# Subcontracting, Assignment, and Substitution for Legal Contracts in Symboleo *

Alireza Parvizimosaed[1], Sepehr Sharifi[1], Daniel Amyot[1], Luigi Logrippo[1,2], and John Mylopoulos[1]

[1] School of EECS, University of Ottawa, Ottawa, Canada
{aparv007, sshar190, damyot, logrippo, jmylopou}@uottawa.ca
[2] Université du Québec en Outaouais, Gatineau, Canada

**Abstract.** *Legal contracts* specify obligations and powers among legal subjects, involve assets, and are constrained to satisfy quality constraints. *Smart contracts* are software systems that monitor the execution of legal contracts by contracting parties to ensure compliance. As a starting point for developing software engineering concepts, tools, and techniques for smart contracts, we have proposed Symboleo, a formal specification language for legal contracts. The complexity of real-life contracts (e.g., in the construction and transportation industries) requires specification languages to support execution-time operations for contracts, such as subcontracting, assignment, delegation, and substitution. This paper formalizes such concepts by proposing for them a syntax and axiomatic semantics within Symboleo. This formalization makes use of primitive operations that support the transfer or sharing of right, responsibility, and performance among contracting and subcontracting parties. A prototype compliance checking tool for Symboleo has also been created to support monitoring compliance for contracts that include subcontracting aspects. A realistic freight contract specified in Symboleo is provided as an illustrative example for our proposal, as well as a preliminary evaluation with positive results.

**Keywords:** Contracts · formal specification languages · legal subcontracts · smart contracts · subcontracting

## 1 Introduction and Motivation

Legal contracts are documents that have been used since antiquity for business transactions to specify obligations and powers among roles. They involve assets, and define constraints enforcing specific modalities. In a world of digital transformations, many aspects of contracts are being automated. In particular, *smart contracts* are software systems that monitor the execution of legal contracts by

* 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"

contracting parties to ensure compliance. Smart contracts have received much attention in the literature and news recently because of their potential application in multiple areas, including Finance, Commerce, Government, and Agriculture. We are interested in developing concepts, tools, and techniques for building monitorable smart contracts. As a starting point for this endeavour, we have proposed *Symboleo*[3], a formal specification language for legal contracts [14].

Real-life contracts (e.g., in the construction and transportation industries) are complex artifacts, based on a rich ontology and an expressive specification language. Moreover, they can change during execution time in the sense that obligations and powers may be cancelled by a party that has the power to do so, and assignments to parties may be changed as well through subcontracting, assignment, delegation, novation, and substitution. Intermediate contractors may further subcontract to third parties, leading to a chain of delegations of performance and responsibility (i.e., who does what and who is responsible for what). For example, large construction projects engage multiple subcontractors in a hierarchy of contracts in order to reduce construction cost and save time [15].

The contributions of this work include (a) a set of execution-time operations that allow the sharing or change of rights, performance responsibilities, and liabilities among contracting parties; (b) a syntax and axiomatic semantics for these operations; (c) the definition of the legal notions of subcontracting, assignment, and substitution in terms of the primitive operations; (d) a preliminary evaluation of the proposal using a realistic freight contract with subcontracting; and (e) a compliance checking tool for Symboleo that includes reasoning with subcontracts, substitutions, and assignments.

The rest of the paper is structured as follows. Section 2 gives a quick overview of Symboleo, while section 3 introduces primitive execution-time operations along with their syntax and semantics, which support the transfer or sharing of performance or responsibility. Section 4 discusses how the legal concepts of subcontracting, assignment, and substitution can be expressed in terms of the proposed primitive operations. In section 5, we adopt a realistic freight contract from the literature, specify it in Symboleo and show how to deal with subcontracting, assignment and others with our proposal. Section 6 highlights how such contract specifications can be analyzed with a compliance checker tool. Section 7 discusses related work, while section 8 concludes.

## 2 Overview of Symboleo

Contracts can be understood as prescriptions of allowable legal process executions. They specify obligations and powers that determine *who* is responsible to *whom* for *what* and *when*. The *how* is left to the responsible party to determine. In this respect, contracts can be seen as outcome-oriented processes, in the sense that they specify what should be the outcome of a contract execution, without specifying the activities that have to be performed. Contracts are very different

[3] From the Greek word $\Sigma\upsilon\mu\beta o\lambda\alpha\iota o$, meaning contract and pronounced '*simvoleo*'

