F. Dignum and R. Kuiper. Obligations and Dense Time for Specifying Deadlines. In *Proceedings of thirty-First HICSS*, Hawaii, 1998.

Obligations and Dense Time for Specifying Deadlines

F. Dignum,
R. Kuiper
Fac. of Maths. & Comp. Sc.,
Eindhoven University of Technology
P.O.box 513, 5600 MB Eindhoven,
The Netherlands, tel.+31-40-472733,
e-mail: {dignum,wsinruur}@win.tue.nl

Abstract

In this paper we consider the notion of deadlines in the context of dense time. We show that obligations and actions are essential elements for the specification of deadlines. These notions can be relatively easy combined when a discrete temporal framework is used. However, we show that, once a dense time is introduced, several problems appear. In solving these problems we cannot use the same framework and definitions as used for the discrete time. In the new framework we use a branching dense temporal framework as a basis to specify both actions and obligations. Finally we show that all types of deadlines that were defined for the discrete temporal framework can also be defined for dense time.

1 Introduction

In [6] we have already argued that it is important to be able to specify deadlines in a declarative way for e.g. agent applications. In this way it is possible for the agent to reason about the deadlines *before* the planning process determines the next (sequence of) action(s).
Deadlines cannot be modeled as hard constraints because:

- It is not always possible to keep all deadlines at the same time (they are logically inconsistent).
- the agent does not always have control over all actions (or conditions).

Therefore we propose the introduction of deontic logic in which it is possible to reason about states in which obligations are not fulfilled.

It is also clear that deadlines involve a temporal element and an action element. Every deadline specifies some temporal condition before which an action has to be performed. Therefore the modeling framework should contain some type of action descriptions as well as some temporal concepts.

In [6] we have shown how temporal operators can be added to the dynamic logic framework to model deadlines. This can be done relatively easy in the case of a discrete temporal framework. In that case actions are performed in lockstep and the duration of an action is one or more time units.

In some cases a discrete temporal framework does not suffice to model the real world. This is the case when actions can occur at random moments in time and are time critical. Sampling at discrete intervals might mean that an action is discovered to late to react or it is not possible to set a new deadline accurately on the basis of the occurrence of the action.
This happens for instance with the application of medicines to patients. It might be that a patient needs to be administered some medicine before 10 minutes have elapsed. After the medicine has been given another medicine should be given within 5 minutes. If we set the time frame to five minutes this simple example can already go wrong.
Although each example can be remedied by choosing the granularity of the time intervals smaller it is often difficult on forehand to determine which is the smallest

interval that will suffice for all cases in a certain application. In these cases it is worthwhile to explore the use of continuous (or dense) time.

In [7] we explored the possibilities to combine obligations with a temporal framework based on continuous time. In this case the Kripke structures that function as semantics for dynamic logic cannot function as a basis for the combination anymore. This is due to their inherent discrete nature. Therefore we have to use a model for continuous temporal logic as a basis for the combination. On top of such a temporal model we have to construct an action logic, because the representation of (combinations of) actions is essential for the description of deadlines.

In [7] we already sketched some other approaches that combine temporal and deontic logics. We will recapitulate them here briefly.
The approaches described in e.g. [18, 13, 10, 8, 9] tend to express the deontic concepts/operators in terms of temporal concepts/operators. This is the most obvious in the approach of [10] where obligation is viewed as a kind of *liveness* condition: something *will* happen sometime in the future. In our view such a reduction is not appropriate, since the notions of obligation and liveness express quite different properties conceptually: liveness expresses that something will definitely happen while an obligation expresses that something should happen (it is *desirable* that something happens) but in fact might never happen at all! Also the other time-based approaches are quite different from what we are aiming at in this paper. For instance, Thomason [18] uses a reduction of obligation to truth in all future courses of action/time that are morally acceptable, and Horty [13] considers a branching-time framework and defines obligations on the basis of the choices available to the agent and 'stit' (seeing to it that) constructs (cf. [4]). Finally, Van Eck [8, 9] employs a temporal necessity operator $\Box_t$ that is evaluated with respect to a *set* of (linear) time structures ('world courses' in the terminology of Van Eck): $\Box_t\varphi$ means that in every world course that is the same up to t (and *not* including t) the formula φ is true. From this definition one easily observes that anything that is true in the past (i.e. before time t) is *necessarily* true. Philosophically this is defended as follows: since the past cannot be changed, there are no other possibilities than what actually happened, so this happened *necessarily*. Furthermore, Van Eck considers in [9] obligations of the form $\varphi O_t \psi$ expressing that in all world courses that are possible from time t onwards and are as perfect as possible–given that φ is the case in them–ψ is the case. Van Eck uses these modalities idea to solve contrary-to-duty paradoxes such as a certain (the 'Suzy Mae' or 'forward') version of the Chisholm paradox [5, 16]. The notions of temporal necessity and obligation as considered by Van Eck are very different from our temporal and deontic operators. Van Eck's obligation may be viewed as a temporal variant of a (deontic) conditional, whereas our obligation operator has no such flavour.

