A Formal Language for Writing Contracts

William M. Farmer
McMaster University
Hamilton, Ontario, Canada
wmfarmer@mcmaster.ca

Qian Hu
McMaster University
Hamilton, Ontario, Canada
huq6@mcmaster.ca

***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 a formal language with a precise semantics for expressing general contracts that may depend on temporally based conditions. We argue that the language is more effective for writing and analyzing contracts than previously proposed formal contract languages.**

***Index Terms*—Contracts, formal languages, simple type theory, observables, deontic logic, conditional agreements, temporally based conditions.**

I. Introduction

A contract records, orally or in writing, a legally binding agreement between two or more parties [1]. Contracts come in many forms and are used for many purposes [1], [2]. 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 [3].

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 II presents a simple example of a dynamic contract. Section III discusses what properties contracts have. An overview of FCL is given in section IV, and the formal semantics of FCL is outlined in section V. Section VI shows how the example from Section II can be expressed in FCL. A more complex example expressed in FCL is given in Section VII. How FCL is related to other formal and informal contract languages is summarized in section VIII, and the paper concludes with section IX.

II. Example

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 [4]. 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.

III. 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 [2, 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" [2, 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 [5]. 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 [3]. 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.

IV. 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 the section V.

A. Underlying Logic

We will assume that the underlying logic of FCL is some version of simple type theory [6]. 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 have 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 IV-B.

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.

Building FCL on simple type theory gives FCL access to the high expressivity and reasoning power of simple type theory [6]. 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.

B. Observables

An *observable* is something that has a variable value that can be observed at a particular time [7], [8]. 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, $o =$ true and $o =$ 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.

$\mathbb{O}(a, \mathcal{T})$	stands for	$\lambda\, t : \mathsf{Time}\ .\ \exists\, u : \mathsf{Time}\ .$ $u \in \mathcal{T} \wedge u \leq t \wedge \mathsf{obs\text{-}event}(u, a).$
$\mathbb{F}(a, \mathcal{T})$	stands for	$\lambda\, t : \mathsf{Time}\ .\ \forall\, u : \mathsf{Time}\ .$ $u \in \mathcal{T} \supset (u \leq t \wedge \neg \mathsf{obs\text{-}event}(u, a)).$

TABLE I
NOTATIONAL DEFINITIONS

C. 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.

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.

D. 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 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.

E. Notational Definitions

We will introduce now the notational definitions in Table I. Let a be an action and $\mathcal{T}$ be a set of times. Expressions of the form $\mathbb{O}(a, \mathcal{T})$ and $\mathbb{F}(a, \mathcal{T})$ are called *obligations* and *prohibitions*. respectively. These two kinds of expressions are used in FCL to represent agreements (which are defined in the next subsection). $\mathbb{O}(a, \mathcal{T})(t)$ asserts that the action a has been observed at some time in $\mathcal{T}$ at or before the time t, and $\mathbb{F}(a, \mathcal{T})(t)$ asserts that action a has not been observed at any time in $\mathcal{T}$ at or before the time t. The operators $\mathbb{O}$ and $\mathbb{F}$ are inspired by the deontic operators for *obligation* and *prohibition* [5].

F. Agreements

An *agreement* is a promise to do or not do a specific action. An *agreement* of FCL is a time predicate p such that $p(t)$

asserts that the promise represented by p has been met at time t. Obligations $\mathbb{O}(a, \mathcal{T})$ and prohibitions $\mathbb{F}(a, \mathcal{T})$, where a is an action and $\mathcal{T}$ is a set of times, are especially useful examples of agreements. $\mathbb{O}(a, \mathcal{T})$ represents the promise that the action a will occur at some time in $\mathcal{T}$, and $\mathbb{F}(a, \mathcal{T})$ represents the promise that the action a will not occur at any time in $\mathcal{T}$.

Notice that agreements in FCL do not include an expression formed using an operator corresponding to the deontic operator for *permission*. Permissions and obligations are really the same thing looked at from different points of view. For example, A's obligation to pay B money, is the same thing as B's permission to be paid by A, and A's permission to receive a payment from B, is the same as B shall pay A the money.

