UNIVERSITY OF OSLO
Department of Informatics

Towards a Formal Definition of Electronic Contracts[1]

Research Report No. 348

Cristian Prisacariu

Gerardo Schneider

Isbn 82-7368-305-2
Issn 0806-3036

January 2007

Towards a Formal Definition of Electronic Contracts[†]

Cristian Prisacariu[‡] Gerardo Schneider[§]

January 2007

Abstract

In this paper we propose a formal language for writing electronic contracts, based on the normative deontic notions of obligation, prohibition, and permission. We take an ought-to-do approach, where the above notions 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 obligation, prohibition and permission, and to express deterministic and concurrent actions. We provide a translation of the contract language into the logic, and we show how the semantics faithfully captures the meaning of the contract language. 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 also discuss informally the main problems in formalizing the above normative deontic notions in particular in the context of electronic contracts. We finally show its applicability on a contract example.

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

[‡]Dept. of Informatics – Univ. of Oslo, P.O. Box 1080 Blindern, N-0316 Oslo, Norway. E-mail: cristi@ifi.uio.no

[§]Dept. of Informatics – Univ. of Oslo, P.O. Box 1080 Blindern, N-0316 Oslo, Norway. E-mail: gerardo@ifi.uio.no

Contents

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 stating penalties in case of contract violations. In the case of a bilateral contract, one usually talks about the roles of *service provider* and *service consumer*; but multi-lateral contracts are also possible where the participants may play other roles. 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 security and quality of service (QoS).

Before a contract is signed it has to go first through a stage of *negotiation*. At this stage, the contract template offered by the service provider has to be analyzed (e.g. by model checking techniques) and changed to suite the needs of both the client and the provider. After each change the new contract is sent to the other party which either accepts it or changes it again. This process goes on until an agreement is achieved.

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 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.

Formal Approaches for Contracts. There are currently several different approaches aiming at defining a formal language for contracts. Some works concentrate on the definition of contract taxonomies [Aag01, BJP99, TP05], while others look for formalizations based on logics (e.g. classical [DKR04], modal [DM01], deontic [GR06, PDK05] or defeasible logic [Gov05, SG05]). Other formalizations are based on models of computation

(e.g. FSMs [MJSSW04] and Petri Nets [Das00]).

In our opinion, the most promising approach is the one based on logic. A logic for contracts not necessarily has to be based on, or extend, deontic logic, but must contain normative deontic notions (obligation, permission, and prohibition) and preserve their intuitive properties, both in the proof system and in its model theory.

Deontic Logic. Formalizing the usual 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 [Mal26][1]. These works have obviously been done much before the concrete problem of defining electronic contracts (e-contracts) and the problems identified still continue to challenge philosophers, logicians and computer scientists.

In early papers (e.g. [Wri51]) the approach was to relate the normative notions of obligation, permission and prohibition in a similar way as the quantifiers (all, some, no) and modalities (necessary, possible, impossible) of classical and modal logic, respectively. This was the bases of the so-called Standard Deontic Logic (SDL) which is built on classical propositional logic, leading to a nice formalization but also to many paradoxes.

One of the first issues to take into account before 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 an *ought-to-do* and the latter as *ought-to-be*. The following is a classical example where "One ought to build a window" can be understood as an ought-to-do sentence, while "There ought to be a window" is an ought-to-be sentence. In many cases it is possible to translate an ought-to-be sentence into its corresponding ought-to-do quite easily, as in the following example: "It ought to be the case that John pays the money to Smith" (ought-to-be) and "John ought to pay the money to Smith" (ought-to-do). In many e-contracts it is more natural to find *ought-to-do* statements; 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 be also cases where an ought-to-be approach gives a more concise expression, like in QoS contracts where we may have statements that express quantitative restrictions like: *The average bandwidth should be more than 20kb/s*. The discussion among philosophers and logicians is far from an end in what concerns the decision of whether one approach is better

[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. Avon Wright [Wri51].

than the other, or even if both should coexist in the same reasoning system. Some authors have oscillated from one side to the other – Avon Wright for instance took an ought-to-be approach in early papers, and later inclined for the ought-to-do (action-based) approach, as stated in [Wri99].

Note that norms (and clauses in contracts), by definition, are violable (if we have the guarantee that nobody will violate the norms, normative systems would be completely useless). Hence, *contrary-to-duty obligations* (or CTDs) and *contrary-to-prohibitions* (or CTPs), concerning the fact that obligations might not be fulfilled and that prohibitions might be violated, are important aspects to be considered. In both cases, we might want to know which is the *reparation* or the *penalty* to be applied. See for instance [PS96] for a discussion on CTDs.

There are many other problems to be considered when formalizing obligation, permission and prohibition. Among others, 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".

The intention of this section is to give an overview of the main problems in deontic logic, and not to discuss the different solutions. See [Wri99] for a nice overview of the history, problems and different approaches on deontic logic. The entry "Deontic Logic" of the Stanford Encyclopedia of Philosophy contains a general description of the topic, mainly the different paradoxes arising under SDL[2]. See also the chapter of McNamara in the Handbook of the History of Logic [McN06].

Our Approach and Contributions. The above discussion should not give the impression that we are trying to solve an old unsolvable problem. We are mainly concerned with formal definition of contracts, and more precisely, of *e-contracts*. By narrowing the scope of application of deontic logic, we are definitely on a terrain where many of the philosophical problems of the logic are not present.

In this paper we take a first step towards the definition of a formal contract language, based on an extension of the μ-calculus. Our starting point is [BWM01], where a fix-point characterization of obligation, permission and prohibition is given, based on the modal μ-calculus. The logic allows to express obligation, permission and prohibition on regular actions, taking thus an *out-to-do* approach.

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

[2] `http://plato.stanford.edu/entries/logic-deontic/index.html`.

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. It is possible to express CTDs and CTPs;
5. 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 insight on how they should be related under 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 present an informal discussion about deontic logic, and the main problems arising when formalizing the notions of obligation, permission and prohibition. In Section 3 we present the most well-known paradoxes as well as a new one we found under certain different interpretation of the normative deontic notions. Based on the two previous sections we present a list of desirable properties for a contract language, in Section 4. In Section 5 we present our formal language for writing contracts, and in Section 6 we present a variant of the μ-calculus, with its syntax and semantics, and we give a translation of the language into the logic. In Section 7 we show that our language avoids the most important paradoxes, and that it satisfies most of the desirable properties described in Section 4. In Section 8 we present an example of a contract written in our language. We briefly describe a related approach also based on a variant of the μ-calculus [BWM01] in Section 9 and we discuss the advantages and disadvantages of the approach in contrast to ours. We conclude in Section 10.

2 Obligation, Permission and Prohibition: Informal Discussion

Capturing the right intuition of normative notions in general, and in particular of obligation, permission and prohibition, is a difficult task. We present in this section an informal discussion about the main ideas to take into account when trying to formalize the above notions. In what follows we use $O(a)$ to denote the obligation of performing a given action a, similarly for permission ($P(a)$) and prohibition ($F(a)$), and $+$ for *choice* among actions. A more precise definition will be given later.

2.1 On the Truth-Value and the Notion of Consistency in Deontic Logic

This section is entirely based on [Wri99]. In the philosophical tradition of Avon Wright's education, norms were seen as subjective, relative and dependent on culture, without any truth-value: "norms, as prescriptions for conduct, simply *are not* true or false" [Wri99]. The apparent problem here is that if one takes this point of view, then it is not possible to study the logical relation between obligation, permission and prohibition, to define a notion of logical consequence or to detect contradictions. Von Wright argues that the above only implies that logic is much more than truth and thus norms are still subject to logical laws. Von Wright makes a difference between *prescriptive* and *descriptive* sentences. In the former the sentence does not have a truth-value, it only enunciates a norm, while in the latter it has a truth-value (it is a *norm-proposition*). In its descriptive interpretation of formulas, deontic logic should aim at a complete and contradiction-free system of norms. Von Wright makes a clear distinction between "ought", *the obligation*, and "must", *the practical necessity*. The first is neither true nor false and it is an ought-to-be, while the second can be true or false depending on the situation and is thus related to something which has to be done (ought-to-do).

Von Wright claims that "a set of norms is consistent if and only if, the conjunction of *all* states pronounced obligatory by the norms with *any one* of the states pronounced permitted is a doable state of affairs, i.e., something which can be achieved through human action." Along these lines, it is possible to define the notion of *normative entailment*: a consistent set of norms entails another one if and only if adding the negation of the latter makes the set inconsistent.

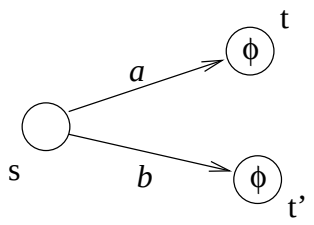

Figure 1: Example of a model for $\langle a\rangle\phi \wedge \langle b\rangle\phi$ but not for $\langle a\&b\rangle\phi$.

2.2 Conjunction in Action Logics

Before explaining why conjunction is problematic when combined with deontic operators, we start by showing some problems when trying to add conjunction to Propositional Dynamic Logic (PDL). If we want to define $\langle a\&b\rangle\phi$ compositionally, it is natural to think that it can be defined as follows:

$$\langle a\&b\rangle\phi = \langle a\rangle\phi \wedge \langle b\rangle\phi.$$

If actions a and b are interpreted as sets of pairs of states (i.e. relations over states) and if conjunction over actions $a\&b$ is interpreted as intersection of sets [BV03] then in PDL extended with action conjunction (denoted as $PDL^{\cap}$) it holds only that $\langle a\&b\rangle\phi \Rightarrow \langle a\rangle\phi \wedge \langle b\rangle\phi$. The converse implication does not hold in $PDL^{\cap}$ because the left side means that there exists a state, say t to which the system may get by performing action a and also by performing action b and the formula ϕ holds in t. On the other hand, the right side means that there exists a state t to which one may get by performing action a and there exists another state t' to which one may get by performing action b, and in both t and t', ϕ holds; but t and t' may be different. Because of these the right side does not imply the left side. Consider the model in Figure 1 which is a model for the formula on the right of the implication but is not a model for the formula on the left of the implication because it does not exist a state to which the system can get by performing both actions a and b.

One solution to the above problem is not to define $\langle\cdot\rangle$ and $[\cdot]$ on conjunction of actions, but to axiomatize the logic giving the desirable properties [BV03]. Another solution is to enhance the logic with *nominals* as in hybrid logics (see for instance [AtC06] and reference therein). Hybrid logics define, besides the sort of propositional variables, a new sort of special propositions called nominals NOM $= \{i, j, k, \ldots\}$ disjoint from the set of propositional variables. The intent of the nominals is to *name states* of a model. The naming of the states is possible because each nominal holds in only one state

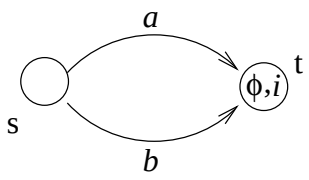

Figure 2: Example of a model for both $\langle a\rangle\phi \wedge \langle b\rangle\phi$ and $\langle a\&b\rangle\phi$.

of the model (i.e. if a nominal i holds in the state s of the model then it is said that the state has the name i; also there can not be another state s' with the same name i). Given a current state, if i is the name of a successor state, then we could write:

$$\langle a\rangle(i \wedge \phi) \wedge \langle b\rangle(i \wedge \phi) \Rightarrow \langle a\&b\rangle(i \wedge \phi),$$

which would force the transitions to have the same source and target states. A model for both formulas on the left and right of the implication arrow is pictured in Figure 2. This, however, does not force the two actions to be performed concurrently. In order to capture *true concurrency* we would need to force having only *one* transition labeled with a and b in an atomic way; we will see a solution in Section 6. An extension of PDL with nominals was first presented in [PT85] (see also [PT91]).

2.3 On the Relationship Between Obligation and Permission

The relation between obligation and permission is rather cumbersome. There is no consensus on how to relate these two notions or if it is possible (or more precisely, natural) to express one in terms of the other. Many researchers argue for defining permission as derived from obligation (or vice-versa): $O(a) \equiv \neg P(\overline{a})$. In [Wri99], von Wright argues for not using the above definition, though he introduced it in his early works; he proposes instead the following two equivalences: $\neg O(a) \equiv P(\overline{a})$ and $O(\overline{a}) \equiv \neg P(a)$.

