$\mathcal{CL}$: An Action-based Logic for Reasoning about Contracts [*]

Cristian Prisacariu and Gerardo Schneider

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

Abstract. This paper presents a new version of the $\mathcal{CL}$ contract specification language. $\mathcal{CL}$ combines deontic logic with propositional dynamic logic but it applies the modalities exclusively over structured actions. $\mathcal{CL}$ features synchronous actions, conflict relation, and an action negation operation. The $\mathcal{CL}$ version that we present here is more expressive and has a cleaner semantics than its predecessor. We give a direct semantics for $\mathcal{CL}$ in terms of *normative structures*. We show that $\mathcal{CL}$ respects several desired properties from legal contracts and is decidable. We relate this semantics with a trace semantics of $\mathcal{CL}$ which we used for run-time monitoring contracts.

1 Introduction

Internet-based negotiation and contracting is becoming more diverse and complex in the e-business and e-government environments. This calls for a more formal apparatus which can be used by the computer to automate and help in some of the tasks involved in e-contracting; e.g. detection of contradictions and inconsistencies in contracts, identification of superfluous clauses, and checking some desired properties on a contract. This contracting style found in e-business and virtual organizations (and inspired from legal contracts) can also be used in service oriented architectures, component based systems [1], and agent societies [2]. In these areas contracts are used to regulate the interaction and exchanges between the parties involved (being that services, components, or agents).

Much research has been invested into giving a formalization of contractual clauses, and also into providing a machine readable language for specifying contracts. 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 becomes a feasible goal. The most promising approaches are based on variants of deontic logic.

In this paper we present a logic (which we call $\mathcal{CL}$) designed to represent and reason about contracts. The goal of $\mathcal{CL}$ is to preserve many of the natural properties and concepts relevant to legal contracts, while avoiding deontic paradoxes,

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

and at the same time to have a suitable language for the specification of software contracts. $\mathcal{CL}$ combines deontic logic [3] with propositional dynamic logic (PDL) [4, 5]. $\mathcal{CL}$ applies the (deontic and dynamic) modalities *exclusively* over actions instead of over formulas (or state of affairs) as is the case in standard deontic logic (SDL) [3]. Therefore, $\mathcal{CL}$ adopts what is known as the ought-to-do approach to deontic logic as opposed to the ought-to-be approach of SDL. The ought-to-do approach has been advocated by von Wright [6] which argued that deontic logic would benefit from a "foundation of actions", and many of the philosophical paradoxes of SDL would be eliminated. Important contributions to this approach were done by Segerberg to introduce actions inside the deontic modalities [7] and by the seminal work of Meyer on dynamic deontic logic (DDL) [8, 9].

$\mathcal{CL}$ extends the (regular) actions, which are usually considered in DDL and PDL, with a concurrency operator to model the notion of actions "*done at the same time*". The model of concurrency that we adopt is the synchrony model of Milner's SCCS [10]. Synchrony is easy to integrate with the other regular operations on actions (choice, sequence, and repetition). Moreover, synchrony is a natural choice for reasoning about the notion "at the same time" for human-like actions that we have in legal contracts.

The notion of synchrony has different meanings in different areas of computer science. Here we take the distinction between *synchrony* and *asynchrony* as presented in the SCCS calculus and later implemented in e.g. the Esterel synchronous programming language [11]. We understand *asynchrony* as when two concurrent systems proceed at indeterminate relative speeds (i.e. their actions may have different non-correlated durations); whereas in the *synchrony* model each of the two concurrent systems instantaneously perform a single action at each time instant. This is an abstract view of the actions found in contracts and is good for reasoning about properties of the contract.

The *synchrony model* of concurrency takes the assumption that time is discrete and that basic actions are instantaneous and represent the time step. Moreover, at each time step all possible actions are performed, i.e. the system is considered *eager* and *active*. 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. The mathematical framework of the synchrony model is much cleaner and more general than the asynchronous interleaving model (SCCS has the (asynchronous) CCS as a subcalculus [10]).

Because of the assumption of an eager behavior for the actions 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 by using temporal operators, like the *always* operator ($\mathcal{CL}$ can encode temporal operators with the dynamic modality [5]). The eagerness assumption facilitates reasoning both about the existence of the deontic modalities as well as about violations of the obligations or prohibitions.

Initial investigations into $\mathcal{CL}$ have been done in [12]. The $\mathcal{CL}$ language presented in this paper is more expressive. A variant of this syntax of the $\mathcal{CL}$ (without the propositional constants) was used in [13] for doing run-time monitoring of electronic contracts. There we used a restricted semantics based on traces of actions. This semantics was specially designed for monitoring the actions of the contracting parties at run-time with the purpose of detecting when a contract is violated. In [12] there was no direct semantics for $\mathcal{CL}$. In the present paper we give full semantics for $\mathcal{CL}$ based on *normative structures* and relate it with the trace based semantics of [13].

In this paper we present theoretical results and thus we use a simple and unrestricted syntax for $\mathcal{CL}$, in practice some syntactic restrictions may be imposed (see [12]). Since our main objective is to analyze contracts through formal tools like model-checking and run-time monitoring, we aim at a tractable language (i.e. decidable and with manageable complexities).

Related work: Compared to [8, 9, 14, 15], which also consider deontic modalities applied over actions, the investigation presented in this paper at the level of the deontic actions is different in several ways. First we add a synchrony operation, and second, we exclude the Kleene star *. None of the few papers that consider repetition as an action combinator under deontic modalities [14, 9] give a precise motivation for having such recurring actions inside obligations, permissions, or prohibitions. Even more, the use of repetition inside the deontic modalities is counter intuitive: take the expression $O(a^*)$ - "One is obliged to not pay, or pay once, or pay twice in a row, or..." - which puts no actual obligations; or take $P(a^*)$ - "One has the right to do any sequence of action a." - which is a very shallow permission and is captured by the widespread *Closure Principle* in jurisprudence where *what is not forbidden is permitted* [7]. Moreover, [9] argues that expressions like $F(a^*)$ and $P(a^*)$ should be simulated with the PDL modalities as $\langle a^*\rangle F(a)$ respectively $[a^*]P(a)$. In our opinion the * combinator under deontic modalities can be captured by using temporal or dynamic logic modalities along with deontic modalities over *-free actions.

Regarding the synchronous actions inside the dynamic modality, $\mathcal{CL}$ is included in the class of extensions of PDL which can reason about concurrent actions: PDL$^\cap$ with intersection of actions [16] which is undecidable for deterministic structures; or concurrent PDL [17]. In contrast, $\mathcal{CL}$ with synchronous composition over deterministic actions (see Definition 3) inside the dynamic modality is decidable. This makes $\mathcal{CL}$ more attractive for automation of reasoning about contracts.

Due to lack of space we do not present full proofs here; please refer to the technical reports [19, 22] for proofs and more technical details.

2 The Contract Language $\mathcal{CL}$

The syntax of $\mathcal{CL}$ is defined by the grammar in Table 1. In what follows we provide intuitions of the $\mathcal{CL}$ syntax and define our notation and terminology.

