A state/event temporal deontic logic

Julien Brunel, Jean-Paul Bodeveix, Mamoun Filali

Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier
118 route de Narbonne, 31062 Toulouse, France
{brunel,bodeveix,filali}@irit.fr

Abstract. This paper studies a logic that combines deontic and temporal aspects. We first present a state/event temporal formalism and define a deontic extension of it. Then, we study the interaction between the temporal dimension and the deontic dimension. We present some logical properties, concerning formulas where deontic and temporal operators are nested, and discuss their intuitive meaning. We focus more particularly on the properties of obligation with deadline and define a specific operator to express this notion.

1 Introduction

Deontic logic is useful for specifying normative systems, i.e., systems which involve obligations, prohibitions, and permissions. Applications can be found in computer security, electronic commerce, or legal expert systems [21].

In this paper, we are interested in applications where both temporal and deontic notions appear. For instance, it may be interesting to express an access control policy in which the permissions depend on time, or events. Such a policy can be called an availability policy [8].

Consider a simple resource monitoring problem. An availability policy consists in giving a set of obligations and permissions for users to use resources.

– *$user_i$ has the permission to use the resource r for 5 time units continuously, and he must be able to access it 15 time units after asking, at the latest*
– *$user_i$ has always the permission to use the resource r, and he has to release it after 5 time units of utilization*
– *If $user_i$ is asking for the resource and he has the permission to use it, then the system has the obligation to give it to him before 5 time units*
– *If $user_i$ uses the resource without the permission, he must not ask for it during 10 time units*

The cases where permissions (idem with obligations and prohibitions) are granted may depend on the temporal events of the system, as in the two first sentences. But in the two last sentences, we see that permissions and prohibitions are granted according to other deontic notions. The last sentence gives for instance a prohibition if an obligation is violated.

Cuppens and Saurel have used in [8] predicate logic. They have exhibited four dedicated predicates to express a policy - which gives a limited expressive power - and about ten formulas to check that the system does not violate the policy. Our goal is to define a language which allows to specify easily availability policies, or other systems in which time and norms play an important role.

We will first present a state/event extension of the temporal logic LTL[18]. Section 3 defines a deontic extension of this formalism. Then, we will discuss in section 4 about properties of formulas where temporal and deontic operators are nested. In section 4, we will particularly focus on obligation with deadline.

2 State/Event extensions of *LTL*

We present here an extension of the state based *Linear Temporal Logic* (LTL)[18]. We will first discuss about the notions of event and action. Then we will present a state/event extension of Kripke structures. We will also present the logic which is interpreted on such structures : *State/Event Linear Temporal Logic*[5] (*SE-LTL*), a state/event extension of LTL. Note that these extensions do not increase the expressiveness, but allow to express behaviours in a much more succinct way (see [5] for more details).

2.1 Events or actions?

The notion of event is close to an action in dynamic logic [12]. But an event is atomic, and has no duration. The actions can be composed with several combinators: sequence, choice, iteration, converse. Some propositions to combine dynamic and temporal logics [13] strengthen the temporal operator until U. $\varphi_1 U^\alpha \varphi_2$ means that $\varphi_1 U \varphi_2$ is satisfied along some path which corresponds to the execution of the action α. This gives a good expressive power, but the specification of properties can be much harder. For instance, consider the sentence "If $user_i$ uses the resource, he will release it". in_use_i is a proposition, and $release_i$ an event (or an atomic action in dynamic logic). In a state/event logic, we express it naturally: $in\_use_i \Rightarrow F\ release_i$
But in a dynamic temporal logic, we cannot put actions and propositions at the same level. We have to use specific operators to introduce actions: $in\_use_i \Rightarrow \top\ U^{\Sigma *; release_i}\ \top$
where Σ represents any atomic action.

Moreover, in many cases the composition of events, can be expressed without using the combinators of dynamic logic. For instance, the formula $e_1 \wedge X e_2$ expresses that the event e_1 happens, followed by e_2, which corresponds to the action $e_1; e_2$ in a dynamic logic. The formula $e_1 U e_2$ means that there is an arbitrary number of executions of e_1, followed by an execution of e_2, and corresponds to the execution of the composed action $e_1*\ ; e_2$.

Besides, the semantics of events is much simpler, and there exists efficient tools for state/event logics [5,4]. In the rest of this paper, we have preferred events to actions.

2.2 Labeled Kripke Structures

We define here a labeled Kripke structure. Although it has the same expressive power than event free Kripke structure, it is an interesting implementation formalism because it allows to express behaviours in a much more succinct way.

Definition 1 (LKS). *A labeled Kripke structure (LKS) over $(P, \mathcal{E})$ is a tuple $(S, I, T, \alpha, \beta, \lambda, \nu)$ where*

- *P is a countable set of atomic propositions that label the states.*
- *$\mathcal{E}$ is a countable set of events that label the transitions.*
- *S is a countable set of states.*
- *$I \subseteq S$ is a set of the initial states.*
- *T is a set of transition identifiers, hereafter simply called transitions.*
- *$\alpha : T \to S$ is a function which associates each transition with its source state.*
- *$\beta : T \to S$ is a function which associates each transition with its destination state.*
- *$\lambda : T \to \mathcal{E}$ is a function which associates each transition with the event performed during the transition.*
  We often write $t : s \xrightarrow{e} s'$, where $t \in T, s, s' \in S$, and $e \in \mathcal{E}$, to mean that $\alpha(t) = s$, $\beta(t) = s'$, and $\lambda(t) = e$
  We suppose that every state has an outgoing transition:
  $\forall s \in S\ \exists (t, e, s') \in T \times \mathcal{E} \times S \quad t : s \xrightarrow{e} s'$[1]
- *$\nu : S \to 2^P$ is a valuation function which associates each state with the set of the atomic propositions it satisfies.*

Definition 2 (Run and trace). *A run $\rho = (s_0, t_0, s_1, t_1 \ldots)$ of an LKS is an infinite alternating sequence of states and transitions such that $s_0 \in I$ and $\forall i \in \mathbb{N} \quad \exists e \in \mathcal{E} \quad t_i : s_i \xrightarrow{e} s_{i+1}$. We can talk about infinite runs because we have supposed that there is a starting transition from every state.*