We claim that none of the above equivalences are natural, at least for our purpose in trying to define a logic for formalizing e-contracts.

First notice that not being obliged to do something does not add any knowledge about what is permitted. Furthermore, in the context of a logic for contracts it does not make much sense to talk about negation of obligations: a contract must specify your rights and obligations, not what you are not obliged to do. Thus the first equivalence above can be discarded. Furthermore, we do not accept the implication $\neg P(a) \Rightarrow O(\overline{a})$ because it is

not natural to infer from not being permitted an action (or equivalently, the action is prohibited) that it is obligatory to perform the negated action. On the other hand, $O(\overline{a}) \Rightarrow \neg P(a)$ might be reasonable only on systems where the presence of $O(a)$ and $O(\overline{a})$ make the system inconsistent. Not everybody agrees on such inconsistency, so we do not consider it in a first instance.

In our opinion the only natural relations between obligation and permission are the following:

$$\begin{array}{l} O(a) \Rightarrow P(a) \\ O(a) \Rightarrow \neg P(\overline{a}) \end{array} \tag{1}$$

where the second implication only holds if there is no contrary-to-duty obligation (CTD) associated with $O(a)$, in which case one *must* take the reparation in case the obligation is not fulfilled.

2.4 Obligations and Permissions

2.4.1 About Disjunction of Actions

We first make a remark about obligation over disjunction of actions. Many papers use the notation $O(a \cup b)$ for obligation of "disjunction" of actions, while in fact they mean "choice", or "exclusive or". Indeed, it does not seem very intuitive to define obligation of classical disjunction of actions, since this is not the usual meaning in natural languages. We will, thus, use the notation $a + b$ for the choice of actions.

We want to define $O(a+b)$ compositionally while avoiding the Ross paradox. In order to do so, we need to have a hierarchical definition of formulas and not allow the $\vee$ on obligation formulas. Instead we add a *XOR* operator ($\oplus$) over obligation formulas to represent the intuitive idea of choice[3]. In this way we have the intuitive meaning of the obligation of a choice:

$$O(a + b) = O(a) \oplus O(b).$$

Many of the problems associated with the choice disappear as soon as a temporal aspect is introduced [PS96], as for example in "You must pay on time or at least give a notice 10 days before the paying date. If you don't pay on time and you don't give notice, you must pay a fine of 1000$".

[3]This operator is not new to logics: it can be defined in classical propositional logic and also has special properties in linear logic.

2.4.2 About Conjunction of Actions

We would like to be able to express obligation of performing concurrent actions, $O(a\&b)$. There are two solutions to do this: (1) using interleaving, and (2) having true concurrency. True concurrency would capture the idea that $O(a) \wedge O(b) \Rightarrow O(a\&b)$. We will propose later a solution based on sets of actions to capture concurrent actions in the logic

Another important aspect to take into account is the difference between permission and obligation over conjunction of actions. Saying that "you are obliged to remain silent and to talk with your lawyer" introduces an inconsistency since there is a requirement to do two contradictory actions. On the other hand, to say that "you have the right to remain silent and to talk with your lawyer" does not introduces any inconsistency. This shows that there is a clear difference between permission and obligation of conjunction of actions. We believe the discussion about the differences between conjunction under permission and under obligations is constructive and sheds some light on problems not always considered by many researchers.

We consider now the problem of understanding $\neg O(a\&b)$[4]. We will give here three different interpretations. We then justify the intuitive solution, and then explain how we can get the right solution by making a distinction between the conjunction of actions under the scope of obligations and permissions.

1. By defining $\neg O(a) = P(\overline{a})$, we get (by applying De Morgan law and the equivalence $O(a\&b) \equiv O(a) \wedge O(b)$):

   $$\neg O(a\&b) = \neg(O(a) \wedge O(b)) = \neg O(a) \vee \neg O(b) = P(\overline{a}) \vee P(\overline{b})$$

   This is completely counter-intuitive since it is not clear what the disjunction over permissions means. We also have disjunction on obligations, which we believe should be forbidden syntactically, though many researchers on deontic logic see disjunction on obligations as natural.

2. One can argue that $\neg O(a) = P(\overline{a}) \wedge P(a)$, since intuitively not being obliged to do something gives you permission to do the contrary, but also the permission of the positive action itself. In this case we have:

   $\neg O(a\&b) = P(\overline{a}) \wedge P(\overline{b}) \wedge P(a) \wedge P(b)$

   We have now two different interpretations (based on the interpretation $P(a\&b) = P(a) \wedge P(b)$ or $P(a + b) = P(a) \wedge P(b)$)

[4]Notice that the discussion about negation of obligations is more philosophical, and included here only for completeness. As discussed in the previous section negation over obligations is not natural in e-contracts.

(a) $\neg O(a\&b) = P(a\&\overline{a}\&b\&\overline{b})$

(b) $\neg O(a\&b) = P(a + \overline{a} + b + \overline{b})$

The first option seems more natural, but this would imply to give special meaning to the &, since intuitively $a\&\overline{a} = \bot$ under obligation, but $a\&\overline{a} \neq \bot$ under permission. The second option has the problem that we cannot do two things at the same time (like $a\&b$, which should be allowed).

All the discussion above lead us to the following conclusions:

1. The action operator & behaves differently under permissions and obligations, hence we need two different action operators (let's call them $\&_o$ and $\&_p$).
2. We need to introduce XOR also for permissions.
3. We must allow negation on actions also under permissions.

Assuming we have a *conflict* relation $\sharp$ between actions, in what follows we propose some laws for getting the above:

1. $\&_o$ can only be used under obligations and must have the following properties:

$$\begin{array}{l} a\&_o\overline{a} = \bot \\ a\&_o b = \bot \quad \text{if } a\sharp b \\ (\overline{a\&_o b}) = \overline{a}\&_o\overline{b} \end{array}$$

We then have that:

$$\begin{array}{l} O(a\&_o b) = \bot \text{ if } a\sharp b \\ O(a\&_o\overline{a}) = \bot \\ O(a\&_o b) = O(a) \wedge O(b) \\ \neg O(a\&_o b) = \neg O(a) \wedge \neg O(b) = P(\overline{a}\&\overline{b}) \end{array}$$

2. $\&_p$ can only be used under permissions and must have the following properties:

$$\begin{array}{l} a\&_p\overline{a} = a + \overline{a} \\ a\&_p b \neq a + b \quad \text{if } a \neq b \wedge \neg(a\sharp b) \\ a\&_p b \rightarrow a + b \quad \text{if } a\sharp b \ \text{ (Here } \rightarrow \text{ means that } a\&_p b \text{ must be replaced by } a + b) \\ \neg(a\&_p b) = \overline{a}\&_p\overline{b} \end{array}$$

We then have that:

$$
\begin{array}{l}
P(a\&_p b) = P(a) \wedge P(b) \quad \text{if } \neg(a \sharp b) \\
P(a\&_p \overline{a}) = P(a + \overline{a}) = P(a) \oplus P(\overline{a}) \\
P(a\&_p b) = P(a + b) = P(a) \oplus P(b) \quad \text{if } a \sharp b \\
P(a\&_p b) = F(a) \wedge F(b)
\end{array}
$$

With these laws, we might get the right interpretation of the $\neg O(a\&b)$.

2.4.3 About the Negation of Actions

Negation introduces new problems and at first it seems enough to consider only negation over atomic actions. We can have "positive" and "negative" atomic actions. One crucial question is: Given an action a, what does it mean by "negation" of a? Does it mean "not doing a", or "doing anything but a"? Do we want to allow both interpretations? If so, we might need to have different notations, like $\overline{a}$ and $\neg a$ for the two different notions. The intuitive meaning of a negative action a is "not performing a". That is, $\overline{a}$ is not defined as "the set of all the actions but a". One intrinsic problem concerning the definition of negative actions is that when performing an action, the current state changes, but what is the effect of *not* perform an action? Is it natural to consider *not* performing an action as being an action itself? For example, if I withdraw money from my personal bank account, then the account changes. On the other hand, if I do not withdraw any money, this negative action has not effect on my bank account. Though we do not have a convincing final solution on how to treat negation, we will see later the approach we take in our contract language.

Besides, the above problem extends to obligation, permission and prohibition over negative actions. For instance "you are not obliged to talk", $\neg O(talk)$, might be interpreted as "you have the right to remain in silence" (which means "you have the right not to talk", i.e., $P(\overline{talk})$). This shows that the intuition of negated actions on permission is in some sense different from those on obligations, and it might be reasonable to allow them under permissions.

2.5 On Obligation, Permission and Prohibition in E-contracts

Many of the research conducted by philosophers and logicians tend to stress differences between "ought" and "must", or to define logical equivalences between obligation and permission, or even to force one notion being dual of

the other and then characterizing the exceptions. Although this is reasonable in a philosophical context or in pure logic, we claim that we can avoid many of the above discussions given that we are restricted to electronic contracts. In what follows we provide arguments for restricting syntactically the occurrence of certain expressions involving obligation (O), permission (P) and prohibition (F) in a e-contracts.

In what follows we resume some of the above discussions, and we introduce new insights of what should and should not be expressible in a contract language.

- We consider statements expressing *one is NOT obliged to do something* is not intuitive in the setting of e-contracts.

$$\neg O(a) \text{ should not occur in a contract}$$

- It is counter intuitive to have iteration of actions under obligation, permission and prohibition; e.g. it is not normal to have in a contract a statement like: *One is obliged to not pay, or pay once, o pay twice, or ...*.

$$O(a^*),\ P(a^*), \text{ or } F(a^*) \text{ are not allowed}$$

- A statement like *one is NOT permitted to do some action* can be rewritten as *one is forbidden to do the action*

$$\neg P(a) \equiv F(a)$$

- A statement like *one is NOT forbidden to do an action* can be rewritten as *one is permitted to do the action*

$$\neg F(a) \equiv P(a)$$

Note that we adhere to the classical definitions of permission and prohibition as one being the negation of the other.

We now discuss some restrictions related to Prohibition (F).

- 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:

$$\begin{aligned} \varphi_{UK} &\Rightarrow F(d_r) \\ \varphi_{REU} &\Rightarrow F(d_l). \end{aligned}$$

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.

- The prohibition of performing an action a should imply the prohibition of any concurrent execution of any set of actions that contain the action a:

$$F(a) \Rightarrow F(a\&b), \tag{2}$$

but the converse implication should not hold:

$$F(a\&b) \not\Rightarrow F(a). \tag{3}$$

3 Puzzles and Paradoxes

In what follows we mention some of the most important paradoxes of deontic logic (see for instance [McN06] for more details), and analyze a new paradox that arises in our approach.

3.1 Classical Paradoxes and Puzzles

Ross's Paradox [Ros41]: In natural language it is expressed as:

1. It is obligatory that one mails the letter.
2. It is obligatory that one mails the letter or one burns the letter.

In Standard Deontic Logic (SDL) these are expressed as:

1. $O(p)$
2. $O(p \vee q)$

The problem is that in SDL one can infer that $O(p) \Rightarrow O(p \vee q)$.

The Good Samaritan Paradox [Pri58]: In natural language we have:

1. It ought to be the case that Jones helps Smith who has been robbed.
2. It ought to be the case that Smith has been robbed.

And one naturally infers that:

> Jones helps Smith who has been robbed if and only if Jones helps Smith and Smith has been robbed.

In SDL the first two are expressed as:

1. $O(p \wedge q)$
2. $O(q)$

The problem is that in SDL one can derive that $O(p \wedge q) \Rightarrow O(q)$ which is counter intuitive in the natural language, as in the example above.

The Free Choice Permission Paradox [Ros41]: In natural language we have:

1. You may either sleep on the sofa or sleep on the bed.
2. You may sleep on the sofa and you may sleep on the bed.

In SDL this is:

1. $P(p \vee q)$
2. $P(p) \wedge P(q)$

The natural intuition tells that $P(p \vee q) \Rightarrow P(p) \wedge P(q)$. In SDL this would lead to $P(p) \Rightarrow P(p \vee q)$ which is $P(p) \Rightarrow P(p) \wedge P(q)$, so $P(p) \Rightarrow P(q)$. As an example: *If one is permitted something, then one is permitted anything.*

Sartre's Dilemma [McN06]: In natural language:

1. It is obligatory to meet Jones now (as promised to Jones).
2. It is obligatory to not meet Jones now (as promised to Smith).