We call a formula $\mathcal{C}$ a (general) *contract clause* (or plainly *contract*). We consider a finite number of propositional constants ϕ drawn from a set Φ_B. We call $O_{\mathcal{C}}(\alpha)$, $P(\alpha)$, and $F_{\mathcal{C}}(\alpha)$ the *deontic modalities*, representing the obligation,

$$\begin{array}{rcl}
\mathcal{C} & := & \phi \mid O_{\mathcal{C}}(\alpha) \mid P(\alpha) \mid F_{\mathcal{C}}(\alpha) \mid \mathcal{C} \rightarrow \mathcal{C} \mid [\beta]\mathcal{C} \mid \ \bot \\
\alpha & := & a \mid \mathbf{0} \mid \mathbf{1} \mid \alpha \times \alpha \mid \alpha \cdot \alpha \mid \alpha + \alpha \\
\beta & := & a \mid \mathbf{0} \mid \mathbf{1} \mid \beta \times \beta \mid \beta \cdot \beta \mid \beta + \beta \mid \beta^* \mid \varphi? \\
\varphi? & := & \phi \mid \mathbf{0} \mid \mathbf{1} \mid \varphi? \vee \varphi? \mid \varphi? \wedge \varphi? \mid \neg \varphi?
\end{array}$$

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

permission, or prohibition of performing a given *action* α. $\mathcal{CL}$ includes directly in the definition of the obligation and prohibition the *reparations* in case of violations. Intuitively $O_{\mathcal{C}}(\alpha)$ states the obligation to perform α, and the reparation $\mathcal{C}$ in case the obligation is *violated*, i.e. whenever α is not performed. The reparation may be any contract clause. The modality $O_{\mathcal{C}}(\alpha)$ (resp. $F_{\mathcal{C}}(\alpha)$) represents what is called contrary-to-duty obligations, CTDs, (resp. contrary-to-prohibitions, CTPs) in dynamic deontic logic.[1] Obligations without reparations are written as $O_{\bot}(\alpha)$ where $\bot$ (and conversely $\top$) is the Boolean `false` (respectively `true`). We usually write $O(\alpha)$ instead of $O_{\bot}(\alpha)$. The prohibition modality $F_{\mathcal{C}}(\alpha)$ states the actual forbidding of the action α together with the reparation $\mathcal{C}$ in case the prohibition is violated. Note that it is possible to express nested CTDs and CTPs. Permissions have no reparations associated because they cannot be violated; permissions can only be exercised.

Throughout the paper we denote by $a, b, c \in \mathcal{A}_B$ the *basic actions* (e.g. "pay", "deliver", or "redraw"), by indexed $\alpha \in \mathcal{A}$ *deontic actions*, and by indexed β the *dynamic actions*. Actions α are used inside the deontic modalities, whereas the (more general) actions β are used inside the dynamic modality. An *action* term α is constructed from the basic actions $a \in \mathcal{A}_B$ and the special actions $\mathbf{0}$ and $\mathbf{1}$ (called the *violating* action and respectively the *skip* action) using the binary constructors: *choice* "+", *sequence* "·", and *synchrony* (or concurrency) "×". Actions β have the extra operators Kleene star * (for *repetition*) and Boolean *test* ?. Tests $\varphi?$ are constructed with the Boolean operators from *basic tests* which in our case are the propositional constants $\phi \in \Phi_B$ (also denoted $\mathcal{A}_B^?$)

We define a symmetric and irreflexive relation over the basic actions $\mathcal{A}_B$, which we call *conflict relation* and denote by $\#_{\mathcal{C}} \subseteq \mathcal{A}_B \times \mathcal{A}_B$. The conflict relation is a notion often found in legal contracts and is given a priori. The intuition of the conflict relation is that if two actions are in conflict then the actions cannot be done at the same time. This intuition explains the need for the following equational implication at the level of the deontic actions: $a \,\#_{\mathcal{C}}\, b \rightarrow a \times b = \mathbf{0}$, $\forall a, b \in \mathcal{A}_B$. 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" should result in a conflict (see Proposition 2-(16)). The second kind of conflicts

[1] The notions of CTD and CTP from $\mathcal{CL}$ are in contrast with the classical notion of CTD as found in the SDL literature [18]. 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 context (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 changed context (next world).

that $\mathcal{CL}$ rules out are: "Obligatory to go west and forbidden to go west" which is a standard requirement on a deontic logic.

The dynamic logic modality $[\cdot]\mathcal{C}$ is parameterized by *actions* β. The expression $[\beta]\mathcal{C}$ is read as: "after the action β is performed $\mathcal{C}$ must hold". Therefore, $\mathcal{CL}$ can reason about synchronous actions inside the dynamic modality. We use the classical Boolean implication operator $\rightarrow$; the other operators $\wedge, \vee, \neg, \leftrightarrow, \top, \oplus$ (exclusive or) are expressed in terms of $\rightarrow$ and $\bot$ as in propositional logic.

In $\mathcal{CL}$ we can write *conditional* obligations, permissions and prohibitions of two different kinds. As an example consider conditional obligations. The first kind is given with the propositional implication: $\mathcal{C} \rightarrow O_{\mathcal{C}}(\alpha)$ which is read as "if $\mathcal{C}$ is the case then it is obligatory that action α" (e.g. "If Internet traffic is high then the Client is obliged to pay"). The second kind is given with the dynamic box modality: $[\beta]O_{\mathcal{C}}(\alpha)$ which is read as "if action β was performed then it is obligatory that action α" (e.g. "after receiving necessary data the Provider is obliged to offer password").

The formalization of the actions has been thoroughly investigated in [19] where interpretations for the actions have been defined, and completeness and decidability results have been established. The semantics of the $\mathcal{CL}$ language is based on the interpretation of the deontic actions as rooted trees with edges labeled by elements of $2^{\mathcal{A}_B}$. Denote by $I(\alpha)$ the tree interpreting the deontic action α. Intuitively, $+$ provides the branching in the tree and $\cdot$ provides the parent-child relation on each branch (see also Theorem 3 in the appendix).

Each dynamic action β denotes a set of *guarded concurrent strings.*

Definition 1 (guarded concurrent strings). *Over the set of basic tests $\mathcal{A}_B^?$ we define* atoms *as functions $\nu : \mathcal{A}_B^? \rightarrow \{0,1\}$ which assign a Boolean value to each basic test. Denote by $Atoms = \{0,1\}^{\mathcal{A}_B^?}$ the set of all such functions. A* guarded concurrent string *(denoted by u, v, w) is a sequence*

$$w = \nu_0 x_1 \nu_1 \ldots x_n \nu_n, \qquad n \geq 0,$$

where $\nu_i \in Atoms$ and $x_i \in 2^{\mathcal{A}_B}$ are sets of basic actions.

