FCL: A Formal Language for Writing Contracts*

William M. Farmer and Qian Hu

McMaster University, Hamilton, Ontario, Canada
{wmfarmer,huq6}@mcmaster.ca

28 July 2017

Abstract. A contract is an artifact that records an agreement made by the parties of the contract. Although contracts are considered to be legally binding and can be very complex, they are usually expressed in an informal language that does not have a precise semantics. As a result, it is often not clear what a contract is intended to say. This is particularly true for contracts, like financial derivatives, that express agreements that depend on certain things that can be observed over time such as actions taken of the parties, events that happen, and values (like a stock price) that fluctuate with respect to time. As the complexity of the world and human interaction grows, contracts are naturally becoming more complex. Continuing to write complex contracts in natural language is not sustainable if we want the contracts to be understandable and analyzable. A better approach is to write contracts in a formal language with a precise semantics. Contracts expressed in such a language have a mathematically precise meaning and can be manipulated by software. The formal language thus provides a basis for integrating formal methods into contracts. This paper outlines FCL, a formal language with a precise semantics for expressing general contracts that may depend on temporally based conditions. We present the syntax and semantics of FCL and give two detailed examples of contracts expressed in FCL. We also sketch a reasoning system for FCL. We argue that the language is more effective for writing and analyzing contracts than previously proposed formal contract languages.

Keywords: Contracts, formal languages, simple type theory, observables, deontic logic, conditional agreements, temporally based conditions.

1 Introduction

A contract records, orally or in writing, a legally binding agreement between two or more parties [22]. Contracts come in many forms and are used for many

 Published in: S.H. Rubin and T. Bouabana-Tebibel, eds., *Quality Software Through Reuse and Integration, Advances in Intelligent Systems and Computing*, Vol. 561, DOI 10.1007/978-3-319-56157-8_9, Springer, 2018 (forthcoming). This is a revised and extended version of [9]. This research was supported by NSERC.

purposes [6, 22]. Written contracts are artifacts that can be stored, analyzed, modified, and reused. As artifacts, contracts are usually expressed informally in a natural language such as English. Since natural language does not have a precise semantics, it can be difficult to write complex ideas in a natural language in a clear and unambiguous way. Thus contracts that embody complex agreements can be very difficult to both write and understand when natural language is used.

The meaning of a contract — that is, what the agreement is — often depends on certain things that can be observed, called *observables*, such as actions taken by the parties of the contract, events that happen, and values (like a stock price) that fluctuate with respect to time. A contract of this kind is *dynamic*: the contract's meaning changes over the course of time. A dynamic contract contains temporally based conditions that trigger changes to the contract's meaning when the conditions become true. Since the structure of these conditions can be very complex, dynamic contracts can be very difficult to understand and analyze. For example, financial derivatives that *derive* their values from fluctuating underlying assets are dynamic contracts that are notorious for being difficult to value [4].

Contracts — in particular, dynamic contracts — are naturally becoming more complex as the complexity of the world and human interaction grows. Continuing to write complex contracts in natural language is not sustainable if we want the contracts to be understandable and analyzable. A better approach is to write contracts in a formal language with a precise semantics. Then a contract becomes a formal object that has a mathematically precise meaning and that can be manipulated by software. A *formal contract* of this kind can be written, analyzed, and manipulated in various ways with the help of sophisticated software tools.

This paper outlines FCL, a Formal Contract Language with a precise semantics for writing general contracts. In FCL, a contract is a set of components (definitions, agreements, and rules) that can refer to observables and can include conditions that depend on observables. The meanings of these components can change when the values of observables mentioned in them change, and new components can be added when conditions become true. Hence the state of a contract as a set of components can evolve over time in much the same way as the state of a computer program evolves over time.

The paper is organized as follows. Section 2 presents a simple example of a dynamic contract. Section 3 discusses what properties contracts have. An overview of FCL is given in section 4, and the formal semantics of FCL is outlined in section 5. Permissions and reparations are discussed in section 6. Section 7 shows how the example from Section 2 can be expressed in FCL. A more complex example expressed in FCL is given in section 8. A system for reasoning about contracts written in FCL is sketched in section 9. How FCL is related to other formal and informal contract languages is summarized in section 10. And the paper concludes with section 11.

2 Example 1: An American Call Option

To illustrate the role of observables and conditions that depend on them in a dynamic contract, we will consider the following simple example.

Example 1. Consider an American call option for purchasing one share of a certain kind of stock on June 30, 2015 for $5. The expiration date of the option is December 17, 2015 (and so the option may be exercised on any date from June 30, 2015 to December 17, 2015). The strike price of the option is $80. The transaction of the sale of the stock must be finished within 30 days of payment.

An *American option* is a contract that gives the owner the right, but not the obligation, to buy or sell a specified asset at a specified price on or before a specified date [11]. This example describes the conditions that are required for the sale of one share of stock. It shows the role that observables and conditions commonly play in contracts. If a payment of $5 on June 30, 2015 is made to the option seller to buy the option (first condition), the option contract will become effective. If the option buyer exercises the option by paying $80 to the option seller on or before December 17, 2015 (the second condition), the option seller will transfer one share of the stock to the option buyer within 30 days after the option is exercised. The payments of $5 and $80 are both observables on which the first and second conditions respectively depend. The transference of the stock is also an observable.

This American call option can be deconstructed into three components:

1. **Condition 1**: The option buyer gives the option seller $5 on June 30, 2015 to buy an American call option consisting of a conditional agreement composed of the following two components.
2. **Condition 2**: The option buyer chooses to exercise the option by paying $80 to option seller on or before December 17, 2015.
3. **Agreement**: The option seller is obligated to transfer one share of stock to the option buyer no later than 30 days after the option is exercised.

The contract thus has the following form:

if **Condition 1**
then
if **Condition 2**
then **Agreement**

Both if-then parts of the contract are conditional agreements. The second conditional agreement consists of **Condition 2** and **Agreement**, and the first conditional consists of **Condition 1** and the second conditional agreement.

The *offer time* of the American call option is the time the option contract is offered by the option seller to possible buyers. At the offer time, the option contract is not a legally binding agreement. It becomes a legally binding agreement

only when the option contract is purchased by the option buyer (i.e., the time when **Condition 1** is satisfied).

A conditional agreement, like either of the two in this example, can be viewed as a "rule" that generates an agreement depending on the values of certain observables. In general, different agreements are generated when observables have different values. Observables determine both the meaning of a contract and how the meaning of the contract evolves over time.

3 What is a Contract?

Before we present FCL, our formal language for writing contracts, we need to discuss what a contract is. A contract is an artifact with certain properties. There is not a clear consensus of which of these properties are necessary and which are optional. We favor the definition of a contract given by Brian Blum in [6, p. 2]. He says a contract must have each of the following properties:

- Is an oral or written agreement.
- Involves at least two parties.
- Includes at least one promise made by the parties.
- Establishes an exchange relationship between the parties.
- Is legally enforceable.