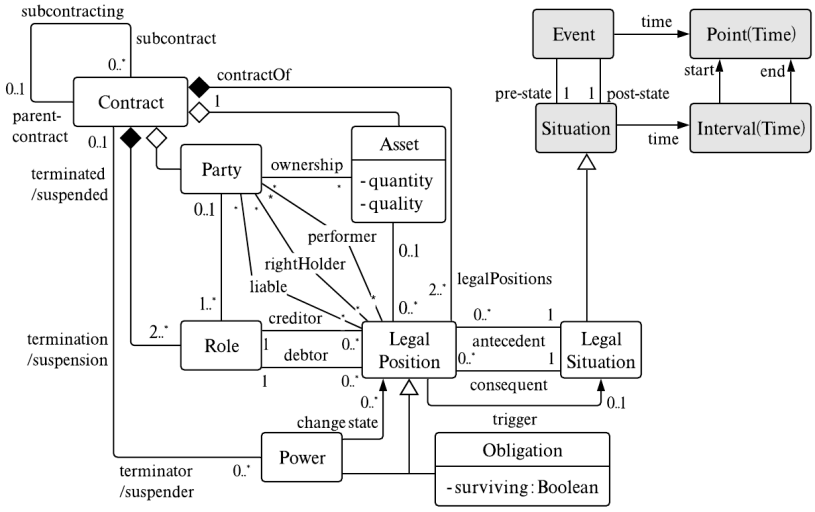


Fig. 1: Symboleo's contract ontology

from business processes in that powers can change the status of obligations, e.g., by cancelling obligations or imposing new ones during contract execution. The concepts of our contract ontology are briefly reviewed in the following. Other definitions can be found in [14].

As shown in Symboleo's ontology (Fig. 1), a legal contract (or just contract henceforth) is defined as a collection of obligations and powers between two or more roles. A contract is concerned with at least one asset (contractual consideration) from each contractual role. For a contract execution, roles are assigned to parties (persons or legal entities) that take part in the contract execution.

A legal position is either an obligation or a power that defines a legal relationship between a debtor and a creditor, has a (possibly null) legal situation as activation condition (antecedent), and obliges the debtor to bring about another legal situation (consequent). Legal positions can be instantiated via triggers. Obligations are legal duties of a debtor towards a creditor to bring about a consequent, while powers define the right of a creditor to create, change, suspend, or cancel legal positions. Antecedents, consequents, and triggers are propositions constraining the occurrence of instantaneous events and situations holding over a time interval. The full ontology of Symboleo, which extends the UFO-L foundational legal ontology [6] (e.g., see shaded concepts in Fig. 1), is described in more detail in [14].

The aim of the Symboleo language is to enable contract creators to specify *parameterized* contract templates that can be instantiated with different parameter values. Symboleo's formal semantics also enables checking contracts for safety and liveness properties, which respectively verify that bad things do not happen (e.g., payment loopholes or privacy violations) and that good things eventually happen (e.g., assets will be delivered and will be paid for) during the execution of a contract instance.

Table 1: Sample sale-of-goods (SOG) contract specification

**Domain** salesD

/∗ *Includes concepts that are specializations of the contract ontology concepts such as Buyer/Seller, Goods and Delivered/Paid, which are specializations of* Role, Asset *and* Event, *respectively.* Additional attributes may also be specified. ∗/
Goods **isA** Asset **with** goodsID: Integer;
...
Delivered **isA** Event **with** delAddress: String, delDueDate: Date;

**endDomain**
**Contract** salesC(seller: Seller, buyer: Buyer, ID: Integer, amnt: Integer, curr: Currency, delAdd, delDd: String)

**Declarations**
/∗ *Here, the values of the parameters are passed on to the variables that were defined in the domain model.* ∗/
goods : Goods **with** goodsID := ID;
...
delivered : Delivered **with** delAddress := delAdd, delDueDate := delDd;
**Preconditions**
isOwner(seller, goods) AND NOT isOwner(buyer, goods);
**Postconditions**
isOwner(buyer, goods) AND NOT isOwner(seller, goods);
**Obligations**
$O_1$ : O(Seller, Buyer, true, **happensBefore**(delivered, delivered.delDueD));
$O_2$ : O(Buyer, Seller, true, **happensBefore**(paid, paid.payDueD));
**Powers**
$P_1$ : **violates**($O_2$, _) → P(Seller, Buyer, true, **terminates**(salesC));
**SurvivingObl**
/∗ *Some obligations will remain active even after the contract has terminated successfully, namely confidentiality obligations.* ∗/
**Constraints**
**not**(isEqual(buyer, seller));

**endContract**

We illustrate the workings of Symboleo using a sale-of-goods example. Suppose there is a contract between a buyer and a seller, consisting of three template clauses, namely two obligations and one power (*right*) guarded by a trigger:

O1. The Seller shall deliver the Goods <*goodsID*> to the Buyer at address <*delAdd*> before the delivery due date <*delDd*>.
O2. The Buyer shall pay the amount of <*amnt*> in currency <*curr*> to the Seller before the payment due date <*payDd*>.
P1. In case of violation of the payment obligation (O2), the Seller has the right to terminate the contract.

A contract specification has a *domain* section and a *contract body* section (Table 1). Domain-dependent concepts and axioms are defined in the domain section as specializations of Symboleo's ontology (Fig. 1). The contract body starts with the contract's *signature*, which contains parameters and their types. Parameter values are used to instantiate a contract. Aside from the specification of obligations, powers, and *surviving obligations* (that persist after the successful