For each dynamic action β we can construct a special two level finite automaton (denoted *GNFA*(β)) which accepts all and only the guarded concurrent strings denoting β. The important detail is that with each state of the automaton of the upper level it is associated a special finite state automaton (denoted $\lceil s \rceil$) which accepts a set of atoms. (An atom ν can be seen as a valuation of a test φ? iff the truth assignments of ν to the basic tests make φ? true; thus, for each test φ? there is a set of all atoms which make it true.) The results of [19] ensure that working with the dynamic actions or with the automata on guarded concurrent strings is the same (they are different notations for the same set).

Proposition 1 (automata for tests). *There exists a class of finite state automata, denoted $\mathcal{M}$, which accept all and only the subsets of Atoms; in notation $\mathcal{L}(M) \in 2^{Atoms}$, $M \in \mathcal{M}$ and $\forall A \in 2^{Atoms}, \exists M \in \mathcal{M}$ s.t. $\mathcal{L}(M) = A$.*

Definition 2 (automata on guarded concurrent strings). *Consider a two level finite automaton GNFA* $= (S, \mathcal{P}(\mathcal{A}_B), S_0, \rho, F, \lceil\cdot\rceil)$*. It consists at the first level of a finite automaton on concurrent strings* $(S, \mathcal{P}(\mathcal{A}_B), S_0, \rho, F)$*, together with a map* $\lceil\cdot\rceil : S \rightarrow \mathcal{M}$*. An automaton on concurrent strings (i.e. the first level automation) consists of a finite set of* states S*, the finite* alphabet $2^{\mathcal{A}_B}$ *(i.e. the powerset of the set of basic actions* $\mathcal{A}_B$*), a set of* initial *states* $S_0 \subseteq S$*, a* transition relation $\rho : 2^{\mathcal{A}_B} \rightarrow S \times S$*, and a set of final states* F*. At the lower level the mapping* $\lceil\cdot\rceil$ *associates with each state of the first level an automaton* $M \in \mathcal{M}$ *as defined in Proposition 1 which accepts a set of atoms denoted* $\mathcal{L}(\lceil s \rceil)$*.*

Intuitively, a *GNFA*(β) accepts a guarded concurrent string $\nu_0 x_1 \nu_1 \dots x_n \nu_n$ if there is a sequence of nodes $s_0 \dots s_n \in S$ with $s_n \in F$ s.t. the x_i label the transitions s_{i-1}, s_i and the atoms ν_i are accepted by the corresponding automata (on the second level) $\lceil s_i \rceil$. The definition with two levels of *GNFA* is needed when defining the special operations for fusion product and synchronous composition corresponding to respectively $\cdot$ and $\times$ (see [19]).

3 Semantics

The formulas $\mathcal{C}$ of the logic are given a model theoretic semantics in terms of (what we define as) *normative structures.*

Definition 3 (normative structure). *A* normative structure *is a tuple denoted* $K^{\mathcal{N}} = (\mathcal{W}, \rho, \mathcal{V}, \varrho)$ *where* $\mathcal{W}$ *is a set of* worlds*,* $\mathcal{V} : \mathcal{W} \rightarrow 2^{\Phi_B}$ *is a* valuation function *returning for each world the set of propositional constants which hold in that world.* $\mathcal{A}_B$ *is a finite set of* basic labels *and* $2^{\mathcal{A}_B}$ *represents the* labels *of the structure as sets of basic labels;* $\rho : 2^{\mathcal{A}_B} \rightarrow 2^{\mathcal{W} \times \mathcal{W}}$ *is a function returning for each label a partial function (therefore for each label from one world there is at most one reachable world), and* $\varrho : \mathcal{W} \rightarrow 2^{\Psi}$ *is a marking function which marks each state with one or several markers from* $\Psi = \{\circ_a, \bullet_a \mid a \in \mathcal{A}_B\}$*. The marking function respects the restriction that no state can be marked by both* $\circ_a$ *and* $\bullet_a$ *(for any* $a \in \mathcal{A}_B$*). A* pointed normative structure *is a normative structure with a designated state* i *(denoted by* $K^{\mathcal{N}}, i$*).*

Notation: We denote by t a node of a tree (or by r the root) and by s (or i for initial) a world of a normative structure. Henceforth we use the more graphical notation $t \xrightarrow{\alpha_\times} t'$ ($s \xrightarrow{\beta_\times} s'$) for an edge (transition) in a tree (normative structure), where $\alpha_\times, \beta_\times \in 2^{\mathcal{A}_B}$ denote labels. Note that we consider both the trees and the normative structures to have the same set of basic labels $\mathcal{A}_B$. For the sake of notation we can view the valuation of one particular world $\mathcal{V}(s)$ as a function from Φ_B to $\{0, 1\}$ where $\forall \varphi \in \Phi_B$, $\mathcal{V}(s)(\varphi) = 1$ iff $\varphi \in \mathcal{V}(s)$ and 0 otherwise. Therefore, we can say that $\mathcal{V}(s)$ is accepted by the automata of the second level of *GNFA* and write $\mathcal{V}(s) \in \mathcal{L}(\lceil s \rceil)$.

One difference from the standard PDL is that we consider *deterministic* structures. 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. The deterministic restriction of Kripke structures was investigated in [20]. Note that deterministic PDL is undecidable if action negation (or intersection of actions) is added [5].

The marking function and the markers are needed to identify obligatory (i.e. $\circ$) and prohibited (i.e. $\bullet$) actions. Markers with different purposes were used in [8] to identify violations of obligations, in [14] to mark permitted transitions, and in [15] to identify permitted events. The labels of the normative structure (and of the trees $I(\alpha)$ or automata $\mathit{GNFA}(\beta)$) are sets of basic labels $\mathcal{A}_B$. Therefore, we can compare them using set inclusion (and call one label *bigger* than another).

Definition 4 (simulation). *For a tree $T = (\mathcal{N}, \mathcal{E}, \mathcal{A}_B)$ and a normative structure $K^{\mathcal{N}} = (\mathcal{W}, \rho, \mathcal{V}, \varrho)$ we define a relation $\mathcal{S} \subseteq \mathcal{N} \times \mathcal{W}$ which we call the* simulation *of the tree node by the state of the structure.*

$$t\,\mathcal{S}\,s \text{ iff } \forall t \xrightarrow{\gamma} t' \in T,\ \exists s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}} \text{ s.t. } \gamma \subseteq \gamma' \wedge t'\mathcal{S}s' \text{ and}$$
$$\forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}} \text{ with } \gamma \subseteq \gamma' \text{ then } t'\,\mathcal{S}\,s'.$$

We say that a tree T, with root r is simulated by a normative structure $K^{\mathcal{N}}$ w.r.t. a state s, denoted $T\ \mathcal{S}_s\,K^{\mathcal{N}}$, iff $r\,\mathcal{S}\,s$.

