. Shaded concepts in the figure are adopted from UFO, while the remaining concepts are either new or specializations of UFO concepts with new association links. It is worth noting that UFO-L is based on Alexy's *Theory of Constitutional Rights* [7], which is based on Hohfeld's theory of *legal positions* [8] but without some of its shortcomings [6]. The concepts of our contract ontology are as follows:

Contract: a collection of obligations and powers between two or more roles, which are assigned to parties during execution, and are concerned with two or more assets (since at least one asset should be associated with each role.

Asset: an owned (tangible or intangible) item of value [9]. Assets include *contractual considerations* [10] that a contract is concerned with. Other kinds of assets can also be used to ensure proper execution of a contract, e.g., a bill of lading for freight contracts or invoices. Asset quantity and quality constraints are typically specified in contracts.

Legal Position: legal positions are the legal relationships between roles. For our purposes, there are just two such relationships: obligations and powers [8].

Obligation: the legal duty of a debtor towards a creditor to bring about a certain legal situation (consequent) when another legal situation (antecedent) holds. Surviving obligations remain in effect after the termination of the contract. A 6-month non-disclosure obligation after the end of the contract is an example of a surviving obligation. Obligations usually concern assets and are instantiated by conditions (trigger)[3].

Legal situation: a type of situation associated with a contract, obligation or power instance. Situations are states of affairs and are comprised of possibly many endurants (including other situations and relata) [12]. A situation *occurs* within a time interval *T*, but not in any of its proper subintervals [13].

Event: a happening that occurs at a time instance, and that cannot change. Events also have pre-state and post-state situations [12], [13]. For example, *delivered* for a product is an event whose pre-state is 'being in transit' and post-state is 'being in the point of destination'.

Power: the right of a party to create, change, suspend or extinguish legal positions. A power is instantiated by a trigger and has an *antecedent* (legal situation) that must be met for it to become in effect.

Role: contractual roles are characterized by collections of obligations and powers they participate in [6].

Party: a legal agent (person or institution) who owns assets and who is assigned roles in contracts.

III. Symboleo: A Contract Specification Language

This section introduces Symboleo with a meat sales example expressed as parameterized natural language in Table I. Producing a formal specification from natural language text involves several key decisions. These decisions can be taken

[3]A trigger is *true* for most obligations. However, *suspensive obligations* need to be triggered explicitly before they are instantiated [11].

TABLE I
SAMPLE CLAUSES OF A MEAT PURCHASE AND SALE CONTRACT

This agreement is entered into as of the date <*effDate*>, between <*party1*> as Seller with the address <*retAdd*>, and <*party2*> as Buyer with the address <*delAdd*>.

1) **Payment & Delivery**
   - 1.1 Seller shall sell an amount of <*qnt*> meat with <*qlt*> quality ("goods") to the Buyer.
   - 1.2 Title in the Goods shall not pass on to the Buyer until payment of the amount owed has been made in full.
   - 1.3 The Seller shall deliver the Order in one delivery within <*delDueDateDays*> days to the Buyer at its warehouse.
   - 1.4 The Buyer shall pay <*amt*> ("amount") in <*curr*> ("currency") to the Seller before <*payDueDate*>.
   - 1.5 In the event of late payment of the amount owed due, the Buyer shall pay interests equal to <*intRate*>% of the amount owed, and the Seller may suspend performance of all of its obligations under the agreement until payment of amounts due has been received in full.
2) **Assignment**
   - 2.1 The rights and obligations are not assignable by Buyer.
3) **Termination**
   - 3.1 Any delay in delivery of the goods will not entitle the Buyer to terminate the Contract unless such delay exceeds 10 Days.
4) **Confidentiality**
   - 4.1 Both Seller and Buyer must keep the contents of this contract confidential during the execution of the contract and six months after the termination of the contract.