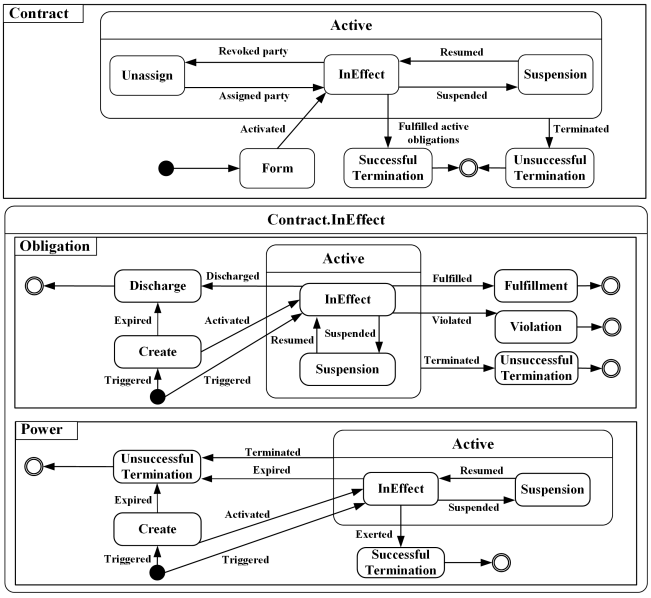


Fig. 2: UML statecharts for obligations, powers, and contracts [14]

termination of a contract, e.g., a non-disclosure clause), pre/post-conditions and constraints on the contract execution are also specified in the contract body.

The first two clauses of this contract are obligations ($O_1$ and $O_2$ respectively), while the third is a power ($P_1$). As seen in the example, legal positions have as signatures *[trigger→] O(debtor, creditor, antecedent, consequent)* for obligations and *[trigger→] P(creditor, debtor, antecedent, consequent)* for powers.

The lifecycle of a contract/obligation/power instance is captured by UML statecharts defined in Fig. 2 [14]. State transitions are events that are recorded on ledgers (preferably with assured integrity as in blockchains) that enable the monitoring function of smart contracts. A contract is initially in its Form state and transitions to the InEffect state when it is signed and its effective date is reached. Since $O_1$ and $O_2$ do not have a trigger (true by default), they transition to the Create state when the contract transitions to the InEffect state. However, $P_1$ will be instantiated whenever its trigger becomes true, i.e., the event *violated($O_2$)* happens or $O_2$ transitions to the Violation state. After becoming InEffect (i.e., the antecedent becomes true), the creditor of $P_1$ has the power to bring about the consequent (exertion of power), i.e., transitioning the contract to the Unsuccessful Termination state, which results in all other active obligations and powers transitioning to their Unsuccessful Termination state. After exertion, the power itself transitions to its Successful Termination state.

The statecharts act as the baseline for Symboleo's semantics. In [14], the semantics of transitions are given in terms of axioms that use the predicates listed in Table 2, inspired by the Event Calculus [13].

Table 2: Primitive predicates of Symboleo

| | |
|---|---|
| $e$ within $s$ | situation $s$ holds when event $e$ happens. |
| occurs($s$, $T$) | situation $s$ holds during the whole interval $T$, but does not occur 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 point $t$. |
| holdsAt($s$, $t$) | situation $s$ holds at time point $t$. |

## 3 Primitive Execution-Time Operations

During the execution of contracts, when specific values are bound to the parameter variables of contract templates, certain *operations* can change the contract state at runtime. The most notable types of legal contract execution-time operations are *subcontracting*, *delegation*, *substitution*, *novation* and *assignment*.

These terms may have different interpretations in different legal jurisdictions, and possibly even within a single legal jurisdiction. For example, while *assignment* is defined as transferring the claims and rights of an assignor to an assignee in the Common Law system, some courts in the USA will also treat it as transferring a contract as a whole, depending on the intentions inferred from the assignment clause [9].

Despite various intention-dependent definitions, the actions underlying these operations can be categorised as sharing or transferring rights, responsibilities, or performance of parties. In this paper, we have extended the original Symboleo ontology [14] with such relationships, defined between Party and Legal Position in Fig. 1. Note that "liable" here is a synonym of "responsible". From a syntactic viewpoint:

– ***rightHolder*(*x*, *p*):** for an obligation/power instance $x$, party $p$ is *rightHolder*.
– ***liable*(*x*, *p*):** for an obligation/power instance $x$, party $p$ is *liable*.
– ***performer*(*x*, *p*):** for an obligation/power instance $x$, party $p$ is *performer*.

These terms are used in **Axioms 1-4** of the augmented axiomatic semantics of Symboleo, based on the predicates of Table 2. For all obligation instances $o$, power instances *pow.* and party instances $p$, there exists a time point $t$ for which the following hold:

$$\begin{aligned} &happens(activated(o), t) \ \wedge \ holdsAt(bind(o.debtor, p), t) \\ &\rightarrow \ initiates(activated(o), liable(o, p)) \wedge \ initiates(activated(o), performer(o, p)) \end{aligned} \tag{1}$$

$$\begin{aligned} &happens(activated(o), t) \ \wedge \ holdsAt(bind(o.creditor, p), t) \\ &\rightarrow \ initiates(happens(activated(o), rightHolder(o, p)) \end{aligned} \tag{2}$$

$$\begin{aligned} &happens(activated(pow), t) \ \wedge \ holdsAt(bind(pow.creditor, p), t) \\ &\rightarrow \ initiates(activated(pow), rightHolder(pow, p)) \\ &\wedge \ initiates(activated(pow), performer(pow, p)) \end{aligned} \tag{3}$$

$$\begin{aligned} &happens(activated(pow), t) \ \wedge \ holdsAt(bind(pow.debtor, p), t) \\ &\rightarrow \ initiates(activated(pow), liable(pow, p)) \end{aligned} \tag{4}$$

