Abstract Specification of Legal Contracts [*]

Cristian Prisacariu
Department of Informatics, University of Oslo,
P.O. Box 1080 Blindern, N-0316 Oslo, Norway.
cristi@ifi.uio.no

Gerardo Schneider
Department of Informatics, University of Oslo,
P.O. Box 1080 Blindern, N-0316 Oslo, Norway.
gerardo@ifi.uio.no

ABSTRACT

The paper presents an action-based formal language called $\mathcal{CL}$ for abstract specification of legal contracts. The purpose of the language is to be used to reason about legal contracts (and electronic contracts on the long run). $\mathcal{CL}$ combines the legal notions obligation, permission, and prohibition from deontic logic with the action modality of propositional dynamic logic (PDL). The deontic modalities are applied only over actions, thus following the ought-to-do approach. The language includes a synchrony operator to model "actions performed at the same time", and a special complementation operation to encode the violation of obligations. The language has a formal semantics in terms of normative structures, specially defined to capture several natural properties of legal contracts. We focus on the informal presentation of the choices made when designing $\mathcal{CL}$, and its semantics.

1. INTRODUCTION

Much research has been invested into giving a formalization of legal contracts, and also into providing a machine readable language for specifying contracts. Among the many approaches the most promising are the ones based on variants of deontic logic. Such a formal language is desired for doing static (like model-checking) or dynamic (like run-time monitoring) analysis of (abstractions of) contracts. Moreover, the automation of the negotiation process for contracts becomes a feasible goal.

We present a formal specification language for contracts, called $\mathcal{CL}$, able to represent obligations, permissions and prohibitions, as well as what happens when obligations or prohibitions are not respected. We focus on the informal explanation of the language (though some formalization is presented as well), and in particular on its design decisions.

The goal of $\mathcal{CL}$ is to faithfully capture several concepts used in the context of legal contracts, to avoid deontic paradoxes, and to preserve many of the natural properties relevant to legal contracts. Since our main objective is to analyze contracts through formal verification techniques, we aim at a tractable language (i.e. decidable and with manageable complexities). In this way, we can use formal tools like model-checking and run-time monitoring.

[*]Partially supported by the Nordunet3 project "COSoDIS – Contract-Oriented Software Development for Internet Services" (`http://www.ifi.uio.no/cosodis/`).

$$\begin{array}{lcl}
\mathcal{C} & := & \phi \mid O_{\mathcal{C}}(\alpha) \mid P(\alpha) \mid F_{\mathcal{C}}(\alpha) \mid \mathcal{C} \to \mathcal{C} \mid [\beta]\mathcal{C} \mid \perp \\
\alpha & := & \mathbf{0} \mid \mathbf{1} \mid a \mid \alpha \times \alpha \mid \alpha \cdot \alpha \mid \alpha + \alpha \\
\beta & := & \mathbf{0} \mid \mathbf{1} \mid a \mid \beta \times \beta \mid \beta \cdot \beta \mid \beta + \beta \mid \beta^* \mid \mathcal{C}?
\end{array}$$

Table 1: Syntax of the contract language $\mathcal{CL}$.

2. THE CONTRACT LANGUAGE

$\mathcal{CL}$ is an *action-based* logic for specifying electronic and legal contracts with model theoretic semantics in terms of normative structures. The syntax of $\mathcal{CL}$ is defined by the grammar in Table 1. Formal definition and technicalities of $\mathcal{CL}$ are published in [6]. In what follows we provide intuitions of the various features and syntax of $\mathcal{CL}$. Some works related to $\mathcal{CL}$ are [1, 3, 4, 7], which also consider deontic modalities (i.e. O, P, and F) applied over actions.

In $\mathcal{CL}$ the deontic modalities are applied only over actions instead of over formulas of the logic. This is known as the ought-to-do approach to deontic logic as opposed to the more classic ought-to-be approach of Standard Deontic Logic (SDL). The ought-to-do approach has been advocated by von Wright [8] which argued that deontic logic would benefit from a "foundation of actions", since many of the philosophical paradoxes of SDL would be removed.