in consultation with all parties involved and contribute to make formally specified contracts more precise and consistent than it is normally the case for conventional contracts. Firstly, we need to decide how generic/specific we want the specification to be. In the example, the contract could apply to a single sale of meat with two specific parties serving as seller and buyer, or to multiple sales of various food assets involving different parties. This decision determines the parameters of the contract specification. Secondly, the specifier needs to consider whether the informal specification is missing important implicit constraints and, if so, include them in the formal specification. For example, does every execution of the contract terminate in a finite amount of time (say, 21 days after the start date), or can it run for an indefinite period because of a missing temporal constraint? Are there sub-contracting constraints? Answers to such questions concern liveness and safety properties for contracts, in a way similar to those defined for distributed systems [14]. To address the above concerns, we propose the structure illustrated in Table II, with an example corresponding to the natural language contract in Table I.

Contract Specification. Consists of two sections: (a) the *domain* section, defining domain-dependent concepts as specializations of Symboleo's concepts, and corresponding to the *definitions* stated in the contract; (b) the *contract body*, corresponding to the *terms and conditions* stated in the contract. The body essentially contains deontic requirements.

Domain. Domain-related concepts are defined as specializations (**isA**) of contract ontology concepts. For instance, *Buyer* and *Seller* are specializations of Role with additional attributes; *Meat* is a specialization of *PerishableGood*, which is a specialization of Asset; and *Paid* specializes Event with attributes *amount* and *currency*.

TABLE II
MEAT SALES CONTRACT PROTOCOL

Domain meatSaleD
  Seller **isA** Role **with** returnAddress: String;
  Buyer **isA** Role **with** warehouse: String;
  Currency **isA** Enumeration('CAD', 'USD', 'EUR');
  MeatQuality **isA** Enumeration('PRIME', 'AAA', 'AA', 'A');
  PerishableGood **isA** Asset **with** quantity: Number, quality: MeatQuality;
  Meat **isA** PerishableGood;
  Delivered **isA** Event **with** item: Meat, deliveryAddress: String, delDueD: Date;
  Paid **isA** Event **with** amount: Number, currency: Currency, from: Role, to: Role, payDueD: Date;
  PaidLate **isA** Event **with** amount: Number, currency: Currency, from: Role, to: Role;
  Disclosed **isA** Event **with** contractID : String;
endDomain

Contract meatSaleC(buyer: Buyer, seller: Seller, qnt: Number, qlt: MeatQuality, amt: Number, curr: Currency, payDueDate: Date, delAdd: String, effDate: Date, delDueDateDays: Number, intRate: Number)
  Declarations
    goods : Meat **with** quantity := qnt, quality := qlt;
    delivered : Delivered **with** item := goods, deliveryAddress := delAdd, delDueD := effDate + delDueDatedays;
    paid : Paid **with** amount := amt, currency := curr, from := buyer, to := seller, payDueD := payDueDate;
    paidLate : PaidLate **with** amount := (1 + intRate/100)×amt, currency := curr, from := buyer, to := seller;
    disclosed : Disclosed **with** contract := **self**;
  Preconditions
    isOwner(goods, seller);
  Postconditions
    isOwner(goods, buyer) AND NOT(isOwner(goods, seller));
  Obligations
    O_1 : O(seller, buyer, true, happensBefore(delivered, delivered.delDueD));
    O_2 : O(buyer, seller, true, happensBefore(paid, paid.payDueD));
    O_3 : **violates**(O_2.**instance**) → O(buyer, seller, true, happens(paidLate, _));
  SurvivingObls
    SO_1: O(seller, buyer, true, not happens(disclosed(**self**), t) AND (t within **activates**(**self**) + 6 months));
    SO_2: O(buyer, seller, true, not happens(disclosed(**self**), t) AND (t within **activates**(**self**) + 6 months));
  Powers
    P_1: **violates**(O_2.**instance**) → P(seller, buyer, true, **suspends**(O_1.**instance**));
    P_2: happensWithin(paidLate, **suspension**(O_1.**instance**)) → P(buyer, seller, true, **resumes**(O_1.**instance**));
    P_3: not(happensBefore(delivered, delivered.delDueDate + 10 days)) → P(buyer, seller, true, **terminates**(**self**));
  Constraints
    NOT(isEqual(buyer, seller));
    forAll o | **self**.obligation.**instance** (CannotBeAssigned(o));
    forAll p | **self**.power.**instance** (CannotBeAssigned(p));
endContract