Table 3: Primitive execution-time operations

| | |
|---|---|
| *shareR*($x$, $p$) | Party $p$ becomes a rightHolder for obligation/power instance $x$. |
| *shareL*($x$, $p$) | Party $p$ becomes liable for obligation/power instance $x$. |
| *shareP*($x$, $p$) | Party $p$ becomes a performer for obligation/power instance $x$. |
| *transferR*($x$,$p_{old}$,$p_{new}$) | Party $p_{new}$ becomes a rightHolder for obligation/power instance $x$ and $p_{old}$ will no longer be a rightHolder for $x$. |
| *transferL*($x$,$p_{old}$,$p_{new}$) | Party $p_{new}$ becomes liable for obligation/power instance $x$ and $p_{old}$ will no longer be liable for $x$. |
| *transferP*($x$,$p_{old}$,$p_{new}$) | Party $p_{new}$ becomes a performer for obligation/power instance $x$ and $p_{old}$ will no longer be a performer for $x$. |

In other words, after the time an obligation instance $o$ is activated, the party bound to the debtor role of $o$ is the *performer* of $o$ and is *liable* for $o$ (**Axiom 1**), and the party bound to the creditor role of $o$ is the *rightHolder* of $o$ (**Axiom 2**).

After the time a power instance *pow* is activated, the party bound to the creditor role of *pow* is the *rightHolder* and the *performer* of *pow* (**Axiom 3**), and the party bound to the debtor role of pow is *liable* for *pow* (**Axiom 4**).

Based on the above axioms, we define a set of primitive contract execution-time operations (Table 3) to express what can happen during the execution of a contract instance. An execution-time operation is initiated/terminated by an event with a corresponding name (e.g., *shareR* is initiated/terminated using event *shared*$\underline{R}$). The semantics of the primitive sharing and transfer operations defined in Table 3 are exemplified with *shareR* and *transferR* (a party can share or transfer her rights under a contract to another party). The semantics of the other four primitive operations are defined with similar axioms not presented here due to space limitations.

**Axiom 5:** Given active obligation/power instance $x$, party $p$, and the fact that *sharedR*($x$, $p$) is the event that initiates the sharing of $x$ with $p$, at some time $t$ the following holds:

$$\begin{aligned} &happens(sharedR(x,p),t) \ \wedge \ holdsAt(active(x),t) \ \rightarrow \\ &\quad initiates(sharedR(x,p), rightHolder(x,p)) \end{aligned} \tag{5}$$

**Axiom 6:** Given active obligation/power instance $x$, party instances $p_{new}$ and $p_{old}$, and the fact that *transferredR*($x$, $p_{old}$, $p_{new}$) is the event that initiates the transfer of rights, there exists a time point $t$ for which the following holds:

$$\begin{aligned} &happens(transferredR(x,p_{old},p_{new}),t) \ \wedge \ holdsAt(active(x),t) \ \wedge \\ &\quad holdsAt(rightHolder(x,p_{old}),t) \ \rightarrow \\ &\qquad initiates(transferredR(x,p_{old},p_{new}), rightHolder(x,p_{new})) \ \wedge \\ &\qquad terminates(transferredR(x,p_{old},p_{new}), rightHolder(x,p_{old})) \end{aligned} \tag{6}$$

These new primitive operation can now be used to implement various interpretations (e.g., from different jurisdictions) of contract execution-time operations. The next section defines three operations for general international law.

## 4 Assignment, Substitution, and Subcontracting

Although execution-time operations can have different meanings according to the practices in different jurisdictions or the intentions of the contractual parties, we focus here on the definitions of *assignment (of rights)*, *substitution (of contractual parties)*, and *subcontracting* due to their more stable and consistent definitions in different contexts and their frequent application in everyday practice.

We formally specify syntax (parametric shorthand) and semantics (axioms) for these operations in Symboleo, to enable runtime monitoring. Shorthands are situations in Symboleo and are captured as Prolog predicates in our tool. In the following axioms, $O$ and $P$ respectively represent the sets of all obligation instances and all power instances in the contract. Also, the dot (.) operator is used in some axioms to navigate our ontology, à la OCL.

**Assignment (of rights)**: $assignR(\{x_1, ..., x_n\}, p_{old}, p_{new})$
**Semantics:** A party can assign the rights that she is entitled to under a contract to a third-party [9]. Its axiom builds upon *transferR* (Axiom 6).
**Axiom 7:** For any set of obligation/power instances $x = \{x_1, ..., x_n\}$ that party $p_{old}$ is the rightHolder of, if $p_{old}$ assigns her rights for $x$ to another party $p_{new}$, then the rights for $x$ are transferred from $p_{old}$ to $p_{new}$. Here, *assignedR(x,p)* is the event that initiates the assignment, leading to many primitive transfers.

$$\begin{aligned}&\forall x \in \mathbb{P}(O \cup P), \forall x_i \in x : happens(assignedR(x, p_{old}, p_{new}), t) \wedge \\ &\quad holdsAt(rightHolder(x_i, p_{old}), t) \rightarrow\ happens(transferredR(x_i, p_{old}, p_{new}), t)\end{aligned} \tag{7}$$