The structured action of $\mathcal{CL}$ are constructed from a finite set of basic actions $a \in \mathcal{A}_B$, $\mathbf{1}$, and $\mathbf{0}$ with the regular operators $+$ and $\cdot$ (for *choice* and *sequence*).[1] We add a concurrency operator $\times$ to model that two actions are *done at the same time*. The model of concurrency that we addopt is the synchrony model of Milner's SCCS [5]. Synchrony is easy to integrate with the other regular operations on actions. In the SCCS calculus the *synchrony* model considers that each of the two concurrent systems instantaneously perform a single action at each time instant. It takes the assumption that time is discrete and that basic actions represent the time step. Moreover, at each time step all possible actions are performed, i.e. the system is considered *eager*. For this reason if at a time point an obligation to perform an action is enabled then this action must be immediately executed so that the obligation is not violated. Because of the assump-

[1]See discussion in [6] for why we exclude the Kleene star *.

tion of an eager behavior the scope of the obligations (and of the other deontic modalities too) is immediate, making them transient obligations which are enforced only in the current world. One can get persistent obligations using temporal logic modalities like the *always*, which are expressed in $\mathcal{CL}$ with the PDL modality $[\beta]$.

We define a *conflict relation* $\#_\mathcal{C}$ over actions which represents the fact that two actions cannot be done at the same time. This is necessary for detecting (and for ruling out) a first kind of *conflicts* in contracts: "Obligatory to go west and obligatory to go east". The second kind of conflicts that the $\mathcal{CL}$ rules out are: "Obligatory to go west and forbidden to go west" which is standard for a deontic logic.

$\mathcal{CL}$ defines an *action complement* operation which encodes the violation of an obligation. Obligations (and prohibitions) can be violated by *not* doing the obligatory action (or *doing* the forbidden action). Intuitively action complement $\overline{\alpha}$ is a function which returns the action given by *all* the immediate actions that *take us outside* α [1].

$\mathcal{CL}$ combines notions from both deontic logic and PDL therefore, considering the *dynamic modality* $[\beta]\mathcal{C}$ which is read as: "after the action β is performed then the contract clause $\mathcal{C}$ should hold in the next world". A first difference from the standard PDL is that we consider *deterministic* actions. This is natural and desired in legal contracts as opposed to the programming languages community where nondeterminism is an important notion. In contracts the outcome of an action like "deposit 100$ in the bank account" is uniquely determined. A second feature of $\mathcal{CL}$ is the introduction of the synchrony operation $\times$ on the actions inside the dynamic modality. Therefore, $\mathcal{CL}$ can reason about synchronous actions and is related to extensions of PDL which can reason about concurrent actions like PDL$^\cap$ with intersection, concurrent PDL, or dynamic deontic logic DDL [4].

In $\mathcal{CL}$ *conditional obligations* (or prohibitions) can be of two kinds. (a) The first kind is given with the propositional implication: $\mathcal{C} \rightarrow O_\mathcal{C}(\alpha)$; "If the internet traffic is high then the Client is obliged to pay". (b) The second kind is given with the dynamic box modality: $[\beta]O_\mathcal{C}(\alpha)$; "After receiving necessary data the Provider is obliged to offer password".

Regarding the deontic modalities $O_\mathcal{C}$ and $F_\mathcal{C}$, $\mathcal{CL}$ includes directly in the definition of the obligation and prohibition the *reparations* $\mathcal{C}$ which hold in case of violations. This models a notion of contrary-to-duty obligation (CTD) which is in contrast with the classical notion of CTD as found in the SDL literature [2]. In SDL, what we call reparations are secondary obligations which hold in the same world as the primary obligation. In our setting where the action changes the world one can see a violation of an obligation (or prohibition) only after the action is performed and thus the reparations are enforced in the next (reachable) world.

The *semantics* of $\mathcal{CL}$ is given in terms of *normative structures* and it is specially defined to capture several properties which we find natural for legal contracts.[2] One of the interesting properties particular to $\mathcal{CL}$ is $O_\mathcal{C}(\alpha) \wedge O_\mathcal{C}(\beta) \rightarrow O_\mathcal{C}(\alpha \times \beta)$ (ex.: "Obliged to delay payment" and also "Obliged to notify by e-mail" then we can conclude that "Obliged to delay and notify at the same time"). Also particular to $\mathcal{CL}$ is $F_\mathcal{C}(\alpha) \rightarrow F_\mathcal{C}(\alpha \times \beta)$ (prohibition of an action α implies that any action that "includes" it is prohibited).