We will use a dynamic obligation based on the one defined in [14]. Also we try to add a temporal dimension to these dynamic obligations instead of expressing the obligations in terms of some temporal operators.

This paper should be viewed as a first experiment to express deadlines in a (branching) dense time. We do not give an axiomatization for this logic or even a set of inference rules. Therefore automatic reasoning with the specifications is not yet possible.

The rest of this paper is organized as follows. First we will discuss the topic of action description in a dense temporal framework where obligations are used. In section 3 we describe the dynamic obligations within a branching dense temporal framework. We also discuss the topic of persistence of violations in this framework. In section 4 we describe the deadlines that were already defined in [6] in this new framework. It appears that some definitions become more perspicious and elegant in this new setting! Section 5 contains some conclusions and directions for further research.

2 Action description in a temporal framework

It is clear from the above that the notion of action is important in our framework. We cannot take just any action representation, because we also want to be able to represent combinations of actions and their effects (as could be done in dynamic logic). This is especially important because we represent the obligation to perform an action in dynamic deontic logic (DDL) as:

$$O(\alpha) =_d ef [\overline{\alpha}]V$$

Which means that α is obliged iff not performing α leads to a state in which V, standing for violation, holds. In order to keep this intuitive definition of the dynamic obligation in the combined logic we have to be able to represent negations of actions.

In a framework with continuous time a common way to represent actions with durations is to define momentaneous actions that mark the start and end point of

the actual action. E.g. writing an article would be represented by the following formula:

$$start(write\_article, t) \wedge$$

$$finish(write\_article, t + duration(write\_article))$$

Some care should be taken to ensure that the action denoting the finishing of the actual action is linked to the right starting action.
However, using this formulation leads to the following representation of the obligation of writing an article:

$$O(start(write\_article, t) \wedge$$

$$finish(write\_article, t + duration(write\_article)))$$

which (in all deontic systems) is equal to:

$$O(start(write\_article, t)) \wedge$$

$$O(finish(write\_article, t + duration(write\_article)))$$

The problem is that the obligation of writing the article is translated to two separate obligations. Do I violate the obligation of writing the article if I do not start it at time t but still adhere to the obligation to finish it in time? If the obligation on the action is geared to an obligation on the result of the action, we are only interested in the finishing action. If the obligation is really meant to enforce performing the action, we are only interested that the obligation to start the action has been fulfilled. It appears that this type of action description is not suitable when we want to express obligation on actions as well.

An alternative approach to describe actions in a temporal framework is given by Allen [1]. He uses temporal intervals as primitives in the logic. Therefore the representation of writing an article becomes:

$$OCCUR(write\_article, t)$$

where t stands for an interval.
Now the obligation on an action is represented in an atomic way:

$$O(OCCUR(write\_article, t)),$$

where we assume O to be a standard 'ought-to-be' obligation operator (as in standard deontic logic, cf. e.g. [15]). However, we now have the problem that the obligation also seems to include the specific interval during which the action is supposed to occur. When is the above obligation violated? Formally this happens when $\neg OCCUR(write\_article, t)$ is true. However, this can be true due to two reasons:

1. The article is not written
2. The interval t is not correct (too small or large)

Intuitively a violation of the obligation to write the article means that the article is not written. The interval (time) that it takes to write the article should be intrinsic to the action. However, formally, it seems difficult to prevent the violation on the basis of a "wrong" interval.