Contract Signature. The second part of a contract specification begins with its name and typed parameters. Parameters consist of at least two roles with optional variables that determine properties of contractual elements. During contract formation, roles are assigned to parties. For instance, *Meat-*

Sale (shown in Table II) is a contract between roles *buyer* and *seller*, where *seller* promises to deliver *qnt* quantity of meat with *qlt* quality to *buyer*; and *buyer* promises to pay the amount owed *amt* with currency *curr* before due date *payDueDate*. The *buyer* and *seller* are assigned (e.g., EatMart and Great Argentinian Meat Company) upon instantiation.

Contract Body. Contracts also contain local variable declarations; preconditions and postconditions; obligations and powers; as well as contract constraints that define liveness and safety properties.

Obligations. The main part of a contract consists of obligations. An obligation is specified as *O_{id}:O(debtor, creditor, antecedent, consequent)*. Debtor and creditor are roles, and antecedent and consequent are legal situations (specified by propositions). Antecedent and consequent propositions describe situations that need to hold for obligations to be fulfilled. Obligations become *InEffect* when their antecedents becomes true. *Suspensive Obligations* require a trigger to be created. Triggers are situations that are stated in terms of propositions and are located on the left side of the '$\rightarrow$' symbol. If there are no triggers mentioned in the specification, an obligation will be instantiated but will take effect only when its antecedent becomes true. In Table II, three obligations are specified for the example contract:

– O_1 obliges the seller towards the buyer to bring about the meat delivery by due date; it should be noted that, since quantity and quality are attributes of the meat, delivery has not occurred if these attributes are not complied with.

– O_2 obliges the buyer towards the seller to bring about payment by its due date.

– O_3 obliges the buyer towards the seller to bring about late payment. O_3 is triggered by the violation of O_2. The amount of late payment is specified in the *Declarations* section.

Surviving Obligations. They are obligations that survive after the *Termination* of a contract. Surviving obligations are usually prohibitions such as non-disclosure clauses (e.g., SO_1 and SO_2 in Table II). They too can have triggers.

Powers. A power is specified as *P_{id}:P(creditor, debtor, antecedent, consequent)*, where the creditor and debtor are roles, the antecedent is a legal situation described as a proposition, and consequent is a proposition describing a legal situation that can be brought about by the *creditor*. In Table II, three powers are specified:

– P_1 allows the seller to suspend delivery (i.e., O_1.instance) if obligation O_2 has been violated.

– P_2 allows the buyer to resume O_1 with a late payment (including interests).

– P_3 allows the buyer to terminate the contract, if meat delivery does not occur within ten days after the delivery due date.

A power entitles the creditor to bring about the consequent. For example, P_1 entitles the seller to perform the suspending action and bring about a *suspends (O_1.instance)* situation. A power is activated whenever its antecedent is true. If a party obtains a power, it can change the states of obligations, powers and contracts as stated in its consequent. For example, P_3 can bring about *unsuccessful termination* of the contract if its antecedent becomes true (which is always true in this case). Just as obligations, powers can be instantiated by triggers.