**Contractual Party Substitution**: $substituteC(c, r, p_{old}, p_{new})$
**Semantics:** A contractual party might decide to leave an ongoing contract and have a third-party replace her in the contract. A party $p_{old}$ who has a role $r$ in contract $c$ can substitute herself with another party $p_{new}$ and transfer all of the rights, responsibilities, and performance of all the active obligations/powers $x$ to $p_{new}$, given the consent of all original parties and of $p_{new}$ [9].
**Axiom 8:** Given the consent of $p_{old}$, $p_{new}$, and other parties of the contract $c$ to $substituteC(c, r, p_{old}, p_{new})$, and given contract $c$, obligation/power $x$, and role $r$, and the fact that $substitutedC(c, r, p_{old}, p_{new})$ is the event that occurs and initiates the substitution, then there exists a time $t$ for which this holds:

$$\begin{aligned}&\forall x \in c.legalPosition : happens(consented(substitutedC(c, r, p_{old}, p_{new})), t) \\ &\wedge\ happens(substitutedC(c, r, p_{old}, p_{new}), t)\ \wedge\ holdsAt(active(c), t) \\ &\wedge\ holdsAt(bind(c.r, p_{old}), t)\ \rightarrow \\ &\quad initiates(substitutedC(c, r, p_{old}, p_{new}), bind(c.r, p_{new})) \\ &\quad \wedge\ terminates(substitutedC(c, r, p_{old}, p_{new}), bind(c.r, p_{old})) \\ &\quad \wedge\ happens(transferredR(c.x, p_{old}, p_{new}), t) \\ &\quad \wedge\ happens(transferredL(c.x, p_{old}, p_{new}), t) \\ &\quad \wedge\ happens(transferredP(c.x, p_{old}, p_{new}), t)\end{aligned} \tag{8}$$

**Subcontracting**: $subcontract\big(\{o_1, ..., o_m\}$ ***to*** $\{\{c_1, pa_1\}, ..., \{c_n, pa_n\}\}$ ***with*** $\{constr_1, ..., constr_n\}\big)$. Subcontracting involves sharing performance of a set of

contractual obligations with one or more other parties through subcontracts $c_1$, ..., $c_n$. Since single contractual counter-party is a simple and popular case of subcontracting, this paper focuses on this case and leaves the generic forms (i.e., multiple multilateral subcontracts) to future work.

**Semantics:** As Axiom 9 indicates, subcontracting is a legal way of granting new parties this privilege. Subcontractors fulfill the subcontracted obligations once they successfully terminate the corresponding well-designed subcontracts, which trigger events that bring about the consequents of the delegated obligations.

For instance, a seller may hire a carrier to transport goods from a warehouse to port A, another one to ship the goods from port A to port B, and a third one to transport the goods from port B to the final destination. In this case, successful termination of three subcontracts fulfills the corresponding obligations of the original contract. However, *violation*, *suspension*, and *unsuccessful termination* of subcontracts do not alter the state of the original contract's obligations since the contractor, as a liable party and primary performer, can run an alternative plan (e.g., subcontractor replacement) and consequently fulfill its original obligations. Contractors may stipulate some constraints to supervise further subcontracts, e.g., to acquire a main contractor's consent to shift its burden to a third party.

**Axiom 9:** For any set of obligation instances $o$ in $O$ that is *subcontracted* out under a set of contracts in $C$ to a set of parties in $PA$ subject to a set of domain assumptions expressed as additional propositional constraints ($\{constr_1, ..., constr_n\}$), then the performance of all subcontracted obligations is shared with all of the (sub)contractual counter-parties.

$$\begin{aligned}&\forall o \in \mathbb{P}(O), \forall cp \in \mathbb{P}(C \times PA) : \\ &happens(subcontracted(o, cp, \{constr_1, ..., constr_n\}), t) \wedge \\ &constr_1 \wedge ... \wedge constr_n \rightarrow \forall o_i \in o, \forall (c, pa) \in cp : happens(sharedP(o_i, pa), t)\end{aligned} \tag{9}$$

## 5 Case Study: Multiple Freights as Subcontracts

The sale-of-goods contract from section 2 has a delivery clause, and there are many examples of businesses subcontracting such obligations to third parties under a separate contract whose post-condition implies the satisfaction of the subcontracted obligation's consequent. One of the results (post-conditions) of a *Freight contract*'s successful completion (e.g., Tables 4 and 5) is that the goods (meat here) to be delivered by the *Shipper* are delivered to the desired delivery address (*delAdd*). Likewise, a precondition bans execution of the freight contract unless the good is ready on the required lading location (*pkAdd*).

Subcontracting of an obligation is the act of delegating the satisfaction of a consequent (*contractual performance*) of that obligation to another party under a new contract [9]. The subcontract, also a contract, can be created at runtime via a power that *implicitly* exists in the contract (as stated in formula 10). Right holders of such powers are restricted to subcontract obligations for which they are liable and all partners consent. The power to assign claims and subcontracts are present for both parties unless explicitly disallowed in the *constraints* part of the contract specification.

