A Formal Language for Electronic Contracts*

Cristian Prisacariu and Gerardo Schneider

Dept. of Informatics – Univ. of Oslo,
P.O. Box 1080 Blindern, N-0316 Oslo, Norway
cristi@ifi.uio.no, gerardo@ifi.uio.no

Abstract. In this paper we propose a formal language for writing electronic contracts, based on the deontic notions of obligation, permission, and prohibition. We take an ought-to-do approach, where deontic operators are applied to actions instead of state-of-affairs. We propose an extension of the μ-calculus in order to capture the intuitive meaning of the deontic notions and to express concurrent actions. We provide a translation of the contract language into the logic, the semantics of which faithfully captures the meaning of obligation, permission and prohibition. We also show how our language captures most of the intuitive desirable properties of electronic contracts, as well as how it avoids most of the classical paradoxes of deontic logic. We finally show its applicability on a contract example.

1 Introduction

With the imminent use of Internet as a means for developing cross-organizational collaborations and virtual communities engaged in business, new challenges arise to guarantee a successful integration and interoperability of such virtual organizations. Service-oriented architectures (SOA) is becoming more and more the trend in this arena. Entities participating in a SOA have no access to complete information, including information for checking the reliability of the service provider and/or service consumer. For instance, a service consumer has no access to the code implementing the service, and is therefore unable to examine, much less verify, the service implementation to have assurance of its compliance with his/her needs. This motivates the need of establishing an agreement before any transaction is performed, through a *contract*, engaging all participants in the transaction under the commitments stipulated in such a document, which must also contain clauses determining penalties in case of contract violations. A service provider may also use a contract template (i.e., a yet-to-be-negotiated contract) to publish the services it is willing to provide. As a service specification, a contract may describe many different *aspects* of a service, including functional properties and also non-functional properties like quality of service (QoS).

In order to advance towards a reliable SOA, we need to be able to write contracts which can be "understood" by the software engaged in the negotiation

* Partially supported by the Nordunet3 project "Contract-Oriented Software Development for Internet Services".

process, and later may be used by virtual organizations responsible for ensuring that the contract is respected. In other words, contracts should be amenable to formal analysis and thus written in a formal language.

There are currently several different approaches aiming at defining a formal language for contracts, the most promising approach, in our opinion, being the one based on logics. A logic for contracts not necessarily has to be based on, or extend, *deontic logic*, but must contain notions like obligation, permission, and prohibition, and preserve their intuitive properties. Formalizing the usual normative (deontic) notions of obligation, permission and prohibition is not an easy task as witnessed by the extensive research conducted by the deontic community both from the philosophical and the logical point of view, starting as early as 1926 [17].[1] In what follows we discuss some of the problems and challenges that appear when defining *electronic contracts* (e-contracts).

In early papers (e.g. [33]) the approach was to relate the normative notions of obligation, permission and prohibition in a similar way as the quantifiers *all*, *some* and *no*, and the modalities *necessary*, *possible* and *impossible*. This was the bases of the so-called Standard Deontic Logic (SDL) which builds up on propositional classical logic, leading to a nice formalization but also to many paradoxes which still continue to challenge philosophers, logicians and computer scientists.

Besides avoiding paradoxes, one of the first issues to take into account when formalizing normative notions is whether we want to represent (names of) human actions or (sentences describing) states of affairs, product of a human action. The former is usually known as *ought-to-do* and the latter as *ought-to-be*. For example "Jones ought to pay the money" is an ought-to-do sentence, while "It ought to be the case that Jones pays the money" is an ought-to-be sentence. In general the relationship between both representations is not as obvious as in the above example and the translation from one to the other is much more involved. The discussion among philosophers and logicians is far from an end in what concerns the decision of whether one approach is better than the other, or even if both should coexist in the same reasoning system. In many e-contracts it is more natural to find statements of *ought-to-do* kind; where the *subject* is stated explicitly (the supplier, the client), the *actions* (that are permitted or forbidden) are visible, and also in many cases there might be an *object*. There may also be cases where an ought-to-be approach gives a more concise expression, like in QoS contracts where we may have statements expressing quantitative restrictions like *the average bandwidth should be more than 20kb/s*.

Contracts contain clauses which by definition are violable (if we have the guarantee that nobody will violate them, contracts would be useless). Hence, *contrary-to-duty obligations* (CTD) and *contrary-to-prohibitions* (CTP) are important aspects to be considered. CTDs are statements that represent the fact that obligations might not be respected where CTPs are similar statements which deal with prohibitions that might be violated. Both constructions specify

[1] Mally's work is considered a precursor of Deontic Logic, though it is widely accepted that modern Deontic Logic started with the work by G.H. von Wright [33].

the obligation/prohibition to be fulfilled and which is the *reparation/penalty* to be applied in case of violation.

Other problems to be considered when formalizing deontic notions are the study of their interrelation (duality and definition in terms of each other), the understanding of their truth-value (even the discussion whether it is reasonable to talk about the truth-value of such notions), and the difference between "must" and "ought".

Since we are concerned with formal definition of e-contracts we are definitely on a terrain where many of the philosophical problems of the deontic logic are not present. In this paper we take a first step towards the definition of a formal contract language following an ought-to-do approach. Our starting point is [5], where a fix-point characterization of obligation, permission and prohibition is given, based on the modal μ-calculus, allowing the definition of the deontic notions over regular actions.

The main contribution of this paper is the definition of a contract language with the following properties:

1. The language avoids most of the classical paradoxes of deontic logic;
2. It is possible to express in the language obligations, permission and prohibition over concurrent actions keeping their intuitive meaning;
3. Obligation of disjunctive and conjunctive actions is defined compositionally;
4. The definition and semantics of obligation does not contain action negation;
5. It is possible to express CTDs and CTPs;
6. The language has a formal semantics given in a variant of the propositional μ-calculus.

Other side contributions are:

1. We revisit the relations between the deontic notions, providing new insights into how they should be related in the context of e-contracts;
2. We give special attention to the disjunction on obligations, to which we provide a natural and precise interpretation;
3. We extend the propositional μ-calculus with the possibility of expressing concurrent and deterministic actions.