It appears from the observations given above that the type of representation that should be used for action descriptions depends on the way the actions are used within obligations. Therefore some of the usual ways to describe actions in a temporal framework are not appropriate when used in combination with obligations. We will show in section 3, how actions can be represented such that the usual properties of obligations are preserved also in this framework. One important aspect in this respect is the use of branching time for the temporal framework. The branching time semantics is needed to "simulate" the branching nature that is inherent in the Kripke models of dynamic logic.

Besides the problem of how to represent "simple" actions in a temporal framework with continuous time in the presence of obligations, we address two questions that did not arise concerning the representation of combinations of actions in a framework with discrete time. The first issue has to do with choices between actions. In a discrete time framework the actions are performed in one step. (Every action that takes more time is split into a sequence of subactions). Therefore, performing $\alpha + \beta$ means that either α has been performed or β has been performed after one time step. However, in a framework with continuous time one might perform $\alpha + \beta$ by starting with α then stopping α and starting β right away, etc. All the while one is performing either α or β but even after the time it would cost to perform α followed by β one might not have finished either one of them.

A second issue is the synchronisation of parallel actions. If all actions have equal length then performing $\alpha \& \beta$ means performing α and β at the same time until they are both finished.
However, if α takes less time to perform than β then it is not possible to say that performing $\alpha \& \beta$ means performing them both at the same time until they are finished. During some time α is finished while β is still performed.

We do not intend to solve all these issues in a generic way in this paper, but will give an intuitive definition of obligations on all types of actions. This will form the

basis of the definition of the deadlines given in section 4.

3 Obligations in a dense temporal logic

In this section we will investigate how the concepts of dynamic deontic logic as defined in e.g. [14] can be expressed in a temporal framework.
Extending the notion of action to dense time means adding duration. This is achieved by introducing two new state predicates, $DOING(\alpha)$, which denotes that action α is being performed and $DONE(\alpha)$ which denotes that in the timepath that led to the state where it is evaluated, α has been performed as the last action. At first sight it might seem superfluous to introduce both a DO and a $DOING$ predicate. Consider, however, the case that $O(\alpha)$ is NOT satisfied, because in the moment following the state where $O(\alpha)$ is demanded the execution of α does not commence. This should be noticable immediately, but because α has duration, the evaluation of $DONE(\alpha)$ can only take place after enough time has expired for α to have completed. Hence the introduction of $DOING(\alpha)$, not so much to enable noticing that α is being executed, but rather to sense that it isn't by means of the expression $\neg DOING(\alpha)$.

The actions that have to be performed before the deadlines can also be composite. The operators that combine the actions are + for a choice between actions, & for parallel execution of actions, $^-$for the negation of an action and ; for the sequence of actions.

Based on these operators and a set $A = \{a, b, c, ...\}$ of basic actions we can define a set Act of action expressions by the following BNF:

$$\alpha ::- \ x|\alpha_1 + \alpha_2|\alpha_1 \& \alpha_2|\overline{\alpha}|\alpha_1; \alpha_2$$

With $x \in A$.

We assume the following equivalences for the $DOING$ and the $DONE$ operator:

Definition 3.1

1. $DOING(\alpha_1 + \alpha_2) = DOING(\alpha_1) \vee DOING(\alpha_2)$
2. $DOING(\alpha_1 \& \alpha_2) = DOING(\alpha_1) \wedge DOING(\alpha_2)$
3. $DOING(\overline{\alpha}) = \neg DOING(\alpha)$
4. $DONE(\alpha_1 + \alpha_2) = DONE(\alpha_1) \vee DONE(\alpha_2)$
5. $DONE(\alpha_1 \& \alpha_2) = DONE(\alpha_1) \wedge DONE(\alpha_2)$
6. $DONE(\overline{\alpha}) = \neg DONE(\alpha)$
7. $DONE(\alpha_1; \alpha_2) = DONE(\alpha_2) \wedge Before(\alpha_2) DONE(\alpha_1)$

We will not make the operator $Before(\alpha)$ explicit in this paper due to space and time constraints. However, it can be expressed in a quite natural manner using standard temporal operators.
Furthermore, to avoid distracting technicalities we assume that each instance of an action is uniquely named. There are various well established ways to deal with duplicate instances of actions; a crude but straightforward way to avoid them is timestamping.

In the following subsection we give the syntax and semantics of the branching dense time temporal logic in which these concepts are embedded. Then we will indicate how the obligation on actions can be defined in this logic and which are the problems.