A contract is created only because the parties reach agreement on the terms of the contract. The *parties* are the people or entities that have mutually agreed to the contract and are bound by its terms and conditions. In the case of written agreements, the parties are typically identified as the people or entities that signed the agreement. For any contract to be valid, there must be at least two parties. Typically, one party makes an offer and the other party accepts it. In addition, to be valid a contract must involve the parties in an *exchange* of something of value such as services, goods, or a promise to perform some action. Note that the exchange of money is not necessary.

A contract involves a *promise* which Blum defines as an "undertaking to act or refrain from acting in a specified way at some future time" [6, p. 5]. We think of "undertaking to act" as the deontic notion of *obligation*. Similarly, we understand "refrain from acting" as the deontic notion of *prohibition*. Obligation and prohibition are concepts studied in deontic logic [17]. They have the distinctive characteristic of being violable. When a promise made in a contract is honored, we say the promise has been *satisfied*. If it has not been honored, we say it has been *violated*. A promise may be restricted by a temporal bound, that is, a period of time during which an obligation or prohibition is in force. For example, a tenant may be obliged to pay rent on the first day of each month.

We will use an expanded definition of a contract that includes "degenerate contracts" that would not be considered contracts according to Blum's definition but are convenient to include in the space of all possible contracts. For example, a contract is *void* if it violates the law [4]. Void contracts are not legally enforceable agreements, so by Blum's definition they are not genuine contracts. We will

consider them to be contracts, but we will designate them as being degenerate. Similarly, we will consider an agreement between two parties that does not include a promise or establish an exchange relationship between the parties as a degenerate contract.

4 Overview of FCL

This section describes the main components of FCL and informally explains their purpose and meaning. The formal semantics of FCL is outlined in section 5.

4.1 Underlying Logic

We will assume that the underlying logic of FCL is some version of simple type theory [8]. Simple type theory is a form of high-order logic with function types, quantification over functions, and function abstraction. The underlying logic must have the following base types:

1. Bool, a type consisting of the boolean values T (true) and F (false).
2. Time, a type consisting of the integers $\mathbb{Z}$. That is, we assume that time is represented as a discrete linearly ordered set of values such that each value has a predecessor and a successor. The values many denote any convenient measure of time such as days, hours, seconds, etc.
3. Event, a type of events. These can be actions performed by the parties of a contract as well as events that the parties have no control over.

The underlying logic must include the following constants:

1. true and false of type Bool.
2. obs-event of type $\mathsf{Time} \times \mathsf{Event} \to \mathsf{Bool}$.

true and false represent the truth values T and F, respectively. obs-event is used to express observations of events as described in section 4.2. The underlying logic must also have the variable X_{time} of type Time. X_{time} is used to instantiate expressions with the current time of a contract.

An *expression* of FCL is any expression in the underlying logic of FCL. A *formula* of FCL is an expression of FCL of type Bool.

Building FCL on simple type theory gives FCL access to the high expressivity and reasoning power of simple type theory [8]. This means that FCL can be developed largely by utilizing the standard machinery of simple type theory without the need to develop new logical ideas.

4.2 Observables

An *observable* is something that has a variable value that can be observed at a particular time [20, 21]. Let us look at a couple of examples. The temperature of a room is an observable. Its value at a given time t is the temperature measured

in the room at t. An event is an observable whose value is either true or false. Its value at a given time t is true [false] if the event occurs [does not occur] at t.

An *observable* of FCL is the application of a constant f of type

$$\mathsf{Time} \times \alpha_1 \times \cdots \times \alpha_n \to \beta$$

where $n \geq 0$. Thus the value of the observable $f(t, a_1, \ldots, a_n)$ depends on time in the sense that it depends on the value of its first argument which is of type Time. The value of $f(t, a_1, \ldots, a_n)$ also depends on the parameters $a_1, \ldots, a_n$. An *observation* of FCL is an atomic formula of the form $o = v$ where o is an observable $f(t, a_1, \ldots, a_n)$ and v is a value in the output type of f. When the output type of f is Bool, $o = \mathsf{true}$ and $o = \mathsf{false}$ can be written as o and $\neg o$, respectively. An *observational statement* of FCL is a formula of the underlying logic of FCL constructed from observations using the machinery of the underlying logic — which includes propositional connectives, quantifiers, and the other usual machinery of simple type theory.

We will show how the two examples of observables mentioned above can be expressed in FCL. Let obs-temp be a constant of type $\mathsf{Time} \to \mathbb{Z}$. Then $\mathsf{obs\text{-}temp}(t) = a$ represents the observation that the temperature in a particular room is a at time t. $\mathsf{obs\text{-}event}(t, e)$ represents the observation that the event e occurs at time t.

4.3 Actions

An *action* is an event that can be performed by the parties of a contract. There are two sorts of entities involved in an action: *subjects* and *objects*. The former are the entities who perform the action, while the latter are the entities that are acted upon by the subjects. An *action* a of type Event is defined as a tuple of the form $(L, \alpha, \mathcal{S}, \mathcal{O})$ where L is the *label* of an action, α is the *act* of the action (i.e., the thing that is performed), $\mathcal{S}$ is the set of *subjects* of the action, and $\mathcal{O}$ is the set of *objects* of the action.

Contracts typically include actions that specify the transfer of resources (money, goods, services, and even pieces of information) between parties. The act of the action would be the transfer of resources from one party (the subject) to another party (the object). Notice that an action of this kind encodes both what is transferred and what parties are involved in the transference.

4.4 Constant Definitions

A *constant definition* of FCL is an expression of the form $c = e$ where c is a new constant or an application of a new constant and e is an expression that defines the value of c. Constant definitions are used, among other things, to define temporally based values.

4.5 Agreements

An *agreement* is a promise to do or not do a specific action. An *agreement* of FCL is an expression of either the form $\mathbb{O}(a, \mathcal{T})$ or the form $\mathbb{F}(a, \mathcal{T})$ where a is an action and $\mathcal{T}$ is a set of times. $\mathbb{O}(a, \mathcal{T})$ is called an *obligation*; it represents the promise that the action a will be observed at some time in $\mathcal{T}$. $\mathbb{F}(a, \mathcal{T})$ is called a *prohibition*; it represents the promise that the action a will not be observed at any time in $\mathcal{T}$. $\mathbb{O}(a, \mathcal{T})$ and $\mathbb{F}(a, \mathcal{T})$ are considered to be *duals* of each other. The operators $\mathbb{O}$ and $\mathbb{F}$ are inspired by the deontic operators for *obligation* and *prohibition* [17].