The paper is organized as follows. In Section 2 we start by presenting an example of a partial contract, we then informally discuss some of the desirable properties a contract language should have, and finally present our formal language for writing contracts. In Section 3 we present a variant of the μ-calculus, with its syntax and semantics, and we give a translation of the contract language into the logic. In Section 4 we show that our language avoids many of the paradoxes and that it satisfies most of the desirable properties listed in Section 2. Before concluding, we present in Section 5 the modeling of the example of Section 2 using our contract language.

2 A Formal Language for Contracts

We start by presenting an example, we then list desirable properties for defining a contract language and we describe informally the kind of actions that are needed for our language. In the last subsection we present the syntax of the language for writing e-contracts and the intuition behind it.

2.1 A Contract Example

In what follows we provide part of a contract between a service provider and a client, where the provider gives access to Internet to the client. We consider two parameters of the service: *high* and *low*, which denote the client's Internet traffic. We abstract away from several technical details as how it is measured the Internet traffic. We will consider only the following clauses of the contract:

1. Whenever the Internet traffic is *high* then the client must pay x \$ immediately, or the client must notify the service provider by sending an e-mail specifying that he will pay later.
2. In case the client delays the payment, after notification he must immediately lower the Internet traffic to the *low* level, and pay later $2 * x$ \$.
3. If the client does not lower the Internet traffic immediately, then the client will have to pay $3 * x$ \$.
4. The provider is forbidden to cancel the contract without previous written notification by normal post and by e-mail.
5. The provider is obliged to provide the services as stipulated in the contract, and according to the law regulating Internet services.

A formalization of the above will be presented in Section 5.

2.2 Desirable Properties of a Language for Contracts

In what follows we use $+$ for *choice* among actions, $O(a)$ to denote the obligation of performing a given action a, and similarly for permission $P(a)$ and prohibition $F(a)$. A more precise definition will be given later.

General Requirements: We list first some general intuitive properties we should have, and others we should avoid, when formalizing deontic notions in contracts.

We want to avoid as many logical paradoxes as possible[2], in particular: the Ross paradox (i.e., $O(a) \Rightarrow O(a + b)$)[3], and the free choice paradox (i.e., $P(a) \Rightarrow P(a + b)$). Syntactically disallow the classical disjunction between deontic modalities. Obligation should be defined only on actions, not on formulas (which, as argued in the deontic community, would avoid several of the present paradoxes). Conjunction on obligations should imply executing the obliged actions at the same time (not to violate any of the obligations). Obligation of a sequence of actions should imply the obligation of all the subsequent actions. Allow specification of reparations for violations of obligations and prohibitions. Allow the definition of conditional obligations (i.e., $\varphi \Rightarrow O(a)$). Obligation

[2] For a list of classical paradoxes see [28].

[3] The symbol "$\Rightarrow$" is not part of our contract language and we use it informally as a shortcut for "if-then" or "implies".

should imply permission. Do not define permission and obligation in terms of each other (see von Wright's argument [34]). Defining permission in terms of prohibition is natural and desirable.

Properties of Electronic Contracts: In the philosophical and pure logic contexts we find many reasonable discussions related to deontic operators, which we claim can be avoided given that we are restricted to e-contracts. In what follows we provide arguments for restricting syntactically the occurrence in the contract language of certain expressions involving obligation, permission and prohibition applied to actions.

It is not natural to have in contracts statements like *one is NOT obliged to perform an action*, thus $\neg O(a)$ should not occur in a contract. A statement like *one is NOT permitted to do something* can be rewritten as *one is forbidden to do something*; $\neg P(a) \equiv F(a)$. Also *one is NOT forbidden to do an action* can be rewritten as *one is permitted to do the action*; thus we should consider $\neg F(a) \equiv P(a)$. We adhere thus to the usual approach of defining permission and prohibition as one being the negation of the other.

It is not intuitive to have the + under the F operator. Consider for example the following norm: *In Europe it is forbidden one of the following actions (but not both): to drive on the left side of the road (d_l), or to drive on the right side (d_r)* which can be represented as $F(d_l + d_r)$. The problem is that it is not clear under which circumstances each one of the actions can be taken. The natural way to exclusively forbid the choice between two actions is to relate each of the actions with its context. So, the above sentence could be rewritten as: *In the United Kingdom it is forbidden to drive on the right side of the road. In the rest of Europe (except United Kingdom) it is forbidden to drive on the left side of the road.* Which can be formalized as: $\varphi_{UK} \Rightarrow F(d_r)$ and $\varphi_{REU} \Rightarrow F(d_l)$. Where φ_{UK} and φ_{REU} are mutually exclusive. On the other hand, it is possible to forbid two actions a and b simultaneously by imposing $F(a) \wedge F(b)$.

Moreover, we argue that in contracts it is not common to find statements that may be formalized using an exclusive OR operator $\oplus$ between prohibitions. If we take the formula $F(a) \oplus F(b)$ to mean that either is forbidden a or forbidden b but not forbidden both then one case of the statement is $F(a) \wedge \neg F(b)$ which, using the above equivalence between P and $\neg F$ is $F(a) \wedge P(b)$. This means that one has the permission to do b. Similar from the second case, one may conclude that it is permitted to do a. In the end, the formula $F(a) \oplus F(b)$ does not explicitly prohibit anything, making its use completely meaningless and dangerous.

2.3 Actions

Our practical requirements to represent actions found in e-contracts force us to make some changes to the classical dynamic algebra [27]. We first drop the Kleene star (iteration) as it is unnatural to have it under the deontic operators. A second difference involves the inclusion of concurrent actions.

Our action algebra has a set of atomic actions denoted by $\mathcal{L}$, a set $\mathcal{B}$ of formulas in the Boolean algebra, and the action operators which define the compound actions: $+$ for *choice* of two actions[4], $\cdot$ for sequence of actions, & for concurrent execution of two atomic actions. The test operator ? is applied to elements of $\mathcal{B}$ and generates actions of $\mathcal{L}$. For brevity we often drop the sequence operator and instead of $\alpha \cdot \beta$ we just write $\alpha\beta$. We also define *action negation* $\overline{\alpha}$ of a compound action α as the action given by all the immediate traces that *take us outside* the trace of α [5] and is formally defined using a canonic form of the actions.

2.4 The Contract Language

We aim at the definition of a precise syntax of a contract language, with a translation into a logic in order to be able to reason about it. We define the contract language $\mathcal{CL}$, and provide a set of rewriting rules in order to simplify and minimize the number of expressions in the language.