We think permission more properly refers to a discretionary authority to trigger an obligation for the other party to perform. In FCL, one party's permission to perform an action a will often be expressed by:

1. The absence of an obligation to perform a;
2. The absence of a prohibition to perform a; and
3. A conditional agreement that says, if a is performed, then the other party is obligated to perform some action b.

Thus, a permitted action will be expressed as a rule that can create the other party's obligations in response to the performance of this action. Consider Example 1 in which the option buyer is permitted to exercise the option. This permission is captured by the conditional agreement that, if the option buyer exercises this option, then the option seller is obligated to sell a share of stock to the option buyer.

G. 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 the underlying logic 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 V that, if φ is satisfied at time t, the members of $\mathcal{B}$ are added to the state of a contract at $t + 1$. 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*.

H. 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 constant definitions, agreements, and rules. The *parties* of C are the parties mentioned in the rules 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.

V. Formal Semantics of FCL

This section presents the highlights of the formal semantics of FCL.

A. Models

A *model* of FCL is a model of the underlying logic of FCL. 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 $a_1, \ldots, a_n$).

B. Agreements

Let A be an agreement of FCL and $t \in \mathbb{Z}$. The *value* of A in $\mathcal{M}$ at time t is $V^{\mathcal{M}}(A(\bar{t}))$, where $\bar{t}$ is some canonical expression whose value is t. An agreement is *satisfied [violated]* in $\mathcal{M}$ at time t if its value in $\mathcal{M}$ at t is T [F].

C. 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 e. Let $\mathcal{B}\sigma = \{e\sigma \mid e \in \mathcal{B}\}$ for $\sigma \in \mathsf{sub}(\varphi, t)$. Then define $\mathsf{new\text{-}items}(R, \mathcal{M}, t)$ to be

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

R is *active* in $\mathcal{M}$ at t if $V^{\mathcal{M}}(\varphi\sigma) = \text{T}$ for some $\sigma \in \mathsf{sub}(\varphi, t)$. R is *defunct* in $\mathcal{M}$ at t if $V^{\mathcal{M}}(\varphi\sigma) = \text{F}$ 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 $u \geq t$.

D. 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 $\mathsf{state}(C, \mathcal{M}, t + 1) =$

$$(\mathsf{state}(C, \mathcal{M}, t) \cup \bigcup_{R \in \mathcal{B}} \mathsf{new\text{-}items}(R, \mathcal{M}, t).$$

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 $\mathcal{M}$ at t and every rule in $\mathsf{state}(C, \mathcal{M}, t)$ is defunct in $\mathcal{M}$ 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 $\mathcal{M}$ 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 $\mathcal{M}$ at t.

C is *perpetually fulfilled [perpetually breached]* in $\mathcal{M}$ if there is a time $t \geq t_{\text{offer}}$ such that C is fulfilled [breached] in $\mathcal{M}$ at every $u \geq t$.

VI. Example 1: An American Call Option

We formalize here the American Call Option introduced in Section II 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 be the day before June 30, 2015 when the seller offered the contract to the buyer. 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:
   $\varphi_1 = \mathsf{obs\text{-}event}(X_{\text{buy}}, e_1) \wedge X_{\text{buy}} = t_{\text{buy}}$.
   $e_1 = (\text{“Buy Option”}, \mathsf{transfer}(\$5), \{\mathsf{buyer}\}, \{\mathsf{seller}\})$.
   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. 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 $t_{\text{exercise}} = \overline{u}$.

VII. Example 2: A Sale of Goods Contract

We previously saw an encoding of the American Call Option in FCL. In this section, we extend an example of a sale of a laser printer contract from [9, p. 5] and provide a formalization of this contract in FCL.

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.