BTLcont syntax

There are various temporal operators possible, of which at the moment we only need the until (U) and the operator A that stands for "all paths to the future". The latter operator is needed to distinguish formulas that hold in all possible future paths and formulas that only hold in some future paths. A distinction typical for branching temporal logics.
Besides the $DOING$ and the $DONE$ operator over actions we also introduce a $PREFER$ operator that indicates of two actions which one is preferred to be performed (at a certain moment). This operator is used to model obligations that have to be performed as soon as possible but not before some more important or "preferred" actions have been executed.

Definition 3.2
The set of BTLcont formulas is defined inductively, given a set PV (of Propositional Variables, including special ones *true*, *false*, and V).

1. every member of PV is a formula;
2. if $\alpha \in Act$ then $DONE(\alpha)$ and $DOING(\alpha)$ are formulas;
3. if $\alpha_1, \alpha_2 \in Act$ then $PREFER(\alpha_1, \alpha_2)$ is a formula;
4. if ϕ and ψ are formulas, then so are $\neg\phi$ and $\phi \vee \psi$;
5. if ϕ, ψ are formulas, then so are $A\phi$ and $(\phi U \psi)$.

As in [6] we can define the derived operators $\Diamond$ (eventually) and $\Box$ (always from now) as follows:

- $\Diamond\phi := true\mathcal{U}\phi$;
- $\Box\phi := \neg\Diamond\neg\phi$;

These operators will be used later in the definition of deadlines.

BTLcont semantics

To obtain the semantics, we extend the evaluation function for one (implicit) world from the propositional case to an evaluation function defined for a tree of dense discrete paths of worlds; the ordering representing time.

Think of, for example, sequences ρ of rational or real numbers as the worlds, the usual order as (time) relation). For a more extensive description of the logic, see [19].

Definition 3.3
Let $M = \langle T, \pi \rangle$ be a *model*, where T is a tree of worlds and π assigns to every w_i a set of propositional variables and formulas of the form $PREFER(\alpha_1, \alpha_2)$, $DONE(\alpha)$ and $DOING(\alpha)$ (that are true in that world). $\rho = (w_0, \ldots)$ is a path through the tree T. We denote the suffix $(w_i, \ldots)$ of ρ by ρ_i.

1. $M, w_i \models q$ iff $q \in \pi(w_i)$, for $q \in PV$;
2. $M, w_i \models \neg\phi$ iff not $M, w_i \models \phi$;

   $M, w_i \models \phi \vee \psi$ iff $M, w_i \models \phi$ or $M, w_i \models \psi$;
3. $M, w_i \models (\phi U \psi)$ iff there is $k \geq i$ such that $M, w_k \models \psi$ and for all j, $i \leq j < k$, $M, w_j \models \phi$.
4. $M, w_i \models A(\phi)$ iff for all ρ_i through T, $(\exists w_j \in \rho_i)\ w_j \models \phi$

An BTLcont formula ϕ is said to be *true in* a model M (written $M \models \phi$) iff $w_0 \models \phi$. And ϕ is *valid* iff for every model M, $M \models \phi$. This notion of validity is called *anchored validity*.

Note that the index i is an element of $R_0{}^+$, the positive real numbers plus 0. The index notation is maintained to provide a close connection to the more widely used integer based framework. Also note that we tacitly assume that technicalities like finite variability ([3]) have been taken care of.

BTLcont definition of obligation

In BTLcont we can now try and propose a definition of obligation in the same spirit of the formula given in section 2 using dynamic logic.

Definition 3.4

$$O(\alpha) =_{def} A(DOING(\alpha)U(DONE(\alpha) \vee V))$$

Intuitively this definition states that an action is obligated iff along every time path it is the case that α is being performed until either α is finished (done) or we are in a violation state. The latter can be read equivalently that one is performing (doing) α until a state is reached such that if there is no violation the action α is finished, or again equivalently until a state is reached such that if the action is not done / finished we are in violation. This last reading is indeed very close to the reading of the definition of obligation in the dynamic logic setting.