Definition 1 (Contract Language Syntax). *A contract is defined by:*

$$\begin{array}{rcl} \mathcal{C}ontract & := & \mathcal{D} \, ; \, \mathcal{C} \\ \mathcal{C} & := & \phi \mid \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle\alpha\rangle\mathcal{C} \mid \mathcal{C}\,\mathcal{U}\,\mathcal{C} \mid \bigcirc\mathcal{C} \\ \mathcal{C}_O & := & O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\ \mathcal{C}_P & := & P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\ \mathcal{C}_F & := & F(\delta) \mid \mathcal{C}_F \vee [\delta]\mathcal{C}_F \end{array}$$

The syntax of $\mathcal{CL}$ closely resembles the syntax of a modal (deontic) logic. Though this similarity is clearly intentional since we are driven by a logic-based approach, $\mathcal{CL}$ is *not* a logic. In what follows we provide an intuitive explanation of the $\mathcal{CL}$ syntax; a more precise meaning will be given later through the translation into an extension of the propositional μ-calculus.

A contract consists of two parts: *definitions* ($\mathcal{D}$) and *clauses* ($\mathcal{C}$). Note that we deliberately let the definitions part underspecified in the syntax above. $\mathcal{D}$ specifies the *assertions* (or conditions) and the atomic actions present in the clauses. ϕ denotes assertions and ranges over Boolean expressions including arithmetic comparisons, like *the budget is more than 200$*. For now we let the atomic actions underspecified, which for our purposes can be understood as consisting of three parts: the proper action, the subject performing the action, and the target of (or, the object receiving) such an action. Note that, in this way, the partners involved in a contract are encoded in the actions.

$\mathcal{C}$ is the general *contract clause*. $\mathcal{C}_O$, $\mathcal{C}_P$, and $\mathcal{C}_F$ denote respectively *obligation*, *permission*, and *prohibition* clauses. $\wedge$ and $\oplus$ may be thought as the classical conjunction and exclusive disjunction, which may be used to combine obligations and permissions. For prohibition $\mathcal{C}_F$ we have $\vee$, again with the classical meaning of the corresponding operator. α is a compound action with syntax as given in Section 2.3, while δ denotes a compound action not containing any occurrence of $+$. Operationally, we consider that atomic actions do not require time for their execution, i.e., the atomic actions are *instantaneous*. A concurrent action is also instantaneous, so it can be seen as atomic. Note that syntactically $\oplus$ cannot

[4] We do not distinguish between internal (free) choice and external (imposed) choice.

Table 1. Compositional rules

(1)	$O(\alpha+\beta)$	$\equiv$	$O(\alpha) \oplus O(\beta)$
(2)	$O(a\&b)$	$\equiv$	$O(a) \wedge O(b)$
(3)	$O(\alpha\beta)$	$\equiv$	$O(\alpha) \wedge [\alpha]O(\beta)$
(4)	$P(\alpha+\beta)$	$\equiv$	$P(\alpha) \oplus P(\beta)$
(5)	$P(\alpha\beta)$	$\equiv$	$P(\alpha) \wedge \langle\alpha\rangle P(\beta)$
(6)	$F(\alpha\beta)$	$\equiv$	$F(\alpha) \vee [\alpha]F(\beta)$

Table 2. Rewriting rules for obligation O

(1)	$O(a) \wedge O(b)$	$\rightsquigarrow$	$O(a\&b)$
(2)	$O(a) \wedge O(a\&b)$	$\rightsquigarrow$	$O(a\&b)$
(3)	$O(a) \wedge (O(a) \oplus O(b))$	$\rightsquigarrow$	$O(a)$
(4)	$O(a) \wedge O(a)$	$\rightsquigarrow$	$O(a)$
(5)	$O(a) \oplus O(a)$	$\rightsquigarrow$	$O(a)$
(6)	$O(c) \wedge (O(a) \oplus O(b))$	$\rightsquigarrow$	$(O(c) \wedge O(a)) \oplus (O(c) \wedge O(b))$
(7)	$(\oplus_i O(a_i)) \wedge (\oplus_j O(b_j))$	$\rightsquigarrow$	$\oplus_{i,j}(O(a_i) \wedge O(b_j))$ $a_i \neq b_j$

appear between prohibitions and $+$ cannot occur under F, as we have discussed in Section 2.2.

We borrow from Propositional Dynamic Logic (PDL) the syntax $[\alpha]\phi$ to represent that after performing α (if it is possible to do so), ϕ must hold. The $[\cdot]$ notation allows having a *test*, where $[\phi?]\mathcal{C}$ must be understood as $\phi \Rightarrow \mathcal{C}$. $\langle\alpha\rangle\phi$ captures the idea that there must be the possibility of executing α, in which case ϕ must hold afterwards. Following temporal logic (TL) [23] notation we have $\mathcal{U}$ (*until*) and $\bigcirc$ (*next*) with intuitive semantics as in TL. Thus $\mathcal{C}_1 \mathcal{U} \mathcal{C}_2$ states that $\mathcal{C}_1$ should hold until $\mathcal{C}_2$ holds. $\bigcirc\mathcal{C}$ intuitively states that the $\mathcal{C}$ should hold in the next moment, usually after something happens. We can define $\Box\mathcal{C}$ (*always*) and $\Diamond\mathcal{C}$ (*eventually*) for expressing that $\mathcal{C}$ holds everywhere and sometimes in the future, respectively.

The rules of Table 1 are guided by common usage in electronic contracts and provides an equivalence relation between different syntactic expressions, which might also be interpreted as a means to define certain constructs compositionally. Note that concurrent actions are compositional only under obligation; there are no similar rules for F and P. Note that F has no rule for $+$ because exclusive choice does not appear under F. For an intuition and examples for these rules we refer to the extensive discussions in the technical report [28].

We give in Table 2 a set of rewriting rules for simplifying $\mathcal{C}_O$ expressions. Rules (1)-(3) are guided by the common examples found in real contracts, rules (4)-(5) are the usual contraction rules, and the rules (6)-(7) basically give the distributivity of conjunction over the exclusive disjunction.