Note two differences with the classical definition of simulation: first, the labels of the normative structure may be bigger than the labels in the tree because respecting an obligatory action means executing an action which includes it (is bigger). We can drop this condition and consider only $\gamma = \gamma'$, in which case we call the relation *strong simulation* and denote by $\mathcal{S}^s$. Second, any transition in the normative structure that can simulate an edge in the tree must enter under the simulation relation. This is because from the state s' onwards we need to be able to continue to look in the structure for the remaining tree (to see that it is simulated). We can weaken the definition by combining this condition with the one before into: $\forall t \xrightarrow{\gamma} t' \in T$, $\forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}}$ with $\gamma \subseteq \gamma'$ then $t'\,\tilde{\mathcal{S}}\,s'$. We call the resulting relation *partial simulation* and denote it by $\tilde{\mathcal{S}}$.

Definition 5 (semantics). *We give in Table 2 a recursive definition of the* satisfaction relation $\models$ *of a formula $\mathcal{C}$ w.r.t. a pointed normative structure $K^{\mathcal{N}}, i$; it is written $K^{\mathcal{N}}, i \models \mathcal{C}$ and is read as "$\mathcal{C}$ is* satisfied *in the normative structure $K^{\mathcal{N}}$ at state i". The notions of* satisfiability *and* validity *are defined as usual.*

$\mathcal{CL}$ has particular properties found in legal contracts (which we list in Propositions 2 and 3). Some intuitive motivation for these properties from the perspective of legal contracts has been given in [12]. Here we explain how these properties influenced our decisions in the design of the semantics of $\mathcal{CL}$.

For the $O_{\mathcal{C}}$ the semantics has basically two parts. The first part (lines one to four) gives the interpretation of the obligation. Line one says how to walk on the structure depending on the tree of the action α. The normative structure must simulate the tree of the action, completely. This is because all the actions (i.e. on all the choices) that are obligatory must appear as transitions in the structure in order to guarantee properties like (21) and (8). Moreover, the simulation relation

$$
\begin{array}{ll}
K^{\mathcal{N}}, i \models \varphi & \text{iff } \varphi \in \mathcal{V}(i). \\
K^{\mathcal{N}}, i \models \mathcal{C}_1 \to \mathcal{C}_2 & \text{iff whenever } K^{\mathcal{N}}, i \models \mathcal{C}_1 \text{ then } K^{\mathcal{N}}, i \models \mathcal{C}_2. \\
K^{\mathcal{N}}, i \models O_{\mathcal{C}}(\alpha) & \text{iff } I(\alpha)\ \mathcal{S}_i\, K^{\mathcal{N}}, \quad \text{and} \\
& \quad \forall t \xrightarrow{\gamma} t' \in I(\alpha),\ \forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}} \text{ s.t. } t\, \mathcal{S}\, s \wedge \gamma \subseteq \gamma' \text{ then} \\
& \qquad \forall a \in \gamma,\ \circ_a \in \varrho(s'), \quad \text{and} \\
& \quad \forall s \xrightarrow{\gamma'} s' \in K_{rem}^{I(\alpha),i},\ \forall a \in \gamma' \text{ then } \circ_a \notin \varrho(s'), \quad \text{and} \\
& \quad K^{\mathcal{N}}, s \models \mathcal{C} \quad \forall s \in N \text{ with } t\, \mathcal{S}^s\, s \wedge t \in \mathit{leaves}(I(\overline{\alpha})). \\
K^{\mathcal{N}}, i \models F_{\mathcal{C}}(\alpha) & \text{iff } I(\alpha)\ \tilde{\mathcal{S}}_i\, K^{\mathcal{N}} \text{ then} \\
& \quad \forall \sigma \in I(\alpha) \text{ a full path s.t. } \sigma\ \mathcal{S}_i\, K^{\mathcal{N}}, \\
& \quad \exists t \xrightarrow{\gamma} t' \in \sigma \text{ s.t. } \forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}} \text{ with } t\, \mathcal{S}\, s \wedge \gamma \subseteq \gamma' \text{ then} \\
& \qquad \forall a \in \gamma',\ \bullet_a \in \varrho(s'), \quad \text{and} \\
& \quad K^{\mathcal{N}}, s \models \mathcal{C} \quad \forall s \in N \text{ with } t\, \mathcal{S}\, s \wedge t \in \mathit{leaves}(\sigma). \\
K^{\mathcal{N}}, i \models P(\alpha) & \text{iff } I(\alpha)\ \mathcal{S}_i\, K^{\mathcal{N}}, \text{ and} \\
& \quad \forall t \xrightarrow{\gamma} t' \in I(\alpha),\ \forall s \xrightarrow{\gamma} s' \in K^{\mathcal{N}} \text{ s.t. } t\, \mathcal{S}^s\, s \wedge t'\, \mathcal{S}^s\, s' \text{ then} \\
& \qquad \forall a \in \gamma,\ \bullet_a \notin \varrho(s'). \\
K^{\mathcal{N}}, i \models [\beta]\mathcal{C} & \text{iff } \forall t \in \mathcal{W} \text{ with } (i,t) \in \rho(\beta) \text{ then } K^{\mathcal{N}}, t \models \mathcal{C}.
\end{array}
$$

$$
\begin{array}{rl}
\rho(\beta) = \{(s,t) \mid & \exists k, \exists \sigma = x_0 \ldots x_k \text{ a final path in } \mathit{GNFA}(\beta), \\
& \exists s_0 \ldots s_k \in \mathcal{W} \text{ s.t. } s_0 = s, s_k = t, \text{ and } \forall i \leq k,\ \mathcal{V}(s_i) \in \mathcal{L}(\lceil x_i \rceil), \text{ and} \\
& \forall 0 \leq i < k \text{ with } x_i \xrightarrow{\beta_\times} x_{i+1} \in \sigma \text{ then } (s_i, s_{i+1}) \in \rho(\beta_\times)\}
\end{array}
$$

Table 2. Semantics for $\mathcal{CL}$.

allowes for the labels of the structure to be bigger than (not necessary the same as) the labels in the tree. This is to guarantee property (7). Intuitively it means that if we do these (bigger) actions the obligation of α is still respected. The second condition of the simulation relation is needed also for properties like (8) because regardless of the way to respect a first obligation we must be able to enforce the subsequent obligations; which is related to property (8).

Lines two and three in the semantics of $O_{\mathcal{C}}$ say how the states must be marked with $\circ$. Note that the markers correspond exactly to the basic actions in the tree and not to the basic actions in the structure (which may be more). This is needed in the proof of the property (7) and property (17) and, in conjunction with the restriction on the normative structures, that no state is marked with both markers $\circ_a$ and $\bullet_a$ for any $a \in \mathcal{A}_B$, also for properties like (6) and (14). Moreover, note that all the relevant transitions in the normative structure must enter under the marking function in order to have properties like (18).

Line four ensures that no other reachable relevant transitions of the structure (i.e. from the non-simulating remainder structure $K_{rem}^{I(\alpha),i}$) are marked with obligation markers $\circ$. (See Definition 8 of $K_{rem}^{I(\alpha),i}$ in Appendix.) This is needed for property (22). It says that all the transitions which are outside the action (say outside α) should not be labeled with $\circ$ markers. Thus transitions from any other actions of a choice (e.g. of α' from $\alpha + \alpha'$) cannot be marked correctly.