In [6] the obligation to perform an action was expressed in terms of an obligation for a certain formula to hold:

$$O(\alpha) =_{def} [\mathbf{any}]O(DONE(\alpha))$$

which means that whatever you do now afterwards the formula $DONE(\alpha)$ is obliged to hold. In this logic we do not have the special violation predicates V, but have an O operator for formulas. Although this logic is richer in principle it also introduces many complications, because we have to define the truth value of $O(\phi)$ for any formula ϕ. This involves the introduction of another semantic relation orthogonal to the temporal dimension.
We avoided these complications through the introduction of the violation predicate V, which can be seen as an abreviation of the formula:

$$O(\phi) \wedge \neg\phi$$

This keeps the logic simpler and more adequate for our present purpose. However, the definition as in [6] could also be expressed easily in our present framework when we extend it with obligations over formulas in the following way:

$$O(\alpha) =_{def} A(DOING(\alpha)UO(DONE(\alpha)))$$

Obligations on compound actions

We will in this section examine briefly how the general definition given in the previous subsection works for

compound actions. That is, for actions of the form $\alpha_1 + \alpha_2$, $\alpha_1 \& \alpha_2$, $\overline{\alpha}$ and $\alpha_1 ; \alpha_2$.

According to the definition above $O(\alpha_1 + \alpha_2)$ is equal to:

$$A(DOING(\alpha_1 + \alpha_2)U(DONE(\alpha_1 + \alpha_2) \vee V))$$

which is equivalent to:

$$A(DOING(\alpha_1) \vee DOING(\alpha_2)U \\ (DONE(\alpha_1) \vee DONE(\alpha_2) \vee V))$$

The only problem with this definition is that it is possible to swap between doing α_1 and doing α_2 until finishing one of them. Intuitively one would think that the choice about which action is performed is made at the start, after which that action should be performed completely. This anomaly cannot be repaired whithin the present framework. It needs an extra operator to enforce such a choice explicitly. We leave this for further research.

Fortunately we do have that

$$O(\alpha) \vee O(\beta) \Rightarrow O(\alpha + \beta)$$

and

$$O(\alpha) \vee O(\beta) \not\Leftarrow O(\alpha + \beta)$$

, as desired. This property is common to almost all deontic systems and is kept in our logic through the use of a branching time.

If we fill in the action $\alpha_1 \& \alpha_2$ in the general definition we get:

$$A(DOING(\alpha_1 \& \alpha_2)U(DONE(\alpha_1 \& \alpha_2) \vee V))$$

which is equivalent to:

$$A(DOING(\alpha_1) \wedge DOING(\alpha_2)U \\ (DONE(\alpha_1) \wedge DONE(\alpha_2) \vee V))$$

However, this is not at all intuitively correct. It implies that the actions α_1 and α_2 are having equal durations and can finish at exactly the same moment. This is not usually the case in a framework with continuous time. The same problem appears in the dynamic logic framework when action sequences with different length are combined. However, this problem is solved in that framework through the definition of the "[ ]" operator. This operator is also defined for conjuntions of sequences of actions of different lengths. Therefore we still have a good definition of $[\alpha]V$ for all α there. We could try the same in the current framework by defining an operator $After(\alpha)\phi$ which indicates that after the action α has been performed the formula ϕ holds. This operator can then take care of actions with different lengths. However, the definition of this operator is not trivial and we will not pursue it in the present paper.

In this place we will just give a new definition for the obligation on parallel actions:

Definition 3.5

$$O(\alpha_1 \& \alpha_2) =_{def} A(\quad DOING(\alpha_1 \& \alpha_2)U \\ (V \vee DONE(sh(\alpha_1, \alpha_2))) \\ \wedge DOING(lo(\alpha_1, \alpha_2))U \\ (DONE(lo(\alpha_1, \alpha_2)) \vee V))$$

where $sh(\alpha_1, \alpha_2)$ gives the shortest of the two actions and lo gives the longest of the two actions.

The above definition states that first α_1 and α_2 have to be performed in parallel until the shortest of the two is finished. After that the longest of the two should be continued until it is also finished.