To express CTDs we provide the following notation, $O_\varphi(\alpha)$, which is syntactic sugar for $O(\alpha) \wedge [\overline{\alpha}]\varphi$ stating the obligation to execute α, and the reparation φ in case the obligation is violated, i.e. α is not performed. The reparation may be any contract clause. Similarly, CTP statements $F_\varphi(\alpha)$ can be defined as $F_\varphi(\alpha) = F(\alpha) \wedge [\alpha]\varphi$, where φ is the penalty in case the prohibition is violated. Notice that it is possible to express nested CTDs and CTPs.

In $\mathcal{CL}$, we can write *conditional* obligations, permissions and prohibitions in two different ways. Just as an example let us consider conditional obligations. The first kind is represented as $[\alpha]O(\beta)$, which may be read as "after performing α, one is obliged to do β". The second kind is modeled using the test operator ?: $[\varphi?]O(\alpha)$, representing "If φ holds then one is obliged to perform α". Similarly for permission and prohibition.

3 The Underlying Logic for the Contract Language

3.1 Yet Another Propositional μ-Calculus

We take the classical propositional μ-calculus as defined by Kozen [13] and we extend it with concurrent actions and special propositional constants. We call this extension $\mathcal{C}\mu$. We consider a special set $\mathcal{L}$, which we call *atomic actions* and denote by $a, b, c, \ldots$. We add a set of propositional constants which we denote by P_c. To capture true concurrency we extend the set $\mathcal{L}$ with *concurrent sets* which are *finite subsets of atomic actions* with the intuitive meaning that all the atomic actions inside a concurrent set are executed concurrently (at the same time).

Definition 2 (concurrent sets). *A concurrent action set, denoted by γ (possibly indexed), is a finite subset of the set of atomic actions $\mathcal{L}$, $\gamma = \{a_1, \ldots, a_n\}$ where $a_i \in \mathcal{L}$. The concurrent sets $\gamma \in 2^{\mathcal{L}}$ are the labels of $\mathcal{C}\mu$.*

The *syntax* of $\mathcal{C}\mu$ is given by:

$$\varphi \;:=\; P \mid Z \mid P_c \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\gamma]\varphi \mid \nu Z.\varphi(Z)$$

where P represents *propositional variables*, Z represents *state variables*, $\top$ is the constant proposition denoting true, and $[\gamma]\varphi$ is the formula stating that after executing the concurrent set γ, φ holds. $\nu Z.\varphi(Z)$ is the greatest fix-point, and the other syntactic constructs come from propositional logic. The constant propositions are added in order be able to capture the deontic operators of $\mathcal{CL}$. P_c contains two distinguished kind of constants: *obligation constants* O_a and *prohibition constants* $\mathcal{F}_a$, which are uniquely indexed by the atomic actions $a \in \mathcal{L}$. The constant propositions are interpreted in the same way as the propositional variables of P as a set of states where the constant proposition holds. The intuition of the obligation constants is that when the system is in a state s and by action a it gets to a state t where O_a holds then we may conclude that in the state s the system has the obligation to execute action a. Similarly, $\mathcal{F}_a$ denotes the fact that action a is prohibited.

Note that $\mathcal{C}\mu$ includes the classical μ-calculus because if $\gamma = \{a\}$ then $[\gamma]\varphi \equiv [a]\varphi$, and P_c can be considered as a subset of P. We also have the usual dualities:

$$\varphi \vee \psi \overset{def}{=} \neg(\neg\varphi \wedge \neg\psi)$$
$$\langle\gamma\rangle\varphi \overset{def}{=} \neg[\gamma]\neg\varphi$$
$$\mu Z.\varphi(Z) \overset{def}{=} \neg\nu Z.\neg\varphi(\neg Z)$$

The interpretation of the above syntactic constructs follows the standard set-theoretical approach [13]. The formulas are interpreted over a structure denoted by $\mathcal{T}$. Given a set $Prop = P \cup P_c$ of propositions, and a set of atomic actions $\mathcal{L}$, $\mathcal{T} = (\mathcal{S}, R_{2^\mathcal{L}}, \mathcal{V}_{Prop}, \mathcal{V})$, where $\mathcal{S}$ is the set of states (worlds), $R_{2^\mathcal{L}} : 2^\mathcal{L} \to \mathcal{S} \times \mathcal{S}$ is the function assigning to each concurrent set γ of $2^\mathcal{L}$ a relation over $\mathcal{S}$ (i.e., $R_{2^\mathcal{L}}(\gamma) \subseteq \mathcal{S} \times \mathcal{S}$, $\gamma \in 2^\mathcal{L}$), $\mathcal{V}_{Prop} : Prop \to 2^\mathcal{S}$ is the interpretation of the propositions, and $\mathcal{V}$ is a valuation function assigning to each state variable a set of states. The valuation $\mathcal{V}[Z := S]$ maps variable Z to the states set S and in the rest it agrees with $\mathcal{V}$. For the sake of notation instead of $R_{2^\mathcal{L}}(\gamma)$ we write R_γ. The semantics of each syntactic construct of $\mathcal{C}\mu$ over a structure $\mathcal{T}$ is:

$$\|\top\|^\mathcal{T}_\mathcal{V} = \mathcal{S} \;\;;\;\; \|P\|^\mathcal{T}_\mathcal{V} = \mathcal{V}_{Prop}(P) \;\;;\;\; \|Z\|^\mathcal{T}_\mathcal{V} = \mathcal{V}(Z) \;\;;\;\; \|P_c\|^\mathcal{T}_\mathcal{V} = \mathcal{V}_{Prop}(P_c)$$
$$\|\neg\varphi\|^\mathcal{T}_\mathcal{V} = \mathcal{S} \setminus \|\varphi\|^\mathcal{T}_\mathcal{V}$$
$$\|\varphi \wedge \psi\|^\mathcal{T}_\mathcal{V} = \|\varphi\|^\mathcal{T}_\mathcal{V} \cap \|\psi\|^\mathcal{T}_\mathcal{V}$$
$$\|[\gamma]\varphi\|^\mathcal{T}_\mathcal{V} = \{s \mid \forall t \in \mathcal{S}.\ (s,t) \in R_\gamma \Rightarrow t \in \|\varphi\|^\mathcal{T}_\mathcal{V}\}$$
$$\|\nu Z.\varphi\|^\mathcal{T}_\mathcal{V} = \bigcup\{S \subseteq \mathcal{S} \mid S \subseteq \|\varphi\|^\mathcal{T}_{\mathcal{V}[Z:=S]}\}$$
$$\|\varphi \vee \psi\|^\mathcal{T}_\mathcal{V} = \|\varphi\|^\mathcal{T}_\mathcal{V} \cup \|\psi\|^\mathcal{T}_\mathcal{V}$$
$$\|\langle\gamma\rangle\varphi\|^\mathcal{T}_\mathcal{V} = \{s \mid \exists t \in \mathcal{S}.\ (s,t) \in R_\gamma \wedge t \in \|\varphi\|^\mathcal{T}_\mathcal{V}\}$$
$$\|\mu Z.\varphi\|^\mathcal{T}_\mathcal{V} = \bigcap\{S \subseteq \mathcal{S} \mid S \supseteq \|\varphi\|^\mathcal{T}_{\mathcal{V}[Z:=S]}\}$$