In SDL this is:

1. $O(p)$
2. $O(\neg p)$

The problem is that in the natural language the two obligations are intuitive and often happen, where the logical formulas are inconsistent when put together (in conjunction) in SDL.

Chisholm's Paradox [Chi63]: In natural language it is expressed as:

1. John ought to go to the party.
2. If John goes to the party then he ought to tell them he is coming.
3. If John does not go to the party then he ought not to tell them he is coming.
4. John does not go to the party.

In Standard Deontic Logic (SDL) these are expressed as:

1. $O(p)$
2. $O(p \Rightarrow q)$
3. $\neg p \Rightarrow O(\neg q)$
4. $\neg p$

The problem is that in SDL one can infer $O(q) \wedge O(\neg q)$ which is due to statement (2).

The Gentle Murderer Paradox [For84]: In natural language it is expressed as:

1. It is obligatory that John does not kill his mother.

2. If John does kill his mother, then it is obligatory that John kills her gently.
3. John does kill his mother.

In Standard Deontic Logic (SDL) these are expressed as:

1. $O(\neg p)$
2. $p \Rightarrow O(q)$
3. p

The problem is that when adding a natural inference like $q \Rightarrow p$ then in SDL one can infer that $O(p)$.

3.2 A new paradox?

Apparently the deontic community does not see, in general, $O(a) \vee O(b)$ as a problematic formula, but we believe it is indeed a problem to have disjunction of obligations – and also of permissions and prohibitions. This might be avoided in different ways depending on the approach, but in the presence of conjunction of actions and some of the usual relations between obligation, permission and prohibition, a new paradox arises. In what follows we explain why we think the above causes problems on the deontic reasoning.

Most of the approaches using logics for formalizing normative deontic notions[5] propose an extension of propositional logic (PL), meaning that the logics include all the tautologies of PL. This naturally includes the following tautology: $A \Rightarrow A \vee B$. We will show in what follows that from $O(a)$ we can derive $P(a) \wedge P(b)$ which is clearly a dangerous paradox ("if I am obliged not to talk in the presence of the Pope, then I am permitted not to talk *and* to kill the Pope"). In our derivation we use the following common relations:

- $O(\cdot) \Rightarrow P(\cdot)$,
- $P(\cdot) \equiv \neg F(\cdot)$.

We also make use of the De Morgan laws and the following intuitive equivalences:

- $P(a\&b) \equiv P(a) \wedge P(b)$,

[5]Usually these notions are formalized as *operators* and in deontic logic are considered to be *modalities*. Though they are not operators in our approach, we keep the terminology whenever no confusion might arise.

- $F(a\&b) \equiv F(a) \wedge F(b)$.

Notice that the above is not "standard" since many approaches do not consider conjunction over actions, but it is very intuitive to interpret permission and prohibition of conjunction of actions as above. We are ready now to show that $O(a)$ implies $P(a) \wedge P(b)$.

First take $O(a) \Rightarrow O(a) \vee O(b)$ (instance of the PL tautology $A \Rightarrow A \vee B$). From $O(a) \Rightarrow P(a)$ and $O(b) \Rightarrow P(b)$, we get that $O(a) \vee O(b) \Rightarrow P(a) \vee P(b)$. But $P(a) \vee P(b) \Rightarrow \neg F(a) \vee \neg F(b)$ and by the De Morgan law we have that $\neg(F(a) \wedge F(b))$ which implies $\neg F(a\&b)$. We then get $P(a\&b)$ which is equivalent to $P(a) \wedge P(b)$.

What is wrong on the above derivation? Some might argue that the equivalences given for permission and prohibition of actions are not universally accepted by the deontic community and that they are not correct. We believe that the cause of the problem relies on accepting certain laws of propositional logic when reasoning about deontic modalities (like de Morgan laws). Moreover, we strongly advocate for the elimination of the classical disjunction on normative deontic notions, given that the intuitive idea in natural language when using the word *or* is usually that of an *exclusive or* ("The client is obliged to pay or to send a notification of delay.", and another example would be: "You have the right to remain silent or anything you say can be used against you in the court of law."). Thus, we claim that a logic of actions (with conjunction of actions) for a correct representation and reasoning of obligation, permission and prohibition should have the following restrictions:

- The De Morgan laws cannot be applied to deontic modalities,
- Use the exclusive or, and disallow (syntactically) the classical disjunction on deontic modalities.

We claim that the right interpretation of $\neg(O(a) \wedge O(b))$ should be $\neg O(a) \wedge \neg O(b)$, which is more intuitive, in case one admits the use of negation over obligations. Similarly for prohibition.

4 Desirable Properties of a Language for Contracts

Before presenting our language we start by listing some of the intuitive properties we should have, and others we should avoid, when formalizing contracts.

(1) Avoid as many deontic logic paradoxes as possible:

   (a) Avoid the Good Samaritan paradox, Satre's dilemma, and the Gentle Murder paradox;

   (b) Avoid Chisholm's paradox. This means obligation should be defined only on actions, not on formulas. In particular do not write formulas of the form $O(\phi \Rightarrow \psi)$;

   (c) Avoid Ross's paradox. This means avoid having (in the classical notation of deontic logic): $O(p) \Rightarrow O(p \vee q)$;

   (d) Avoid the Free Choice Permission paradox (i.e. do not allow the following implication: $P(p) \Rightarrow P(p \vee q)$);

   (e) Avoid the new paradox described in Section 3.2; i.e., syntactically disallow the classical disjunction between deontic modalities.

(2) Use the *XOR* logical connective instead of the classical disjunction between modalities;

(3) Allow concurrent actions and keep the intuition of conjunction on obligations; i.e., $O(a\&b) = O(a) \wedge O(b)$.

(4) Some intuitive desirable relations on obligations:

   (a) $O(a;b) = O(a) \wedge [a]O(b)$

   (b) Allow CTD (reparation)

(5) Allow the definition of conditional obligations, i.e., formulas of the form $\psi \Rightarrow O(a)$.

(6) Have the following: $O(a) \Rightarrow P(a)$.

(7) Do not define permission and obligations in terms of each other (for instance, do not define obligation as $O(a) = \neg P(\neg a)$).

(8) Some intuitive desirable relations on permissions:

   (a) $P(a;b) = P(a) \wedge [a]P(b)$

   (b) $P(a+b) = P(a) \oplus P(b)$[6]

(9) Some intuitive desirable relations on prohibitions:

   (a) $F(a) = \neg P(a)$

[6]Many authors prefer to have $P(a+b) = P(a) \wedge P(b)$ (see for instance [BWM01]).

(b) $F(a;b) = F(a) \vee \langle a \rangle F(b)$

(c) $F(a+b) = F(a) \wedge f(b)$

(d) $F(a) \Rightarrow F(a\&b)$

(e) $F(a\&b) \not\Rightarrow F(a)$

(f) Allow contrary-to-prohibition.

5 A Specification Language for Contracts

This section contains the definition of our specification language for writing e-contracts. The first two subsections are meant as a technical preamble to subsection 5.3 where the language is defined. If the reader is more or less familiar with the concept and the intuition of an *action* (from dynamic logics for example) then she may skip directly to subsection 5.3. Subsection 5.2 is intended to define the concept of *action negation*. This section can also be skipped in a first reading.

5.1 Action Algebra

Some of the most well known and studied action algebras come from the work on dynamic logics [Pra76]. We base our work on Pratt and Kozen's dynamic algebra [Pra80, Koz80]. This algebra is built on top of Kleene algebra which was introduced in 1956 and further developed by Conway in [Con71]. For references and an introduction to both Kleene and dynamic algebra see the extensive work of Kozen [Koz81, Koz90, Koz97].

In these research efforts the authors used, for example, regular languages as the objects of the algebra, or relations over a fixed set (as we have in dynamic logic) and analyzed properties like completeness [Koz94], complexity [CKS96] and applications [Coh94] of variants of Kleene algebra. Some variants include the test operator ?, and others discard the iteration operator *. Many insights can be drawn from this extensive work related to our need of action algebra.

We define an algebraic structure similar to dynamic algebra, modified so that it complise with the intuition drawn from e-contracts. A first change is in dropping the Kleene star (iteration) as it is unnatural to have it under obligation, permission and prohibition of the Contract Language (see discussion in Section 2.5). A second change involves the concurrency of two or more actions, and it consists of defining a special operator for the algebra to model truly concurrent actions. For example, we need to express that *The client is obliged to do actions a and b at the same time.*

We recall that a *Kleene algebra* is a structure $\mathcal{K} = \{\mathbf{K}, +, \cdot, \mathbf{0}, \mathbf{1}, ^*\}$ with the properties that $(\mathbf{K}, +, \mathbf{0})$ is a commutative monoid with the identity element $\mathbf{0}$, and $(\mathbf{K}, \cdot, \mathbf{1})$ is a monoid with the identity element $\mathbf{1}$. Moreover, operator $+$ is idempotent and thus it is possible to define a partial order $\leq$ on $\mathbf{K}$ thus having that $(\mathbf{K}, +, \mathbf{0})$ is a semilatice. The * is a unary operator which respects a set of axioms with the intuition that $a^* = 1 + a + a \cdot a + \ldots$. In programming theory it is usual to interpret $+$ as *choice*, $\cdot$ as *sequence* and * as *iteration*.

A *dynamic algebra* is a rather more complex structure $\mathcal{D} = (\mathcal{K}, \mathcal{B}, \langle\cdot\rangle)$ where $\mathcal{K}$ is a Kleene algebra, $\mathcal{B}$ is a Boolean algebra, and $\langle\cdot\rangle$ a scalar multiplication defined as $\langle\cdot\rangle : \mathcal{K} \times \mathcal{B} \rightarrow \mathcal{B}$ respecting the usual rules.

Our action algebra has a set of atomic actions denoted $\mathcal{A}$ and the action operators which form the compound actions: $+$ for *choice* of two actions, $\cdot$ for sequence of actions (or concatenation; in PDL we find this operator denoted as ;), & for concurrent execution of two atomic actions, and the test operator ? (we will see later how with test operator we can simulate implication over formulas [HKT00]). The three operators $+$, $\cdot$, and & are binary operators. Choice ($+$) is applied to compound actions and is associative and commutative. Concurrency (&) operator is applied to atomic actions only and is associative and commutative. The sequence ($\cdot$) operator is applied to compound actions and is right-associative and non-commutative. For brevity we often drop the sequence operator and instead of $\alpha \cdot \beta$ we just write $\alpha\beta$. The operators $+$, $\cdot$, and & are applied to elements of $\mathcal{A}$ (actions).

In dynamic algebra, the elements of the boolean algebra are called tests and are included in the set of actions of the Kleene algebra (i.e. tests are special actions)[7]. With the test operator the *skip* action (denoted $\mathbf{1}$ above) is defined as $\top?$, where $\top$ is the special proposition that holds in every world. $\mathbf{1}$ is interpreted in PDL as the identity relation over the set of worlds. It has the meaning that when executing the *skip* atomic action the system goes to the same state. With *skip* the actions a and $a \cdot \mathbf{1}$ have the same set of traces, and *skip* has also the property that $\mathbf{1}^* = \mathbf{1}$.

We do not study in this paper properties of this action algebra but at a first look the $+$ and $\cdot$ operators obey the same properties as the operators of Kleene algebra. It is left to investigate the properties of & operator and its relations with the other operators. Adding the test operator we obtain an action algebra with tests [Koz97] and we expect to have similar properties.

[7] To be more formal and to have a syntax more closer to the syntax used in PDL we use the ? operator and call it *test* operator. The test operator is special in the sense that it is applied to elements of $\mathcal{B}$ (i.e. formulas in the boolean algebra) and generates actions of $\mathcal{A}$ (i.e. $? : \mathcal{B} \rightarrow \mathcal{A}$). Basically ? generates the set of actions called the set of *tests* included in $\mathcal{A}$.

5.2 Action Normal Formal

It is known that for regular expressions there is no standard normal form; for example, see the *Starr-Height problem* [Egg63] which looks at regular expressions normal forms from the perspective of Kleene star.