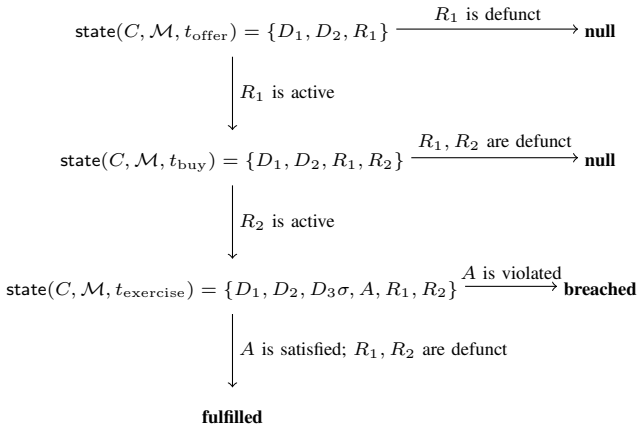

Fig. 1. Execution of the American Call Option C

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.

Although this example contract is very simple, two points should be noticed. First, a contract usually specifies actions to be taken in case of the violation of a part of the contract. In the deontic literature [5], this is known as *contrary-to-duty obligations (CTDs)* or *reparational obligations*. Clause 4 of this contract is such an example in which a potential violation can generate obligations to “repair” the violation. Second, consider the total amount 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 not the buyer’s intention. In fact, the total amount to repay is the amount that the buyer has already paid the seller.

Now we formalize this contract in FCL to explain how we deal with the above problems. 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 ten 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:*
   $R_2 = \varphi_2 \mapsto \{D_2, D_3, R_3, R_4, R_5, R_6\}$.
   $\varphi_2 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_2)$
   $\wedge\, t_{\text{order}} \leq X_{\text{time}} \leq t_{\text{order}} + 22$.
   $D_2 : t_{\text{deliver}} = X_{\text{time}}$.
   $D_3 : \mathsf{obs\text{-}total\text{-}paid}(X_{\text{time}}) = 0$.
   R_3, R_4, R_5 and R_6 are defined below.
3. *Rule for Returning the Printer:*
   $R_3 = \varphi_3 \mapsto \{D_4, A_2\}$.
   $\varphi_3 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_3)$
   $\wedge\, t_{\text{deliver}} \leq X_{\text{time}} \leq t_{\text{deliver}} + 14$.
   $e_3 = (\text{“Return Printer”}, \mathsf{transfer}(\mathsf{printer}), \{\mathsf{buyer}\}, \{\mathsf{seller}\})$.
   $D_4 : t_{\text{return}} = X_{\text{time}}$.
   $A_2 = \mathbb{O}(e_4(\mathsf{obs\text{-}total\text{-}paid}(X_{\text{time}})), [t_{\text{return}}, t_{\text{return}} + 7])$.
   $e_4 = \lambda X_{\text{total}}.(\text{“Return Payment”}, \mathsf{transfer}(X_{\text{total}}), \{\mathsf{seller}\}, \{\mathsf{buyer}\})$.
4. *Rules for Recording the Payment:*
   $R_4 = \varphi_4 \wedge \varphi_5 \mapsto \{D_5\}$.
   $R_5 = \neg\varphi_4 \wedge \varphi_5 \mapsto \{D_6\}$.
   $\varphi_4 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_5(X_{\text{payment}}))$
   $\wedge\, t_{\text{deliver}} \leq X_{\text{time}}$.
   $e_5 = \lambda X_{\text{payment}}.(\text{“Pay Seller”}, \mathsf{transfer}(X_{\text{payment}}), \{\mathsf{buyer}\}, \{\mathsf{seller}\})$.
   $\varphi_5 = \neg\mathsf{obs\text{-}event\text{-}before}(X_{\text{time}}, e_3)$.
   $D_5 : \mathsf{obs\text{-}total\text{-}paid}(X_{\text{time}}) =$
   $\mathsf{obs\text{-}total\text{-}paid}(X_{\text{time}} - 1) + X_{\text{payment}}$.
   $D_6 : \mathsf{obs\text{-}total\text{-}paid}(X_{\text{time}}) = \mathsf{obs\text{-}total\text{-}paid}(X_{\text{time}} - 1)$.
5. *Rules for Making Payments:*
   $R_6 = \varphi_6 \mapsto \{A_3, R_7\}$.
   $\varphi_6 = \neg\mathsf{obs\text{-}event\text{-}before}(t_{\text{delivery}} + 1, e_3)$.
   $A_3 = \mathbb{O}(e_5(200/2), [t_{\text{deliver}}, t_{\text{deliver}} + 1])$.
   $R_7 = \varphi_7 \wedge \varphi_5 \mapsto \{D_7, R_8, R_9, R_{10}\}$.
   $\varphi_7 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_5(200/2))$
   $\wedge\, t_{\text{deliver}} \leq X_{\text{time}} \leq t_{\text{deliver}} + 1$.
   $D_7 : t_{\text{first}} = X_{\text{time}}$.
   $R_8 = \varphi_8 \wedge \varphi_5 \mapsto \{A_4\}$.
   $\varphi_8 = \neg\mathsf{obs\text{-}event\text{-}before}(X_{\text{time}}, e_5(200/2))$
   $\wedge\, t_{\text{first}} + 1 \leq X_{\text{time}} \leq t_{\text{deliver}} + 14$.
   $A_4 = \mathbb{O}(e_5(200/2), [t_{\text{deliver}} + 15, t_{\text{deliver}} + 44])$.
   R_9, R_{10} are defined below.
6. *Rules for Paying Fine for a Late Payment:*
   $R_9 = \varphi_9 \wedge \varphi_5 \mapsto \{D_8\}$.
   $R_{10} = \varphi_{10} \mapsto \{A_5\}$.
   $\varphi_9 = \mathsf{obs\text{-}event}(X_{\text{time}}, e_5(200/2))$
   $\wedge\, t_{\text{first}} \leq X_{\text{time}} \leq t_{\text{deliver}} + 44$.
   $D_8 : t_{\text{second}} = X_{\text{time}}$.
   $\varphi_{10} = t_{\text{deliver}} + 31 \leq t_{\text{second}} \leq t_{\text{deliver}} + 44$.
   $A_5 = \mathbb{O}(e_5(10\% * 200/2), [t_{\text{second}}, t_{\text{second}}])$.