Other properties come from SDL, like: $O_\mathcal{C}(\alpha) \rightarrow P(\alpha)$ or $P(\alpha) \rightarrow \neg F_\mathcal{C}(\alpha)$. Moreover, $\mathcal{CL}$ has properties from DDL, like: $O_\mathcal{C}(\alpha \cdot \beta) \leftrightarrow O_\mathcal{C}(\alpha) \wedge [\alpha]O_\mathcal{C}(\beta)$, $F(\alpha \cdot \beta) \leftrightarrow F(\alpha) \vee \langle\alpha\rangle F(\beta)$, or $F_\mathcal{C}(\alpha + \beta) \leftrightarrow F_\mathcal{C}(\alpha) \wedge F_\mathcal{C}(\beta)$ (and two similar properties for permissions).

In the design decisions for $\mathcal{CL}$ we give special attention to what we call *unwanted implications*.[3] Some are related to synchrony: $F_\mathcal{C}(\alpha \times \beta) \not\rightarrow F_\mathcal{C}(\alpha)$, $P(\alpha \times \beta) \not\rightarrow P(\alpha)$, $O_\mathcal{C}(\alpha) \not\rightarrow O_\mathcal{C}(\alpha \times \beta)$, or $O_\mathcal{C}(\alpha \times \beta) \not\rightarrow O_\mathcal{C}(\alpha)$; whereas other relate to choices of actions: $O_\mathcal{C}(\alpha) \not\rightarrow O_\mathcal{C}(\alpha + \beta)$, $O_\mathcal{C}(\alpha + \beta) \not\rightarrow O_\mathcal{C}(\alpha)$, $O_\mathcal{C}(\alpha + \beta) \not\rightarrow O_\mathcal{C}(\alpha \times \beta)$ or $O_\mathcal{C}(\alpha \times \beta) \not\rightarrow O_\mathcal{C}(\alpha + \beta)$.

With the formal apparatus of $\mathcal{CL}$ one could automatically answer questions like (after writing a contract as a $\mathcal{CL}$ formula): *(a)* Are there superfluous clauses? That is, could we simplify the contract if one clause entails another, or if some clauses would never be enacted due to unreachable conditions? *(b)* What are the client's (provider's) obligations or rights? *(c)* Are client's (provider's) interests met by the contract? The interests of one party (and not only) are expressed as properties about the contract. These can be written in $\mathcal{CL}$ or temporal logic for that matter, and properties may be checked on the abstract model of the contract. Otherwise, one could do inference reasoning using a proof system for $\mathcal{CL}$. *(d)* Is the contract contradiction-free?

3. FINAL REMARKS

With $\mathcal{CL}$ our main interest is not the development of yet-another-language for the specification and representation of normative notions, but a formal system (based on logic) to reason automatically about contracts. $\mathcal{CL}$ is not too expressive as to capture all the features of legal contracts, but it is expressive enough as to capture many useful abstractions of legal contracts. This lack of expressiveness should not be seen as negative since it will allow us to perform automatic analysis of some properties of contracts, as we have presented in this paper.

4. REFERENCES

[1] J. Broersen, R. Wieringa, and J.-J. C. Meyer. A fixed-point characterization of a deontic logic of regular action. *Fundam. Inf.*, 48(2-3):107–128, 2001.

[2] J. Carmo and A. Jones. Deontic logic and contrary-to-duties. In *Handbook of Philosophical Logic*, pages 265–343. Kluwer, 2002.

[3] P. F. Castro and T. Maibaum. A complete and compact propositional deontic logic. In *ICTAC'07*, volume 4711 of *LNCS*, pages 109–123, 2007.

[4] J.-J. C. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame J. Formal Logic*, 29(1):109–136, 1988.

[5] R. Milner. Calculi for synchrony and asynchrony. *Theorethical Computer Science*, 25:267–310, 1983.

[6] C. Prisacariu and G. Schneider. CL: An Action-based Logic for Reasoning about Contracts. In *WOLLIC'09*, volume 5514 of *LNCS*. Springer, 2009.

[7] R. Van der Meyden. Dynamic logic of permission, the. In *LICS'90*, pages 72–78. IEEE, 1990.

[8] G. H. Von Wright. *An Essay in Deontic Logic and the General Theory of Action.* North Holland, 1968.

[2]These intuitive properties are validities in the $\mathcal{CL}$ logic.

[3]The unwanted implications ($\not\rightarrow$) are non-validities in $\mathcal{CL}$.