Although not easy to prove we still have that:

$$O(\alpha_1 \& \alpha_2) \Leftrightarrow O(\alpha_1) \wedge O(\alpha_2)$$

Next we look at the general definition of obligation applied to the negation of an action. The definition of $O(\overline{\alpha})$ becomes:

$$A(DOING(\overline{\alpha})U(DONE(\overline{\alpha}) \vee V))$$

which is equivalent to:

$$A(\neg DOING(\alpha)U(\neg DONE(\alpha) \vee V))$$

Although the definition contains the performance of $\overline{\alpha}$ it seems not correct that this should only hold until $\neg DONE(\alpha)$. This formula will hold almost always, except at the moment that α has just been performed. Therefore this definition cannot be correct. A better definition for $O(\overline{\alpha})$ and the one we will use is:

Definition 3.6

$$A(\neg(DOING(\alpha)U(DONE(\alpha)) \vee V))$$

Finally we check the general definition for the sequence of actions. Using the definition we get that $O(\alpha ; \beta)$ is equivalent to:

$$DOING(\alpha) \vee (DOING(\beta) \wedge Before(\beta)DONE(\alpha)) \\ U(V \vee (DONE(\beta) \wedge Before(\beta)DONE(\alpha))$$

Because we did not give a formal definition for the operator $Before(\alpha)$ we can not formally show that it is equal to the following intuitive definition of the obligation on a sequence of actions:

$O(\alpha ; \beta) =_d efDOING(\alpha)U$
$(V \vee (DONE(\alpha) \wedge DOING(\beta)U(V \vee DONE(\beta))))$

It should be clear, however, that the two definitions are intuitively equivalent.

3.1 Persistence of violations

In the introduction we have argued that it is important to be able to reason about situations in which deadlines (i.e. obligations) have been violated. However, we have not said anything about the persistence of the violation predicates. In the framework as it is now, the violation predicate V is true whenever an obligation has been violated. However, this is only the case for the moment directly after the violation occurred. This seems a bit strange, because it suggests that violations will disappear by themselves again, while in real life usually some "repair" action is needed to dissolve the violation.

The violations can be made persistent in two ways. First it is possible to regenerate the violation for every point in time until the repair action is performed. This can be done by adding the following formula :

$$V \rightarrow O(repair)$$

as an axiom to the logic.
Now, when a violation occurs it implies an obligation to repair the violation. If the repair action is not performed this leads to a violation again, which in its turn leads to a new obligation, etc.

One will notice right away that introducing persistence for the violation predicate also raises the question about the relation between the violation and the obligation that causes the violation. If several obligations are violated before any of them is repaired, we should have several violation predicates related to the different obligations. This can be easily remedied by making the violation predicate into a predicate over the action that causes the violation. I.e. we change the definition of the obligation to:

$$O(\alpha) =_{def} A(DOING(\alpha)U(DONE(\alpha) \vee V(\alpha)))$$

Now the violations are unique determined and related to the event that causes them.
The axiom above should then also be adjusted to read:

$$\forall \alpha \in A \; V(\alpha) \rightarrow O(repair(V(\alpha)))$$

and we can close this by stating that:

$$V(repair(\alpha)) \leftrightarrow V(\alpha)$$

Although the above construction is technically sound it looks a bit counterintuitive. Intuitively the violations are not generated constantly but just persist until the repair action is performed. We can catch this notion quite naturally and adequately by changing the definition of the obligation in the following way:

Definition 3.7
$O(\alpha) =_{def} A(DOING(\alpha)U(DONE(\alpha) \vee (V_p(\alpha)))).$

Where $V_p(\alpha) =_{def} (V(\alpha)U \; DONE(repair(\alpha)))$.
This is the definition as we will use it in the next section where we define deadlines in terms of the obligations that are introduced in this section.

4 Obligations and deadlines

The first thing that can be expressed using the temporal operators is an obligation to perform an action at some time in the future instead of right away:

$$A(\Diamond(DONE(\alpha)) \vee V_p(\alpha))$$

This means that along all possible futures either I will have performed α at some point or I have a violation (now).
Usually this definition of an obligation is to weak. It only states that the action should be performed sometimes, but this can be after an infinite time. It resembles the "liveness" property as described in [10]. We can strengthen this definition in several ways. The first is to demand that the action is performed before a certain condition becomes true. The definition for this type of obligation can be given as follows:

Definition 4.1
$O(\alpha < \psi) =_d efA(\neg\psi U(DONE(\alpha) \vee V_p(\alpha)))$

It states that ψ can not hold true before α is performed.