t_{order}, t_{deliver}, t_{return}, t_{first}, and t_{second} 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.

$\mathsf{obs\text{-}total\text{-}paid}(t)$ 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. $\mathsf{obs\text{-}event\text{-}before}(t, e)$ represents the observation that the event e occurred before 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, R_9, and R_{10} work together for the reparation purpose if the violation occurs. 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_{10} is active, then an additional fine must be paid.

VIII. 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) [7], [8].
- A. Goodchild, C. Herring, and Z. Milosevic (GHM) [10].
- G. Governatori and Z. Milosevic (G&M) [11], [12], [13].
- J. Andersen, E. Elsborg, F. Henglein, J.G. Simonsen, and C. Stefansen (AEHSS) [14].
- C. Prisacariu and G. Schneider (P&S) [12], [15], [16], [17], [18].
- P. Bahr, J. Berthold, M. Elsman (BBE) [19].
- LegalRuleML Technical Committee (TC) [20], [21].

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 contents 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 [10], [13], [12], 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 [14] to model contracts. J&E's functional programming based language [7], [8] and BBE's cash-flow trace based approach [19] use the idea of observables to specify events. P&S introduce in [15], [16], [17], [18] 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 add 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 idea of observables, we introduce in FCL the concept of a *rule* that is 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. The LegalRuleML introduces in [20] a suborder list that is a list of deontic formulas to model penalies. 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.

With the exception of works provided by AEHSS, P&S, and LegalRuleML TC, all of the languages above are informal. GHM and G&M 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 [8] and BBE in [19] are based on stochastic processes. But since both of the two approaches pay more attention to finding the monetary value of contracts, they consider the semantic meaning of a contract to be its cash-flow gains 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 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 both contracts with any number of parties and 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 has not only relative temporal constraints, but also absolute temporal constraints.

IX. 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. FCL can be extended to express laws and regulations and to thus identify agreements of a contract that are in conflict with the underlying laws and regulations of the contract. 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. 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) developing 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 [22], [23], [24], 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] J. Poole, *Textbook on Contract Law*, 11th ed. Oxford University Press, 2012.

[2] B. A. Blum, *Contracts: Examples and Explanations*, 4th ed. Aspen Publishers, 2007.

[3] R. S. Attorney, *Contracts: The Essential Business Desk Reference*. Nolo, 2010.