For the set of action operators ($+$, $\cdot$, *, ?) of the algebra defined in Section 5.1 we have the following definition of action normal form. For the semantics of actions given with traces, as in process logics [Pra79], we obtain all the traces of the action.

Definition 5.1 (action normal form for $+$)**.** *For actions defined with the operators $+$, $\cdot$, *, ? we have an action normal form denoted by ANF^+ and defined as*

$$\alpha = \underset{\rho \in R}{+} \rho \cdot \alpha'$$

where α is a compound action, ρ represents either an atomic action or a test, and R is a subset of atomic actions and tests.

Theorem 5.1. *For every action in the algebra of Section 5.1 we have a corresponding ANF^+.*

A natural and useful view of *action negation* when we consider actions interpreted as traces is to say that the negation $\overline{\alpha}$ of action α is the action given by all the immediate traces that *take us outside* the trace of α [BWM01]. With ANF^+ it is easy to formally define $\overline{\alpha}$.

Definition 5.2 (action negation)**.** *The action negation is denoted by $\overline{\alpha}$ and is defined for any action α in ANF^+ as:*

$$\overline{\alpha} = \overline{\underset{\rho \in R}{+} \rho \cdot \alpha'} = \underset{b \in \mathcal{A} \setminus R}{+} b \quad + \quad \underset{\rho \in R}{+} \rho \cdot \overline{\alpha'}$$

where α' is also in ANF^+, and R is a set of the atomic actions or tests. Note that b is only an atomic action[8] of $\mathcal{A}$, which means that the action negation does not take into consideration the tests.

5.3 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 a Contract Language ($\mathcal{CL}$), and provide a set of rewriting rules in order to simplify and minimize the number of expressions in the language.

[8]When we remove from the set of atomic actions $\mathcal{A}$ the set R which contains both atomic actions and tests, the resulting set will contain only the actions of $\mathcal{A}$ which are not in R.

Definition 5.3 (Contract Language Syntax)**.** *The syntax of the contract language is:*

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

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 a 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 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 5.1, 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 appear between prohibitions and $+$ cannot occur under F, as we have discussed in Section 2.5.

We borrow from PDL the syntax $[\alpha]\mathcal{C}$ (also called *dynamic box*) to represent that after performing α, $\mathcal{C}$ should be the case. Intuitively, one may think of $[\cdot]$ as the $\forall$ quantifier in the sense that either the action is not performed or if it is performed then the clause after it should be enforced. The $[\cdot]$ notation allows having a *test* inside, where the syntax $[\phi?]\mathcal{C}$ must be understood as $\phi \Rightarrow \mathcal{C}$. $\langle\alpha\rangle\mathcal{C}$ (also known as *dynamic diamond*) captures the idea that there

$$
\begin{array}{lrcl}
(1) & O(\alpha+\beta) & = & O(\alpha)\oplus O(\beta) \\
(2) & O(a\&b) & = & O(a)\wedge O(b) \\
(3) & O(\alpha\beta) & = & O(\alpha)\wedge [\alpha]O(\beta) \\
(4) & P(\alpha+\beta) & = & P(\alpha)\oplus P(\beta) \\
(5) & P(\alpha\beta) & = & P(\alpha)\wedge \langle\alpha\rangle P(\beta) \\
(6) & F(\alpha\beta) & = & F(\alpha)\vee [\alpha]F(\beta)
\end{array}
$$

Table 1: Compositional rules

must exist the possibility of executing α, in which case $\mathcal{C}$ will be enforced afterwards. In the contract language we do not relate the dynamic box to the dynamic diamond. They are related in μ-calculus, through their translation of Section 6.3. Following temporal logic (TL) [Pnu77] notation we have $\mathcal{U}$ (*until*) and $\bigcirc$ (*next*) with the intuitive behavior 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 compound actions have a compositional behavior in $\mathcal{CL}$ when they appear under obligation O. For choice of actions we have

$$O(\alpha+\beta) = O(\alpha)\oplus O(\beta) \tag{4}$$

with the intuition (drawn from the world of contracts) that *If one is obliged to choose between doing one action or doing another action*, then one should regard it as being *either obliged to do the first action or as being obliged to do the second action.*

For concurrent actions we have

$$O(a\&b) = O(a)\wedge O(b) \tag{5}$$

with the intuition that, regarding atomic actions *If one is obliged to do an atomic action a and is also obliged to do another atomic action b* then one should conclude that *one is obliged to do the two atomic actions at the same time.*

For the sequence of actions we have

$$O(\alpha\beta) = O(\alpha)\wedge [\alpha]O(\beta) \tag{6}$$

which intuitively means that if one is obliged to do a sequence of actions then one should be obliged to do the first action, and after doing the first action one should also be obliged to do the second action.

The compound actions under permition are similar to the ones under obligation. The choice of actions is also exclusive choice and we still have compositionallity of P:

$$P(\alpha + \beta) = P(\alpha) \oplus P(\beta) \tag{7}$$

which intuitively means that if one is permitted to choose between doing one of the actions α or β then, one is either permitted the first action or is permitted the second action.

For concurrency under permission we do not find any compositionallity rule. A clause $P(a\&b)$ stating that it is permitted to do the two actions at the same time, does not give any information about the individual actions. Moreover, the permission of the individual actions can not give information about the permission of the concurrent execution of the two actions.

For the sequence of actions under permission we have:

$$P(\alpha\beta) = P(\alpha) \wedge \langle\alpha\rangle P(\beta) \tag{8}$$

with the intuition that if one is permitted to do the sequence of actions then one may conclude that one is both permitted the first action and also there exists a way of doing the first action and afterwords one would be permitted the second action.

Compound actions under prohibition do not behave the same as under obligation or permission. For concurrency under prohibition we do not find any compositionallity rule; (see equations (2), and (3) of Section 2.5).

For the sequence of actions under prohibition we have

$$F(\alpha\beta) = F(\alpha) \vee [\alpha]F(\beta) \tag{9}$$

with the intuition that if one is forbidden to do the sequence of actions then one may conclude that one is either forbidden the first action or, if the first action is performed the second action is forbidden.

The main difference between modal logic (where the modality denotes necessity) and deontic logic (where the modality denotes obligation) is in the fact that the deontic modality can be violated. For example, if in modal logic one can make the inference: $\Box p$ then p (if it is necessary that p, then p is true), in deontic logic the inference is no longer possible because O can be violated (see Section 2.1 for a discussion). Related to this we constantly find in contracts the *contrary to duty* (CTD) and *contrary to prohibition* (CTP) formulas. CTDs express what happens if an obligation is violated. In our case, if we have the obligation to do an action then the violation of the obligation is the execution of the negation of the action. CTDs are added to the contract language with the following syntax:

$$O_{\varphi}(\alpha)$$

stating the obligation to execute the compound action α and the reparation formula φ which should hold in case the obligation is violated. The reparation may be either another obligation, a prohibition, a stand alone assertion, or even another CTD which should be enforced after the violation occurs. The above is syntactic sugar for the following $\mathcal{CL}$ formula:

$$O_{\varphi}(\alpha) = O(\alpha) \wedge [\overline{\alpha}]\varphi \tag{10}$$

stating the obligation $O(\alpha)$ which should hold in the current world and if the negation of α is executed (meaning that the obligation is violated) the reparation φ should be enforced.

One might suggest that just the action negation as defined in Section 5.2 does not capture the intuition of violation of an obligation of an action. One may say that for an action a (e.g. *deposit money in the bank account*) a violating action may be just the negative action $\neg a$ (*NOT deposit money in the bank account*). In this paper we do not consider negative actions; for a discussion about our decision see Section 2.4.3. A second argument for our decision is that negative actions may be expressed in other ways. For example, in order to say *obliged NOT to do* one can say *forbidden to do*.

Contrary to Prohibition statements explicitly provide the reparation formula which should hold in case the prohibition is violated. For example *if the forbidden action α is executed (the prohibition is violated) then a reparation formula φ should be enforced.* The CTPs (denoted as $F_{\varphi}(\alpha)$) are abbreviations of the $\mathcal{CL}$ formulas:

$$F_{\varphi}(\alpha) = F(\alpha) \wedge [\alpha]\varphi \tag{11}$$

With the dynamic box syntax we can model in $\mathcal{CL}$ conditional obligations, permissions, and prohibitions (see *Dyadic Deontic Logic* for an introduction to the formalism that has introduced conditional obligations [PS97]). We may have two kinds of conditional expressions; let us take an example for obligation. Conditional obligations can depend on both the execution of an action, or on an assertion which holds in the current state. Intuitively, conditional obligations related to actions state that after executing an action, a certain obligation is the case. We represent such conditional obligation as:

$$[\alpha]O(\beta) \tag{12}$$

where α is the conditioning action and $O(\beta)$ is the obligation enforced by the conditioning action. Often in contracts we find obligations triggered by some assertion that holds in the current world. Intuitively, if the assertion

$$
\begin{array}{lrl}
(1) & O(a) \wedge O(b) & \rightarrow O(a\&b) \\
(2) & O(a) \oplus O(a\&b) & \rightarrow O(a) \\
(3) & O(a) \wedge O(a\&b) & \rightarrow O(a\&b) \\
(4) & O(a) \wedge (O(a) \oplus O(b)) & \rightarrow O(a) \\
(5) & O(a) \wedge O(a) & \rightarrow O(a) \\
(6) & O(a) \oplus O(a) & \rightarrow O(a) \\
(7) & O(c) \wedge (O(a) \oplus O(b)) & \rightarrow (O(c) \wedge O(a)) \oplus (O(c) \wedge O(b)) \\
(8) & (\oplus_i O(a_i)) \wedge (\oplus_j O(b_j)) & \rightarrow \oplus_{i,j}(O(a_i) \wedge O(b_j)) \quad a_i \neq b_j
\end{array}
$$

Table 2: Rewriting rules for obligation O

φ holds in the current world then the obligation should be enforced in the current world. We model this by using the test operator ?:

$$[\varphi?]O(\alpha) \tag{13}$$

The formula φ represents any contract formula $\mathcal{C}$ specified in the Contract Language or a stand alone assertion ϕ like: *the budget is more than 200$*.

We aim at translating into the logic of Section 6.2 as few constructs from the contract language as possible. For this we give first a set of rewriting rules for the $\mathcal{CL}$ obligation formulas which lead to an *obligation normal form* which is much easier to translate. The rewriting rules are also useful for giving several restrictions on the formulas of $\mathcal{CL}$ drawn from real contracts in practice. In the Table 2 the rules (1)-(4) are guided by the common examples found in real contracts, rules (5)-(6) are the usual contraction rules, and the rules (7)-(8) basically give the distributivity of the conjunction operator over the exclusive disjunction operator. Note that the rules (1)-(8) are applied only to obligation operator over atomic or concurrent actions.

For formulas involving just the obligation construct and the $\wedge$ and $\oplus$ over obligations we can write them in the following obligation normal form. Note that it is applied only to obligations of atomic or concurrent actions, thus giving a normal form only for the first step in the traces of the compound actions. We do not take into consideration the $\cdot$ sequence syntax.

$$\bigoplus_{i=1}^{n}(O(\&_{i=1}^{m} a_{i,j}))$$

where for a fixed i, and $\forall j$, $a_{i,j}$ are different one from another. Because of the normal form, all we need to translate for obligations into the extended μ-calculus is: $O(a)$, $O(a\&b)$, and the $\oplus$ syntactic constructs.

6 The Underlying Logic for the Contract Language

6.1 Propositional μ-calculus: Syntax and Semantics

We take the classical propositional μ-calculus as defined in [Koz83] (a very nice introduction can be found in [BS01], where the authors call the logic *modal μ-calculus*). μ-calculus has nice properties: it is decidable [KP83] and has a complete [Wal95] axiomatic system and a complete Gentzen-style deduction system [Wal93].

μ-calculus defines a special set $\mathcal{L}$ of labels, which we call *atomic actions* and denote them by small letters from the beginning of the Latin alphabet $a, b, c, \ldots$. The syntax of propositional μ-calculus is:

> P, Z, and $\top$ are μ-formulas; where P represents the *propositional variables*, Z represents the *state variables*, and $\top$ is the constant proposition denoting true.
>
> If φ and ψ are μ-formulas then $\neg\varphi$, $\varphi \wedge \psi$, and $[a]\varphi$ are μ-formulas where $a \in \mathcal{L}$ are *labels*.
>
> If φ is μ-formula and ν denotes the greatest fix-point then $\nu Z.\varphi(Z)$ is a μ-formula.