Table 4: Freight contract template example

| |
|---|
| Agreement is entered into effect between <*party1*> as Shipper, and <*party2*> as Carrier. |
| O1 The Carrier agrees to transport the goods as stated in tender sheet (<*qnt*> of <*qlty*> quality meat, in proper refrigerated conditions, from <*pkAdd*>, to <*delAdd*> on <*delDueDate*>). |
| O2 The Shipper should pay <*amt*>("amount") in <*curr*>("currency") to the Carrier for its services within 3 days after delivery of goods. |
| O3 The Shipper is additionally subjected to <*intRate*>% interest rate on the amount due if payment is breached. |

Table 5: Freight contract specification in Symboleo

**Domain** freightD

Shipper **isA** Role **with** pickupAddress: String;
Carrier **isA** Role **with** office: String;
Meat **isA** PerishableGood **isA** Asset **with** quantity: Integer, quality: MeatQuality;
Paid **isA** Event **with** amount: Integer, currency: Currency, from: Role, to: Role, payDueDate: Date;
Delivered **isA** Event **with** item : Meat, delAddress: String, delDueDate: Date;
MeatQuality **isA** Enumeration('PRIME', 'AAA', 'AA', 'A');
**teminates**{delivered, paid};

**endDomain**

**Contract** freightC(shipper: Shipper, carrier: Carrier, effDate: Date, qnt: Integer, qlty: MeatQuality, amt: Integer, curr: Currency, delAdd: String, delDd: Date, pkAdd: String, intRate: Integer)

**Declarations**
goods : Meat **with** quantity := qnt, quality := qlty;
paid : Paid **with** amount := amt, currency := curr, from := shipper, to := carrier, dueD:=payDueDate;
paidLate : Paid **with** amount := amt*(1 + intRate/100), currency := curr, from := shipper, to := carrier;
delivered : Delivered **with** item := goods, delAddress := delAdd, delDueDate := delDd;
atLocation : Situation **with** what : Asset, where : String; *// External situ. monitoring*
**Preconditions**
atLocation(goods, pkAdd)
**Postconditions**
atLocation(goods, delAdd)
**Obligations**
$O_1$ : O(carrier, shipper, true, happensBefore(delivered, delivered.delDueDate));
$O_2$ : **happens**(delivered, t) → O(shipper, carrier, true, happensBefore(paid, t + 3 days));
$O_3$ : **violates**($O_2$) → O(shipper, carrier, true, **happens**(paidLate, _));
**Powers** *// None*
**SurvivingObls** *// None*
**Constraints**
**not**(isEqual(shipper, carrier));

**endContract**

$$
\begin{aligned}
pow_{\mathrm{x}} : P\big(&creditor, debtor, rightHolder(pow_{\mathrm{x}}) = Liable(o_1) = ... = Liable(o_{\mathrm{m}}) \wedge \\
&\big(\forall c \in \{c_1, ..., c_{\mathrm{n}}\}, \exists r \in c.Role, bind(r, rightHolder(pow_{\mathrm{x}}))\big) \wedge \\
&\big(\forall p \in PA, \forall o \in \{o_1, .., o_{\mathrm{m}}\} : p = Liable(o) \rightarrow \\
&\quad happens(consented(p, subcontracted\big(o, \{(c_1, p_1), ..., (c_{\mathrm{n}}, p_{\mathrm{n}})\}\big), \_))\big), \\
&happens(subcontracted\big(\{o_1, ..., o_{\mathrm{m}}\}, \{(c_1, p_1), ..., (c_{\mathrm{n}}, p_{\mathrm{n}})\}\big), \_)\big)
\end{aligned}
\tag{10}
$$

The contract in Table 4 is a freight agreement between a shipper of goods (meat) and a carrier who provides shipping services. Table 5 contains a (non-instantiated) specification that will act as a template for the subcontract(s) of the delivery obligation of the sample contract introduced in section 2.

Assume the seller's warehouse of the sales-of-goods (SOG) example from Table 1 is located in Buenos Aires (Argentina) and the buyer's warehouse is located in Ottawa (Canada). The seller might decide not to fulfill the delivery obligation by himself, but rather would subcontract it to three different carriers: one to $carrier_{BA}$, for freight from the seller's warehouse to the port of Buenos Aires; one to $carrier_{Hal}$, for freight from Buenos Aires to Halifax; and one to $carrier_{Ott}$ for freight from Halifax to the buyer's warehouse in Ottawa. Notice that the pre/post-conditions of the freight contract specification ensure that all three freight contracts are executed sequentially. For example, the freight contract from Halifax to Ottawa is not executed before the goods are delivered to Halifax as a result of the successful execution of the contract with $carrier_{Hal}$.

## 6 Analysis

Contracts can be very complex artifacts that hide unwelcome consequences for some of their parties. To mitigate this risk, we developed an analysis tool[4] that takes as input a set of scenarios (each consisting of a sequence of events), along with the expected final states of the contract for each scenario, and actually runs each scenario to validate that it does end in the expected final state. The tool was implemented by using an existing reactive event calculus tool (jREC [10]), written in Java and Prolog, which was extended to support the Symboleo semantics and performs abductive reasoning on given scenarios. We designed six scenarios and corresponding test cases (Table 6) combining the SOG and Freight contracts. All tests involve meat sales between a seller in Argentina and a buyer in Ottawa, with freight subcontracting to a carrier. These test cases cover many possible states of obligations, powers, and contracts, especially boundaries cases.