A trace $\tau = (\tau_{pr}, \tau_{ev})$ over a run $\rho = (s_0, t_0, s_1, t_1 \ldots)$ is defined as follows

- *$\tau_{pr} : seq(2^P)$ is the sequence[2] of the atomic proposition sets associated with the states of the run*
  $\forall i \ \tau_{pr}(i) = \nu(s_i)$
- *$\tau_{ev} : seq(\mathcal{E})$ is the sequence of the events associated with the transitions of the run*
  $\forall i \ \tau_{ev}(i) = e_i$ *where* $t_i : s_i \xrightarrow{e_i} s_{i+1}$

2.3 *SE-LTL*

We present now the syntax and the semantics of the specification formalism *SE-LTL* [5]. It is an extension of the state based logic *LTL* which takes into account both events and propositions.

[1] this is equivalent to $range(\alpha) = S$

[2] $seq(E) \overset{def}{=} \mathbb{N} \to E$ is the set of sequences of elements of E

Definition 3 (Syntax of *SE-LTL*). *Given a countable set P of atomic propositions, and a countable set $\mathcal{E}$ of events, a well-formed formula of SE-LTL is defined by:*

$$\varphi ::= p \in P \mid e \in \mathcal{E} \mid \bot \mid \varphi \Rightarrow \varphi \mid \varphi U^{+} \varphi$$

$\varphi_1 U^{+} \varphi_2$ means that φ_2 will hold at some point m in the strict future, and φ_1 will hold from the next moment until the moment before m.

We can define some usual abbreviations:

$\neg\varphi \overset{def}{=} \varphi \Rightarrow \bot$	$\top \overset{def}{=} \neg\bot$
$\varphi_1 \vee \varphi_2 \overset{def}{=} \neg\varphi_1 \Rightarrow \varphi_2$	$\varphi_1 \wedge \varphi_2 \overset{def}{=} \neg(\varphi_1 \Rightarrow \neg\varphi_2)$
$X\ \varphi \overset{def}{=} \bot\ U^{+}\ \varphi$	$\varphi_1\ U\ \varphi_2 \overset{def}{=} \varphi_2 \vee (\varphi_1\ \wedge\ \varphi_1\ U^{+}\ \varphi_2)$
$F\ \varphi \overset{def}{=} \top U \varphi$	$G\ \varphi \overset{def}{=} \neg F\ \neg\varphi$

The timed operators (with discrete time) are defined as follows:

$$\varphi_1\ \mathcal{U}_{\leqslant k} \varphi_2 \overset{def}{=} \begin{cases} \varphi_2 & \text{if } k = 0 \\ \varphi_2 \vee (\varphi_1 \wedge X\ (\varphi_1\ \mathcal{U}_{\leqslant k-1}\ \varphi_2)) & \text{else} \end{cases}$$

$$\varphi_1\ \mathcal{U}_{=k} \varphi_2 \overset{def}{=} \begin{cases} \varphi_2 & \text{if } k = 0 \\ \varphi_1 \wedge X\ (\varphi_1\ \mathcal{U}_{=k-1}\ \varphi_2) & \text{else} \end{cases}$$

We can now define $F_{\leqslant k}\varphi \overset{def}{=} \top\ U_{\leqslant k}\ \varphi$, $F_{=k}\ \varphi \overset{def}{=} \top\ U_{=k}\ \varphi$, and $G_{\leqslant k}\varphi \overset{def}{=} \neg F_{\leqslant k}(\neg\varphi)$.

Definition 4 (Satisfaction). *A formula φ of SE-LTL is interpreted on a trace of an LKS. Given a trace $\tau = (\tau_{pr}, \tau_{ev})$, a natural i, and a formula φ, we can define the satisfaction relation $\models$ by induction on φ:*

$(\tau, i) \models p$	*iff*	$p \in \tau_{pr}(i)$	*where* $p \in P$
$(\tau, i) \models e$	*iff*	$e = \tau_{ev}(i)$ (*e will be performed next*)	*where* $e \in \mathcal{E}$
$(\tau, i) \not\models \bot$			
$(\tau, i) \models \varphi_1 \Rightarrow \varphi_2$	*iff*	$(\tau, i) \models \varphi_1$ *implies* $(\tau, i) \models \varphi_2$	
$(\tau, i) \models \varphi_1\ U^{+}\ \varphi_2$	*iff*	$\exists j > i\ ((\tau, j) \models \varphi_2$ *and* $\forall\ i < k < j\ \ (\tau, k) \models \varphi_1)$	

We say that a trace τ satisfies a formula φ ($\tau \models \varphi$) if the first state of τ satisfies φ.

$$\tau \models \varphi \quad \textit{iff} \quad (\tau, 0) \models \varphi$$

We can easily extend the satisfaction relation to labeled Kripke structures, which are preferred to sequences in order to model programs.

Definition 5 (Satisfaction by a model and validity). *A labeled Kripke structure $\mathcal{M}$ is called a model of a formula φ, and we write $\mathcal{M} \models \varphi$, if all the traces τ of $\mathcal{M}$ satisfy φ.*
A formula φ is said to be valid if every LKS satisfy it.

Remark 1 (Extension to concurent events). We have chosen a state/event point of view, as $SE\text{-}LTL$ to build our temporal and deontic language. However, we have considered that several events may happen simultaneously. It follows that from each state of a sequence, a set of events can be performed. This reveals to be interesting to model true concurrency (also called non-interleaved concurrency). For instance , if the model of the system contains connected events, they may happen simultaneously. It may be the case with the events $start\_sound$ and $start\_video$ in a multimedia context. In this case, the formula $start\_sound \wedge start\_video$ has to be satisfiable.