[4] M. B. Finan, *A Discussion of Financial Economics in Actuarial Models: A Preparation for Exam MFE/3F.*, Arkansas Tech University, 2015.

[5] P. McNamara, "Deontic logic," in *The Stanford Encyclopedia of Philosophy*, winter 2014 ed., E. N. Zalta, Ed., 2014.

[6] W. M. Farmer, "The seven virtues of simple type theory," *Journal of Applied Logic*, vol. 6, pp. 267–286, 2008.

[7] S. L. Payton Jones, "Composing contracts: An adventure in financial engineering," in *Proceedings of the International Symposium of Formal Methods Europe on Formal Methods for Increasing Software Productivity*, ser. Lecture Notes in Computer Science, vol. 2021. Springer, 2001, p. 435.

[8] S. L. Peyton Jones and J. M. Eber, "How to write a financial contract," in *The Fun of Programming*, ser. Cornerstones in Computing, J. Gibbons and O. de Moor, Eds. Palgrave, 2003, pp. 105–130.

[9] T. Hvitved, F. Klaedtke, and E. Zălinescu, "A trace-based model for multiparty contracts," *Journal of Logic and Algebraic Programming*, 2011.

[10] A. Goodchild, C. Herring, and Z. Milosevic, "Business contracts for b2b," in *Proceedings of the CAISE00 Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing, Stockholm, Sweden*, 2000.

[11] G. Governatori and Z. Milosevic, "A formal analysis of a business contract language," *International Journal of Cooperative Information Systems*, vol. 15, no. 04, pp. 659–685, 2006.

[12] G. Governatori and A. Rotolo, "Logic of violations: a gentzen system for reasoning with contrary-to-duty obligations," *Australasian Journal of Logic*, vol. 4, pp. 193–215, 2005.

[13] P. F. Linington, Z. Milosevic, J. Cole, S. Gibson, S. Kulkarni, and S. Neal, "A unified behavioural model and a contract language for extended enterprise," *Data & Knowledge Engineering*, vol. 51, no. 1, pp. 5–29, 2004.

[14] J. Andersen, E. Elsborg, F. Henglein, J. Simonsen, and C. Stefansen, "Compositional specification of commercial contracts," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 8, no. 6, pp. 485–516, 2006.

[15] C. Prisacariu and G. Schneider, "An algebraic structure for the action-based contract language cl-theoretical results," Technical Report 361, Department of Informatics, University of Oslo, Oslo, Norway, Tech. Rep., 2007.

[16] ——, "A formal language for electronic contracts," *Formal Methods for Open Object-Based Distributed Systems*, pp. 174 – 189, 2007.

[17] ——, "Towards a formal definition of electronic contracts," Technical Report 348, Department of Informatics, University of Oslo, Oslo, Norway, Tech. Rep., 2007.

[18] ——, "A dynamic deontic logic for complex contracts," *The Journal of Logic and Algebraic Programming*, pp. 458–490, 2012.

[19] P. Bahr, J. Berthold, and M. Elsman, "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*, 2015, pp. 315–327. [Online]. Available: http://doi.acm.org/10.1145/2784731.2784747

[20] T. Athan, H. Boley, G. Governatori, M. Palmirani, A. Paschke, and A. Wyner, "Oasis legalruleml." in *ICAIL*, E. Francesconi and B. Verheij, Eds. ACM, 2013, pp. 3–12.

[21] T. Athan, G. Governatori, M. Palmirani, A. Paschke, and A. Z. Wyner, "LegalRuleML: Design principles and foundations," in *The 11th Reasoning Web Summer School*, Wolfgang Faber and Adrian Pashke, Ed. Berlin, Germany: Springer, jul 2015, pp. 151–188.

[22] A. Bove, P. Dybjer, and U. Norell, "A brief overview of Agda — A functional language with dependent types," in *TPHOLs*, vol. 9. Springer, 2009, pp. 73–78.

[23] U. Norell, "Towards a practical programming language based on dependent type theory," Ph.D. dissertation, Chalmers University of Technology, 2007.

[24] ——, "Dependently typed programming in Agda," in *TLDI*, A. Kennedy and A. Ahmed, Eds. ACM, 2009, pp. 1–2.