The second part of the semantics of $O_{\mathcal{C}}$ is just the last line and handles our notion of reparation in case of violations of obligations. The negation of a compound action $\overline{\alpha}$ encodes all possible ways of violating the obligation $O(\alpha)$.

Therefore, at the end of each of these possible ways of violation the reparation must be enforced. (See the Definition 9 of $\overline{\alpha}$ in Appendix.)

The conflict relation $\#_{\mathcal{C}}$ with the equational implication $a \#_{\mathcal{C}} b \rightarrow a \times b = \mathbf{0}$ and properties (1) and (7) is needed to prove how $\mathcal{CL}$ avoids conflicts like (16).

In the semantics of $F_{\mathcal{C}}$ the first condition uses partial simulation. This is because we want to capture the assumption that if a transition is not present in the normative structure then this is forbidden by default. In the second condition we consider all the paths in the tree which are simulated by the structure (i.e. come from the partial simulation) to have the property (10); prohibition of a choice must prohibit all. Note that we are interested only in *full* paths simulated by the structure because for the other paths some of the transitions are missing in the structure and thus there is some action on the sequence which is forbidden. On the other hand, in the third condition we consider at least one of the transitions on this full path in order to have the property (11); forbidding a sequence means forbidding some action on that sequence. Moreover, note that the $\bullet$ markers are associated with the labels in the normative structure and not with the labels in the tree (as for $O_{\mathcal{C}}$) in order to capture the property (9); i.e., forbidding an action implies forbidding any action that is bigger. Also note that all the transitions in the normative structure that simulate the chosen transition in the path are considered so that to capture the property (23). The last condition takes care to put the reparations where a violation of a prohibition is observed; i.e. after executing one full path in the tree of the prohibited action.

The semantics of P is similar to that of $O_{\mathcal{C}}$. It considers the simulation relation in order to have properties like (12). The difference is that the states *must not* be marked with $\bullet$ markers in order to have properties (15), (3), and (4); and to capture the principle that *what is not forbidden is permitted*.

The semantics of the dynamic modality $[\beta]\mathcal{C}$ is classical; it checks that the clause $\mathcal{C}$ holds at all β-reachable states. The reachability function ρ is extended to all compound actions β. Special for $\mathcal{CL}$ is that we extend the regular actions with synchrony and thus we define the reachability function for compound actions using the associated automata $GNFA(\beta)$, in the style of APDL [21].

4 Properties

Proposition 2 (validities). *The following statements hold:*

$$\models \neg O_{\mathcal{C}}(\mathbf{0}) \tag{1}$$

$$\models O_{\mathcal{C}}(\mathbf{1}) \tag{2}$$

$$\models P(\alpha) \rightarrow \neg F_{\mathcal{C}}(\alpha) \tag{3}$$

$$\models O_{\mathcal{C}}(\alpha) \rightarrow P(\alpha) \tag{4}$$

$$\textit{if } \alpha = \alpha' \textit{ then } \models O_{\mathcal{C}}(\alpha) \leftrightarrow O_{\mathcal{C}}(\alpha') \tag{5}$$

$$\models O_{\mathcal{C}}(\alpha) \rightarrow \neg F_{\mathcal{C}}(\alpha) \tag{6}$$

$$\models O_{\mathcal{C}}(\alpha) \wedge O_{\mathcal{C}}(\alpha') \rightarrow O_{\mathcal{C}}(\alpha \times \alpha') \tag{7}$$

$$\models O_{\mathcal{C}}(\alpha \cdot \alpha') \leftrightarrow O_{\mathcal{C}}(\alpha) \wedge [[\alpha]]O_{\mathcal{C}}(\alpha') \tag{8}$$

$$\models F_{\mathcal{C}}(\alpha) \rightarrow F_{\mathcal{C}}(\alpha \times \alpha') \tag{9}$$

$$\models F_{\mathcal{C}}(\alpha + \alpha') \leftrightarrow F_{\mathcal{C}}(\alpha) \wedge F_{\mathcal{C}}(\alpha') \tag{10}$$

$$\models F_{\mathcal{C}}(\alpha \cdot \alpha') \leftrightarrow F_{\mathcal{C}}(\alpha) \vee \langle\langle\alpha\rangle\rangle F_{\mathcal{C}}(\alpha') \tag{11}$$

$$\models P(\alpha + \alpha') \leftrightarrow P(\alpha) \wedge P(\alpha') \tag{12}$$

$$\models P(\alpha \cdot \alpha') \leftrightarrow P(\alpha) \wedge [\alpha]P(\alpha') \tag{13}$$

The following point out conflicts that are avoided in $\mathcal{CL}$:

$$\models \neg(O_{\mathcal{C}}(\alpha) \wedge F_{\mathcal{C}}(\alpha)) \tag{14}$$

$$\models \neg(P(\alpha) \wedge F_{\mathcal{C}}(\alpha)) \tag{15}$$

$$\textit{if } \alpha \,\#_{\mathcal{C}}\, \alpha' \textit{ then } \models \neg(O_{\mathcal{C}}(\alpha) \wedge O_{\mathcal{C}}(\alpha')) \tag{16}$$

The symbols $[[\cdot]]$ and $\langle\langle\cdot\rangle\rangle$ are the corresponding of the dynamic modalities $[\cdot]$ and $\langle\cdot\rangle$ only that they consider any action bigger than the action inside the modality. With the semantics of Table 2 we cannot prove validity of expressions like $F(a) \wedge (\phi \rightarrow P(a))$ which may be intuitive for the reader; e.g. some action a is forbidden and only in exceptional cases (when φ holds) it is permitted. The formula is not satisfied in a model which has a state s s.t. $\bullet_a \in \varrho(s)$ (from $F(a)$) and where ϕ holds and thus from the semantics of $P(a)$ is required that $\bullet_a \notin \varrho(s)$ which is impossible. Nevertheless, the example can be modelled in $\mathcal{CL}$ as $(\neg\varphi \rightarrow F(a)) \wedge (\varphi \rightarrow P(a))$ which is also more natural.

The following result shows that the semantics avoids *unwanted implications*.