4.6 Rules

A *rule* R of FCL is inductively defined as an expression of the form

$$\varphi \mapsto \mathcal{B}$$

where φ is a formula of FCL and $\mathcal{B}$ is a set of constant definitions, agreements, and rules. We assume that each free variable occurring in a constant definition or an agreement in $\mathcal{B}$ also occurs in φ. We will see in section 5 that, if φ is satisfied at time t, the members of $\mathcal{B}$ are added to the state of a contract at time $t + 1$. Thus a rule can dynamically change the meaning of a contract.

A rule of the form $\varphi \mapsto \{A\}$, where A is an agreement, represents a *conditional agreement*.

4.7 Contracts

A *contract* C of FCL is a pair $(t_{\text{offer}}, \mathcal{B})$ where t_{offer} is a time and $\mathcal{B}$ is a set of closed constant definitions, closed agreements, and rules. The *parties* of C are the parties mentioned in the agreements in $\mathcal{B}$. t_{offer} is the time the contract is offered to the parties.

As we will see in the next section, a contract has a state consisting of a set of constant definitions, agreements, and rules. The state evolves over time like the state of a program evolves over time. A contract is *fulfilled* when all the agreements in its state are satisfied and all the rules in its state are no longer applicable. A contract is *breached* when some agreement in its state is violated.

5 Formal Semantics of FCL

This section presents the formal semantics of FCL.

5.1 Models

A *model* of FCL is a model of STT. Throughout this section let $\mathcal{M}$ be a model of FCL. Let $V^{\mathcal{M}}$ be the valuation function of $\mathcal{M}$ that assigns each (closed) expression of FCL a value in $\mathcal{M}$. In particular, $V^{\mathcal{M}}$ assigns each observable $f(t, a_1, \ldots, a_n)$ a value for all times t (and parameters $a_1, \ldots, a_n$). Thus a model includes the values for all observables over all time.

5.2 Agreements

Let $t \in \mathbb{Z}$ and $\bar{t}$ be some canonical expression whose value is t. The *value of an obligation* $\mathbb{O}(a, \mathcal{T})$ *in* $\mathcal{M}$ *at time* t is $V^{\mathcal{M}}(\varphi)$ where φ is the formula

$$\exists\, u : \mathsf{Time} \,.\, u \in \mathcal{T} \wedge u \leq \bar{t} \wedge \mathsf{obs\text{-}event}(u, a).$$

The *value of a prohibition* $\mathbb{F}(a, \mathcal{T})$ *in* $\mathcal{M}$ *at time* t is $V^{\mathcal{M}}(\psi)$ where ψ is the formula

$$\forall\, u : \mathsf{Time} \,.\, u \in \mathcal{T} \supset (u \leq \bar{t} \wedge \neg \mathsf{obs\text{-}event}(u, a)).$$

An agreement is *satisfied* in $\mathcal{M}$ at time t if its value in $\mathcal{M}$ at t is T. An agreement is *violated* in $\mathcal{M}$ at time t if the value of its dual in $\mathcal{M}$ at t is T. We will occasionally use an agreement $\mathbb{O}(a, \mathcal{T})$ or $\mathbb{F}(a, \mathcal{T})$ as a formula of FCL whose meaning is φ or ψ, respectively.

5.3 Rules

Let $R = \varphi \mapsto \mathcal{B}$ be a rule of FCL and $t \in \mathbb{Z}$. Define $\mathsf{sub}(\varphi, t)$ to be the set of substitutions σ that map the free variables in φ to appropriate expressions such that $\sigma(X_{\text{time}}) = \bar{t}$. The variable X_{time} is used to instantiate a rule with the current time of a contract. For any expression e and substitution $\sigma \in \mathsf{sub}(\varphi, t)$, let $e\sigma$ be the result of applying σ to each free variable in e if e is not a rule and to each free variable in e except X_{time} if e is a rule. Then define $\mathsf{new\text{-}items}(R, \mathcal{M}, t)$ to be

$$\{e\sigma \mid \sigma \in \mathsf{sub}(\varphi, t) \wedge V^{\mathcal{M}}(\varphi\sigma) = \mathrm{T} \wedge e \in \mathcal{B}\}.$$

R is *active* in $\mathcal{M}$ at t if $V^{\mathcal{M}}(\varphi\sigma) = \mathrm{T}$ for some $\sigma \in \mathsf{sub}(\varphi, t)$. R is *defunct* in $\mathcal{M}$ at t if $V^{\mathcal{M}}(\varphi\sigma) = \mathrm{F}$ for all $u \geq t$ and all $\sigma \in \mathsf{sub}(\varphi, u)$. If R is defunct in $\mathcal{M}$ at t, then R is not active in $\mathcal{M}$ at u and $\mathsf{new\text{-}items}(R, \mathcal{M}, u) = \emptyset$ for all $u \geq t$.

5.4 Contracts

Let $C = (t_{\text{offer}}, \mathcal{B})$ be a contract. The *state* of C in $\mathcal{M}$ at time $t \geq t_{\text{offer}}$, written $\mathsf{state}(C, \mathcal{M}, t)$, is the set of constant definitions, agreements, and rules defined inductively as follows:

1. $\mathsf{state}(C, \mathcal{M}, t_{\text{offer}}) = \mathcal{B}$.
2. If $t \geq t_{\text{offer}}$, then $\mathsf{state}(C, \mathcal{M}, t+1) =$

$$(\mathsf{state}(C, \mathcal{M}, t)) \cup \bigcup_{R \in \mathcal{B}} \mathsf{new\text{-}items}(R, \mathcal{M}', t)$$

where $\mathcal{M}'$ is the smallest expansion of $\mathcal{M}$ such that $V^{\mathcal{M}}(\psi) = \mathrm{T}$ for each constant definition $\psi \in \mathsf{state}(C, \mathcal{M}, t)$.

The model $\mathcal{M}'$ in clause 2 is called the *C-expansion of $\mathcal{M}$ at time t*. A *model of C at time t* is any C-expansion of a model of FCL at time t.

C is *fulfilled* in $\mathcal{M}$ at time $t \geq t_{\text{offer}}$ if every agreement in $\mathsf{state}(C, \mathcal{M}, t)$ is satisfied in the C-expansion of $\mathcal{M}$ at t and every rule in $\mathsf{state}(C, \mathcal{M}, t)$ is defunct in the C-expansion of $\mathcal{M}$ at t. C is *breached* in $\mathcal{M}$ at time $t \geq t_{\text{offer}}$ if there is an agreement in $\mathsf{state}(C, \mathcal{M}, t)$ that is violated in the C-expansion of $\mathcal{M}$ at t. C is *null* in $\mathcal{M}$ at time $t \geq t_{\text{offer}}$ if $\mathsf{state}(C, \mathcal{M}, t)$ contains no agreements and every rule in $\mathsf{state}(C, \mathcal{M}, t)$ is defunct in the C-expansion of $\mathcal{M}$ at t.