Note that $R_{2^\mathcal{L}}$ for singleton concurrent sets behaves the same as $R_\mathcal{L}$ for actions of μ-calculus. In this case, for the sake of brevity instead of $R_{\{a\}}$ we just write R_a. Also, we often use as shorthand for a concurrent set inside dynamic operators and we write $[a, b]\varphi$ instead of $[\{a, b\}]\varphi$. Furthermore, we have the following restriction for the constant propositions of the form $\mathcal{F}_a$ and O_a: *Constants $\mathcal{F}_a$ and O_a are incompatible, that is their interpretations as sets is disjoint:*

$$\|\mathcal{F}_a\|^\mathcal{T}_\mathcal{V} \cap \|O_a\|^\mathcal{T}_\mathcal{V} = \emptyset, \quad \forall a \in \mathcal{L}. \tag{1}$$

The intuition drawn from e-contracts is that it is not possible to be obliged to do something and at the same time be forbidden to do the same thing. The above description gives the following natural result: 1) $O_a \Rightarrow \neg\mathcal{F}_a$ and 2) $\mathcal{F}_a \Rightarrow \neg O_a$.

Action logics like PDL, and consequently propositional μ-calculus, are usually non-deterministic. From the point of view of modeling contracts it is natural to adopt a deterministic variant of an action logic because it does not make sense to specify different outcomes for the same action in a contract. The determinism of $\mathcal{C}\mu$ requires to have only one transition from one state labeled with a concurrent set. Formally we restrict $R_{2^\mathcal{L}}$ to assign to each concurrent set only partial

Table 3. The translation function $f^{\mathcal{T}}$ from $\mathcal{CL}$ to $\mathcal{C}\mu$

(1)	$f^{\mathcal{T}}(O(\&_{i=1}^{n} a_i))$	$=$	$\langle\{a_1,\ldots,a_n\}\rangle(\wedge_{i=1}^{n} O_{a_i})$
(2)	$f^{\mathcal{T}}(\mathcal{C}_O \oplus \mathcal{C}_O)$	$=$	$f^{\mathcal{T}}(\mathcal{C}_O) \wedge f^{\mathcal{T}}(\mathcal{C}_O)$
(3)	$f^{\mathcal{T}}(P(\&_{i=1}^{n} a_i))$	$=$	$\langle\{a_1,\ldots,a_n\}\rangle(\wedge_{i=1}^{n} \neg\mathcal{F}_{a_i})$
(4)	$f^{\mathcal{T}}(\mathcal{C}_P \oplus \mathcal{C}_P)$	$=$	$f^{\mathcal{T}}(\mathcal{C}_P) \wedge f^{\mathcal{T}}(\mathcal{C}_P)$
(5)	$f^{\mathcal{T}}(F(\&_{i=1}^{n} a_i))$	$=$	$[\{a_1,\ldots,a_n\}](\wedge_{i=1}^{n} \mathcal{F}_{a_i})$
(6)	$f^{\mathcal{T}}(F(\delta) \vee [\beta]F(\delta))$	$=$	$f^{\mathcal{T}}(F(\delta)) \vee f^{\mathcal{T}}([\beta]F(\delta))$
(7)	$f^{\mathcal{T}}(\mathcal{C}_1 \wedge \mathcal{C}_2)$	$=$	$f^{\mathcal{T}}(\mathcal{C}_1) \wedge f^{\mathcal{T}}(\mathcal{C}_2)$
(8)	$f^{\mathcal{T}}(\bigcirc\mathcal{C})$	$=$	$[\mathbf{any}]f^{\mathcal{T}}(\mathcal{C})$
(9)	$f^{\mathcal{T}}(\mathcal{C}_1 \mathcal{U} \mathcal{C}_2)$	$=$	$\mu Z.f^{\mathcal{T}}(\mathcal{C}_2) \vee (f^{\mathcal{T}}(\mathcal{C}_1) \wedge [\mathbf{any}]Z \wedge \langle\mathbf{any}\rangle\top)$
(10)	$f^{\mathcal{T}}([\&_{i=1}^{n} a_i]\mathcal{C})$	$=$	$[\{a_1,\ldots,a_n\}]f^{\mathcal{T}}(\mathcal{C})$
(11)	$f^{\mathcal{T}}([(\&_{i=1}^{n} a_i)\alpha]\mathcal{C})$	$=$	$[\{a_1,\ldots,a_n\}]f^{\mathcal{T}}([\alpha]\mathcal{C})$
(12)	$f^{\mathcal{T}}([\alpha+\beta]\mathcal{C})$	$=$	$f^{\mathcal{T}}([\alpha]\mathcal{C}) \wedge f^{\mathcal{T}}([\beta]\mathcal{C})$
(13)	$f^{\mathcal{T}}([\varphi?]\mathcal{C})$	$=$	$f^{\mathcal{T}}(\varphi) \Rightarrow f^{\mathcal{T}}(\mathcal{C})$

functions (not relations), i.e., for any $(s,t),(s,t') \in R_\gamma$ then $t = t'$. Naturally a compound action may have several ending worlds, both in the interpretation of the *actions as relations* [9] or the *actions as trajectories* [26]. Note that $(s,t) \in R_a$ and $(s,t') \in R_{\{a,b\}}$ does not introduce non-determinism.