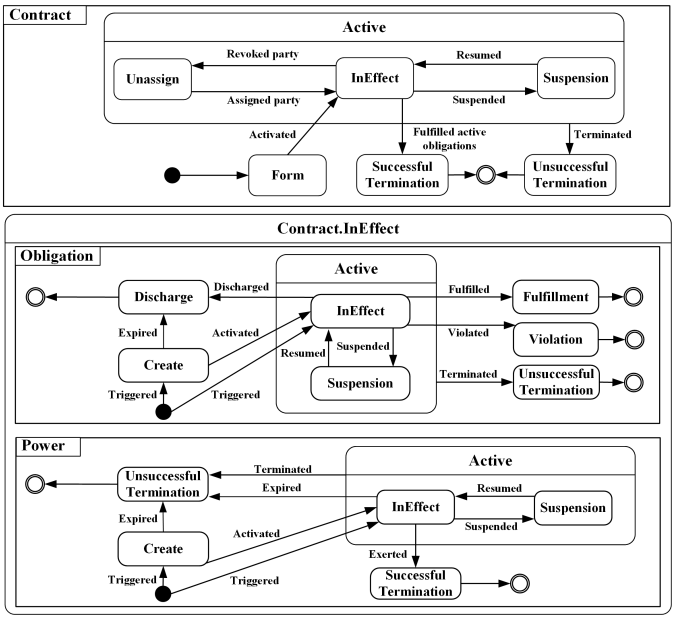

Fig. 2. SDs of the contract, obligation and power concepts

Constraints. Liveness constraints ensure that every contract execution terminates in a bounded amount of time, while safety constraints ensure that bad things do not happen during any execution. The following are safety constraints: *CannotBeAssigned(o)* disallows assignment of obligation instance *o* during the execution of a contract, whereas *not(isEqual(seller, buyer))* prohibits any party from being assigned to both roles at the same time.

IV. Syntax and Semantics

Syntax. The syntax of Symboleo is defined in terms of an Extended BNF grammar, for which we have an editor prototype (based on Xtext) [15].

Semantics. The most important aspect of the semantics of Symboleo concerns instances of contracts, obligations and powers that have a lifecycle that can be described in terms of statecharts (Fig. 2). A change of state for any contract, obligation or power instance is marked by an event. By recording events, for example in a blockchain ledger, smart contracts can monitor contract execution, ensure compliance to the contract, and determine violations and violators.

An obligation (instance) comes into existence as soon as the contract forms, but a suspended obligation depends on a trigger to resume. In addition, the proposed statecharts capture dependencies among the lifecycles of obligations, powers and contract. For example, when an active contract terminates unsuccessfully, e.g., because one of the parties exerts its power to terminate (cancel) it, all active obligations and powers transition to their *unsuccessful termination* state.

After contract formation, parties are bound to the contract

but the contract only becomes active on its effective date. During assignment of a contract [10], a contract may enter the *Unassign* state when the assigner withdraws, and then will remain in that state until an assignee is assigned. A contract may also be *suspended* if one of the parties exerts its suspension powers, or if a force majeure occurs, e.g., a natural catastrophe. Upon suspension, all obligation and power instances associated with the suspended contract are suspended as well. The suspended contract waits for an event that resumes it, such as a suspension deadline or an action performed by some party. After resumption, all instances of suspended obligations and powers return to the *InEffect* state. A contract successfully terminates (*SuccessfulTermination*) if all of its active obligations, except surviving ones, are fulfilled. In other cases, namely termination due to the exertion of a power or contract expiration while in the *Active* superstate, the contract and its active obligations and powers terminate unsuccessfully (*UnsuccessfulTermination*). If a material obligation is violated (material breach of the contract), Contract Law usually allows the damaged party to terminate the contract even if such power is not explicitly specified in the contract. *Renegotiation* and *renewal* are expressed in terms of implicit powers for every contract specification that can be activated upon all contractual parties' agreement and will be further explored in future work.