In a more concise notation the syntax of μ-calculus is:

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

We also have the usual dualities:

$$\varphi \vee \psi \stackrel{def}{=} \neg(\neg\varphi \wedge \neg\psi)$$

$$\langle a\rangle\varphi \stackrel{def}{=} \neg[a]\neg\varphi$$

$$\mu Z.\varphi(Z) \stackrel{def}{=} \neg\nu Z.\neg\varphi(\neg Z)$$

In the following we give the standard semantics of the operators of propositional μ-calculus. The semantic interpretation of the above syntactic constructs follows the classical set-theoretical approach [Koz83]. The formulas are interpreted over a structure (similar to a labelled transition system) denoted $\mathcal{T}$. $\mathcal{T}$ is defined with respect to a set of propositions P and a set of labels $\mathcal{L}$ and is $\mathcal{T} = (\mathcal{S}, R_{\mathcal{L}}, \mathcal{V}_P, \mathcal{V})$. $\mathcal{S}$ is the set of states (worlds), $R_{\mathcal{L}}$ is a function assigning to each action in $\mathcal{L}$ a relation over $\mathcal{S}$ (i.e. $R_{\mathcal{L}}(a) \subseteq \mathcal{S} \times \mathcal{S}$,

$a \in \mathcal{L}$), $\mathcal{V}_P : P \to 2^{\mathcal{S}}$ is the interpretation of the propositions as subsets of states where the propositions hold. $\mathcal{V}$ 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 $R_{\mathcal{L}}(a)$ we write R_a.

Some of the papers in the literature present the semantics of μ-calculus as a Labeled Transition System (LTS) [BS01]. The difference between a LTS and the present structure $\mathcal{T}$ is that in place of a labelled transition relation $\rightarrow \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$ we associate for each action of $\mathcal{L}$ a set of transitions between two states. This set of pairs of states gives for each action a relation over $\mathcal{S}$.

The semantics of μ-calculus is:

$$\|\top\|^{\mathcal{T}}_{\mathcal{V}} = \mathcal{S}$$

$$\|P\|^{\mathcal{T}}_{\mathcal{V}} = \mathcal{V}_P(P)$$

$$\|Z\|^{\mathcal{T}}_{\mathcal{V}} = \mathcal{V}(Z)$$

$$\|\neg\varphi\|^{\mathcal{T}}_{\mathcal{V}} = \mathcal{S} \setminus \|\varphi\|^{\mathcal{T}}_{\mathcal{V}}$$

$$\|\varphi \wedge \psi\|^{\mathcal{T}}_{\mathcal{V}} = \|\varphi\|^{\mathcal{T}}_{\mathcal{V}} \cap \|\psi\|^{\mathcal{T}}_{\mathcal{V}}$$

$$\|[a]\varphi\|^{\mathcal{T}}_{\mathcal{V}} = \{s \mid \forall t \in \mathcal{S}.\ (s,t) \in R_a \Rightarrow t \in \|\varphi\|^{\mathcal{T}}_{\mathcal{V}}\}$$

$$\|\nu Z.\varphi\|^{\mathcal{T}}_{\mathcal{V}} = \bigcup\{S \subseteq \mathcal{S} \mid S \subseteq \|\varphi\|^{\mathcal{T}}_{\mathcal{V}[Z:=S]}\}$$

$$\|\varphi \vee \psi\|^{\mathcal{T}}_{\mathcal{V}} = \|\varphi\|^{\mathcal{T}}_{\mathcal{V}} \cup \|\psi\|^{\mathcal{T}}_{\mathcal{V}}$$

$$\|\langle a\rangle\varphi\|^{\mathcal{T}}_{\mathcal{V}} = \{s \mid \exists t \in \mathcal{S}.\ (s,t) \in R_a \wedge t \in \|\varphi\|^{\mathcal{T}}_{\mathcal{V}}\}$$

$$\|\mu Z.\varphi\|^{\mathcal{T}}_{\mathcal{V}} = \bigcap\{S \subseteq \mathcal{S} \mid S \supseteq \|\varphi\|^{\mathcal{T}}_{\mathcal{V}[Z:=S]}\}$$

It is known that propositional μ-calculus is more expressive than PDL and can embed PDL [BWM01]. Therefore we define the following *syntactic shortcuts* which capture the behavior of the action algebra we have in PDL.

We denote by $[\alpha;\beta]\varphi$ the following μ-formula $[\alpha][\beta]\varphi$

We denote by $[\alpha \cup \beta]\varphi$ the following μ-formula $[\alpha]\varphi \wedge [\beta]\varphi$

We denote by $[\alpha^*]\varphi$ the following μ-formula $\nu Z.\varphi \wedge [\alpha]Z$

We denote by $[\psi?]\varphi$ the following μ-formula $\psi \Rightarrow \varphi$

A simple example of a compound action in PDL is $[\psi?; a]\varphi$ which means that if in the current state ψ holds then we may continue and execute action a and every time the action terminates it will terminate in a state satisfying formula φ. Guided by the definitions of the above syntactic shortcuts we get a μ-formula:

$$[\psi?; a]\varphi \stackrel{def}{=} [\psi?][a]\varphi \stackrel{def}{=} \psi \Rightarrow [a]\varphi$$

This formula expresses the partial correctness assertion of Hoare logic $\{\psi\}a\{\varphi\}$ which means that if a program starts with the input ψ (in a state satisfying ψ) then, whenever the program ends it will end in a state satisfying φ.

6.2 Yet another propositional μ-calculus

In this section we give a variant of the propositional μ-calculus specially tailored for our needs to have a formal framework to reason about contracts specified in $\mathcal{CL}$. We take the syntax of the propositional μ-calculus as defined in Section 6.1, and we modify the set of actions $\mathcal{L}$, and the set of propositions P by adding a set of *propositional constants* which we denote by P_c included in P. The set of state variable remains also unchanged. We call the extended logic $\mathcal{C}\mu$.

The interpretation of the operators remains the same. We only give the semantics for our extension part.

We need first to be able to deal with true concurrency. Instead of the labels representing atomic actions we have *finite subsets of atomic actions* with the intuitive meaning that all the atomic actions in the *set* are executed concurrently.

Definition 6.1 (concurrent sets)**.** *A concurrent set denoted by γ (possible indexed) is a finite subset of the set of atomic actions $\mathcal{L}$, $\gamma = \{a_1, \ldots, a_n\}$ where $a_i \in \mathcal{L}$. These concurrent sets are considered the labels of $\mathcal{C}\mu$. The structure of the new logic is interpreted over $2^{\mathcal{L}}$ instead of $\mathcal{L}$.*

Inside the *box* operator we now have concurrent sets γ instead of atomic actions ($[\gamma]\varphi$). Note that $\mathcal{C}\mu$ subsumes the classical μ-calculus by taking the actions of μ-calculus to be singleton concurrent sets ($\gamma = \{a\}$). We change the $R_{\mathcal{L}}$ function of μ-calculus into $R_{2^{\mathcal{L}}}$ which is applied to concurrent sets of $2^{\mathcal{L}}$ instead of atomic actions of $\mathcal{L}$. $R_{2^{\mathcal{L}}} : 2^{\mathcal{L}} \rightarrow \mathcal{S} \times \mathcal{S}$ is a function assigning to each concurrent set γ of $2^{\mathcal{L}}$ a relation over $\mathcal{S}$ (i.e., $R_{2^{\mathcal{L}}}(\gamma) \subseteq \mathcal{S} \times \mathcal{S}$, $\gamma \in 2^{\mathcal{L}}$). Note also that $R_{2^{\mathcal{L}}}$ for singleton concurrent sets behaves the same as $R_{\mathcal{L}}$ for

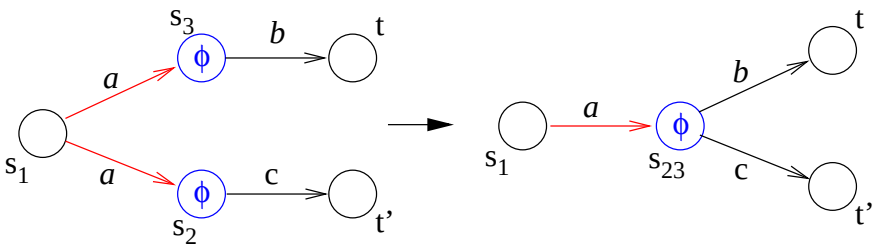

Figure 3: The intuition for the determinism in $\mathcal{C}\mu$.

actions of μ-calculus. For the sake of notation instead of $R_{2^{\mathcal{L}}}(\gamma)$ we write R_γ. In the case of singleton concurrent sets instead of $R_{\{a\}}$ we just write R_a when there is no chance of confusion with the one from propositional μ-calculus. Also, we often use as shorthand for a concurrent set inside dynamic operators just the syntax $[a, b]\varphi$ instead of $[\{a, b\}]\varphi$.

Non-determinism in action logics like PDL and consequently propositional μ-calculus refers to the actions. Actions are considered non-deterministic because from one world/state, by performing an action, the system may go to several other worlds/states.

On the other hand deterministic variants of the above logics have been investigated. Among the first approaches was DPDL of Ben-Ari, Halpern, and Pnueli [BAHP81] where the intuition is that an action started in the current state may terminate in only one final state. The determinism is naturally defined for atomic actions. Formally, the relation R_a that interprets the atomic action a becomes a *partial function* $\rho(a)$, i.e., for any $(s, t), (s, t') \in \rho(a)$ then $t = t'$. Naturally a compound action may have several ending worlds, both in the interpretation of the *actions as relations* [FL77] or the *actions as trajectories* [Pra78].

In the nice essay [PT91] on Combinatory Dynamic Logic (PDL is extended with nominals; which are special constant propositions valid in only one state) determinism is defined by an axiom:

$$(det) \quad \vdash \langle a\rangle\varphi \Rightarrow [a]\varphi$$

The intuition is that an action may end up in several worlds but in all the ending worlds we have the same set of propositions holding. This means that in a Kripke structure we can merge all the arrows labelled with our action into one arrow and all the states that the arrows end up in, into one state (for an example consider the picture in Figure 3). Note that the authors also relate the determinism to the atomic actions.

From the point of view of modelling contracts it is natural to adopt the deterministic variant of an action logic. Usually the aim of a contract clause

is to explicitly state what is the outcome of performing an action. Non-determinism is not desirable because we would be able to model actions which have no clear single outcome.

The determinism that we have presented above extends to the concurrent sets by requiring to have only one transition from one state labelled with a concurrent set. Formally we take the approach of DPDL and restrict $R_{2^{\mathcal{C}}}$ to assign to each concurrent set only partial functions (not relations). For example, if $(s,t),(s,t') \in R_{\{a,b\}}$, and $s,t,t' \in \mathcal{S}$ then $t = t'$. Note that if $(s,t) \in R_a$ and $(s,t') \in R_{\{a,b\}}$ it does not mean that for action a we have non-determinism. This is because one may either perform action a and have a formula holding after, or may perform the concurrent action $a\&b$ and have some other outcome (other formula holding) in the state after. For example one may consider $O(a\&b) \oplus O(a)$ to generate non-determinism. A closer analysis of the above example shows that it does not make sense to *choose* between $O(a\&b)$ and $O(a)$, since if it is *my* choice, then I would choose the lest restrictive for me (i.e. $O(a)$), and if the choice is external (or imposed) it may be the contrary.

Note that the action normal form ANF^+ defined in Section 5.2 merges together several arrows labelled with the same action into one arrow, which goes well with our deterministic variant of μ-calculus.

In order to translate obligation, permission and prohibition syntax of $\mathcal{CL}$ into the new logic we need to extend the propositional μ-calculus with a new set P_c of constant propositions. The constant propositions are interpreted, the same as the propositional variables of P, as a set of states where the constant proposition holds. We define the *obligation constants* $O_a \in P_c$ which are indexed by the atomic actions of $\mathcal{L}$. Similarly we define the *prohibition constants* $\mathcal{F}_a \in P_c$ which are also indexed by the atomic actions.