Notice that we are employing a very simple model of concurrency in our semantics for contracts: At each time t, all active rules are applied simultaneously and the resulting new components – constant definitions, agreements, and rules — are added to the state of the contract at the next point in time. There is no opportunity for the application of rules to interfere with each other. In particular, a component can never be removed from the state once it is added to it. It is possible, however, for a contract state to be produced that contains contradictions, but this would be caused by a flaw in the contract, not a flaw in the conceptual framework.

6 Two Additional Concepts

This section explains how permissions and reparations can be expressed in FCL.

6.1 Permissions

Agreements of the form $\mathbb{O}(a, \mathcal{T})$ and $\mathbb{F}(a, \mathcal{T})$ are used in FCL to represent promises in the form of obligations and prohibitions. Notice that agreements in FCL do not include expressions formed using an operator corresponding to the deontic operator for *permission*. Unlike an obligation or a prohibition, a permission is not a promise.

Some kinds of permissions can be expressed in FCL. For example, the permission to exercise a call option is expressed by adding a rule $\varphi \mapsto \mathcal{B}$ to the contract's state where the condition φ holds if it is observed that the buyer of the option exercises the option and $\mathcal{B}$ includes an obligation that the seller of option sells to the buyer the goods specified by the option. See Section 7 for details.

6.2 Reparations

A contract usually specifies actions to be taken in case of the violation of a part of the contract. A conditional obligation arising in response to a violated agreement is considered as a *reparational agreement*. We extend the example of a sale of a laser printer contract from [15, p. 5] to explain how a violation that arises in contract can be "repaired".

Example 2. The contract consists of five clauses:

1. Seller agrees to transfer and deliver to Buyer one laser printer within 22 days after an order is made.
2. Buyer agrees to accept the goods and to pay a total of $200 for them according to the terms further set out below.
3. Buyer agrees to pay for the goods half upon receipt, with the remainder due within 30 days of delivery.
4. If Buyer fails to pay the second half within 30 days, an additional fine of 10% has to be paid within 14 days.
5. Upon receipt, Buyer has 14 days to return the goods to Seller in original, unopened packaging. Within 7 days thereafter, Seller has to repay the total amount to Buyer.

Note that clause 3 of this example is a primary obligation, saying that the buyer is obligated to pay the second half within 30 days of delivery. Clause 4 of is an example of *reparational obligation* in which an unfulfilled obligation can generate obligations to "repair" this violation. It says what the buyer is obligated to do if he or she violates the primary obligation.

Similar to the reparational obligation, a *reparational prohibition* is a conditional agreement arising in response to a violated prohibition. Both reparational obligations and reparational prohibitions are reparational agreements. In FCL, a reparational agreement is expressed as a rule that creates other agreements or rules in response to the violation of the primary agreement. To express the potential violations, we introduce $\mathsf{obs\text{-}event\text{-}during}(t, \mathcal{T}, e)$ to represent the observation at time t that the event e occurred during the time period $\mathcal{T}$.

In FCL, a reparational obligation of $\mathbb{O}(a, \mathcal{T})$ is expressed as a rule of the form

$$\neg\mathsf{obs\text{-}event\text{-}during}(X_{\text{time}}, \mathcal{T}, a) \mapsto \mathcal{B}.$$

$\neg\mathsf{obs\text{-}event\text{-}during}(X_{\text{time}}, \mathcal{T}, a)$ represents a potential violation of agreement $\mathbb{O}(a, \mathcal{T})$. If an obligation is satisfied, the rule to "repair" this obligation will never be active. Similarly, a reparational prohibition of $\mathbb{F}(a, \mathcal{T})$ is expressed as a rule of the form

$$\mathsf{obs\text{-}event\text{-}during}(X_{\text{time}}, \mathcal{T}, a) \mapsto \mathcal{B}.$$

A rule that represents a reparational prohibition of $\mathbb{F}(a, \mathcal{T})$ will always be defunct if the agreement $\mathbb{F}(a, \mathcal{T})$ is satisfied.

Consider clause 4 of Example 2, the reparational obligation of the primary obligation given in clause 3 is the conditional agreement that, if the second half of payment has not been observed within 30 days of delivery, then the buyer has to pay an additional fine of 10% within 14 days. How this conditional agreement is expressed in FCL is shown in section 8.

7 Example 1 Formalized: An American Call Option

We formalize here the American Call Option introduced in Section 2 as a contract C of FCL. C has two parties: a seller and a buyer. The unit of time is one

day. Let the offer time t_{offer} of the contract, the time the seller offered the contract to the buyer, be some day before June 30, 2015. C is defined as the pair $(t_{\text{offer}}, \{D_1, D_2, R_1\})$ where:

$D_1 : t_{\text{buy}} = 0$ (June 30, 2015).
$D_2 : t_{\text{expire}} = 170$ (December 17, 2015).
R_1 is defined below.

C is constructed from two rules R_1 and R_2:

1. *Rule for Buying the Option:*
   $R_1 = \varphi_1 \mapsto \{R_2\}$ where:
   $\varphi_1 = \mathsf{obs\text{-}event}(t_{\text{buy}}, e_1)$.
   $e_1 = (\text{“Buy Option”}, \mathsf{transfer}(\$5), \{\mathsf{buyer}\}, \{\mathsf{seller}\})$.
   R_2 is defined below.
2. *Rule for Exercising the Option:*
   $R_2 = \varphi_2 \mapsto \{D_3, A\}$ where:
   $\varphi_2 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_2) \wedge t_{\text{buy}} \leq X_{\text{time}} \leq t_{\text{expire}}$.
   $e_2 = (\text{“Exercise Option”}, \mathsf{transfer}(\$80), \{\mathsf{buyer}\}, \{\mathsf{seller}\})$.
   $D_3 : t_{\text{exercise}} = X_{\text{time}}$.
   $A = \mathbb{O}(e_3, [t_{\text{exercise}}, t_{\text{exercise}} + 30])$.
   $e_3 = (\text{“Transfer Stock”}, \mathsf{transfer}(\mathsf{stock}), \{\mathsf{seller}\}, \{\mathsf{buyer}\})$.

t_{buy}, t_{expire}, and t_{exercise} are new constants of type Time. $[t_{\text{exercise}}, t_{\text{exercise}} + 30]$ is the interval representing the set of times $\{t_{\text{exercise}}, t_{\text{exercise}} + 1, \ldots, t_{\text{exercise}} + 30\}$.

Each of the three events e_1, e_2, and e_3 are actions by one of the two parties. The three events are tied to the contract. For example, a more exact name for “Buy Option” would be “Buy Option Described by Contract C”. We assume that each of the three events can happen as most once. φ_1 asserts the option is bought on June 30, 2015, and φ_2 asserts the option is exercised at a time after the option is bought and before the option expires.