From now, each transition of an LKS is labeled with a set of events. ($\lambda : T \rightarrow 2^{\mathcal{E}}$.) A trace over a run $\rho = (s_0, t_0, s_1, t_1, \ldots)$ is a pair $\tau = (\tau_{pr}, \tau_{ev})$ where $\tau_{pr} \in seq(2^P)$ is defined as for $SE\text{-}LTL$, and $\tau_{ev} \in seq(2^{\mathcal{E}})$ is now a sequence of sets of events. We call the extension of $SE\text{-}LTL$ to concurrent events *State/Concurrent Events LTL* ($SCE\text{-}LTL$).

Definition 6 (Syntax of $SCE\text{-}LTL$). *The syntax of $SCE\text{-}LTL$ is the same as the syntax of $SE\text{-}LTL$. Only the semantics of the events differs.*

Definition 7 (Semantics of $SCE\text{-}LTL$). *The semantics of an event in $SCE\text{-}LTL$ is defined by*

$$\tau, i \models e \quad \textit{iff} \quad e \in \tau_{ev}(i) \qquad \textit{where } e \in \mathcal{E}$$

e is satisfied if it is one of the events that are going to be performed simultaneously.
The other formulas have the same semantics as in $SE\text{-}LTL$.

3 Deontic extension

We define here a deontic extension of the state/event formalism described in section 2. In *Standard Deontic Logic* (SDL) [23,17], the semantics of deontic modalities is given by a relation on states (also called worlds). We extend this relation to combine both deontic and temporal aspects.

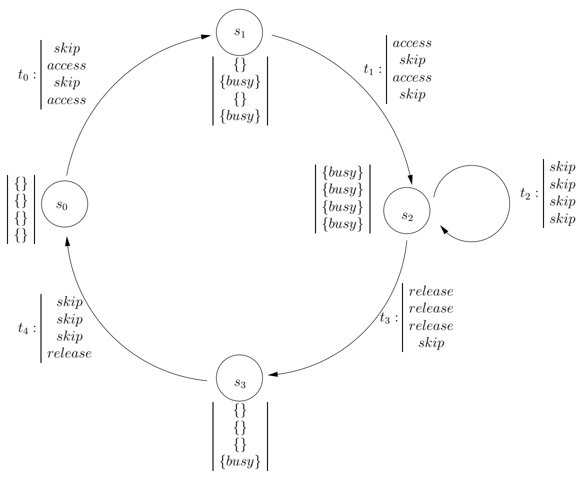
We first present deontic labeled Kripke structures (DLKS). Then, we define the logic *State/Event Deontic Linear Temporal Logic* ($SED\text{-}LTL$), extension of $SCE\text{-}LTL$ with a deontic modality. And in the last part, we discuss about some logic properties in $SED\text{-}LTL$.

3.1 Deontic Labeled Kripke Structures

We present here a deontic extension of a labeled Kripke structure. In our framework, which describes temporal behaviours, a world is a LKS. We call these worlds the alternatives, because they represent different possible behaviours. They all have the same states and transitions, but the labels (on both states and transitions) differ from one alternative to another. Thus, we extend the deontic relation to be a relation on alternatives.

Definition 8 (Deontic labeled Kripke structure). *A deontic labeled Kripke structure over* $(P, \mathcal{E})$ *is a tuple* $(A, a_0, S, I, T, \alpha, \beta, \lambda, \nu, R_o)$ *where*

- A *is a (countable) set of alternative names (hereafter called alternatives).*
- $a_0 \in A$ *is the alternative that corresponds to the real behaviour. The other alternatives are needed to model the deontic aspects.*
- $S, I, T, \alpha,$ *and* β *are defined as in part* 2.2.
- $\lambda : A \times T \to 2^{\mathcal{E}}$ *is the valuation function which associates each transition of an alternative with its label (a set of events that occur simultaneously during the transition). If* $t \in T$, $s, s' \in S$, $E \subseteq \mathcal{E}$, $a \in A$, *we often write* $t : s \to s'$ *to mean that* $\alpha(t) = s$ *and* $\beta(t) = s'$, *and* $t : s \overset{E}{\underset{a}{\to}} s'$ *to mean that* $\alpha(t) = s$, $\beta(t) = s'$, *and* $\lambda(a, t) = E$.
- $\nu : A \times S \to 2^P$ *is the valuation function that associates each state of an alternative with a set of atomic propositions.* $\nu(a, s)$ *represents the set of the atomic propositions satisfied by* $s \in S$ *in the alternative* $a \in A$.
- $R_o \subseteq A \times A$ *is the deontic relation which associates each alternative with the set of its good alternatives.* R_o *is supposed to be serial.*



s_0
s_1
s_2
s_3
t_0 : skip access skip access
t_1 : access skip access skip
t_2 : skip skip skip skip
t_3 : release release release skip
t_4 : skip skip skip release
{} {} {} {}
{} {busy} {} {busy}
{busy} {busy} {busy} {busy}
{} {} {} {busy}

Fig. 1. Example of a DLKS

The figure 1 shows an example of a DLKS in which there are four alternatives. Thus, each transition and each state has four labels. We have not shown the relation R_o.

Definition 9 (Run and trace). *As in part 2.3, we define a run* $\rho = (s_0, t_0, s_1, t_1, \ldots)$ *of a DLKS as an alternating sequence of states and transitions, such that* $s_o \in I$, *and* $\forall i \in \mathbb{N}$ $t_i : s_i \to s_{i+1}$. *A trace* τ *of a run* $\rho = (s_0, t_0, s_1, t_1, \ldots)$ *is a pair*

(τ_{pr}, τ_{ev}) where the sequences τ_{pr} and τ_{ev} are now indexed by the alternative. Indeed, through one run, the labels of states and transitions may differ from one alternative to another.