The intuition of the obligation constants is that when the system is in a state s and $\exists t \in \mathcal{S}$ with $(s,t) \in R_a$ and $t \in \|O_a\|^{\mathcal{T}}_{\mathcal{V}}$ then we may conclude that in the current state s the system has the obligation to execute action a.

A first reason for having a set of obligation constants indexed by the actions is that we want to capture in the logic the compositionallity of the obligation construct of the $\mathcal{CL}$ over the concurrent actions. Another reason for indexing the obligation constants is that in each state we need to know which incoming actions are obligation actions; i.e. if we would have only one constant proposition O denoting obligation then if O holds at a state t, and two actions $a = (s,t)$ and $b = (s',t)$ enter the state t then both actions have to be obligatory actions.

For the obligation and prohibition constants we choose to have a restriction on their semantics.

$$
\begin{array}{lrcl}
(1) & f^{\mathcal{T}}(O(a)) & = & \langle a\rangle O_a \\
(2) & f^{\mathcal{T}}(O(a\&b)) & = & \langle\{a,b\}\rangle(O_a \wedge O_b) \\
(3) & f^{\mathcal{T}}(\mathcal{C}_O \oplus \mathcal{C}_O) & = & f^{\mathcal{T}}(\mathcal{C}_O) \wedge f^{\mathcal{T}}(\mathcal{C}_O) \\
(4) & f^{\mathcal{T}}(P(\&_{i=1}^n a_i)) & = & \langle\{a_1,\ldots,a_n\}\rangle(\wedge_{i=1}^n \neg\mathcal{F}_{a_i}) \\
(5) & f^{\mathcal{T}}(\mathcal{C}_P \oplus \mathcal{C}_P) & = & f^{\mathcal{T}}(\mathcal{C}_P) \wedge f^{\mathcal{T}}(\mathcal{C}_P) \\
(6) & f^{\mathcal{T}}(F(\&_{i=1}^n a_i)) & = & [\{a_1,\ldots,a_n\}](\wedge_{i=1}^n \mathcal{F}_{a_i}) \\
(7) & f^{\mathcal{T}}(F(\delta) \vee [\beta]F(\delta)) & = & f^{\mathcal{T}}(F(\delta)) \vee f^{\mathcal{T}}([\beta]F(\delta))
\end{array}
$$

Table 3: The Translation Function for $\mathcal{C}_O, \mathcal{C}_P$ and $\mathcal{C}_F$

Definition 6.2 (constants incompatibility)**.** *We define the constant propositions $\mathcal{F}_a$ and the constant obligations O_a, with $a \in \mathcal{L}$ to be incompatible, meaning that their interpretations as sets of states must be disjoint:*

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

The intuition drawn from electronic contracts is that we want to disallow having in a certain world the obligation to do an action and prohibition of the same action. Note that the above definition gives the following natural result:

Proposition 6.1 (constants implication)**.** *We have the following implications holding:*

1. $O_a \Rightarrow \neg\mathcal{F}_a$
2. $\mathcal{F}_a \Rightarrow \neg O_a$

6.3 Translating the language into the logic

Because of the special status of the concurrent actions we choose to translate both $O(a)$ and $O(a\&b)$. Because of this and of the equation (5) of Section 5.3 we do not translate into $\mathcal{C}\mu$ the $\wedge$ conjunction over obligations. Nevertheless, we translate the *choice* and the *dynamic box*.

We consider a translation function $f^{\mathcal{T}}$ applied to formulas of $\mathcal{CL}$ which generates formulas of $\mathcal{C}\mu$.

Translation of the obligation to do an atomic action a is:

$$f^{\mathcal{T}}(O(a)) = \langle a\rangle O_a$$

Translation of the obligation to do both actions a and b *at the same time* uses the concurrent sets:

$$f^{\mathcal{T}}(O(a\&b)) = \langle\{a, b\}\rangle(O_a \wedge O_b)$$

Note that the conjunction $\wedge$ on the right side of the definition is the conjunction operator from propositional μ-calculus (with the usual interpretation).

The two translations above can be generalized and combined into the following concise notation:

$$f^{\mathcal{T}}(O(\&_{i=1}^{n} a_i)) = \langle\{a_1 \ldots a_n\}\rangle(\wedge_{i=1}^{n} O_{a_i}) \tag{14}$$

where O_{a_i} are the special constant propositions of $\mathcal{C}\mu$, and concurrency of only one atomic action (i.e. $\&_{i=1}^{1} a_i$) represents the execution of only that specific atomic action (a_1).

The translation of the exclusive or $\oplus$ over obligations is:

$$f^{\mathcal{T}}(\mathcal{C}_O \oplus \mathcal{C}_O) = f^{\mathcal{T}}(\mathcal{C}_O) \wedge f^{\mathcal{T}}(\mathcal{C}_O) \tag{15}$$

There is no translation for the conjunction operator $\wedge$ over obligations because this is handled by the rewriting rule (1) of Table 2.

The translation of the permission operator is similar to the translation of the obligation operator.

$$f^{\mathcal{T}}(P(\&_{i=1}^{n} a_i)) = \langle\{a_1 \ldots a_n\}\rangle(\wedge_{i=1}^{n} \neg\mathcal{F}_{a_i}) \tag{16}$$

And the translation of the $\oplus$ over permition is:

$$f^{\mathcal{T}}(\mathcal{C}_P \oplus \mathcal{C}_P) = f^{\mathcal{T}}(\mathcal{C}_P) \wedge f^{\mathcal{T}}(\mathcal{C}_P) \tag{17}$$

We need to translate both prohibition over atomic actions and prohibition over concurrent actions; i.e., $F(a)$ and $F(a\&b)$.

$$\begin{array}{c} f^{\mathcal{T}}(F(a)) = [a]\mathcal{F} \\ f^{\mathcal{T}}(F(a\&b)) = [\{a, b\}]\mathcal{F} \end{array} \tag{18}$$

The disjunction $\vee$ over prohibition is translated naturally to its corresponding operator of propositional μ-calculus.

$$f^{\mathcal{T}}(F(\alpha) \vee [\beta]F(\gamma)) = f^{\mathcal{T}}(F(\alpha)) \vee f^{\mathcal{T}}([\beta]F(\gamma)) \tag{19}$$

Regarding general contract clauses $\mathcal{C}$, the conjunction is translated as the corresponding conjunction operator of propositional μ-calculus, and *until* $\mathcal{U}$, and *next* $\bigcirc$ operators are translated using fix-point expressions.

$$
\begin{array}{rcl}
f^{\mathcal{T}}(\mathcal{C}_1 \wedge \mathcal{C}_2) & = & f^{\mathcal{T}}(\mathcal{C}_1) \wedge f^{\mathcal{T}}(\mathcal{C}_2) \\
f^{\mathcal{T}}(\bigcirc \mathcal{C}) & = & [\mathbf{any}] f^{\mathcal{T}}(\mathcal{C}) \\
f^{\mathcal{T}}(\mathcal{C}_1 \mathcal{U} \mathcal{C}_2) & = & \mu Z. f^{\mathcal{T}}(\mathcal{C}_2) \vee (f^{\mathcal{T}}(\mathcal{C}_1) \wedge [\mathbf{any}] Z \wedge \langle \mathbf{any} \rangle \top)
\end{array}
\tag{20}
$$

where **any** is the special action which is interpreted as the union of all actions in $\mathcal{L}$; the intuition is *doing any action*.

Because α inside the *dynamic box* $[\alpha]\mathcal{C}$ is a compound action obtained by applying the operators of the action algebra of Section 5.1 and in $\mathcal{C}\mu$ we have only concurrent sets of atomic actions, we have to give separate translations for each compound action. We give the translation of the compound actions under the *dynamic box* operator from $\mathcal{CL}$ into $\mathcal{C}\mu$ as follows:

$$
\begin{array}{lrcl}
(1) & f^{\mathcal{T}}([\&_{i=1}^{n} a_i]\mathcal{C}) & = & [\{a_1, \ldots, a_n\}] f^{\mathcal{T}}(\mathcal{C}) \\
(2) & f^{\mathcal{T}}([(\&_{i=1}^{n} a_i)\alpha]\mathcal{C}) & = & [\{a_1, \ldots, a_n\}] f^{\mathcal{T}}([\alpha]\mathcal{C}) \\
(3) & f^{\mathcal{T}}([\alpha + \beta]\mathcal{C}) & = & f^{\mathcal{T}}([\alpha]\mathcal{C}) \wedge f^{\mathcal{T}}([\beta]\mathcal{C}) \\
(4) & f^{\mathcal{T}}([\varphi?]\mathcal{C}) & = & f^{\mathcal{T}}(\varphi) \Rightarrow f^{\mathcal{T}}(\mathcal{C})
\end{array}
\tag{21}
$$

7 Properties of the Contract Language

We show here some of the good properties $\mathcal{CL}$ enjoys, as well as that the language avoids most important deontic paradoxes and the undesirable properties listed in Section 4.

Proposition 7.1 ensures that it is not needed to use negation on deontic operators, while Proposition 7.2 establishes the standard relation between obligations and permissions.

Proposition 7.1. *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 easy from the translation of the P and F operators into the logic and the duality of the μ-calculus operators $[\cdot]$ and $\langle \cdot \rangle$. □

Proposition 7.2. *The following statement is valid in* $\mathcal{CL}$*:*

$$O(\alpha) \Rightarrow P(\alpha)$$

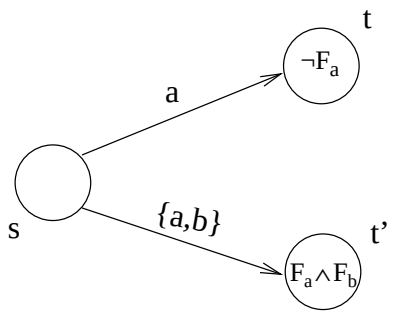

Figure 4: A model M in the $\mathcal{C}\mu$.

Proof: The proof follows from the similar translations of the O and P into the logic. Moreover, the proof makes use of the Definition 6.2 of the incompatibility of O_a and $\mathcal{F}_a$ constants. □

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

Proposition 7.3. *The following statement does not hold in* $\mathcal{CL}$*:*

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

Proof: We give a counter example to show that the implication is not possible. In our case we should 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.

Consider $(s,t) \in R_a$ and $(s,t') \in R_{\{a,b\}}$ with $t \notin \|\mathcal{F}_a\|^{\mathcal{T}}_{\mathcal{V}}$ and $t' \in \|\mathcal{F}_a\|^{\mathcal{T}}_{\mathcal{V}} \cap \|\mathcal{F}_b\|^{\mathcal{T}}_{\mathcal{V}}$. Consider the model M in Figure 4 which has states $\mathcal{S} = \{s,t,t'\}$ two relations: one for action a, $R_a = \{(s,t)\}$ and one 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. □

Proposition 7.4. *The following statement does not hold in* $\mathcal{CL}$*:*

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

Proof: The proof is based again on giving a counterexample. We change the example of Proposition 7.3 such that $t \in \|\mathcal{F}_a\|^{\mathcal{T}}_{\mathcal{V}}$ and $t' \notin \|\mathcal{F}_a\|^{\mathcal{T}}_{\mathcal{V}}$. M is in this case a model of the first formula but is not a model of the second formula. □

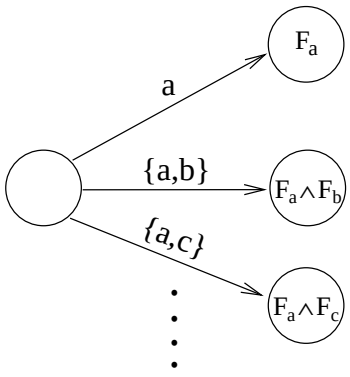

Figure 5: A model in $\mathcal{C}\mu$ for the $\mathcal{CL}$ prohibition expression $F(a)$.