Proposition 3 (unwanted implications). *The following statements hold:*

$$\not\models O_{\mathcal{C}}(\alpha) \rightarrow O_{\mathcal{C}}(\alpha \times \alpha') \tag{17}$$

$$\not\models O_{\mathcal{C}}(\alpha \times \alpha') \rightarrow O_{\mathcal{C}}(\alpha) \tag{18}$$

$$\not\models O_{\mathcal{C}}(\alpha + \alpha') \rightarrow O_{\mathcal{C}}(\alpha \times \alpha') \tag{19}$$

$$\not\models O_{\mathcal{C}}(\alpha \times \alpha') \rightarrow O_{\mathcal{C}}(\alpha + \alpha') \tag{20}$$

$$\not\models O_{\mathcal{C}}(\alpha) \rightarrow O_{\mathcal{C}}(\alpha + \alpha') \tag{21}$$

$$\not\models O_{\mathcal{C}}(\alpha + \alpha') \rightarrow O_{\mathcal{C}}(\alpha) \tag{22}$$

$$\not\models F_{\mathcal{C}}(\alpha \times \alpha') \rightarrow F_{\mathcal{C}}(\alpha) \tag{23}$$

$$\not\models P(\alpha \times \alpha') \rightarrow P(\alpha) \tag{24}$$

$$\not\models O_{\mathcal{C}}(\alpha) \oplus O_{\mathcal{C}}(\alpha') \rightarrow O_{\mathcal{C}}(\alpha + \alpha') \tag{25}$$

$$\not\models O_{\mathcal{C}}(\alpha + \alpha') \rightarrow O_{\mathcal{C}}(\alpha) \oplus O_{\mathcal{C}}(\alpha') \tag{26}$$

We show that $\mathcal{CL}$ is decidable by showing that it has the *finite model property*. We do this by first showing that $\mathcal{CL}$ has the *tree model property*. For proving the finite model property we use the *selection* technique [23, sec.2.3] because it is hard to use the filtration technique in our case as we do not know what are subformulas of an obligation of a complex action like $O_{\mathcal{C}}(a \cdot (b + c))$. (For proofs see technical report [22].)

Theorem 1 (decidability). *$\mathcal{CL}$ with the semantics of Table 2 is decidable.*

In Appendix A.1 we give the relation between the full semantics presented so far and a trace semantics for $\mathcal{CL}$ from [13]. The results hold in a slightly restricted setting which is due to the restrictions on $\mathcal{CL}$ coming from the trace semantics and the run-time monitoring setting where it is used.

5 Conclusion

Some technical details of the interpretation of the actions and of the $\mathcal{CL}$ formulas have been omitted due to space restrictions in favor of intuitive explanations; the interested reader can find these details in [22]. The $\mathcal{CL}$ logic presented here has a decidable satisfiability problem. The technical difficulties in the underlying semantics come from the many properties that the logic needs to capture. Some of the novelties of $\mathcal{CL}$ are the use of synchronous actions and the definitions of the conflict relation and the normative structures.

References

1. Owe, O., Schneider, G., Steffen, M.: Components, objects, and contracts. In: SAVCBS'07. ACM Digital Library, Dubrovnik, Croatia (September 2007) 91–94
2. van der Torre, L.: Contextual deontic logic: Normative agents, violations and independence. Ann. Math. Artif. Intell. **37**(1-2) (2003) 33–63
3. von Wright, G.H.: Deontic logic. Mind **60** (1951) 1–15
4. Fischer, M.J., Ladner, R.E.: Propositional modal logic of programs. In: STOC'77, ACM (1977) 286–294
5. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
6. von Wright, G.H.: An Essay in Deontic Logic and the General Theory of Action. North Holland Publishing Co., Amsterdam (1968)
7. Segerberg, K.: A deontic logic of action. Studia Logica **41**(2) (1982) 269–282
8. Meyer, J.J.C.: A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. Notre Dame Journal of Formal Logic **29**(1) (1988) 109–136
9. Broersen, J., Wieringa, R., Meyer, J.J.C.: A fixed-point characterization of a deontic logic of regular action. Fundam. Inf. **48**(2-3) (2001) 107–128
10. Milner, R.: Calculi for synchrony and asynchrony. TCS **25** (1983) 267–310
11. Berry, G.: The foundations of Esterel. In: Proof, language, and interaction: essays in honour of Robin Milner. MIT Press (2000) 425–454
12. Prisacariu, C., Schneider, G.: A formal language for electronic contracts. In: FMOODS'07. Volume 4468 of LNCS., Springer (2007) 174–189
13. Kyas, M., Prisacariu, C., Schneider, G.: Run-time monitoring of electronic contracts. In: ATVA'08. Volume 5311 of LNCS., Springer-Verlag (2008) 397–407
14. Van der Meyden, R.: Dynamic logic of permission, the. In: LICS'90, IEEE Computer Society Press (1990) 72–78
15. Castro, P.F., Maibaum, T.: A complete and compact propositional deontic logic. In: ICTAC'07. Volume 4711 of LNCS., Springer-Verlag 109–123
16. Harel, D.: Recurring dominoes: Making the highly undecidable highly understandable. In: FCT'83. Volume 158 of LNCS., Springer (1983) 177–194
17. Peleg, D.: Concurrent dynamic logic. In: STOC'85, ACM (1985) 232–239
18. Prakken, H., Sergot, M.: Dyadic deontic logic and contrary-to-duty obligation. In: Defeasible Deontic Logic. Kluwer Academic Publishers (1997) 223–262
19. Prisacariu, C.: Extending Kleene Algebra with Synchrony – technicalities. Technical Report 376, Univ. Oslo (2008)
20. Ben-Ari, M., Halpern, J.Y., Pnueli, A.: Finite models for deterministic propositional dynamic logic. In: ICALP'81. Volume 115 of LNCS., Springer (1981) 249–263
21. Harel, D., Sherman, R.: Propositional dynamic logic of flowcharts. In: FCT'83. Volume 158 of LNCS., Springer (1983) 195–206
22. Prisacariu, C., Schneider, G.: CL: A Logic for Reasoning about Legal Contracts –Semantics. Technical Report 371, Univ. Oslo (2008)
23. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Volume 53 of Cambridge Tracts in Theoretical Computer Science. Cambridge Univ. Press (2001)

A Additional proofs and definitions

This Appendix contains formal definitions for some of the concepts given in the paper, as well as more thorough presentation of the results. All these, and more detailed explanations, examples, and full proofs can be found in the technical report [22].

The Definition 9 of *action negation* and the tree interpretation of the deontic actions in Theorem 3 are based on a notion of *canonical form*.

Definition 6 (canonical form). *We say that an action α is in* canonical form *denoted by $\underline{\alpha}$ iff it has the following form:*

$$\underline{\alpha} = +_{i \in I}\ \alpha^i_\times \cdot \underline{\alpha^i}$$

where $\alpha^i_\times \in \mathcal{A}^\times_B$ and $\underline{\alpha^i} \in \mathcal{A}$ is an action in canonical form. The indexing set I is finite as $\alpha^i_\times \in \mathcal{A}^\times_B$ are finite; therefore there is a finite number of application of the $+$ combinator. Actions $\mathbf{0}$ *and* $\mathbf{1}$ *are in canonical form.*

Theorem 2 ([19]). *For every action α there exists a corresponding action $\underline{\alpha}$ in canonical form and equivalent to α.*

Definition 7 (rooted tree). *A* rooted tree *is an acyclic connected graph $(\mathcal{N}, \mathcal{E})$ with a designated node r called* root *node. $\mathcal{N}$ is the set of* nodes *and $\mathcal{E}$ is the set of* edges *(where an edge is an ordered pair of nodes (n, m)). We consider* rooted trees with labeled edges *and denote the labeled directed edges with (n, α, m) and the tree with $(\mathcal{N}, \mathcal{E}, \mathcal{A}_B)$. The labels $\alpha \in 2^{\mathcal{A}_B}$ are* sets of basic labels*; e.g. $\alpha_1 = \{a, b\}$ or $\alpha_2 = \{a\}$ with $a, b \in \mathcal{A}_B$. Labels are compared for set equality (or set inclusion). Note the special* empty set *label. We consider a special label Λ to stand for an impossible label. We restrict our presentation to* finite *rooted trees (i.e., there is no infinite path in the graph starting from the root node). The set of all such defined trees is denoted $\mathcal{T}$.*

Theorem 3 (interpretation of deontic actions). *For any action α there exists a tree representation corresponding to the canonical form $\underline{\alpha}$.*

Proof. The *representation* is an interpretation function $I : \mathcal{A} \to \mathcal{T}$ which interprets all action terms as trees. More precisely, given an arbitrary action of $\mathcal{A}$, the canonical form is computed first and then I generates the tree representation.