- *$\tau_{pr} : A \rightarrow seq(2^P)$ associates each alternative with a sequence of proposition sets.*
  $\forall(i, a) \in \mathbb{N} \times A \quad \tau_{pr}(a)(i) = \nu(a, s_i)$
- *$\tau_{ev} : A \rightarrow seq(2^E)$ associates each alternative with a sequence of event sets.*
  $\forall(i, a) \in \mathbb{N} \times A \quad \tau_{ev}(a)(i) = \lambda(a, t_i)$

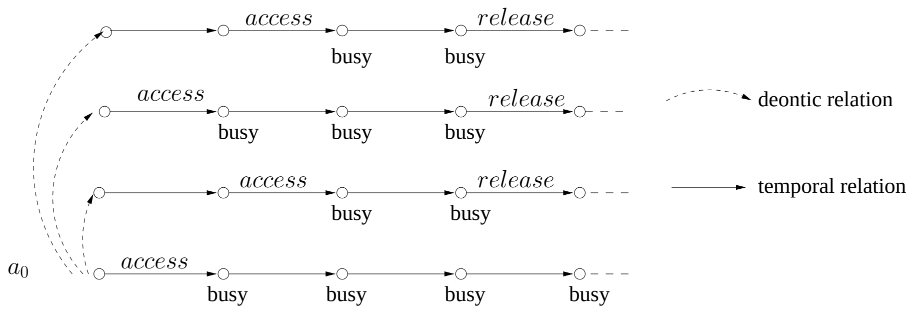


Fig. 2. Example of a DLKS trace

The figure 2 shows an illustration of the trace over the run $(s_0, t_0, s_1, t_1, s_2, t_2, s_2, t_3, s_3, \ldots)$ of the DLKS of the figure 1.

3.2 Syntax and semantics

We present here the syntax of our specification formalism *State/Event Deontic Linear Temporal Logic* (*SED-LTL*), and its semantics.

Definition 10 (Syntax of *SED-LTL*). *Given a countable set P of atomic propositions, and a countable set $\mathcal{E}$ of events, a well-formed formula of SED-LTL is defined by:*

$$\varphi ::= p \in P \mid e \in \mathcal{E} \mid \bot \mid \varphi \Rightarrow \varphi \mid \varphi\ U^+\ \varphi \mid \mathbf{O}(\varphi)$$

The deontic modality $\boldsymbol{O}$ represents the obligation. We can define the permission ($\boldsymbol{P}$) and the prohibition ($\boldsymbol{F}$) as the following abbreviations:

$$\boldsymbol{P}(\varphi) \overset{def}{=} \neg\boldsymbol{O}(\neg\varphi) \quad \boldsymbol{F}(\varphi) \overset{def}{=} \boldsymbol{O}(\neg\varphi)$$

We define some usual operators, and timed operators (with discrete time) as for *SE-LTL* (cf part (2.3)).

We define here the semantics of the formulas in *SED-LTL*.

Definition 11 (Satisfaction). *A formula φ is interpreted on a trace $\tau = (\tau_{pr}, \tau_{ev})$ of a DLKS. Given a DLKS $\mathcal{M}$, an alternative a, a trace τ, an integer i, and a formula φ, we can define the satisfaction relation $\models$ by induction on φ:*

$a, \tau, i \models p$	*iff*	$p \in \tau_{pr}(a)(i)$	*where* $p \in P$
$a, \tau, i \models e$	*iff*	$e \in \tau_{ev}(a)(i)$	*where* $e \in \mathcal{E}$
$a, \tau, i \not\models \bot$			
$a, \tau, i \models \varphi_1 \Rightarrow \varphi_2$	*iff*	$a, \tau, i \models \varphi_1$ *implies* $a, \tau, i \models \varphi_2$	
$a, \tau, i \models \varphi_1\ U^+\ \varphi_2$	*iff*	$\exists i'' > i$ *such that* $a, \tau, i'' \models \varphi_2$ *and* $\forall i'\ i < i' < i'' \ \Rightarrow\ a, \tau, i' \models \varphi_1$	
$a, \tau, i \models \mathbf{O}\varphi$	*iff*	$\forall a'$ *such that* $(a, a') \in R_o\ \ a', \tau, i \models \varphi$	

An alternative in a trace satisfies a formula if its first state satisfies it.

$$a, \tau \models \varphi \quad \textit{iff} \quad a, \tau, 0 \models \varphi$$

A DLKS $\mathcal{M}$ satisfies a formula if the alternative a_0 in the trace of any run satisfies it.

$$\mathcal{M} \models \varphi \quad \textit{iff} \quad \forall \tau \textit{ trace of } \mathcal{M} \ \ a_0, \tau \models \varphi$$

A formula is valid if all the DLKS satisfy it.

$$\models \varphi \quad \textit{iff} \quad \forall \mathcal{M}\ \mathcal{M} \models \varphi$$

Remark 2 (Product). This formalism is very closed to a product of a temporal and a deontic logic. However, in this case, the definition of a product [11] does not match because of the use of events that makes the temporal semantics more complex than in LTL. We plan to study the adaptation of decidability results to $SED\text{-}LTL$.

Let us consider the trace of figure 2. The behaviour of the alternative a_0 accesses to the resource while it is permitted, and does not release it after three time units although it is obliged. This deontic trace satisfies the following formulas, which express the violation of some obligations.

$a_0, \tau \models F_{=3}(\boldsymbol{O}\ release\ \wedge\ \neg release)$
$a_0, \tau \models \boldsymbol{O}\ (access \Rightarrow F_{\leqslant 3} release)\ \wedge\ (access \wedge \neg F_{\leqslant 3} release)$
$a_0, \tau \models G\boldsymbol{O}\ (busy \Rightarrow F_{\leqslant 3} \neg busy)\ \wedge\ X(busy \wedge \neg F_{\leqslant 3} \neg busy)$

3.3 Expression of availability policies

We express here in our formalism the policies given in the introduction.