Remark: We may give an alternative translation of the prohibition operator F so that the above implication holds in $\mathcal{CL}$. The translation we give for $F(a)$ respects equations (2) and (3) and represents also $F(a\&b)$:

$$f^{\mathcal{T}}(F(a)) = \bigwedge_{\gamma \subseteq \mathcal{L}} [\gamma](\wedge_{a_i \in \gamma} \mathcal{F}_{a_i}) \tag{22}$$

where $\mathcal{F}_{a_i}$ are the special constant propositions and γ is a concurrent set which contains action a, i.e., $\gamma = \{a\} \cup \gamma', \quad \gamma' \subseteq \mathcal{L} \setminus \{a\}$. For a pictured intuition of this translation consider Figure 5.

If we were to consider only one constant proposition $\mathcal{F}$ instead of the actions indexed constants $\mathcal{F}_a$ then the translation above is more concise and also respects the above implication and equations (2) and (3).

$$f^{\mathcal{T}}(F(a)) = \bigwedge_{\gamma \subseteq \mathcal{L}} [\gamma]\mathcal{F} \tag{23}$$

Note that this translation of prohibition goes well with the desiderata from Broersen et al. [BWM01]. If $F(a\&b)$ than we can not say that $F(a)$ but we may conclude that we are forbidden to do any other concurrent actions which involves the $a\&b$.

Proposition 7.5. *The following statements do not hold in $\mathcal{CL}$:*

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

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

Proof: Proof proceeds similar to the proofs of the propositions above by giving a counterexample. □

7.1 Paradoxes

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

Proposition 7.6. *Ross's paradox does not hold in $\mathcal{CL}$.*

Proof: Basically, Ross's paradox says that it is counter intuitive to have $O(a) \Rightarrow O(a+b)$; i.e., *Obligation to drink implies obligation to drink or to kill.* In $\mathcal{CL}$ this inference is not possible. The first formula is translated into $\mathcal{C}\mu$ as $\langle a\rangle O_a$. For the second formula we have $O(a+b) \equiv O(a) \oplus O(b) \overset{f^{\mathcal{T}}}{=} \langle a\rangle O_a \wedge \langle b\rangle O_b$. We have in the logic that $\langle a\rangle O_a \not\Rightarrow \langle a\rangle O_a \wedge \langle b\rangle O_b$. □

Proposition 7.7. *The Free Choice Permission paradox does not exist in $\mathcal{CL}$.*

Proof: The Free Choice Permission paradox basically says that from having one permission we may infer that we have any permission. That is: $P(a) \Rightarrow P(a+b)$ or $P(a) \Rightarrow P(a) \wedge P(b)$.

Neither of the two implications hold in our approach. The second one is obvious. The first one is based on the second one because $P(a+b) \equiv P(a) \oplus P(b)$ which translates in the logic with the conjunction operator. □

Proposition 7.8. *Sartre's Dilemma is not expressable in our approach.*

Proof: Sartre's dilemma can be rewritten in contracts terminology as: *Obliged to meet John and Forbidden to meet John.* This is formally written in $\mathcal{CL}$ as $O(a) \wedge F(a)$ which is a well formed formula. The translation into $\mathcal{C}\mu$ would result in a contradiction because we would have a state t with $(s,t) \in R_a$ and $t \in \|\mathcal{F}_a\|^{\mathcal{T}}_{\mathcal{V}}$ and $t \in \|O_a\|^{\mathcal{T}}_{\mathcal{V}}$. This means that $\|\mathcal{F}_a\|^{\mathcal{T}}_{\mathcal{V}} \cap \|O_a\|^{\mathcal{T}}_{\mathcal{V}} \neq \emptyset$ which is a contradiction with the semantics of the two constant propositions in the logic (see Definition 6.2). So this paradox is dealt with at the semantic level, in $\mathcal{C}\mu$. □

Proposition 7.9. *The Good Samaritan paradox can not be expressed like in SDL, which means we do not have this paradox.*

Proof: The Good Samaritan paradox uses *ought-to-be* and is more delicate to transform it into our *ought-to-do* approach. The transformation looks like: $\psi \Rightarrow O(h)$ which means that *If Smith has been robbed then John is obliged to help Smith.* Where ψ is *Smith has been robbed*, $\Rightarrow$ is $if \ldots then$, and h is the action *John helps Smith.* We can not express in $\mathcal{CL}$ obligation

over conjunction of two actions that are not performed concurrently as this paradox is expressed in SDL; i.e., we cannot express $O(a \wedge b)$. Also, with our representation of the paradox we cannot infer that ψ holds; i.e., infer that *Smith has been robbed.* □

Proposition 7.10. *The Chisholm's paradox is avoided in* $\mathcal{CL}$.

Proof: The propositions of the Chisholm's paradox are expressed in $\mathcal{CL}$ as:

1. $O(a)$
2. $[a]O(b)$
3. $[\overline{a}]O(\overline{b})$

Note first that formulas (1) and (3) give the CTD formula $O_{\varphi}(a)$ of $\mathcal{CL}$ where $\varphi = O(\overline{b})$. The problem in SDL was that one may infer both $O(b)$ and $O(\overline{b})$ holding in the same world. This is not our case because $O(b)$ holds only after doing action a, where $O(\overline{b})$ holds only after doing the contradictory action $\overline{a}$. In the model of the above representation we can not have in the same world both $O(b)$ and $O(\overline{b})$.

□

Proposition 7.11. *The Gentle Murderer paradox is avoided in* $\mathcal{CL}$.

Proof: The propositions of the Gentle Murderer paradox are expressed in $\mathcal{CL}$ as:

1. $F(a)$
2. $[a]O(b)$

Note first that the above two formulas give the CTP formula $F_{\varphi}(a)$ where $\varphi = O(b)$. The problem in the paradox comes from the fact that in SDL it is possible to express the natural implication $b \Rightarrow a$ which in common language is *If John kills the mother gently then it implies that John kills the mother.* This is not the case in $\mathcal{CL}$ because we do not have implication among actions.

On the other hand we could consider that the action of killing gently implies the action of killing by giving a formula in $\mathcal{CL}$ which represents implication of actions:

$$[b]\varphi \Rightarrow [a]\varphi \tag{24}$$

The expression above intuitively says that whenever after executing action b and formula φ holds then it must be the case that whenever after executing

the action a the same formula φ holds. In other words all the effects of action b are also the effects of action a but there may be effects of action a that are not effects of action b.

Still with this definition of implication among actions we can not infer $O(b) \Rightarrow O(a)$, which in SDL lead to the problem of the paradox. This is because by considering action b to imply action a we have the following: $O(b) \overset{f^{\mathcal{T}}}{=} \langle b \rangle O_b \overset{(24)}{\Rightarrow} \langle a \rangle O_b \neq O(a)$ □

8 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.

We here formalize this partial contract, showing the $\mathcal{CL}$ formula for each of the five clauses above. Let us first define the different propositions and actions:

φ = the Internet traffic is high
p = client pays x\$
d = client delays payment
n = client notifies by e-mail
l = client downs 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 following is the contract written in $\mathcal{CL}$:

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

Remarks

1. The formulas 2. and 3. may be combined in a single formula using CTDs:

   23 $\Box([d,n](O_{\varphi}(l) \wedge [l]\Diamond(O(p) \wedge [p]O(p))$ where $\varphi = O(p) \wedge [p]O(p) \wedge [p \cdot p]O(p)$.

2. Formulas 2 and 3 are rather long because we can not represent in $\mathcal{CL}$ quantitative information like *pay two times* $(2 * x\$)$. It might be more natural to use the & operator over actions with the same intuition as in logics of resources (e.g. linear logic [Gir87]) and for *obliged to pay twice* we could write in $\mathcal{CL}$ $O(p\&p)$ instead of $O(p) \wedge [p]O(p)$. Formulas 2 and 3 above would become:

   2' $\Box([d,n](O(l) \wedge [l]\Diamond(O(p\&p))))$
   3' $\Box([\{d,n\} \cdot \bar{l}\,](\Diamond O(p\&p\&p)))$

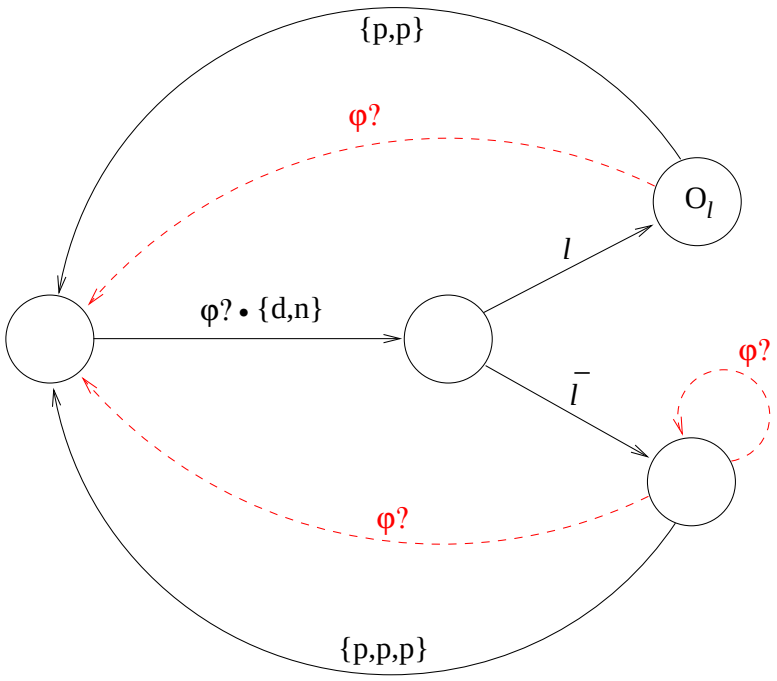

Figure 6: A model for statements 2' and 3' of the contract example.

For these two formulas written in this concise syntax we give the example model in Figure 6. The model as it is allows unwanted traces which are pictured in dashed labeled arrows. A discussion and a solution to this follows.

3. The above example shows the importance of being able to model check a contract. Notice that the contract allows the client to go from low to high Internet traffic many times and pay the penalty ($2 * x\$$) only once. The problem is that after the client downs the Internet traffic, he might get a high traffic again and delay 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 delays 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:

   2" $\Box([d,n](O(l) \wedge [l]\neg\varphi\, \mathcal{U}\, (O(p) \wedge [p]O(p)))$

   3" $\Box([\{d,n\} \cdot\ \bar{l}\ ](\neg\varphi\, \mathcal{U}\, (O(p) \wedge [p]O(p) \wedge [p \cdot p]O(p)))$

   In Figure 7 we give a model for the new statements 2" and 3". Note that the dashed arrows from the previous model have changed into the

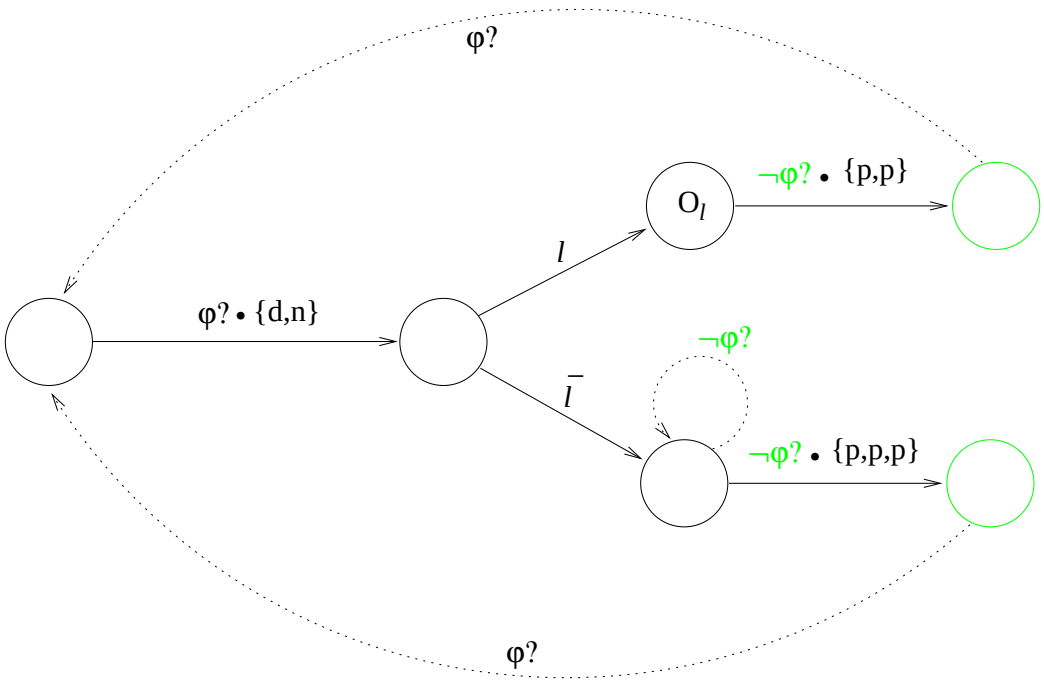

Figure 7: A model for the corrected example.

dotted arrows, and we have also added the negative guards $\neg\varphi?$ so that the until $\mathcal{U}$ formulas are satisfied. Also the change in the statements required two more states to be added to the model.