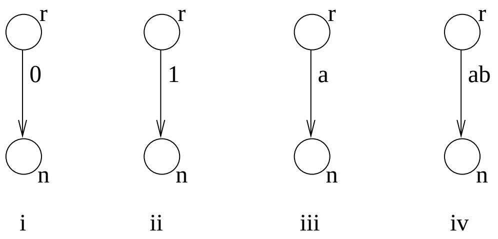

Fig. 1. Trees corresponding to $\mathbf{0}$, $\mathbf{1}$, $a \in \mathcal{A}_B$, and $a \times b \in \mathcal{A}^\times_B$.

The function I is defined inductively. The basis is to interpret each concurrent action of $\mathcal{A}_B^{\times}$ as a tree with labeled edges from $2^{\mathcal{A}_B}$ as pictured in Fig.1. Note that actions of $\mathcal{A}_B^{\times} \cup \{\mathbf{0}, \mathbf{1}\}$ are in canonical form. For a general action in canonical form $\underline{\alpha} = +_{i \in I} \ \alpha_{\times}^i \cdot \underline{\alpha^i}$ the tree is generated by adding one branch to the root node for each element $\alpha_{\times}^i$ of the top summation operation. The label of the branch is the set $\{\alpha_{\times}^i\}$ corresponding to the concurrent action. The construction continues inductively by attaching at the end of each newly added branch the tree interpretation of the smaller action $\underline{\alpha^i}$.

The semantics for $O_{\mathcal{C}}$ relies on a notion of non-simulating reminder structure and on the operation of action negation for deontic actions.

Definition 8 (non-simulating reminder). *Whenever $T \ \mathcal{S}_i \ K^{\mathcal{N}}$ then we call the* maximal simulating structure w.r.t. T and i, *and denote it by* $K_{max}^{T,i} = (\mathcal{W}', \rho', \mathcal{V}', \varrho')$ *the sub-structure of* $K^{\mathcal{N}} = (\mathcal{W}, \rho, \mathcal{V}, \varrho)$ *s.t.:*

1. $i \in \mathcal{W}'$
2. $\mathcal{V}' = \mathcal{V}|_{\mathcal{W}'}$ *and* $\varrho' = \varrho|_{\mathcal{W}'}$
3. $\forall t \xrightarrow{\gamma} t' \in T$ *then* $\forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}}$ *s.t.* $t \, \mathcal{S} \, s \wedge \gamma \subseteq \gamma' \wedge t' \, \mathcal{S} \, s'$ *do add* s' *to* $\mathcal{W}'$ *and add* $s \xrightarrow{\gamma'} s'$ *to* ρ'.

We call the non-simulating remainder of $K^{\mathcal{N}}$ w.r.t. T and i *the sub-structure* $K_{rem}^{T,i} = (\mathcal{W}'', \rho'', \mathcal{V}'', \varrho'')$ *of* $K^{\mathcal{N}}$ *s.t.:* $s \xrightarrow{\gamma} s' \in \rho''$ *iff* $s \xrightarrow{\gamma} s' \notin K_{max}^{T,i} \wedge s \in K_{max}^{T,i} \wedge \exists s \xrightarrow{\gamma} s'' \in K_{max}^{T,i}$*; and* $s \in \mathcal{W}''$ *iff* s *is part of a transition in* $K_{rem}^{T,i}$*; and* $\mathcal{V}'' = \mathcal{V}|_{\mathcal{W}''}$ *and* $\varrho'' = \varrho|_{\mathcal{W}''}$.

Definition 9 (action negation). *The* action negation *is denoted by* $\overline{\alpha}$ *and is defined as a function* $^{-} : \mathcal{A} \to \mathcal{A}$ *(i.e. action negation is not a principal combinator for the actions) and works on the equivalent canonical form* $\underline{\alpha}$ *as:*

$$\overline{\alpha} = \overline{\underset{i \in I}{+} \ \alpha_{\times}^i \cdot \underline{\alpha^i}} = \underset{\beta_{\times} \in \overline{R}}{+} \beta_{\times} \ + \ \underset{j \in J}{+} \ \gamma_{\times}^j \cdot \overline{\underset{i \in I'}{+} \underline{\alpha^i}}$$

Consider $R = \{\alpha_{\times}^i \mid i \in I\}$*. The set* $\overline{R}$ *contains all the concurrent actions* $\beta_{\times}$ *with the property that* $\beta_{\times}$ *is not bigger than any of the actions* $\alpha_{\times}^i$*:*

$$\overline{R} = \{\beta_{\times} \mid \beta_{\times} \in \mathcal{A}_B^{\times} \text{ and } \forall i \in I, \alpha_{\times}^i \not\subseteq \beta_{\times}\};$$

and $\gamma_{\times}^j \in \mathcal{A}_B^{\times}$ *and* $\exists \alpha_{\times}^i \in R$ *s.t.* $\alpha_{\times}^i \subseteq \gamma_{\times}^j$*. The indexing set* $I' \subseteq I$ *is defined for each* $j \in J$ *as:*

$$I' = \{i \in I \mid \alpha_{\times}^i \subseteq \gamma_{\times}^j\}.$$

A.1 Relations with the trace semantics of $\mathcal{CL}$

In [13] we presented a trace semantics for $\mathcal{CL}$ with the goal of monitoring electronic contracts at run-time. This semantics is intended for identifying the *respecting* and *violating* traces of actions for a $\mathcal{CL}$ clause. Here we just present