– *user$_i$ has the permission to use the resource r for 5 time units continuously, and he must be able to access it 15 time units after asking, at the latest* The first part of this sentence has an ambiguity. It is not clear whether it is forbidden to use the resource after 5 time units of utilization. The first formula does not consider it is the case, whereas the second one does.

$$G\ ((access \Rightarrow G_{\leqslant 5}(\boldsymbol{P} use)) \wedge (request \Rightarrow \boldsymbol{O}(F_{\leqslant 15}\ access)))$$

$$G\ ((access \Rightarrow F_{\leqslant 5}\boldsymbol{O}(\neg use)) \wedge (request \Rightarrow \boldsymbol{O}(F_{\leqslant 15}\ access)))$$

– *$user_i$ has always the permission to use the resource r, and he has to release it after 5 time units of utilization*

$$G(\boldsymbol{P}use) \wedge G(access \Rightarrow \boldsymbol{O}(F_{\leqslant 5}release))$$

– *If $user_i$ is asking for the resource and he has the permission to use it, then the system has the obligation to give it to him before 5 time units*

$$G((request \wedge \boldsymbol{P}use) \ \Rightarrow \ \boldsymbol{O}(F_{\leqslant 5}access))$$

– *If $user_i$ uses the resource without the permission, he must not ask for it during 10 time units*

$$G((use \wedge \boldsymbol{O}(\neg use)) \ \Rightarrow \ \boldsymbol{O}(G_{\leqslant 10}\neg request))$$

We have shown that the translation from natural language to our formalism is easy for such sentences, that describe availability policies. We now focus on the logical properties of $SED\text{-}LTL$, and discuss their intuitiveness.

3.4 Finiteness of the set of alternatives

We have not discussed yet about the nature of the set A of alternatives. Our first idea was to consider A countable, by analogy with a usual Kripke structure. But in order to develop a decision procedure, or to reason on some examples, as the figure 2, it would be simpler to have a finite set of alternatives.

Let us exhibit a formula which may be satisfiable or unsatisfiable, whether we consider A finite or countable. Consider the following statement *"p is always permitted, and p ought to happen at most once"*. It corresponds to the formula $\varphi \overset{def}{=} G\boldsymbol{P}p \wedge \boldsymbol{O}(\text{AtMostOne}(p))$, where $p \in P$ is an atomic proposition, and AtMostOne(p) is the abbreviation of $G(p \Rightarrow XG\neg p)$, which means *$p$ happens at most once*. If we consider the class of models such that A is finite, φ is unsatisfiable. Indeed a model of φ necessarily contains a infinite number of good alternatives, since each point of the alternative a_0 satisfies $\boldsymbol{P}p$. For each of these future points, there is an alternative satisfying p different from the other alternatives. It follows that there are as many alternatives as points in the future. The figure 3 shows the construction of a model of φ.

In the same way, φ is clearly satisfiable if we consider that A can be infinite. This shows that our formalism has not the finite model property.

At a first sight, there is no direct contradiction in φ. Thus, it is in the favour of an infinite set A of alternatives.

3.5 Properties

In this part, we will focus on the properties of formulas in which temporal and deontic modalities are nested. For instance, $\boldsymbol{P}X\varphi$, or $\boldsymbol{O}G\varphi$.

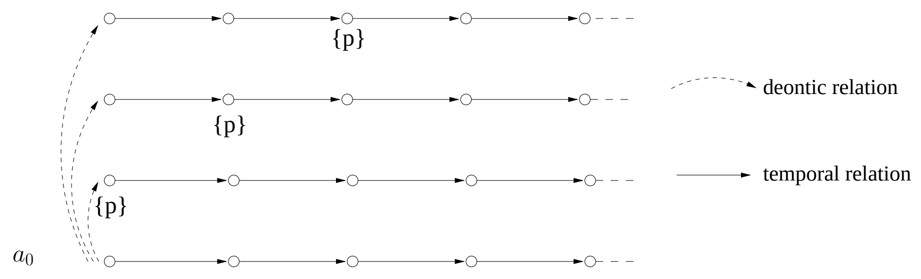


Fig. 3. A model for $G\boldsymbol{P}p \wedge \boldsymbol{O}(\text{AtMostOne}(p))$

The fact that the deontic relation R_o is a relation on sequences, and not on states, has consequences on such formulas. Indeed, the good alternatives of a_0 can be seen as the rules to be obeyed. What is obligatory is then what is true in all the rules, and what is permitted is what is true in at least one rule. Each of these rules is consistent from a temporal point of view. For instance, if $X\varphi$ is permitted in one state (a_0, τ, i), then there exists a good state (a_1, τ, i) which satisfies $X\varphi$. Since $(a_1, \tau, i+1)$ is a good state of $(a_0, \tau, i+1)$, (a_0, τ, i) also satisfies $X\boldsymbol{P}\varphi$. Then, for any alternative a, trace τ, and natural i, we have
$(a, \tau, i) \models \boldsymbol{P}X\varphi \Rightarrow X\boldsymbol{P}\varphi$
In the same way, we prove the converse property $(a, \tau, i) \models X\boldsymbol{P}\varphi \Rightarrow \boldsymbol{P}X\varphi$
Therefore,

$$\models \boldsymbol{P}X\varphi \;\Leftrightarrow\; X\boldsymbol{P}\varphi$$

This is in accordance with one intuition. Indeed, the natural language gives some ambiguities to the expression "I have the permission to check in tomorrow". One can understand that this permission can be withdrawn before tomorrow. That is not our point of view. We consider the permissions globally, and not from an operational point of view. In other words, we do not model changing norms. If the permission to check in does not exist tomorrow, then we cannot say today that we have the permission to check in tomorrow.

The same reasoning shows that the two modalities $\boldsymbol{O}$ and X commute.

$$\models \boldsymbol{O}X\varphi \;\Leftrightarrow\; X\boldsymbol{O}\varphi$$

As a consequence, the deontic modalities commute with the timed operator $F_{=k}$ (cf. part 2.3).

$$\models \boldsymbol{O}F_{=k}\;\varphi \;\Leftrightarrow\; F_{=k}\;\boldsymbol{O}\varphi$$

$$\models \boldsymbol{P}F_{=k}\;\varphi \;\Leftrightarrow\; F_{=k}\;\boldsymbol{P}\varphi$$

This is not the case with the operators F, i.e., the property $\boldsymbol{O}F\varphi \;\Leftrightarrow\; F\boldsymbol{O}\varphi$ is not valid, which is intuitive. Indeed, having the obligation to meet φ in the future does not imply that there will be an immediate obligation to satisfy φ. On the other hand, we would like the obligation to continue while φ is not satisfied.