The *state* of C in a model $\mathcal{M}$ at time $t \geq t_{\text{offer}}$, written as $\mathsf{state}(C, \mathcal{M}, t)$, evolves over time as indicated in Figure 1. How the state of C evolves depends on the observables specified by $\mathcal{M}$. In the figure, let u be the time that R_2 becomes active, i.e, when the buyer exercises the option. Let σ be the substitution that maps X_{time} to $\overline{u}$. Applying σ has the effect of replacing X_{time} with $\overline{u}$, whose value is the time u. $D_3\sigma$ is thus the equation $t_{\text{exercise}} = \overline{u}$, but $A\sigma$ is A since X_{time} does not occur in A.

8 Example 2 Formalized: a Sale of Goods Contract

We previously saw an encoding of the American Call Option in FCL. In this section we formalize the sale of the printer contract introduced in Section 6.2. Although this example contract is very simple, two points should be noticed. First, as illustrated in Section 6.2, this contract includes a reparational agreement that can used to repair a potential violation. Second, consider the total amount

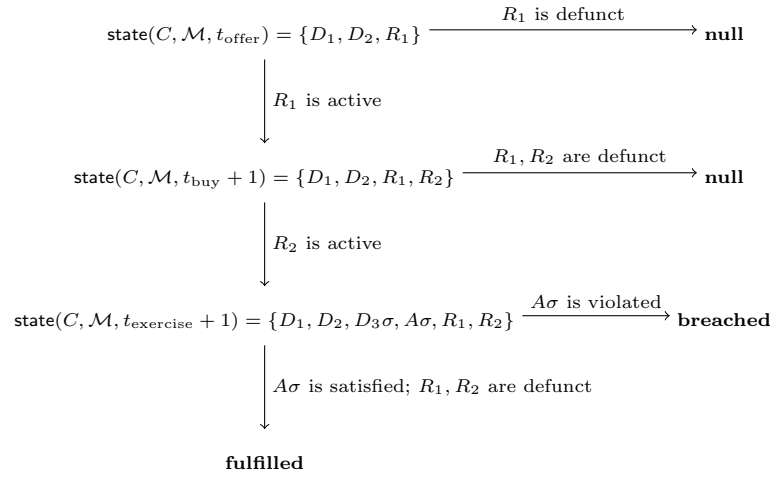

Fig. 1. Execution of the American Call Option C

specified in clause 5. Taken literally, it would imply that the total amount the seller must repay to buyer in case of a return of the printer should be $200 as stated in clause 2. Actually, this is certainly not the seller's intention. In fact, the total amount to be repaid should be the amount that the buyer has already paid the seller (which may not be the full $200).

Now we formalize this contract in FCL to explain how we deal with the problems mentioned above. Let C be this contract expressed in FCL. C has two parties: a seller and a buyer. The unit of time is one day. C is defined as the pair $(t_{\text{offer}}, \{R_1\})$ where t_{offer} is the time when the seller offered the contract to the buyer.

C is constructed from the following nine rules:

1. *Rule for Ordering a Printer:*
   $R_1 = \varphi_1 \mapsto \{D_1, A_1, R_2\}$.
   $\varphi_1 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_1) \wedge t_{\text{offer}} \leq X_{\text{time}}$.
   $e_1 = (\text{“Order Printer”}, \mathsf{transfer}(\mathsf{order}), \{\mathsf{buyer}\}, \{\mathsf{seller}\})$.
   $D_1 : t_{\text{order}} = X_{\text{time}}$.
   $A_1 = \mathbb{O}(e_2, [t_{\text{order}}, t_{\text{order}} + 22])$.
   $e_2 = (\text{“Deliver Printer”}, \mathsf{transfer}(\mathsf{printer}), \{\mathsf{seller}\}, \{\mathsf{buyer}\})$.
   R_2 is defined below.
2. *Rule for Delivering the Printer:*
   $R_2 = \varphi_2 \mapsto \{D_2, D_3, R_3, R_4, R_5, R_6\}$.
   $\varphi_2 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_2) \wedge t_{\text{order}} \leq X_{\text{time}} \leq t_{\text{order}} + 22$.
   $D_2 : t_{\text{deliver}} = X_{\text{time}}$.
   $D_3 : \mathsf{total\text{-}payment}(X_{\text{time}}) = 0$.
   R_3, R_4, R_5 and R_6 are defined below.

3. *Rule for Returning the Printer:*
   $R_3 = \varphi_3 \mapsto \{D_4, A_2\}$.
   $\varphi_3 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_3) \wedge t_{\text{deliver}} \leq X_{\text{time}} \leq t_{\text{deliver}} + 14$.
   $e_3 = (\text{“Return Printer”}, \mathsf{transfer}(\mathsf{printer}), \{\mathsf{buyer}\}, \{\mathsf{seller}\})$.
   $D_4 : t_{\text{return}} = X_{\text{time}}$.
   $A_2 = \mathbb{O}(e_4(\mathsf{total\text{-}payment}(X_{\text{time}})), [t_{\text{return}}, t_{\text{return}} + 7])$.
   $e_4 = \lambda X_{\text{total}}.(\text{“Return Payment”}, \mathsf{transfer}(X_{\text{total}}), \{\mathsf{seller}\}, \{\mathsf{buyer}\})$.
4. *Rules for Recording the Payment:*
   $R_4 = \varphi_4 \wedge \neg\varphi_5 \mapsto \{D_5\}$.
   $R_5 = \neg\varphi_4 \wedge \neg\varphi_5 \mapsto \{D_6\}$.
   $\varphi_4 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_5(X_{\text{payment}})) \wedge t_{\text{deliver}} \leq X_{\text{time}}$.
   $e_5 = \lambda X_{\text{payment}}.(\text{“Pay Seller”}, \mathsf{transfer}(X_{\text{payment}}), \{\mathsf{buyer}\}, \{\mathsf{seller}\})$.
   $\varphi_5 = \mathsf{obs\text{-}event\text{-}before}(X_{\text{time}}, X_{\text{time}}, e_3)$.
   $D_5 : \mathsf{total\text{-}payment}(X_{\text{time}}) = \mathsf{total\text{-}payment}(X_{\text{time}} - 1) + X_{\text{payment}}$.
   $D_6 : \mathsf{total\text{-}payment}(X_{\text{time}}) = \mathsf{total\text{-}payment}(X_{\text{time}} - 1)$.
5. *Rules for Making Payments:*
   $R_6 = \neg\varphi_6 \mapsto \{A_3, R_7\}$.
   $\varphi_6 = \mathsf{obs\text{-}event\text{-}before}(X_{\text{time}}, t_{\text{deliver}} + 1, e_3)$.
   $A_3 = \mathbb{O}(e_5(200/2), [t_{\text{deliver}}, t_{\text{deliver}} + 1])$.
   $R_7 = \varphi_7 \wedge \neg\varphi_5 \mapsto \{D_7, R_8\}$.
   $\varphi_7 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_5(200/2)) \wedge t_{\text{deliver}} \leq X_{\text{time}} \leq t_{\text{deliver}} + 1$.
   $D_7 : t_{\text{first}} = X_{\text{time}}$.
   $R_8 = \neg\varphi_8 \wedge \neg\varphi_5 \mapsto \{R_9\}$.
   $\varphi_8 = \mathsf{obs\text{-}event\text{-}during}(X_{\text{time}}, [t_{\text{first}} + 1, t_{\text{deliver}} + 14], e_5(200/2))$.
   R_9 is defined below.
6. *Rules for Paying Fine for a Late Payment:*
   $R_9 = \neg\varphi_9 \mapsto \{A_4, A_5\}$.
   $\varphi_9 = \mathsf{obs\text{-}event\text{-}during}(X_{\text{time}}, [t_{\text{deliver}} + 15, t_{\text{deliver}} + 30], e_5(200/2))$.
   $A_4 = \mathbb{O}(e_5(200/2), [t_{\text{deliver}} + 31, t_{\text{deliver}} + 44])$.
   $A_5 = \mathbb{O}(e_5(10\% * 200/2), [t_{\text{deliver}} + 31, t_{\text{deliver}} + 44])$.