The next type of obligation that we will describe is the periodic obligation. This obligation returns every time a certain condition holds true and should be fulfilled before another condition holds true. E.g. an order should be placed after the stock of computers has fallen below 15 and before the level has dropped below 5. The condition that the stock falls below a certain level will be true periodically (one hopes) and every time this happens an order for replenishment should be made. The periodic obligation is described as follows:

Definition 4.2
$PO(\phi < \alpha < \psi) =_d efA(\Box[\phi \longrightarrow O(\alpha < \psi)])$

The box operator forces the obligation to be periodic. Every time (from now on) if ϕ becomes true the obligation arises to perform α before ψ.

An alternative definition would be:
$PO(\phi < \alpha < \psi) =_d ef$
$A(\Box[(\phi \rightarrow \neg\psi)U(DONE(\alpha) \vee V_p(\alpha))])$

This is closely related to the problems of conditional obligations. There are two ways to express conditional obligations:

$$\phi \rightarrow O(\psi)$$
$$O(\phi \rightarrow \psi)$$

Both have their merits and problems. We chose for the first formalization, because it seems most natural in our applications and causes less problems. The only counterintuitive aspect of this definition is that the starting condition of the obligation lays outside the scope of the O operator while the end condition of the obligation lays within the scope of the O operator. See [12] for a more thorough discussion on this topic.

In most cases the deadline is enforced only once (or explicitly reenforced every time it is needed). To ensure that the deadline only becomes active the first time ϕ becomes true we extend the definition with an until clause that states that the obligation finishes after the first time ϕ becomes true:

Definition 4.3
$O(\phi < \alpha < \psi) =_d ef A([\phi \longrightarrow O(\alpha < \psi)]U\phi)$

A second way to oblige an action to happen somewhere in the future without having to wait indefinite is to demand that the action is performed as soon as nothing "more important" or preferred is performed.

If quantification over actions is added to the language, this can be defined as follows:

Definition 4.4
$O?(\alpha) =_d ef$
$\forall\beta O(true < \alpha < (DOING(\beta) \wedge PREFER(\alpha, \beta)))$

This obligation is interpreted as meaning that the action should be performed as soon as no other actions with a higher "preference" are performed. This obligation can be used when no strict deadline is given, but we want the action to be performed at some time. I.e. it has to be performed before an action with lesser importance is performed.

Up till now we only described deadlines with an implicit time. Real time, i.e., a quantitative time treatment allowing us to specify numeric deadlines can be added to the above approach by keeping an explicit clock variable that stores the real time (cf. [11, 17]). We distinguish between relative and absolute time conditions. For the absolute time conditions we introduce a special variable *time*. Using this variable we define the general obligation with pure absolute temporal deadline as follows:

Definition 4.5
$O(time = t_1 < \alpha < time = t_2) =_d ef$
$A(\Box[time = t_1 \rightarrow (time < t_2 U(DONE(\alpha) \vee V_p(\alpha))])$

This definition differs only from the general definition in the fact that $\neg(time = temp_2)$ is interpreted as $time < temp_2$.
For deadlines that are given relative to the present time the definition is as follows:

Definition 4.6
$O(now + t_1 < \alpha < (now + t_1) + t_2) =_d ef$
$A(time = k \rightarrow$
$\Box[time = k + t_1 \rightarrow time < k + t_1 + t_2 U$
$(DONE(\alpha) \vee V_p(\alpha))])$

This concludes the definition of all the types of deadlines that were also defined in [6]. In that paper we have already shown that these definitions are sufficient to model most common deadlines.

5 Conclusions

Deadlines play an important role in flexible transactions. In situations where several systems have to cooperate deadlines are a means to specify expectations of the behaviour of the other parties. In some situations the transactions are time critical (e.g. medical applications). For these situations a model with continuous time is needed to describe the deadlines. In this paper we have investigated what are the consequences of using dense (continuous) time for the specification of deadlines.

It appears that we can no longer use the model of dynamic logic as a basis for the combined model in which temporal, action and deontic concepts have to be modeled. In the case of dense time we have to take the temporal logic as a basis. Therefore we have to find a representation of the actions in this logic. The fact that we also want to represent obligations over actions restricts the way the actions can be represented.

The logic that was presented in this paper complies to all constraints. However, the definition of obligation on actions is not compositional. I.e. there is no general definition that can be used for compound actions and that can be decomposed into obligations on the basic actions. That compositionality of operators on actions is a difficult problem in a dense time framework can also be seen in [2]. This problem is caused by the differing durations of the actions within the compound action. A solution might be to introduce an extra operator $After(\alpha)\phi$ which would be the equivalent to the

dynamic logic operator $[\alpha]\phi$. This operator can be used (as is done in dynamic logic) to isolate the problem of combinations of actions with different length from the obligation operator.