Conditional obligations are created (instantiated) when their triggers become true[4]. However, a trigger transitions an unconditional obligation (whose antecedent is always *true*) to the *InEffect* state directly. A conditional obligation is not activated until its antecedent becomes true. In the case of antecedent expiration, the obligation is discharged, since there is no possibility for it to become true after it has expired. Discharged obligations are cancelled obligations rather than unsuccessfully terminated ones. When an obligation instance becomes *InEffect*, its debtor can fulfill it by bringing about its consequent. The breach (transitioning to the *violation* state) of an obligation instance, e.g., because a deadline has passed, may trigger a power that entitles its creditor to suspend, terminate or discharge one or more *InEffect* obligation instances, or may trigger another obligation[5]. In the case of suspension, the debtor is not responsible against the creditor to bring about the obligation until an event, e.g., the fulfillment of another obligation, resumes it.

Powers are instantiated and activated in the same way as obligations. In many cases, events such as violations of obligations trigger them to become *InEffect*. A power might have a deadline for exertion, i.e., a deadline in its antecedent. After the deadline, the power expires thus entering the *Unsuccessful Termination* state.

The formal semantics of contract, obligation and power instance lifecycles is specified through 27 axioms. Due to space constraints, we present here three of these axioms in eqs. 1-3, while the rest are available in [17]. The axioms' five primitive predicates are listed in Table III. Since Symboleo supports both temporal interval and point expressions, some predicates are adopted from Allen [13], namely *occurs(s,T)*, while *initiates(e,s)*, *terminates(e,s)*, *happens(e,s)* and *holdsAt(s,t)* are adopted from the event calculus [18]. Moreover, as a shorthand, we allow events to be used in place of points in time expressions, and situations in place of intervals, as in *'e within s'*, where event *e* represents a time point and situation *s* represents a time interval.

TABLE III
PRIMITIVE PREDICATES OF SYMBOLEO

e within s	situation *s* holds when event *e* happens.
occurs(s, T)	situation *s* holds during the whole interval *T*, not just in any of its subintervals.
initiates(e, s)	event *e* brings about situation *s*.
terminates(e, s)	event *e* terminates situation *s*.
happens(e, t)	event *e* happens at time instance *t*.

Axiom 1 (Create a conditional obligation): for all conditional triggered obligations o of contract c, if o is triggered while c is in effect, then o is created. **Assumption:** $o.ant$ denotes the antecedent of obligation o.

$$\begin{aligned} &happens(triggered(o), \_) \wedge \\ &(triggered(o) \textbf{ within } InEffect(c)) \wedge \neg(o.ant = true) \\ &\quad \rightarrow initiates(triggered(o), create(o)) \end{aligned} \tag{1}$$

Axiom 2 (Terminate an obligation by a power): for any obligation o and power p of contract c, if the consequent of p implies that o is terminated and p is exerted while p is in effect, then o is terminated unsuccessfully.

$$\begin{aligned} &(e = terminated(o)) \wedge (e \textbf{ within } active(o)) \wedge \\ &(e \textbf{ within } InEffect(p)) \wedge (e \textbf{ within } InEffect(c)) \wedge \\ &(p.cons \rightarrow happens(terminated(o), \_)) \\ &\quad \rightarrow initiates(e, unsuccessfulTermination(o)) \wedge \\ &terminates(e, active(o)) \wedge happens(terminated(o), \_) \end{aligned} \tag{2}$$

Axiom 3 (Suspend an obligation by contract suspension): for any obligation o of contract c, if c is suspended while o is in effect; then o is suspended.

$$\begin{aligned} &(e = suspended(c)) \wedge happens(e, \_) \wedge \\ &(e \textbf{ within } InEffect(o)) \wedge (e \textbf{ within } InEffect(c)) \rightarrow \\ &\quad initiates(e, suspension(o)) \wedge terminates(e, InEffect(o)) \end{aligned} \tag{3}$$

We have tested these axioms through a Prolog-based prototype reasoning tool by checking the sample Sales-of-goods contract. The tool and the test cases (with successful results) are also available [17].