t_{order}, t_{deliver}, t_{return}, and t_{first} are new constants of type Time. Each of the five events e_1, e_2, e_3, e_4, and e_5 are actions by one of the two parties. We assume that the events e_1, e_2, e_3, e_4 can happen at most once and the “Pay Seller” event e_5 can happen at most twice.

$\mathsf{total\text{-}payment}(t)$ represents the total amount that the buyer has been observed to have paid the seller at time t. When rule R_4 is active, D_5 is generated. D_5 is used to add a payment to the total amount paid at the previous time point. D_5 and D_6 work together to record the happenings of the “Pay Seller” event e_5 in the timeline. $\mathsf{obs\text{-}event\text{-}before}(t, t', e)$, a constant of type $(\mathsf{Time} \times \mathsf{Time} \times \mathsf{Event}) \to \mathsf{Bool}$, represents the observation at time t that the event e occurred on or before the time t'.

We identify that the buyer has the following options to choose from after he has accepted the printer and made the first payment:

1. Buyer makes a return within 14 days after the delivery is made.

2. Buyer makes the second payment within 14 days after the delivery made.
3. Buyer makes the second payment between 15 to 30 days after the delivery made.
4. Buyer makes the second payment with an additional fine between 31 to 44 days after the delivery made.

R_8 and R_9 work together as a reparation if the second payment is not made on time. Within 14 days the buyer has the first and second options to choose from. R_8 says if the first two options have not been chosen, then between 15 to 44 days the buyer is obligated to make the second payment. If it is paid late, which means R_9 is active, then an additional fine must be paid.

9 A Reasoning System

In this section we will sketch a reasoning system for FCL which is an extension of a proof system for simple type theory. Let $\bar{t}$ be an expression of type Time whose value is the time t, φ be a formula, Γ be a set of formulas, A be an agreement, R be a rule, and C be a contract. For a model $\mathcal{M}$ of FCL, $\mathcal{M} \models \varphi$ means $V^{\mathcal{M}}(\varphi) = \mathrm{T}$ and $\mathcal{M} \models \Gamma$ means $\mathcal{M} \models \psi$ for all $\psi \in \Gamma$.

A reasoning system for simple type theory (and other traditional logics) has a judgment of the form $\Gamma \vdash \varphi$ that asserts φ logically follows from Γ, i.e, $\mathcal{M} \models \Gamma$ implies $\mathcal{M} \models \varphi$ for all models $\mathcal{M}$ of FCL. Since constant definitions, agreements, and rules are not expressions of simple type theory, we need the following additional judgments in a reasoning system for FCL:

1. $\Gamma \vdash_{C,\bar{t}} \varphi$ asserts that φ logically follows from Γ and C at t, i.e., $\mathcal{M} \models \Gamma$ implies $\mathcal{M} \models \varphi$ for all models $\mathcal{M}$ of C at t.
2. $\mathsf{Agreement}[\Gamma, A, C, \bar{t}]$ asserts that A is in the state of C at t with respect to Γ, i.e., $\mathcal{M} \models \Gamma$ implies $A \in \mathsf{state}(C, \mathcal{M}, t)$ for all models $\mathcal{M}$ of FCL.
3. $\mathsf{Rule}[\Gamma, R, C, \bar{t}]$ asserts that R is in the state of the C at t with respect to Γ, i.e., $\mathcal{M} \models \Gamma$ implies $R \in \mathsf{state}(C, \mathcal{M}, t)$ for all models $\mathcal{M}$ of FCL.
4. $\mathsf{Satisfied}[\Gamma, A, C, \bar{t}]$ asserts that A is satisfied at t with respect to Γ, i.e., $\mathcal{M} \models \Gamma$ implies that A is satisfied in $\mathcal{M}$ at t for all models $\mathcal{M}$ of C at t.
5. $\mathsf{Violated}[\Gamma, A, C, \bar{t}]$ asserts that A is violated at t with respect to Γ, i.e., $\mathcal{M} \models \Gamma$ implies A is violated in $\mathcal{M}$ at t for all models $\mathcal{M}$ of C at t.
6. $\mathsf{Defunct}[\Gamma, R, C, \bar{t}]$ asserts that R is defunct at t with respect to Γ, i.e., $\mathcal{M} \models \Gamma$ implies R is defunct in $\mathcal{M}$ at t for all models $\mathcal{M}$ of C at t.
7. $\mathsf{Fulfilled}[\Gamma, C, \bar{t}]$ asserts that C is fulfilled at t with respect to Γ, i.e., $\mathcal{M} \models \Gamma$ implies C is fulfilled in $\mathcal{M}$ at t for all models $\mathcal{M}$ of C at t.
8. $\mathsf{Breached}[\Gamma, C, \bar{t}]$ asserts that C is breached at t with respect to Γ, i.e., $\mathcal{M} \models \Gamma$ implies C is breached in $\mathcal{M}$ at t for all models $\mathcal{M}$ of C at t.

The role of Γ is to specify the models in which C will be considered.

The reasoning system has several rules of inference including the usual rules of inference for simple type theory. There is a rule of inference that shows $\Gamma \vdash_{C,\bar{t}} \varphi$ extends $\Gamma \vdash \varphi$:

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash_{C,\bar{t}} \varphi}.$$

The rule of inference says that if φ follows from Γ, then φ follows from Γ and C at t for any contract C and time t.