Model checking is out of the scope of this paper and will be consider in future works.

4. Notice that our contract language lacks the possibility of expressing timing constraints and 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 special variables and simulating 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.

9 Other Approaches

In this section we contrast our approach in detail with the work by Broersen *et al* [BWM01]. 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.

1. There is no notion of *contract language*, only characterization of obligation, permission and prohibition in the logic.

2. - The only deontic primitive is permission over atomic actions;
   - Obligation is defined as an infinite conjunction of negation of permissions 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.
   - Obligation ($O(\cdot)$) and prohibition ($F(\cdot)$) are defined in terms of permissions (e.g. $F(\alpha) = \neg P(\alpha)$).

3. 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.

4. Obligation on the choice of actions is not compositional; it is compositional in our case.

5. There is no conjunction over actions, i.e., it is not possible to express concurrent actions, which is the case in our approach.

6. The approach uses disjunction over actions. We have decided to use the exclusive or instead.

7. 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, and we have a special definition for negation of actions.

8. CTDs cannot be defined unless an extension of the μ^a-calculus is considered. In our setting both CTDs and CTPs are easily defined.

9. 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.

The idea of using a propositional constant in an action-based logic for giving semantics to the deontic notions was first presented in [Mey88], where the special constant V was added to denote an "undesirable state-of-affairs" in the current state.

10 Conclusion

In this paper we have presented a formal language for writing contracts, and provided a formal semantics through the translation of the language into a variant of the propositional μ-calculus extended with concurrent actions. The language avoids most of the classical paradoxes, and enjoys all the nice properties listed in Section 4. 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.

10.1 Further 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 of performing 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 works. Concerning actions, we got inspiration from the works on dynamic logics [Pra76]. 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. 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 [KP83], has a complete axiomatic system [Wal95], and a complete Gentzen-style deduction system [Wal93]. We

want to explore the proof system of the logic, and to extend existing model checkers [Bie97, MS03] to analyze contracts as mentioned in the remarks of our example (Section 8).

We would like to be able to extract a contract monitor from the Kripke structure of a given contract. Notice that this is not easy in general since there are many models for a particular contract. As an example consider a contract containing the following clauses:

$$\Box(\phi_1 \Rightarrow \Diamond O(p_1))$$
$$\Box(\phi_2 \Rightarrow \Diamond O(p_2))$$

where ϕ_1 and ϕ_2 represent conditions on receiving a service with certain quality q_1 and q_2 respectively. Depending to the quality of the service we must pay a difference price p_1 or p_2. It is possible to get a service with quality q_1 once, and then with quality q_2 n times. The above conditional obligations establish that we must pay p_1 only once and n times p_2. This shows that the monitor should keep track of the correlation between the different occurrences of the services and matching with the corresponding payment. Though the run-time monitor is much more complex than the Kripke structure, we believe that the latter can be the bases for constructing the monitor, which would have to be enhanced with some counters and maybe additional data structure. The addition of real-time (e.g. clocks) will simplify many of these kind of problems.

Acknowledgements

We would like to thank Joseph Okika for his comments on an early draft of this paper.

References

[Aag01] J. Aagedal. *Quality of Service Support in Development of Distributed Systems*. PhD thesis, Dept. of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo, 2001.

[AtC06] Carlos Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*. Elsevier, 2006.

[BAHP81] Mordechai Ben-Ari, Joseph Y. Halpern, and Amir Pnueli. Finite models for deterministic propositional dynamic logic. In

Shimon Even and Oded Kariv, editors, *8th Colloquium On Automata, Languages and Programming (ICALP'81)*, volume 115 of *Lecture Notes in Computer Science*, pages 249–263. Springer, 1981.

[Bie97] Armin Biere. μcke - efficient μ-calculus model checking. In Orna Grumberg, editor, *9th International Conference on Computer Aided Verification (CAV'97)*, volume 1254 of *Lecture Notes in Computer Science*, pages 468–471. Springer, 1997.

[BJP99] Antoine Beugnard, Jean-Marc Jézéquel, and Noël Plouzeau. Making components contract aware. *IEEE Computer*, 32(7):38–45, 1999.

[BS01] Julian Bradfield and Colin Stirling. Modal logics and mu-calculi: an introduction. In *Handbook of Process Algebra*, pages 293–330. Elsevier, 2001.

[BV03] Philippe Balbiani and Dimiter Vakarelov. PDL with intersection of programs: A complete axiomatization. *Journal of Applied Non-Classical Logics*, 13(3-4):231–276, 2003.

[BWM01] Jan Broersen, Roel Wieringa, and John-Jules Ch. Meyer. A fixed-point characterization of a deontic logic of regular action. *Fundam. Inf.*, 48(2-3):107–128, 2001.

[Chi63] Roderick M. Chisholm. Supererogation and offence: A conceptual scheme for ethics. *Ratio Juris*, 5:1–14, 1963.

[CKS96] Ernie Cohen, Dexter Kozen, and Frederick Smith. Complexity of kleene algebra with tests, the. Technical report, Cornell University, Ithaca, NY, USA, 1996.

[Coh94] Ernie Cohen. Using kleene algebra to reason about concurrency control. Technical report, Telcordia, Morristown, N.J., 1994.

[Con71] John Horton Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, London, UK, 1971.

[Das00] Aspassia Daskalopulu. Model Checking Contractual Protocols. In Leenes Breuker and Winkels, editors, *Legal Knowledge and Information Systems, JURIX 2000: The 13th Annual Conference*, Frontiers in Artificial Intelligence and Applications Series, pages 35–47. IOS Press, 2000.

[DKR04] Hasan Davulcu, Michael Kifer, and I. V. Ramakrishnan. CTR-S: A Logic for Specifying Contracts in Semantic Web Services. In *Proceedings of WWW2004*, pages 144–153, May 2004.

[DM01] Aspassia Daskalopulu and T. S. E. Maibaum. Towards Electronic Contract Performance. In *Legal Information Systems Applications, 12th International Conference and Workshop on Database and Expert Systems Applications*, pages 771–777. IEEE C.S. Press, 2001.

[Egg63] L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Mathematical Journal*, 10(4):385–397, 1963.

[FL77] Michael J. Fischer and Richard E. Ladner. Propositional modal logic of programs. In *9th ACM Symposium on Theory of Computing (STOC'77)*, pages 286–294. ACM, 1977.

[For84] J. W. Forrester. Gentle murder or the adverbial samaritan. *Journal of Philosophy*, 1981:193–197, 1984.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gov05] Guido Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14:181–216, 2005.

[GR06] Guido Governatori and Antonino Rotolo. Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations. *Australatian Journal of Logic*, 4:193–215, 2006.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[Koz80] Dexter Kozen. A representation theorem for models of *-free pdl. In J. W. de Bakker and Jan van Leeuwen, editors, *7th Colloquium on Automata, Languages and Programming (ICALP'80)*, volume 85 of *Lecture Notes in Computer Science*, pages 351–362. Springer, 1980.

[Koz81] Dexter Kozen. On the duality of dynamic algebras and kripke models. In *Logic of Programs, Workshop*, volume 125 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1981.

[Koz83] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[Koz90] Dexter Kozen. On kleene algebras and closed semirings. In Branislav Rovan, editor, *Mathematical Foundations of Computer Science (MFCS'90)*, volume 452 of *Lecture Notes in Computer Science*, pages 26–47. Springer, 1990.

[Koz94] Dexter Kozen. A completeness theorem for kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[Koz97] Dexter Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems (TOPLAS'97)*, 19(3):427–443, 1997.

[KP83] Dexter Kozen and Rohit Parikh. A decision procedure for the propositional μ-calculus. In Edmund M. Clarke and Dexter Kozen, editors, *4th Workshop on Logics of Programs*, volume 164 of *Lecture Notes in Computer Science*, pages 313–325. Springer, 1983.

[Mal26] Ernst Mally. *Grundgesetze des Sollens. Elemente fer Logik des Willens*. Graz: Leuschner & Lubensky, 1926.

[McN06] Paul McNamara. Deontic logic. In Dov M. Gabbay and John Woods, editors, *Handbook of the History of Logic*, volume 7, pages 197–289. North-Holland Publishing, 2006.

[Mey88] J.-J. Ch. Meyer. 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.

[MJSSW04] Carlos Molina-Jiménez, Santosh K. Shrivastava, Ellis Solaiman, and John P. Warne. Run-time Monitoring and Enforcement of Electronic Contracts. *Electronic Commerce Research and Applications*, 3(2), 2004.

[MS03] Radu Mateescu and Mihaela Sighireanu. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. *Science of Computer Programming*, 46(3):255–281, 2003.

[PDK05] Adrian Paschke, Jens Dietrich, and Karsten Kuhla. A Logic Based SLA Management Framework. In *4th Semantic Web Conference (ISWC 2005)*, 2005.

[Pnu77] Amir Pnueli. Temporal logic of programs, the. In *Proceedings of the 18th IEEE Symposium On the Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Computer Society Press, 1977.

[Pra76] Vaughan R. Pratt. Semantical considerations on floyd-hoare logic. In *IEEE Symposium On Foundations of Computer Science (FOCS'76)*, pages 109–121, 1976.

[Pra78] Vaughan R. Pratt. A practical decision method for propositional dynamic logic: Preliminary report. In *10th ACM Symposium on Theory of Computing (STOC'78)*, pages 326–337. ACM Press, 1978.

[Pra79] Vaughan R. Pratt. Process logic. In *6th Symposium on Principles of Programming Languages (POPL'79)*, pages 93–100. ACM, 1979.

[Pra80] Vaughan R. Pratt. Dynamic algebras and the nature of induction. In *12th ACM Symposium on Theory of Computing (STOC'80)*, pages 22–28. ACM, 1980.

[Pri58] A. N. Prior. Escapism: The logical basis of ethics. In A. I. Melden, editor, *Essays in Moral Philosophy*, volume 33 of *Journal Symbolic Logic*, pages 135–146. Association for Symbolic Logic, 1958.

[PS96] Henry Prakken and Marek Sergot. Contrary-to-duty obligations. *Studia Logica*, 57(1):91–115, 1996.

[PS97] Henry Prakken and Marek Sergot. Dyadic deontic logic and contrary-to-duty obligation. In Donald Nute, editor, *Defeasible Deontic Logic*, pages 223–262. Kluwer Academic Publishers, 1997.

[PT85] Solomon Passay and Tinko Tinchev. PDL with data constants. *Information Processing Letters*, 20:35–41, 1985.

[PT91] Solomon Passay and Tinko Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.

[Ros41] Alf Ross. Imperatives and logic. *Theoria*, 7:53–71, 1941.

[SG05] Ishu Song and Guido Governatori. Nested rules in defeasible logic. In *RuleML*, volume 3791 of *Lecture Notes in Computer Science*, pages 204–208, 2005.

[TP05] Vladimir Tosic and Bernard Pagurek. On Comprehensive Contractual Descriptions of Web Services. In *IEEE International Conference on e-Technology, e-Commerce, and e-Service*, pages 444–449. IEEE, 2005.

[Wal93] Igor Walukiewicz. *A Complete Deductive System for the μ-Calculus*. PhD thesis, Institute of Informatics, Warsaw University, 1993.

[Wal95] Igor Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. In *10th IEEE Symposium on Logic in Computer Science (LICS'95)*, pages 14–24. IEEE Computer Society, 1995.

[Wri51] Georg Henrik Von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

[Wri99] Georg Henrik Von Wright. Deontic logic: A personal view. *Ratio Juris*, 12(1):26–38, 1999.