The general expression of this propagation property is :

$$\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg \boldsymbol{O}\varphi \wedge \neg\varphi \;\Rightarrow\; X\,\boldsymbol{O}\psi \tag{1}$$

We need $\boldsymbol{O}(\varphi \vee X\psi) \wedge \neg \boldsymbol{O}\phi$ in the hypothesis of (1) rather than only $\boldsymbol{O}(\varphi \vee X\psi)$. Indeed, we want to express the fact that the disjunction $\varphi \vee X\psi$ is obligatory, but not φ.

A consequence of the simpler formula $\boldsymbol{O}(\varphi \vee \psi) \wedge \neg\varphi \;\Rightarrow\; X\,\boldsymbol{O}\psi$ would be $\boldsymbol{O}(\varphi) \wedge \neg\varphi \;\Rightarrow\; X\,\boldsymbol{O}\psi$, where ψ can be any formula, which is of course not desirable.

However, the property (1) is not valid because of the semantics of $\boldsymbol{O}$ given by the deontic relation R_o. The problem does not rely on the addition of the temporal dimension, but on the interaction between obligation and disjunction. The problem raised by propagation comes from the fact the following property is not valid in SDL:

$$\boldsymbol{O}(\varphi_1 \vee \varphi_2) \wedge \neg \boldsymbol{O}\varphi_1 \wedge \neg\varphi_1 \;\Rightarrow\; \boldsymbol{O}\varphi_2$$

The intuition suggests a strong interaction between what happens ($\neg\varphi_1$ in this formula) and what is obliged ($\varphi_1 \vee \varphi_2$). This interaction does not exist in SDL, and thus does not exist either in our formalism. The obligation we model is a norm that does not depend on the facts.

We have the same problem with the obligation with deadline ($\boldsymbol{O}F_{\leqslant k}\varphi$), because $F_{\leqslant k}\varphi$ is equivalent to $\varphi \vee XF_{k-1}\varphi$ (if $k \geqslant 1$). We will focus on the notion of obligation with deadline, which is closer to real cases than the obligation to satisfy φ in the future without deadline.

So, we would like the following property to be satisfied

$$\boldsymbol{O}\ F_{\leqslant k}\varphi \wedge \neg \boldsymbol{O}\varphi \wedge \neg\varphi \;\Rightarrow\; X\,\boldsymbol{O}F_{\leqslant k-1}\ \varphi$$

As explained, this property is not satisfied. To overcome this problem, we introduce a new operator in the next section.

4 Obligation with deadline

In this part, we will focus on the notion of obligation with deadline. We want to specify that it is obliged to satisfy φ before k time units. We will use the notation $\mathcal{O}_k(\varphi)$.

4.1 A first definition

The natural way to specify it is : $\mathcal{O}_k(\varphi) \overset{def}{=} \boldsymbol{O}(F_{\leqslant k}\ \varphi)$

Do we have properties that are in accordance with our intuition? First of all, we have the property of monotonicity with respect to the deadline:

$$\models \mathcal{O}_k(\varphi) \Rightarrow \mathcal{O}_{k'}(\varphi) \quad \text{where } k \leqslant k'$$

It is in accordance with the first intuition we have of obligation with deadline. Indeed, if it is obliged to meet φ before k time units, then it is also obliged to meet φ before a greater deadline k'.

Another property is the property of monotonicity with respect to φ:

$$\models \mathcal{O}_k(\varphi_1 \wedge \varphi_2) \Rightarrow \mathcal{O}_k(\varphi_1) \quad \text{and} \quad \models \mathcal{O}_k(\varphi_1) \ \Rightarrow \ \mathcal{O}_k(\varphi_1 \vee \varphi_2)$$

However, as we said in part (3.5), the property of propagation of an obligation with deadline is not satisfied. It expresses that if it is obliged to satisfy φ before k time units, and if φ is not satisfied now, then in one time unit, it will be obliged to satisfy φ before $k-1$ time units. It corresponds to the formula:

$$\mathcal{O}_k(\varphi) \wedge \neg \boldsymbol{O}\varphi \wedge \neg\varphi \ \Rightarrow \ X\mathcal{O}_{k-1}(\varphi)$$

This is not a valid formula in *SED-LTL*. Indeed, a model satisfies $\mathcal{O}_k\varphi$ if all the good alternatives of a_0 satisfy φ before k time units. But one of these good alternatives may satisfy φ now, and $\neg\varphi$ thereafter. In this case, the obligation does not hold in one time unit, even if φ has not been satisfied.

4.2 A new operator for obligation with deadline

The fact obligation with deadline is not propagated while it is not complied with implies that it can be violated without having an immediate obligation at any moment. In other words, $\boldsymbol{O}(F_{\leqslant k}\varphi) \wedge \neg F_{\leqslant k}\varphi \ \wedge \ G\neg \boldsymbol{O}\varphi$ is satisfiable. A solution is to have a "counter" which is set to k when $\boldsymbol{O}F_{\leqslant k}\ \varphi$ holds, and decremented while φ is not satisfied. So, we have to define a new operator dedicated to the obligation with deadline, $\mathcal{O}_k(\varphi)$, which means that there is an obligation to meet φ before k which has not been fulfilled yet.

$$\begin{array}{llll} a,\tau,i \models \mathcal{O}_k(\varphi) & \text{iff} & \exists k' \in \mathbb{N} & a,\tau,i-k' \models \boldsymbol{O}F_{\leqslant k+k'}\ \varphi \ \wedge \ \neg\varphi U_{=k'}\ \top \\ & & \wedge\ \nexists k'' < k+k' & a,\tau,i-k' \models \boldsymbol{O}F_{\leqslant k''}\ \varphi \end{array}$$

More precisely, $\mathcal{O}_k(\varphi)$ may be read as *"k' time units ago, there were an obligation to satisfy φ before $k+k'$, and the obligation has not been fulfilled yet"*. The second part of the definition (on the second line) means that k' time units ago, there was no obligation to satisfy φ before a shorter deadline. Otherwise, an immediate obligation $\boldsymbol{O}\varphi$ would imply $\mathcal{O}_k\varphi$ (for any k) while φ is not satisfied.