The following three rules of inference show how a rule changes the state of a contract:

$$\frac{\mathsf{Rule}[\Gamma, R, C, \bar{t}], \Gamma \vdash_{C,\bar{t}} \varphi\sigma}{\Gamma \vdash_{C,\bar{t}+1} \psi_1\sigma, \ldots, \Gamma \vdash_{C,\bar{t}+1} \psi_k\sigma}$$

$$\frac{\mathsf{Rule}[\Gamma, R, C, \bar{t}], \Gamma \vdash_{C,\bar{t}} \varphi\sigma}{\mathsf{Agreement}[\Gamma, A_1\sigma, C, \bar{t}+1], \ldots, \mathsf{Agreement}[\Gamma, A_m\sigma, C, \bar{t}+1]}$$

$$\frac{\mathsf{Rule}[\Gamma, R, C, \bar{t}], \Gamma \vdash_{C,\bar{t}} \varphi\sigma}{\mathsf{Rule}[\Gamma, R_1\sigma, C, \bar{t}+1], \ldots, \mathsf{Rule}[\Gamma, R_n\sigma, C, \bar{t}+1]}$$

where

$$R = \varphi \mapsto \{\psi_1, \ldots, \psi_k, A_1, \ldots, A_m, R_1, \ldots, R_n\},$$

$\psi_1, \ldots, \psi_k$ are constant definitions, $A_1, \ldots, A_m$ are agreements, $R_1, \ldots, R_n$ are rules, and $\sigma \in \mathsf{sub}(\varphi, t)$

There is a rule of inference for satisfied agreements:

$$\frac{\mathsf{Agreement}[\Gamma, A, C, \bar{t}], \Gamma \vdash_{C,\bar{t}} A}{\mathsf{Satisfied}[\Gamma, A, C, \bar{t}]}.$$

There are similar rules of inference for violated agreements, defunct rules, fulfilled contracts, and breached contracts.

A reasoning system of this kind can be used to both prove statements about a contract and to simulate the unfolding of a contract over time. The latter is done by using Γ to specify the observations that are expected over the course of the contract. The reasoning system can be strengthened by introducing temporal operators that enable one to say, for example, that it follows from Γ that C will be *eventually* be fulfilled.

10 Related Work

Several formal languages for writing contracts have been proposed. Our language FCL is most closely related to the following work:

- S. L. Peyton Jones and J. M. Eber (J&E) [20, 21].
- A. Goodchild, C. Herring, and Z. Milosevic (GHM) [12].
- G. Governatori and Z. Milosevic (G&M) [13, 14, 16].

- J. Andersen, E. Elsborg, F. Henglein, J. G. Simonsen, and C. Stefansen (AEHSS) [1].
- C. Prisacariu and G. Schneider (P&S) [10, 23–26].
- P. Bahr, J. Berthold, M. Elsman (BBE) [5].
- LegalRuleML Technical Committee (TC) [2, 3].

The domains of these approaches are varied: J&E's and BBE's works are restricted to financial contracts; GHM builds a domain-specific language for business contracts; AEHSS is concerned with formalizing commercial contracts; and the LegalRuleML TC focuses on the creation of machine-readable forms of the content of legal texts, such as legislation, regulations, contracts, and case law, for different concrete Web applications. Same as P&S's work, our proposed language FCL considers the formalization of general contracts that are agreements written by and for humans.

Several techniques are employed in the literature for developing a precise formal language for specifying contracts. Most of the techniques, such as those given in [12, 13, 16], belong to the event-condition-action (ECA) based scheme. GHM and G&M model contracts as sets of policies. A policy specifies that a legal entity is either forbidden or obliged to perform an action under certain event-based conditions. AEHSS provide an action-trace based language [1] to model contracts. J&E's functional programming based language [20, 21] and BBE's cash-flow trace based approach [5] use the idea of observables to specify events. P&S introduce in [23–26] a contract language $\mathcal{CL}$ for expressing electronic contracts based on a combination of concepts from deontic, dynamic, and temporal logic. $\mathcal{CL}$ restricts deontic modalities to ought-to-do statements and adds the modalities of dynamic logic to be able to reason about what happens after an action is performed. Rather than providing a logical language for contracts, the LegalRuleML TC extends RuleML to provide a rule interchange language with formal features specific for the legal domain. This enables implementers to structure the contents of the legal texts in a machine-readable format by using the representation tools. Motivated by the ECA-based formalisms and the idea of observables, we introduce in FCL the concept of a *rule* that is (in its simplest form) a conditional agreement that depends on certain observations. The use of observables to determine both the meaning of a contract and how the meaning of the contract evolves over time provides a basis for monitoring the dynamic aspects of a contract.

Only P&S's $\mathcal{CL}$ language and LegalRuleML can specify reparation clauses. $\mathcal{CL}$ language incorporates the notions of contrary-to-duty and contrary-to-prohibition by explicitly attaching to the deontic modalities a reparation which is to be enforced in case of violations. LegalRuleML introduces in [2] a suborder list that is a list of deontic formulas to model penalties. We think $\mathcal{CL}$'s and LegalRuleML's use of only contrary-to-duty obligations to recover a contract when it is breached is too limited. There is no provision provided for recovery from technical or business-related issues. In FCL, we interpret an agreement in a contract in terms of the deontic concepts of *obligation* and *prohibition*. These concepts are applied in expressions to actions that are executed by the parties of

the contract. Thus, the concepts express what a party *ought to do* and or *ought not do*. FCL rules can also be used for reparational purposes when an agreement is violated (see subsection 6.2).

With the exception of approaches provided by AEHSS, BBE, P&S, and LegalRuleML TC, all of the languages above are informal. The work of both AEHSS and P&S include a trace-based reduction semantics model for contracts. These two approaches provide a run-time monitoring of the fulfillment and breach of a contract since the state of a contract at a time is determined by the events that have happened. LegalRuleML utilizes the defeasible deontic logic to reason about violations of obligations. Both GHM's work and G&M's work lack a formal semantics and a reasoning system even though they provide a good framework for monitoring contracts. The semantics provided by J&E in [21] is based on stochastic processes. J&E's approach provides the ability to perform compositional analysis of monetary values of contracts. This work can estimate the expected value of financial contracts. BBE's trace-based semantics allows the modification of a contract according to the passage of time and the values of observables. But since both of the approaches provided by J&E and BBE pay more attention to finding the monetary value of contracts, they consider the semantic meaning of a contract to be its cash-flow gain or loss, which is too limited for general contracts from our point of view. We find this lack of work on formal semantics surprising since one of the main benefits of defining a contract language to be formal is to enable the language to have a precise, unambiguous semantics.

Although the languages of AEHSS and P&S provide a formal mathematical model for contracts with a formal semantics and are able to express some important features of contracts, they are not as expressive as FCL. For example, in the case where a contract is breached, the monitor should not only report a breach of contract, but also who among the contract parties is responsible (blame assignment). Except for the languages provided by BBE and LegalRuleML TC, all the other contracts covered by these approaches, including the work of AEHSS and P&S, are two-party contracts in which the parties are implicit. These approaches are not able to determine who is to be blamed when a contract is breached. Our proposed language provides explicit participants and thus provides the possibility of having contracts with both an unrestricted number of parties and with blame assignment.