V. RELATED WORK

There has been much work on the formalization of legal concepts. Logicians model legal concepts with variants of Deontic Logic, such as Standard Deontic Logic [19] and Defeasible Logic [20]. Event Calculus [21] and Linear Temporal Logic [22] have been used to formalize obligations, permissions and powers [23]. Such approaches have not addressed many aspects of contracts, focusing instead on modelling legal relations (legal positions). A process view of contracts was proposed in [4], [5] where contracts are modelled as finite state machines (FSMs). This enables normative monitoring of

[4]In some cases, triggers can always be *true*, e.g. *O1* in Table II.

[5]This is also known as a Contrary to Duty (CTD) Obligation [16].

contracts [5]. Our approach considerably extends theirs, by using a FSM model that has separate machines for the contract as a whole, obligations, and powers. Likewise, Chesani et al. [3] propose an event calculus-based axiomatization that formally specifies FSM of time-aware commitments in order to monitor business processes, but do not address powers and surviving obligations. These axioms have been extended by Günay et al. [24] to include conditional commitments where there are antecedents with consequents.

Our view of contracts-as-processes is in line with that of Azzura [25]. However, these processes should be defined declaratively in terms of outcomes, rather than operationally in terms of activities. We have mentioned here and will show in future work that outcomes are events that are the result of obligations and powers being acted upon and can be recorded in logs or in the blockchain. Ladleif and Weske [26] recently proposed a unifying model of legal smart contracts based on UFO-L, where legal relations enable actions when their conditions occur and are updated by actions. This is similar to our notion of power. However, their model only has primitive time and does not support several elements of legal contracts.

Various smart contract languages have been introduced for distributed ledger systems. Many of them are implementation dependent and do not completely capture the concepts involved in legal contracts. The languages that have come closest to addressing both these issues are the Accord Project (https://www.accordproject.org), DAML (https://daml.com) and CSL (https://www.deondigital.com). Although these are not as widely used as Solidity (https://solidity.readthedocs.io), they abstract out the storage layer from the language, making it implementation agnostic. Still, these languages only capture basic legal notions (obligations but not power) and their development is in early stages. These languages are really programming languages, rather than analyzable specification languages such as Symboleo.

VI. Conclusions and Future Work

We have sketched the fundamental elements of Symboleo, a formal specification language for contracts and their requirements. In the spirit of disciplined Software Engineering principles, such specifications are intended to serve as starting point towards smart contract implementations on platforms offering adequate distributed and secure ledger functionalities, such as blockchain platforms. Our language is based on concepts that considerably extend the state of the art, including an extensible ontology, an extensible state machine model, logic specification by axioms, and Prolog prototyping.

For future work, we plan to (a) develop a tool-supported process for transforming contract specifications into smart contract code, e.g., in DAML; (b) develop formal analysis methods for contract specifications, likely with SMT solvers; (c) improve the usability of the syntax; (d) develop tools for the semi-automatic transformation of legal contract texts into formal specifications (e.g., to analyze existing contracts); (e) extend Symboleo to support advanced subcontracting features, and (f) perform case studies. An Xtext-based editor for (a) and a Prolog-based tool for (b) are currently under development [15], [17]. We have also identified case studies in the energy and service sectors, with industrial partners.

References

[1] H. Shahid, "How smart contracts are transforming the landscape of insurance industry," 2019, accessed 2019-02-05. [Online]. Available: http://bit.ly/32MWZJc

[2] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997.

[3] F. Chesani, P. Mello, M. Montali, and P. Torroni, "Representing and monitoring social commitments using the event calculus," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 85–130, 2013.

[4] A. Daskalopulu, "Modelling legal contracts as processes," in *Database and Expert Systems Applications, 2000. 11th Int. Workshop*. IEEE, 2000, pp. 1074–1079.