Because of this last point, the new operator cannot be deduced from $\boldsymbol{O}(F_{\leqslant k}\varphi)$. Indeed, if $\boldsymbol{O}\varphi$ is satisfied at some moment, then $\boldsymbol{O}(F_{\leqslant k}\varphi)$ is also true, whereas $\mathcal{O}_k(\varphi)$ may not be true. There exists a, τ, i such that

$$a,\tau,i \nvDash \boldsymbol{O}F_{\leqslant k}(\varphi) \Rightarrow \mathcal{O}_k(\varphi)$$

In the same way, the property of monotonicity with respect to the deadline does not hold either: there exists a, τ, i such that

$$a,\tau,i \nvDash \mathcal{O}_k(\varphi) \ \Rightarrow \ \mathcal{O}_{k'}(\varphi) \quad \text{where } k \leqslant k'$$

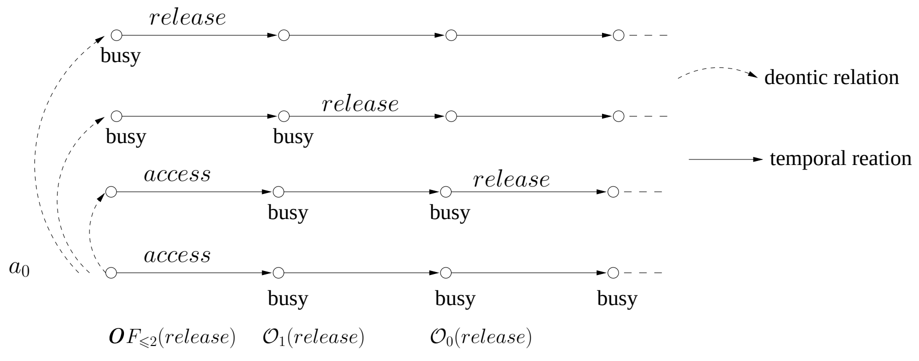


Fig. 4. Illustration of the obligation with deadline operator

Indeed, unlike the first definition (cf section 4.1), the new definition of $\mathcal{O}_k(\varphi)$ considers an exact deadline k: what is obligatory is to meet φ before k time units, not to meet φ before $k+1$, or $k-1$ time units. Of course, the fact that $\mathcal{O}_k(\varphi)$ is complied with, which corresponds to $F_{\leqslant k}\varphi$, has the monotonicity property with respect to k.

With a deadline 0, the new operator can be seen as a generalization of the obligation $\boldsymbol{O}$. Indeed, when φ is obligatory, in the sense of $\boldsymbol{O}$, then $\mathcal{O}_0(\varphi)$ holds, but the converse property is not true. For any a, τ, i

$$a, \tau, i \models \boldsymbol{O}(\varphi) \Rightarrow \mathcal{O}_0(\varphi)$$

But there exists a, τ, i such that $a, \tau, i \not\models \mathcal{O}_0\varphi \Rightarrow \boldsymbol{O}\varphi$.
Thus, $\mathcal{O}_0$ can be seen as a generalised immediate obligation. Indeed, it holds if

- there is an explicit immediate obligation $\boldsymbol{O}$
- or there is an obligation with deadline which has not been fulfilled, and we have reached the deadline

For instance, consider the case where we check that a violation condition for a formula φ does not hold. $\mathcal{O}_0(\varphi) \wedge \neg\varphi$ is much stronger that the usual violation condition $\boldsymbol{O}(\varphi) \wedge \neg\varphi$.

Note that since the semantics of the operator $\mathcal{O}_k$ involves the past, we give properties for any step i, given a trace τ and an alternative a. Until this point, the temporal operators only involved the future of the current state, so the properties was given at step 0, and thus expressed as validity properties.

Figure 4 shows an illustration of obligation with deadline. In the first state of a_0, there is an obligation to release the resource before three time units. Indeed, each of the three good alternatives of a_0 do so. In the second state, the resource has not been released, but the obligation does not hold anymore because one of the good alternatives has already released the resource. But our new operator $\mathcal{O}_k$ holds until the deadline is reached. And in the fourth state, there is a generalised immediate obligation to release the resource.

5 Related work

In this section, we compare our formalism with some existing formalisms. We have proposed to define both deontic and temporal concepts using a Kripke model. In [16,9], Meyer proposes a definition of deontic notions in dynamic logic, based on the reduction of deontic logic to alethic modal logic by Anderson [1]. The idea is to introduce a new proposition V that represents the violation of an obligation, and to define obligation in terms of this violation proposition. This differs from our formalism since we have defined deontic concepts by a relation on states of a Kripke model, and then the violation of some obligation $\boldsymbol{O}(\varphi)$ as $\boldsymbol{O}(\varphi) \wedge \neg\varphi$. A consequence is that reductionist approaches cannot distinguish the violation of a formula φ_1 from the violation of another formula φ_2.

Broersen et al. also suggests in [3] a reductionist approach. The goal is to model obligations with deadline in the branching time logic CTL (*Computation Tree Logic* [6,19]), using a violation constant and an ideality constant. The deadline is a formula and not a concrete duration as suggested in this paper. This is an interesting point of view which can be considered as a more abstract point of view as ours. Dignum et al. [10] have also proposed a reductionist approach to study the specification of deadlines.

Some other formalisms represent both time and deontic notions using relations on states in Kripke models. For instance, in [22,20] there is no constraint that link the two relations (deontic and temporal). This is another possible interpretation of the natural language, which corresponds to the case where the obligations can change over time. We have chosen here to model obligations that cannot be withdrawn, or added if they are in contradiction with previous obligations. For instance, $\boldsymbol{O}Gp \wedge X\neg\boldsymbol{O}p$ and $\boldsymbol{O}Gp \wedge X\,\boldsymbol{O}(\neg p)$ are both unsatisfiable in $SED\text{-}LTL$. If one gets the obligation to always establish p, this obligation cannot be withdrawn tomorrow, and a new obligation cannot be added if it is in contradiction with the previous one.