In addition, because time constraints are implicit in P&S's $\mathcal{CL}$ language, it only has relative deadlines where one party's commitment to do something depends on when the other party has performed an action. Our proposed language FCL has not only relative temporal constraints, but also absolute temporal constraints.

11 Conclusion and Future Work

In this paper we have presented FCL, a formal language for writing contracts that may contain temporally based conditions. Changes to the meaning of a

FCL contract are triggered when the conditions in it become true. FCL admits agreements that correspond to the deontic notions of obligation and prohibition, can express conditions that depend on events and other observables, and include condition-based rules to define new constants and introduce new agreements and rules. We have sketched a reasoning system for FCL. To our knowledge, no other formal contract language is as expressive as FCL.

FCL offers three advantages to the contract writer. First, since FCL has a precise semantics, contracts written in FCL have an unambiguous meaning. Since the underlying logic of FCL is simple type theory, the semantics of contracts written in FCL is based on very well understood ideas and the logical tools for writing contracts in FCL are very expressive. Third, since FCL is a formal language, software-implemented formal methods can be used to assist in the writing and analysis of FCL contracts. In particular, we can use software tools to check whether an action in a contract has been performed or not, to report whether a contract has been fulfilled or violated, to compute the value of a contract, etc. We can also use software tools to reason about possible future outcomes of a contract and about the relationship between different contracts.

Our future work will include (1) extending the design of FCL, (2) writing several additional examples of contracts in FCL, (3) finishing the development of a reasoning system for FCL, (4) designing a module system for building contracts out of contract modules, and (5) integrating FCL with contract law and regulations. We will also validate FCL by implementing it in Agda [7, 18, 19], a dependently typed functional programming language. In a future paper, we will give a full presentation of FCL and its implementation in Agda.

Acknowledgments

The authors are grateful to the reviewers for their comments and suggestions. This research was supported by NSERC.

References

1. Andersen, J., Elsborg, E., Henglein, F., Simonsen, J., Stefansen, C.: Compositional specification of commercial contracts. International Journal on Software Tools for Technology Transfer (STTT) 8(6), 485–516 (2006)
2. Athan, T., Boley, H., Governatori, G., Palmirani, M., Paschke, A., Wyner, A.: OASIS LegalRuleML. In: Francesconi, E., Verheij, B. (eds.) ICAIL. pp. 3–12. ACM (2013)
3. Athan, T., Governatori, G., Palmirani, M., Paschke, A., Wyner, A.Z.: LegalRuleML: Design principles and foundations. In: Wolfgang Faber and Adrian Pashke (ed.) The 11th Reasoning Web Summer School. pp. 151–188. Springer, Berlin, Germany (July 2015)
4. Attorney, R.S.: Contracts: The Essential Business Desk Reference. Nolo (2010)

5. Bahr, P., Berthold, J., Elsman, M.: Certified symbolic management of financial multi-party contracts. In: Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming, ICFP 2015, Vancouver, BC, Canada, September 1-3, 2015. pp. 315–327 (2015)
6. Blum, B.A.: Contracts: Examples and Explanations. Aspen Publishers, fourth edn. (2007)
7. Bove, A., Dybjer, P., Norell, U.: A brief overview of Agda — A functional language with dependent types. In: TPHOLs. vol. 9, pp. 73–78. Springer (2009)
8. Farmer, W.M.: The seven virtues of simple type theory. Journal of Applied Logic 6, 267–286 (2008)
9. Farmer, W.M., Hu, Q.: A formal language for writing contracts. In: 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI 2016). pp. 134–141. IEEE (2016)
10. Fenech, S., Pace, G.J., Schneider, G.: Automatic conflict detection on contracts. In: International Colloquium on Theoretical Aspects of Computing. pp. 200–214. Springer (2009)
11. Finan, M.B.: A discussion of financial economics in actuarial models: A preparation for exam mfe/3f (2015), prepared for Arkansas Tech University
12. Goodchild, A., Herring, C., Milosevic, Z.: Business contracts for B2B. In: Proceedings of the CAISE00 Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing, Stockholm, Sweden (2000)
13. Governatori, G., Milosevic, Z.: A formal analysis of a business contract language. International Journal of Cooperative Information Systems 15(04), 659–685 (2006)
14. Governatori, G., Rotolo, A.: Logic of violations: a gentzen system for reasoning with contrary-to-duty obligations. Australasian Journal of Logic 4, 193–215 (2005)
15. Hvitved, T., Klaedtke, F., Zălinescu, E.: A trace-based model for multiparty contracts. Journal of Logic and Algebraic Programming (2011)
16. Linington, P.F., Milosevic, Z., Cole, J., Gibson, S., Kulkarni, S., Neal, S.: A unified behavioural model and a contract language for extended enterprise. Data & Knowledge Engineering 51(1), 5–29 (2004)
17. McNamara, P.: Deontic logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Winter 2014 edn. (2014)
18. Norell, U.: Towards a Practical Programming Language based on Dependent Type Theory. Ph.D. thesis, Chalmers University of Technology (2007)
19. Norell, U.: Dependently typed programming in Agda. In: Kennedy, A., Ahmed, A. (eds.) Proceedings of the 4th International Workshop on Types in Language Design and Implementation. pp. 1–2. No. 2 in TLDI '09, ACM, New York, NY, USA (2009)
20. Peyton Jones, S.L.: Composing contracts: An adventure in financial engineering. In: Proceedings of the International Symposium of Formal Methods Europe on Formal Methods for Increasing Software Productivity. Lecture Notes in Computer Science, vol. 2021, p. 435. Springer (2001)
21. Peyton Jones, S.L., Eber, J.M.: How to write a financial contract. In: Gibbons, J., de Moor, O. (eds.) The Fun of Programming, pp. 105–130. Cornerstones in Computing, Palgrave (2003)
22. Poole, J.: Textbook on Contract Law. Oxford University Press, 11 edn. (2012)
23. Prisacariu, C., Schneider, G.: An algebraic structure for the action-based contract language cl-theoretical results. Tech. rep., Technical Report 361, Department of Informatics, University of Oslo, Oslo, Norway (2007)

24. Prisacariu, C., Schneider, G.: A formal language for electronic contracts. In: International Conference on Formal Methods for Open Object-Based Distributed Systems. pp. 174 – 189. Springer (2007)
25. Prisacariu, C., Schneider, G.: Towards a formal definition of electronic contracts. Tech. rep., Technical Report 348, Department of Informatics, University of Oslo, Oslo, Norway (2007)
26. Prisacariu, C., Schneider, G.: A dynamic deontic logic for complex contracts. The Journal of Logic and Algebraic Programming 81, 458–490 (2012)