3.2 Translating the Language into the Logic

Because of the special status of the concurrent actions, the compositionality rules of Table 1, and the rewriting rules of Table 2, we choose to translate O, P, and F over both atomic actions a and concurrent actions $a\&b$. We also need to translate the $\oplus$ over obligation and permission as well as the $\vee$ operator over prohibition.

We consider a translation function $f^{\mathcal{T}}$ from expressions of $\mathcal{CL}$ into formulas of $\mathcal{C}\mu$. In Table 3 lines (1)-(6) we give the translation of the basic deontic constructs of $\mathcal{CL}$ ($\mathcal{C}_O, \mathcal{C}_P$ and $\mathcal{C}_F$). Note that the translation of concurrent actions $a\&b$ uses concurrent sets and we use a concise notation which, for example, for atomic actions under O would give $f^{\mathcal{T}}(O(a)) = \langle a\rangle O_a$ —we abuse the notation and denote the atomic actions as conjunction over only one a_i. Lines (7)-(13) show the translation of the other $\mathcal{CL}$ expressions, where **any** is the special action which is interpreted as the union of all actions in $\mathcal{L}$ with the intuition of *doing any action.* The conjunction is translated as the corresponding conjunction operator of $\mathcal{C}\mu$, *next* $\bigcirc$ uses the action **any**, and *until* $\mathcal{U}$ is translated using a fix-point expression as usual. We give separate translations for each compound action inside the *dynamic box* operator of $\mathcal{CL}$. The translation is similar to the translation of PDL into μ-calculus.

Note that with this translation one cannot give a truth value to an obligation $O(a)$ of an action (or a permition or prohibition), because the truth value of its translation $\langle a\rangle O_a$ can be determined only after the execution of the action. This is in accordance with the classical semantics of the deontic modalities [33].

4 Properties of the Contract Language

In this section we show some of the properties $\mathcal{CL}$ enjoys, as well as how the language avoids most of the important deontic paradoxes and the undesirable properties listed in Section 2.2. Most of the proofs are omitted and can be found in [28].

Proposition 1 ensures that it is not needed to use negation on deontic operators, while Proposition 2 establishes the standard relation between obligations and permissions.

Proposition 1. *The following statements are valid in $\mathcal{CL}$:*

a) $P(\alpha) \equiv \neg F(\alpha)$
b) $F(\alpha) \equiv \neg P(\alpha)$.

Proof: The proof follows from the translation of $P(\alpha)$ and $F(\alpha)$ into the logic and the duality between the μ-calculus operators $[\cdot]$ and $\langle\cdot\rangle$. $\square$

Proposition 2. *The following statement is valid in $\mathcal{CL}$: $O(\alpha) \Rightarrow P(\alpha)$.*

Proof: The proof follows from the translations of $O(\alpha)$ and $P(\alpha)$ into the logic. Moreover, the proof makes use of the equation (1) of the incompatibility of O_a and $\mathcal{F}_a$ constants. $\square$

The following two results express that $\mathcal{CL}$ does not allow the derivation of certain undesirable properties.

Proposition 3. *The following implications do not hold in $\mathcal{CL}$:*

a) $P(a) \Rightarrow P(a\&b)$
b) $F(a) \Rightarrow F(a\&b)$.

Proof: We give a counter example to show that the implication is not possible, i.e., we give a model in the logic which is a model for the translation of the first $\mathcal{CL}$ formula and is not a model for the translation of the second $\mathcal{CL}$ formula.

For a) consider $(s,t) \in R_a$ and $(s,t') \in R_{\{a,b\}}$ with $t \notin \|\mathcal{F}_a\|_{\mathcal{V}}^{\mathcal{T}}$ and $t' \in \|\mathcal{F}_a\|_{\mathcal{V}}^{\mathcal{T}} \cap \|\mathcal{F}_b\|_{\mathcal{V}}^{\mathcal{T}}$. Consider the model M which has states $\mathcal{S} = \{s,t,t'\}$ and two relations: for action a, $R_a = \{(s,t)\}$ and for action $\{a,b\}$, $R_{\{a,b\}} = \{(s,t')\}$. M is a model for the first formula but is not a model of the second formula.

For b) we change the above model such that $t \in \|\mathcal{F}_a\|_{\mathcal{V}}^{\mathcal{T}}$ and $t' \notin \|\mathcal{F}_a\|_{\mathcal{V}}^{\mathcal{T}}$. M is a model of the first formula but is not a model for the second formula. $\square$

Proposition 4. *The following implications do not hold in $\mathcal{CL}$:*

a) $F(a\&b) \Rightarrow F(a)$
b) $P(a\&b) \Rightarrow P(a)$.

Proof: The proof is similar to the proposition above. □

The following proposition expresses that the most important paradoxes of deontic logic are avoided in our contract language, either because there are not expressible in the language or because they are simply excluded by the translation into the underlying logic.

Proposition 5. *The following paradoxes are avoided in $\mathcal{CL}$:*

- *Ross's paradox*
- *The Free Choice Permission paradox*
- *Sartre's dilemma*
- *The Good Samaritan paradox.*
- *Chisholm's paradox*
- *The Gentle Murderer paradox*

5 Example

We formalize here the example introduced in Section 2.1. As part of the formalization of a contract in $\mathcal{CL}$ we first have to define the assertions and actions:

ϕ = the Internet traffic is high
p = client pays x \$
d = client delays payment
n = client notifies by e-mail
l = client lowers the Internet traffic
s = provider provides the service as stipulated in the contract
c = provider cancels the contract
e = provider sends a written notification to the client by e-mail
w = provider sends a written notification to the client by normal post

The five clauses of the example are written in $\mathcal{CL}$ as follows:

1. $\Box(\phi \Rightarrow O(p + (d \& n)))$
2. $\Box([d, n](O(l) \wedge [l]\Diamond(O(p) \wedge [p]O(p))))$
3. $\Box([\{d, n\} \cdot \overline{l}\,]\Diamond(O(p) \wedge [p]O(p) \wedge [p \cdot p]O(p)))$[5]
4. $\Box(F(c) \wedge [w, e]P(c))$
5. $\Box O(s)$.