In [7], Cuppens et al. present the formalism $NOMAD$ which can be considered as a starting point of our study. Nevertheless, the only temporal operator is “next”, and there is a distinction between two kinds of obligations: contextual obligations, and effective obligations. Moreover, they have chosen a representation of actions which differs from our concurrent events. Actions have a duration, and they are observed via the three predicates *start*, *done*, and *doing*.

Aqvist proposes in [2] a logic that combines temporal and deontic modalities. His model differs from ours. The deontic relation associates two states of two histories only if they share the same past. This implies an interesting interaction between the two dimensions. The temporal operator is the “absolute” operator R_t (“it holds at the time t that”) instead of the “relative” operators X and U, which are more expressive.

We have talked in this paper neither about the entities who give the obligations, nor about those who are concerned by the obligations. For instance, [15,20] present a formalism where the deontic modalities are indexed by agents. There are in the model as many deontic relations as there are agents. It would be interesting to integrate these aspects which would give a greater expressiveness.

6 Conclusion and future work

We have proposed a deontic and temporal logic that takes into account states and events, following the idea of *SE-LTL*[5]. We have focused on the interaction between the temporal dimension and the deontic dimension. Thus, we have discussed the intuitiveness of some properties of formulas where deontic and temporal modalities are nested, and we have studied more particularly obligation with deadline.

We plan to study a modification of the semantics of obligation that would take into account an interaction between the facts and the rules, in order to have the propagation property. We also want to study a (sound and complete) axiomatization of *SED-LTL*, and its decidability. Indeed, a decision procedure would allow to check the internal coherency of an availability policy expressed in this formalism. Another direction for future work is to check if a system does not violate an obligation within a policy. This last problem, which requires to reason on both a pure temporal behaviour (the system) and a deontic temporal formula (the policy), need to be formalised.

As we said in section 5, it would be interesting to integrate the notion of agents. We have begun to model access control policies which can be described in the OrBAC[14] model, where the notion of agent appear. Following [20] in which roles and groups of agents are represented, we are studying an agent based extension of *SED-LTL* in order to express such policies.

References

1. A. R. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, pages 100–103, 1958.
2. L. Aqvist. Combinations of tense and deontic logic. *Journal of Applied Logic*, 3:421–460, 2005.
3. J. Broersen, F. Dignum, V. Dignum, and J.-J. C. Meyer. Designing a deontic logic of deadlines. In *7th International Workshop on Deontic Logic in Computer Science (DEON'04)*, Madeira, Portugal, 26-28 May 2004.
4. S. Chaki, E. Clarke, O. Grumberg, J. Ouaknine, N. Sharygina, T. Touili, and H. Veith. State/event software verification for branching-time specifications. In *Fifth International Conference on Integrated Formal Methods (IFM 05)*, volume 3771 of *Lecture Notes in Computer Science*, pages 53–69, 2005.
5. S. Chaki, E. M. Clarke, J. Ouaknine, N. Sharygina, and N. Sinha. State/event-based software model checking. In E. A. Boiten, J. Derrick, and G. Smith, editors, *Proceedings of the 4th International Conference on Integrated Formal Methods (IFM '04)*, volume 2999 of *Lecture Notes in Computer Science*, pages 128–147. Springer-Verlag, April 2004.
6. E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proceedings of the 3rd Workshop of Logic of Programs (LOP'81)*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.
7. F. Cuppens, N. Cuppens-Boulahia, and T. Sans. Nomad: a security model with non atomic actions and deadlines. In *Proceedings of the 18th IEEE Computer Security Foundations Workshop*, June 2005.

8. F. Cuppens and C. Saurel. Towards a formalization of availability and denial of service. In *Information Systems Technology Panel Symposium on Protecting Nato Information Systems in the 21st Century*, Washington, 1999.
9. P. d'Altan, J.-J. C. Meyer, and R. Wieringa. An integrated framework for ought-to-be and ought-to-do constraints. *Artif. Intell. Law*, 4(2):77–111, 1996.
10. F. Dignum and R. Kuiper. Obligations and dense time for specifying deadlines. In *Thirty-First Annual Hawaii International Conference on System Sciences (HICSS)-Volume 5*, 1998.
11. D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyachev. *Many-Dimensional Modal Logics: Theory and Applications.* Elsevier, 2003.
12. D. Harel, D. Kozen, and J. Tiuryn. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*, pages 497–604. D. Reidel Publishing Company: Dordrecht, The Netherlands, 1984.
13. J. G. Henriksen and P. S. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96(1-3):187–207, 1999.
14. A. A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization based access control. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003)*, Lake Come, Italy, June 2003.
15. A. Lomuscio and M. Sergot. On multi-agent systems specification via deontic logic. In *Agent Theories Languages, and Architectures*, volume 2333 of *Lecture Notes in Artificial Intelligence*, Seattle, Springer 2001. Springer Verlag.
16. J.-J. C. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 1988.
17. J.-J. C. Meyer, R. Wieringa, and F. Dignum. The role of deontic logic in the specification of information systems. In *Logics for Databases and Information Systems*, pages 71–115, 1998.
18. A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.
19. J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. In *Proceedings of the 5thInternational Symposium on Programming (SOP'82)*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351, 1982.
20. L. van der Torre, J. Hulstijn, M. Dastani, and J. Broersen. Specifying multiagent organizations. In *Seventh International Workshop on Deontic Logic in Computer Science (DEON'04)*, volume 3065 of *Lecture Notes in Computer Science*, pages 243–257, 2004.
21. R. J. Wieringa and J.-J. C. Meyer. *Applications of Deontic Logic in Computer Science: A Concise Overview*, pages 17–40. John Wiley & Sons, 1993.
22. B. Wožna, A. Lomuscio, and W. Penczek. Bounded model checking for deontic interpreted systems. In *Electronic Notes in Theoretical Computer Science*, volume 126, pages 93–114, march 2005.
23. G. H. V. Wright. Deontic logic. *Mind*, 1951.