The use of an explicit violation predicate related to the violated action makes it possible to reason about states in which deadlines are not kept. It is also easy to make these violations persistent. They will only disappear after a special "repair" action has been performed, which seems to comply with the intuitive meaning of violation and punishment.

An important area for further research is of course to define an axiomatization of the logic and give some inference rules. Only when these things are achieved it will be possible to use this logic in practice.

References

[1] J.F. Allen. An Interval-based Representation of Temporal Knowledge. *Proc. 7th Int. Joint Conf. on Artificial Intelligence*, Vancouver, Canada, 1981, pp. 221–226.

[2] H.Barringer, R.Kuiper, A. Pnueli, Now you may compose temporal logic specifications, *Proc. 16th ACM Symposium on Theory of Computing*, pp51-63, 1984.

[3] Barringer, H., Kuiper, R., Pnueli, A., A really abstract concurrent model and its temporal logic, *Proc. of the 13th ACM Symposium on Pinciples of Programming Languages* pp. 173-183,1986

[4] N. Belnap and M. Perloff. Seeing to it that: a canonical form to agentives. *Theoria* 54, 1988, pp. 175–199.

[5] R.M. Chisholm. Contrary-to-Duty Imperatives and Deontic Logic. *Analysis* 24, 1963, pp. 33–36.

[6] F. Dignum and R. Kuiper. Combining dynamic deontic logic and temporal logic for the specification of deadlines. In Jr. R. Sprague, editor, *Proceedings of thirtieth HICSS*, Wailea, Hawaii, 1997.

[7] F. Dignum, R. Kuiper and J.-J.Ch. Meyer Obligations in a temporal framework *submitted to DEON98*

[8] J.A. van Eck. A System of Temporally Relative Modal and Deontic Predicate Logic and Its Philosophical Applications (1), *Logique et Analyse 99*, 1982, pp. 249–290.

[9] J.A. van Eck. A System of Temporally Relative Modal and Deontic Predicate Logic and Its Philosophical Applications (2), *Logique et Analyse 100*, 1982, pp. 339–381.

[10] J. Fiadeiro and T. Maibaum. Temporal Reasoning over Deontic Specification. In *Journal of Logic and Computation*, 1 (3), 1991.

[11] E. Harel, O. Lichtenstein and A. Pnueli. Explicit clock temporal logic. In *Proceedings Symposium on Logic in Computer Science*, pages 402-413, 1990.

[12] J. Hintikka. Some Main Problems of Deontic Logic. In R. Hilpinen (ed.), *Deontic Logic: Introductory and Systematic Readings*, pages 59-103, Reidel, 1971.

[13] J.F. Horty. Combining Agency and Obligation. In M. Brown and J. Carmo (eds.), *Deontic Logic, Agency and Normative Systems*, pages 98-122, Springer-Verlag, Berlin, 1996.

[14] J.-J.Ch. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. In *Notre Dame Journal of Formal Logic*, vol.29, pages 109–136, 1988.

[15] J.-J. Ch. Meyer and R.J. Wieringa. Deontic Logic: A Concise Overview. In J.-J. Ch. Meyer and R.J. Wieringa (eds.), *Deontic Logic in Computer Science: Normative System Specification*, John Wiley & Sons Ltd., Chichester, 1993, pp. 3–16.

[16] J.-J. Ch. Meyer, R.J. Wieringa and F.P.M. Dignum. The Role of Deontic Logic in the Specification of Information Systems, Techn. Report UU-CS-1996-55, Utrecht University, 1996.

[17] J. Ostroff. *Temporal Logic for Real-Time Systems*. Advanced Software Development Series. Research Studies Press, 1989.

[18] R. Thomason. Deontic Logic as founded on tense logic. In R. Hilpinen, editor, *New Studies in Deontic Logic*, pages 165- 176, D.Reidel Publishing Company, 1981.

[19] P. Zhou, J. Hooman and R. Kuiper. Compositional Verification of Real-Time Systems with Explicit Clock Temporal Logic. In *Formal Aspects of Computing 8:294-323*, 1996.