Remarks: 1) Formulas 2. and 3. are rather long because we can not represent in $\mathcal{CL}$ quantitative information like *pay two times*. We could use the & operator over actions with the same intuition as in logics of resources (e.g. linear logic [10]) and for *obliged to pay twice* we could write in $\mathcal{CL}$ $O(p\&p)$ instead of $O(p) \wedge [p]O(p)$ which is more concise and natural.

[5] The formulas 2 and 3 may be combined in a single formula using CTDs: $\Box([d, n](O_\varphi(l) \wedge [l]\Diamond(O(p) \wedge [p]O(p))$ where $\varphi = O(p) \wedge [p]O(p) \wedge [p \cdot p]O(p)$.

2) Though it is not apparent at first sight the contract allows the client to go from low to high Internet traffic many times and pay the penalty ($2 * x$ \$) only once.[6] The problem is that after the client lowers the Internet traffic, he might get a high traffic again and postpone the payment till a future moment. To avoid this situation we should add a clause specifying that "after getting a high Internet traffic, if the client postpones the payment then he can get a high traffic again only after having paid". In $\mathcal{CL}$ this might be expressed by changing formulas 2 and 3 above as:

2' $\Box([d,n](O(l) \wedge [l]\neg\phi\,\mathcal{U}\,(O(p) \wedge [p]O(p)))$
3' $\Box([\{d,n\} \cdot \; \bar{l}\;](\neg\phi\,\mathcal{U}\,(O(p) \wedge [p]O(p) \wedge [p \cdot p]O(p)))$.

This example shows the importance of being able to model check a contract, which may be done only if the contract is written in a formal language, e.g. $\mathcal{CL}$.

3) Notice that our contract language lacks the possibility of expressing time constraints. More involved clauses like *the client must pay within 7 days*, or *the client is forbidden to pass more than 10 times per month from low to high Internet traffic*, can only be expressed here by introducing time, special variables and simulate a counter. For model checking purposes we would like to include the possibility to express these properties directly in the logic and an extension with real-time would be desirable.

6 Conclusion

In this paper we have presented a formal language for writing contracts, and have provided a formal semantics through the translation of the language into a variant of the propositional μ-calculus extended with concurrent actions. The use of a variant of the μ-calculus as a semantic framework for our language is not casual. The logic has nice properties: it is decidable [15], has a complete axiomatic system [32], and a complete Gentzen-style proof system [31]. Our language avoids most of the classical paradoxes, and enjoys all the nice properties listed in Section 2.2. To our knowledge no other work in the field has achieved such goals. Given that our application domain is that of electronic contracts, we have also given arguments for restricting syntactically and semantically certain uses of (and relations between) obligations, permissions and prohibitions, usually considered in philosophical and logical discussions.

Related Work: There are currently several different approaches aiming at defining a formal language for contracts. Some works concentrate on the definition of contract taxonomies [1,2,30], while others look for formalizations based on logics (e.g. classical [8], modal [7], deontic [12,22] and defeasible logic [11,29]). Other formalizations are based on models of computation (e.g. FSMs [21] and Petri Nets [6]). None of the above has reached enough maturity as to be considered

[6] See the technical report [28] for a more detailed explanation.

the solution to the problems of formal definition of contracts. Some provide a good framework for monitoring but lack a formal semantics and a reasoning system; others have nice proof systems and model theory, but not mechanism for monitoring or negotiation; many of the deontic-based approaches put too much emphasis on the logical properties and neglect the practical side, including monitoring. None of them captures all the intuitive properties of e-contracts we have described, while avoiding the most important paradoxes.

The idea of using a propositional constant in an action-based logic for giving semantics to the deontic notions was first presented in [19], where the special constant V (corresponding to our $\mathcal{F}_a$) was added to denote an "undesirable state-of-affairs" in the current state. We have, in addition, the constants O_a which are used to define obligation not in terms of action negation but using the diamond modal operator, deviating from other approaches (e.g., [4,19]).

Our work is closely related to those based on logic, and in particular to [5]. Due to lack of space and since part of the motivation of our work is to overcome some of the problems of the approach of Broersen *et al* in [5], we contrast our approach in detail only w.r.t. this paper. Broersen *et al* introduce a very interesting characterization of obligation, permission and prohibition by following an *ought-to-do* approach based on a deontic logic of regular actions. The idea is to use the μ^a-calculus as a basis and then define obligation, permission and prohibition over regular expressions on actions. The main differences w.r.t. our approach are the following. (a) There is no notion of *contract language*, only characterization of obligation, permission and prohibition in the logic. (b) The only deontic primitive is permission over atomic actions; obligation is defined as an infinite conjunction of negation of permission over actions not in the scope of the negation. We avoid this infinite conjunction by defining both prohibition and obligation as primitive (and using the propositional constants O_a and $\mathcal{F}_a$ at the semantic level) and prohibition as negation of permission. (c) All the deontic operators are defined over regular actions, including the Kleene star. We consider it is not natural to have starred actions under the deontic notions, we have thus dropped it. (d) Obligation on the choice of actions is not compositional; it is compositional in our case. (e) There is no conjunction over actions, i.e., it is not possible to express concurrent actions, which is the case in our approach. (f) The approach uses disjunction over actions. We have decided to use the exclusive or instead. (g) Negation on actions (meaning "not performing an action") is defined as a complement of the (infinite) set of actions. In our case the set of actions is finite, at the language level. (h) CTDs cannot be defined unless an extension of the μ^a-calculus is considered. In our setting both CTDs and CTPs are easily defined. (i) The semantics of obligation, permission and prohibition is given in terms of properties over traces, instead of over an extension of the Kripke structure as in our case.

For a nice overview of the history, problems and different approaches on deontic logic see [34]. The chapter of McNamara in the Handbook of the History of Logic contains a general description of the topic, mainly the different paradoxes arising under SDL [20]. For a discussion on CTDs see [24] and references therein.