In Table 6 and Fig. 3, V=Violation, F=Form, Fu=Fulfillment, I=InEffect, A=Active, UT=Unsuccessful Termination, and ST= Successful Termination of a contractual clause (i.e., states from Fig. 2). For example, the first test case violates the first obligation(V1) of Freight and (V1) of SOG, but fulfills SOG's second obligation (Fu2). In Fig. 3, the vertical axis shows the states of the contracts and their clauses (O1, O2, O3, P1), and the horizontal axis characterizes events over time (with time units between brackets). The delivery obligation is subcontracted to the Fedex carrier (SOG_subcontFedex) through a freight contract. However, in Test Case 6, after consent, Fedex assigns its payment rights to Walmart. As the freight contract proceeds independently, the delivery obligation of the freight contract stays active after the termination of SOG until its due date arrives and violates the obligation at time 9. Our tool monitors runtime responsibility, right, and performance relationships of parties. The results indicate that the execution of these tests complies with expected results,

[4] The tool is available at `https://sites.google.com/uottawa.ca/csmlab`

Table 6: Test Cases

| Test Case | Freight | SOG |
|---|---|---|
| **1.** Buyer pays off order but Carrier delivers the meat under inappropriate conditions resulting in spoiled meat. | V1 | V1, Fu2 |
| **2.** Carrier's transport is unable to ship loaded meat, and instead the shipper (i.e., Seller) delivers it himself to the Buyer under proper conditions before due date, and gets paid. | V1 | Fu1, Fu2 |
| **3.** Buyer refuses payment and neither Carrier nor Shipper delivers the meat till 10 days after due date. | V1 | V1, V2, A3 |
| **4.** Carrier delivers meat while Shipper awaits more than 10 days for Buyer's payment. | V2, A3 | Fu1, V2, A3 |
| **5.** Buyer refuses to pay off the agreed amount before due date and then the Seller terminates the contract and does not allow unloading the good at due location. | V1 | V2, ST3, $UT_{SOG}$ |
| **6.** Buyer pays original Seller after assigning payment rights to a third party. | - | V2, A3 |

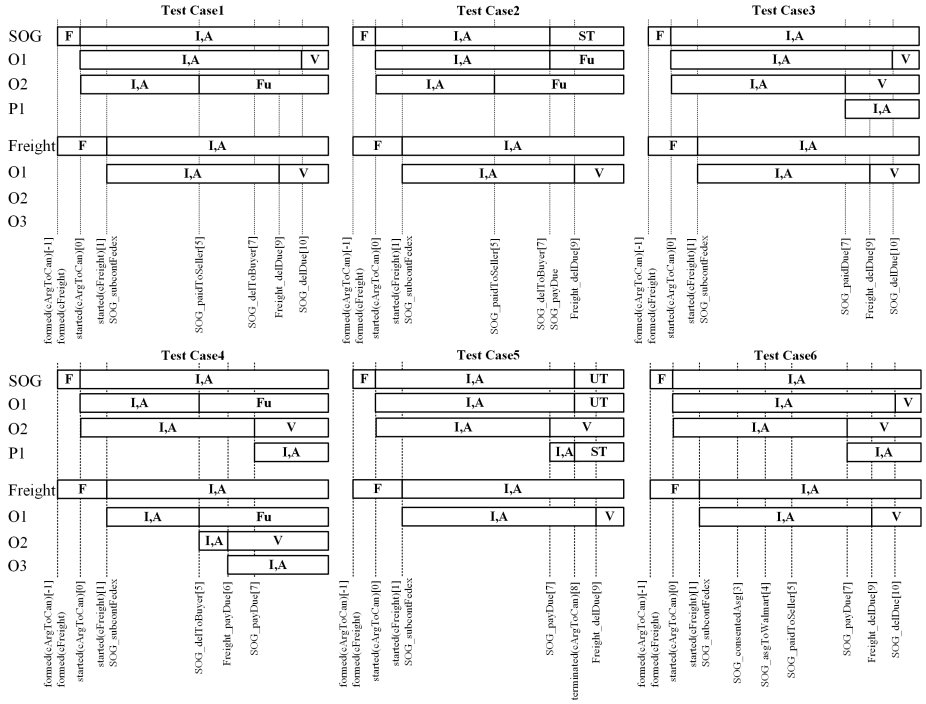


Fig. 3: Test results showing the states of contracts/clauses over events[time]

which partially validates Symboleo's axioms and our new subcontracting and substitution operations.

## 7 Related Work

Multi-agent systems investigate runtime commitment operations, namely delegation and assignment. Kafalı and Torroni [7] propose eight forms of social commitment delegations by discharging and instantiating commitments. Implicit and explicit delegations partially express semantics of obligation delegation and substitution operations respectively. Implicit operation generates a commitment between a party and a third party while keeping the original commitment. Explicit operations cancel the original commitment and then create the new commitment. They also introduce causal delegation chains and delegation trees to perform reasoning on sequences of delegated commitments [8]. Similar to explicit operations, Chesani et al. [3] and Dalpiaz et al. [5] formalize commitment delegation and assignment by means of debtor and creditor replacement axioms, respectively. This delegation transfers responsibility. In contrast, the approaches of Chopra and Singh [4] and Yolum and Sing [16] hold the responsibility of the original debtor. These operations, compared to Symboleo's, shift liability and performance altogether and deal only with social norms. Delegation semantics are incomplete since the fulfillment/violation influence of an implicit delegation on the original commitment is not defined.