For $\top, \bot, \rightarrow, \wedge, \vee, \oplus$	take a standard LTL-style semantics.
$\sigma \models [\alpha_\times]\mathcal{C}$	if $\alpha_\times = \sigma(0)$ and $\sigma(1..) \models \mathcal{C}$, or $\alpha_\times \neq \sigma(0)$.
$\sigma \models [\beta \cdot \beta']\mathcal{C}$	if $\sigma \models [\beta][\beta']\mathcal{C}$.
$\sigma \models [\beta + \beta']\mathcal{C}$	if $\sigma \models [\beta]\mathcal{C}$ and $\sigma \models [\beta']\mathcal{C}$.
$\sigma \models [\beta^*]\mathcal{C}$	if $\sigma \models \mathcal{C}$ and $\sigma \models [\beta][\beta^*]\mathcal{C}$.
$\sigma \models [\mathcal{C}_1?]\mathcal{C}_2$	if $\sigma \not\models \mathcal{C}_1$, or $if \sigma \models \mathcal{C}_1$ and $\sigma \models \mathcal{C}_2$.
$\sigma \models O_\mathcal{C}(\alpha_\times)$	if $\alpha_\times \subseteq \sigma(0)$, or if $\sigma(1..) \models \mathcal{C}$.
$\sigma \models O_\mathcal{C}(\alpha \cdot \alpha')$	if $\sigma \models O_\mathcal{C}(\alpha)$ and $\sigma \models [[\alpha]]O_\mathcal{C}(\alpha')$.
$\sigma \models O_\mathcal{C}(\alpha + \alpha')$	if $\sigma \models O_\bot(\alpha)$ or $\sigma \models O_\bot(\alpha')$ or $\sigma \models [\overline{\alpha + \alpha'}]\mathcal{C}$.
$\sigma \models F_\mathcal{C}(\alpha_\times)$	if $\alpha_\times \not\subseteq \sigma(0)$, or if $\alpha_\times \subseteq \sigma(0)$ and $\sigma(1..) \models \mathcal{C}$.
$\sigma \models F_\mathcal{C}(\alpha \cdot \alpha')$	if $\sigma \models F_\bot(\alpha)$ or $\sigma \models [[\alpha]]F_\mathcal{C}(\alpha')$.
$\sigma \models F_\mathcal{C}(\alpha + \alpha')$	if $\sigma \models F_\mathcal{C}(\alpha)$ and $\sigma \models F_\mathcal{C}(\alpha')$.

Table 3. Trace semantics of $\mathcal{CL}$.

briefly the trace semantics and then concentrate on relating it with the full semantics of Table 2.

Consider an infinite *trace* denoted $\sigma = a_0, a_1, \ldots$ as a map $\sigma : \mathbb{N} \rightarrow \mathcal{A}_B^\times \cup \{\mathbf{1}\}$ from natural numbers (denoting positions) to concurrent actions from $\mathcal{A}_B^\times$. We denote by $\sigma(i)$ the *element* of a trace at position i, by $\sigma(i..j)$ a finite *subtrace*, and by $\sigma(i..)$ the infinite subtrace starting at position i in σ.

Consider the *satisfaction relation* $\models_t$ defined over pairs $(\sigma, \mathcal{C})$ of a trace and a contract which we write $\sigma \models_t \mathcal{C}$ and read as "trace σ *respects* the contract (clause) $\mathcal{C}$". For a brief definition of $\models_t$ see Table 3 and for details see [13].

The standard interpretation of $[\alpha_\times]\mathcal{C}$ is over branching structures as we did in Table 2. Here we interpret the two dynamic modalities over linear structures, i.e. over traces. A trace σ respects the formula $[\alpha_\times]\mathcal{C}$ if either the first (set of actions) element of the trace $\sigma(0)$ is not equal with the (set of basic actions forming the) action $\alpha_\times$ or otherwise $\sigma(0)$ is the same as action $\alpha_\times$ and $\mathcal{C}$ is respected by the rest of the trace (i.e. $\sigma(1..) \models \mathcal{C}$).

A trace σ respects an obligation $O_\mathcal{C}(\alpha_\times)$ if any of the two complementary conditions is satisfied. The first condition deals with the obligation itself: $O(\alpha_\times)$ is respected if the first action of the trace includes $\alpha_\times$. Otherwise, in case the obligation is violated,[2] the only way to fulfill the contract is by respecting the reparation $\mathcal{C}$; i.e. $\sigma(1..) \models \mathcal{C}$.

A folk technique called *linearization* takes (in our case) a pointed normative structure and returns all the (in)finite traces that start in the designated state i of the pointed structure. Denote this set of traces by $\|K^\mathcal{N}, i\|$. We use the notation $\sigma \in \|K^\mathcal{N}, i\|$ to mean that the trace σ is part of the traces of $K^\mathcal{N}$ starting as state i. Therefore, the following statement is obvious: *For any trace σ we can find a normative structure $K^\mathcal{N}$ and a state i s.t. $\sigma \in \|K^\mathcal{N}, i\|$.*

[2] Violation of an obligatory action is encoded by the action negation.

When not mentioned otherwise, the following results hold for a restricted syntax of $\mathcal{CL}$ which does not consider negation of clauses, nor tests inside the dynamic modalities. These syntactic restrictions are enough for doing run-time monitoring. For proofs see technical report [22].

Lemma 1. *For any $K^{\mathcal{N}}$ and $i \in K^{\mathcal{N}}$, if $K^{\mathcal{N}}, i \models \mathcal{C}$ then $\forall \sigma \in \|K^{\mathcal{N}}, i\|\ \sigma \models_t \mathcal{C}$.*

Corollary 1.

$$\bigcup_{K^{\mathcal{N}}, i \models \mathcal{C}} \|K^{\mathcal{N}}, i\| \subseteq \{\sigma \mid \sigma \models_t \mathcal{C}\}$$

Proposition 4. *With the general syntax of $\mathcal{CL}$ from Table 1 we can find a contract clause $\mathcal{C}$ s.t. $\exists \sigma$ s.t. $\sigma \models_t \mathcal{C}$ and $\nexists K^{\mathcal{N}}, \nexists i \in K^{\mathcal{N}}$ s.t. $K^{\mathcal{N}}, i \models \mathcal{C}$.*

Lemma 2 states the relation between the satisfiability in the trace semantics and satisfiability in the branching semantics. It says that if a contract clause is respected by some trace of actions then the contract is satisfiable in the branching semantics.

Lemma 2. *If $\exists \sigma$ s.t. $\sigma \models_t \mathcal{C}$ then $\exists K^{\mathcal{N}}, \exists i \in K^{\mathcal{N}}$ s.t. $K^{\mathcal{N}}, i \models \mathcal{C}$.*

Lemma 3. *If $\sigma \models_t \mathcal{C}$ then $\exists K^{\mathcal{N}}, \exists i \in K^{\mathcal{N}}$ s.t. $K^{\mathcal{N}}, i \models \mathcal{C}$ and $\sigma \in \|K^{\mathcal{N}}, i\|$.*

Corollary 2.

$$\{\sigma \mid \sigma \models_t \mathcal{C}\} \subseteq \bigcup_{K^{\mathcal{N}}, i \models \mathcal{C}} \|K^{\mathcal{N}}, i\|$$

The two corollaries relate the validities in the two semantics (under the restricted syntax). If a clause $\mathcal{C}$ is valid w.r.t. the trace semantics (i.e. there is no way of doing a sequence of actions to violate the contract) then the clause is valid in the full semantics (i.e. any model of the contract also entails the existence of this contract clause).