Future Work: Our work is a first step towards a more ambitious task, and we believe the formalism chosen will allow us to achieve the following goals. The first extension is to add real-time to be able to express and reason about contracts with deadlines. Other immediate extension is the syntactic distinction in the signature of the definition part of $\mathcal{CL}$ between subjects, proper actions and objects. This would permit to make queries (and model check properties) for instance about all the rights and obligations of a given subject, or determine under which conditions somebody is obliged/forbidden to perform something. We have not considered in this paper the problem of negotiation nor monitoring of contracts. We believe these are important features of a contract language which must be taken into account in future work. Concerning actions, we got inspiration from the works on dynamic logics [25]. We would like to deepen the study of the action algebra to make the distinction between the intuitive meaning of conjunction under obligation, permission and prohibition. Further investigation is also needed to characterize negation on actions, both for capturing and distinguishing the ideas of "not doing something" and "doing something but a given action", which are not differentiated in our current approach. We want to explore the proof system of the $\mathcal{C}\mu$ logic, and to extend existing model checkers for μ-calculus [3,18] to analyze contracts as mentioned in the remarks of our example.

References

1. Aagedal, J.: Quality of Service Support in Development of Distributed Systems. PhD thesis, Dept. of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo (2001)
2. Beugnard, A., Jézéquel, J.M., Plouzeau, N.: Making components contract aware. IEEE 32, 38–45 (1999)
3. Biere, A.: mu-cke - efficient mu-calculus model checking. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 468–471. Springer, Heidelberg (1997)
4. Broersen, J.: Action negation and alternative reductions for dynamic deontic logics. J. Applied Logic 2, 153–168 (2004)
5. Broersen, J., Wieringa, R., Meyer, J.J.C.: A fixed-point characterization of a deontic logic of regular action. Fundam. Inf. 48, 107–128 (2001)
6. Daskalopulu, A.: Model Checking Contractual Protocols. In: Breuker, L.R., Leenes, R., Winkels, R. (eds.) Legal Knowledge and Information Systems, JURIX 2000. Frontiers in Artificial Intelligence and Applications, vol. 48, pp. 35–47. IOS Press, Amsterdam, Trento, Italy (2000)
7. Daskalopulu, A., Maibaum, T.S.E.: Towards Electronic Contract Performance. In: Legal Information Systems Applications, 12th International Conference and Workshop on Database and Expert Systems Applications, pp. 771–777. IEEE, NJ, New York (2001)
8. Davulcu, H., Kifer, M., Ramakrishnan, I.V.: CTR-S: A Logic for Specifying Contracts in Semantic Web Services. In: WWW04. pp. 144–153 (2004)
9. Fischer, M.J., Ladner, R.E.: Propositional modal logic of programs. In: STOC'77, pp. 286–294. ACM Press, New York (1977)
10. Girard, J.Y.: Linear logic. Theor. Compu. Sci. 50, 1–102 (1987)
11. Governatori, G.: Representing business contracts in RuleML. International Journal of Cooperative Information Systems 14, 181–216 (2005)

12. Governatori, G., Rotolo, A.: Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations. Australian Journal of Logic 4, 193–215 (2006)
13. Kozen, D.: Results on the propositional mu-calculus. Theor. Comput. Sci. 27, 333–354 (1983)
14. Kozen, D.: Kleene algebra with tests. ACM Transactions on Programming Languages and Systems (TOPLAS'97) 19, 427–443 (1997)
15. Kozen, D., Parikh, R.: A decision procedure for the propositional μ-calculus. In: Clarke, E.M., Kozen, D. (eds.) 4th Workshop on Logics of Programs. LNCS, vol. 164, pp. 313–325. Springer, Heidelberg (1983)
16. Kozen, D.: On kleene algebras and closed semirings. In: Rovan, B. (ed.) Mathematical Foundations of Computer Science 1990. LNCS, vol. 452, pp. 26–47. Springer, Heidelberg (1990)
17. Mally, E.: Grundgesetze des Sollens. Elemente fer Logik des Willens. Graz: Leuschner & Lubensky (1926)
18. Mateescu, R., Sighireanu, M.: Efficient on-the-fly model-checking for regular alternation-free μ-calculus. Sci. Comp. Program. 46(3), 255–281 (2003)
19. Meyer, J.J.C.: A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. Notre Dame Journal of Formal Logic 29, 109–136 (1988)
20. McNamara, P.: Deontic logic. In: Gabbay, D.M., Woods, J. (eds.) Handbook of the History of Logic, vol. 7, pp. 197–289. North-Holland Publishing, Amsterdam (2006)
21. Molina-Jimenez, C., Shrivastava, S., Solaiman, E., Warne, J.: Run-time Monitoring and Enforcement of Electronic Contracts. Electronic Commerce Research and Applications 3, 108–125 (2004)
22. Paschke, A., Dietrich, J., Kuhla, K.: A Logic Based SLA Management Framework. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, Springer, Heidelberg (2005)
23. Pnueli, A.: Temporal logic of programs. In: FOCS'77, pp. 46–57. IEEE, NJ, New York (1977)
24. Prakken, H., Sergot, M.: Contrary-to-duty obligations. Studia Logica 57, 91–115 (1996)
25. Pratt, V.R.: Semantical considerations on floyd-hoare logic. In: FOCS'76, pp. 109–121. IEEE, NJ, New York (1976)
26. Pratt, V.R.: A practical decision method for propositional dynamic logic: Preliminary report. In: STOC'78, pp. 326–337. ACM Press, New York (1978)
27. Pratt, V.R.: Dynamic algebras and the nature of induction. In: STOC'80, pp. 22–28. ACM Press, New York (1980)
28. Prisacariu, C., Schneider, G.: Towards a formal definition of electronic contracts. Technical report 348, Department of Informatics, University of Oslo (2007)
29. Song, I., Governatori, G.: Nested rules in defeasible logic. In: Adi, A., Stoutenburg, S., Tabet, S. (eds.) RuleML 2005. LNCS, vol. 3791, pp. 204–208. Springer, Heidelberg (2005)
30. Tosic, V.: On Comprehensive Contractual Descriptions of Web Services. In: IEEE International Conference on e-Technology, e-Commerce, and e-Service, pp. 444–449. IEEE, NJ, New York (2005)
31. Walukiewicz, I.: A Complete Deductive System for the μ-Calculus. PhD thesis, Warsaw University (1993)
32. Walukiewicz, I.: Completeness of Kozen's axiomatisation of the propositional μ-calculus. In: LICS'95, pp. 14–24. IEEE, NJ, New York (1995)
33. Wright, G.H.V.: Deontic logic. Mind 60, 1–15 (1951)
34. Wright, G.H.V.: Deontic logic: A personal view. Ratio Juris 12, 26–38 (1999)