[5] G. Governatori, F. Idelberger, Z. Milosevic, R. Riveret, G. Sartor, and X. Xu, "On legal contracts, imperative and declarative smart contracts, and blockchain systems," *Artificial Intelligence and Law*, vol. 26, no. 4, pp. 377–409, 2018.

[6] C. Griffo, J. P. A. Almeida, and G. Guizzardi, "Towards a legal core ontology based on Alexy's theory of fundamental rights," in *Multilingual Workshop on Artificial Intelligence and Law (ICAIL)*, 2015.

[7] R. Alexy, *A theory of constitutional rights*. Oxford University Press, USA, 2010.

[8] W. N. Hohfeld, "Some fundamental legal conceptions as applied in judicial reasoning," *Yale Lj*, vol. 23, p. 16, 1913.

[9] Wikipedia contributors, "Asset — Wikipedia, the free encyclopedia," https://bit.ly/35TjZrn, 2019, [Online; accessed 21-October-2019].

[10] B. H. Bix, *Contract law: rules, theory, and context*. Cambridge University Press, 2012.

[11] M.-P. Allard, "The retroactive effect of conditional obligations in tax law," *Canadian Tax Journal*, vol. 49, no. 6, pp. 1726–1839, 2001.

[12] G. Guizzardi, G. Wagner, J. P. A. Almeida, and R. S. Guizzardi, "Towards ontological foundations for conceptual modeling: the unified foundational ontology (UFO) story," *Applied ontology*, vol. 10, no. 3-4, pp. 259–271, 2015.

[13] J. F. Allen, "Towards a general theory of action and time," *Artificial intelligence*, vol. 23, no. 2, pp. 123–154, 1984.

[14] E. Kindler, "Safety and liveness properties: A survey," *Bulletin of the European Association for Theoretical Computer Science*, vol. 53, no. 268-272, p. 30, 1994.

[15] S. Sharifi, T. Paul, and A. Parvizimosaed, "Symboleo-IDE," Jun. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3903951

[16] H. Prakken and M. Sergot, "Contrary-to-duty obligations," *Studia Logica*, vol. 57, no. 1, pp. 91–115, 1996.

[17] A. Parvizimosaed and S. Sharifi, "Symboleo Compliance Checker," Jun. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3903954

[18] M. Shanahan, "The event calculus explained," in *Artificial intelligence today*. Springer, 1999, pp. 409–430.

[19] W. M. Farmer and Q. Hu, "FCL: A formal language for writing contracts," in *Quality Software Through Reuse and Integration*. Springer, 2016, pp. 190–208.

[20] I. A. Letia and A. Groza, "Running contracts with defeasible commitment," in *IEA/AIE 2006*, ser. LNAI, vol. 4031. Springer, 2006, pp. 91–100.

[21] A. D. Farrell *et al.*, "Performance monitoring of service-level agreements for utility computing using the event calculus," in *First IEEE Int. Workshop on Electronic Contracting*. IEEE, 2004, pp. 17–24.

[22] H. L. Cardoso and E. Oliveira, "Directed deadline obligations in agent-based business contracts," in *Coordination, Organizations, Institutions and Norms in Agent Systems V*. Springer, 2010, pp. 225–240.

[23] A. J. Jones and M. Sergot, "A formal characterisation of institutionalised power," *Logic Journal of the IGPL*, vol. 4, no. 3, pp. 427–443, 1996.

[24] A. Günay and P. Yolum, "Detecting conflicts in commitments," in *International Workshop on Declarative Agent Languages and Technologies*. Springer, 2011, pp. 51–66.

[25] F. Dalpiaz, E. Cardoso, G. Canobbio, P. Giorgini, and J. Mylopoulos, "Social specifications of business processes with azzurra," in *9th International Conference on Research Challenges in Information Science (RCIS)*. IEEE CS, 2015, pp. 7–18.

[26] J. Ladleif and M. Weske, "A unifying model of legal smart contracts," in *Conceptual Modeling*. Cham: Springer, 2019, pp. 323–337.