Legal liability, right, and delegation concepts have been studied through temporal logics. Sartor [12] develops notions of obligative and permissive rights, which express the right of debtors and creditors, respectively, regarding Hohfeldian concepts. These legal positions are manipulated at runtime by means of potestative right and legal power normative operations. Norman and Reed [11] adopt tense logic axiomatization to specify the semantics of responsibility and performance transmission and sharing during obligation delegation. In a similar fashion, these legal notions are formally expressed by a CTL*-based logic [1]. These languages typically specify primary legal norms such as right holder and responsibility delegation, whereas Symboleo considers runtime operations at the level of substitution, assignment, and subcontracting via primary operations.

## 8 Conclusions and Future Work

This paper advances the state-of-the-art by extending Symboleo with execution-time operations supporting dynamic assignment of rights, consensual substitution of a contractual party, and subcontracting of obligations. Primitive operations for the sharing and transfer of right, responsibility, and performance of legal positions enable the support of higher-level operations in specific jurisdictions. Axiomatic semantics were defined and prototyped in a compliance checker, which enabled some initial validation for various scenarios involving a sale-of-good contract and a freight sub-contract. These contributions open the door to powerful and necessary capabilities for monitoring legal contracts.

For future work, we intend to further generalize our language and axioms to support multiple multilateral subcontracts, and to improve Symboleo's syntax to make it more usable by legal experts. We will also make our compliance

checker more general and robust. Moreover, we propose to convert Symboleo specifications to nuXmv [2], to model check the properties on contracts.

**Acknowledgment**. The authors thank E. Jonchères, V. Callipel, D. Restrepo Amariles, P. Bacquero, F. Gélinas, S. Giovanni, T. van Engers, and T. van Binsbergen (lawyers and professors from the Autonomy Through Cyberjustice Technologies project) for their feedback on Symboleo and guidance on subcontracting, as well as A. Roudak for his feedback on our compliance checker.

## References

1. Aldewereld, H., Dignum, V., Vasconcelos, W.W.: Group norms for multi-agent organisations. ACM Transactions on Autonomous and Adaptive Systems (TAAS) **11**(2), 1–31 (2016)
2. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv symbolic model checker. In: CAV 2014. LNCS, vol. 8559, pp. 334–342 (2014)
3. Chesani, F., Mello, P., Montali, M., Torroni, P.: Representing and monitoring social commitments using the event calculus. Autonomous Agents and Multi-Agent Systems **27**(1), 85–130 (2013)
4. Chopra, A.K., Singh, M.P.: Multiagent commitment alignment. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. pp. 937–944. FAAMAS (2009)
5. Dalpiaz, F., Cardoso, E., Canobbio, G., Giorgini, P., Mylopoulos, J.: Social specifications of business processes with Azzurra. In: 9th International Conference on Research Challenges in Information Science (RCIS). pp. 7–18. IEEE CS (2015)
6. Guizzardi, G., Wagner, G., Almeida, J.P.A., Guizzardi, R.S.: Towards ontological foundations for conceptual modeling: the unified foundational ontology (UFO) story. Applied ontology **10**(3-4), 259–271 (2015)
7. Kafalı, Ö., Torroni, P.: Social commitment delegation and monitoring. In: Int. W. on Computational Logic in Multi-Agent Systems. pp. 171–189. Springer (2011)
8. Kafalı, Ö., Torroni, P.: Comodo: collaborative monitoring of commitment delegations. Expert Systems with Applications **105**, 144–158 (2018)
9. Kirby, J.: Assignments and transfers of contractual duties: Integrating theory and practice. Victoria U. Wellington L. Rev. **31**, 317 (2000)
10. Montali, M.: jREC. `https://www.inf.unibz.it/~montali/tools.html` (2016)
11. Norman, T.J., Reed, C.: A logic of delegation. Artificial Intelligence **174**(1), 51–71 (2010)
12. Sartor, G.: Fundamental legal concepts: A formal and teleological characterisation. Artificial Intelligence and Law **14**(1-2), 101–142 (2006)
13. Shanahan, M.: The event calculus explained. In: Artificial intelligence today, pp. 409–430. Springer (1999)
14. Sharifi, S., Parvizimosaed, A., Amyot, D., Logrippo, L., Mylopoulos, J.: Symboleo: A specification language for smart contracts. In: 28th IEEE International Requirements Engineering Conference (RE'20). IEEE CS (2020), *(to appear)*
15. Tam, V.W., Shen, L., Kong, J.S.: Impacts of multi-layer chain subcontracting on project management performance. Int. J. Proj. Manag. **29**(1), 108–116 (2011)
16. Yolum, P., Singh, M.P.: Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. Annals of Mathematics and Artificial Intelligence **42**(1-3), 227–253 (2004)