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.

Combining Dynamic Deontic Logic and Temporal Logic for the Specification of 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

Intelligent agents have an agenda that is monitored continuously to decide what action is to be performed. Formally, an agenda is a set of deontic temporal constraints. Deontic, since the agenda specifies what the agent should do. Temporal, since the obligation is usually to be performed before a certain deadline, or as soon as possible. In this paper, we investigate the concepts necessary to describe deadlines. We describe a temporal deontic logic that facilitates reasoning about obligations and deadlines. The logic is a combination of temporal logic and deontic dynamic logic. We describe extensively which choices have to be made in combining temporal and dynamic aspects into one system. In the new logic, we can uniformally specify that an obligation starts at a certain time or event, that it must be done immediately, as soon as possible, before a deadline, or periodically.

1 Introduction

It is not very difficult to develop a program that checks whether deadlines are met. The main idea is to wait until the deadline has passed, which usually can be checked easily, and then check whether a certain action has taken place. However, many difficulties arise when one tries to transform this procedural account of deadlines into a formal one. In an intelligent system (or agent) one would like to be able to reason about this type of constraints in order to check whether they can be fulfilled at all. This holds especially for combinations of different deadlines. Constraints can also be used to influence the behaviour of the agent. The combination of deadlines can be used to plan the actions of an agent. Of course this is only possible if the system has some formal description (besides a procedural one) of the deadlines.

The ideas expressed above are illustrated with the following example.

1. Ernie has to pay the mortgage for his house every month.

2. Ernie borrowed some money from Bert, which he has to repay as soon as he is able to do so.

3. The roof of the house of Ernie started leaking. It has to be repaired before the October rains start. (It is now September).

4. Ernie wants to go on a midweek holiday. He has an offer to rent a cottage for relatively little money, which has to be paid within 30 days after the reservation has been made.

Several things play a role in this example. First, Ernie still has to decide whether he should go on a holiday. The first three constraints are all fixed. The obligation to pay for the holiday only arises if Ernie decides to book it. Secondly, Ernie has to decide whether the obligations are conflicting. If Ernie has enough money

he can fulfill the three obligations (1,2,3) and book the holiday without any problem. If Ernie has not enough money he has to decide which are the most important obligations to keep. He might still decide to repair the roof and go on a holiday in the meanwhile risking a penalty for not paying the mortgage in time and a fight with Bert for not repaying his debt.

The latter remarks already point to an important difference with approaches such as those from Haddawy [12] or Boutilier [4]. In these approaches the deadlines are modelled as goals that can be (partially) fulfilled. However, it is then very difficult to reason about the situations in which the deadlines are not met and new obligations arise. E.g., the fact that a penalty has to be paid when the mortgage is not paid in time. This new obligation can influence the planning of the agent because it influences the utility of the states in which the mortgage is not paid in time. In the decision theoretic approaches these new obligations have to be incorporated somehow in the utility functions of the agents. In our approach we are also able to reason explicitly about the situations after a deadline has not been met and can also represent explicitly which new obligations arise from violating the deadline.

Whether a deadline is met depends on an action that must have taken place. In case this action only depends on the agent itself one might force the agent to perform the action before the deadline is reached. In this case the deadline would be used as part of the planning system of the agent. The deadlines would not have to be checked afterwards because they would be met by default (if possible of course). In case the deadlines cannot be met the best possible plan could be chosen.

Two more problems can arise when deadlines are seen only as goals that should be reached. The first problem arises with deadlines of which it is not known beforehand when they will be reached. E.g. (the third deadline above) Ernie should repair the roof before it starts raining. It is not known at what point in time it will start raining, therefore it is very difficult to incorporate this deadline in a utility function for the agent. However, it is possible to check the deadline and specify new obligations that arise when the deadline is violated. If the violation of the deadline has severe consequences the agent might decide to "repair the roof" before anything else.

We argue that the enforcement of the deadline and the planning problem are two separate issues and the enforcement of deadlines should not be implemented by a planning procedure. Of course we do acknowledge that the deadlines influence the planning of the actions of the intelligent system.

A second problem that arises with the enforcement of deadlines is that the agent is not always capable of enforcing the performance of a certain action. E.g. upon delivery of the product the customer has to pay the bill within 30 days. The agent can base its plans on the fact that the customer has paid within 30 days, but it has no way to enforce this payment (directly). This shows that deadlines cannot always be enforced. Therefore an agent should not reach a state of inconsistency whenever a deadline is not met. Rather it should arrive at a state in which it is clear that a deadline has passed, but other (corrective) actions are still possible. (In case of the customer the agent could send a reminder or a court order for payment).

A last problem that we like to mention is the case where no specific deadline has been set. A certain action should take place "as soon as possible". E.g. after an accident has been reported the ambulance has to go to the place of the accident as soon as possible. However, it might be that the ambulance first has to deliver another patient at the hospital or that the accident is not very serious and the ambulance does not switch on its siren. In these cases there is not a definite point in time where one can check whether the action has been performed or not.

The long term aim of our research is to develop a system in which the deadlines can be formally (and declaratively) described and which is capable of reasoning about these deadlines. For this purpose we want to develop a logic which is subsequently implemented. In this paper we aim at describing the syntax and semantics of the concepts that have to be used in such a logic. Specifically we will describe the choices that have to be made when combining the temporal and dynamic aspects that are necessary to specify deadlines. The outcome is a multi-modal logic with temporal, dynamic and deontic operators. These are also exactly the main components of deadlines; *actions* that are *obliged* to occur before a certain *time*. The concepts used in dynamic logic and deontic logic are orthogonal and can therefore easily be combined. The addition of temporal logic requires careful consideration of the intervention of the concept of time and the existing relations in the dynamic deontic logic; performing an action implies passage of time.

Although other approaches exist that combine temporal and deontic logics (e.g. [24, 16, 10]) these approaches tend to express the deontic concepts in terms of the temporal operators. In this paper we take a different approach. We actually "add" the temporal

operators to the deontic logic that is used as a basis. Also many systems have been described to specify deadlines using temporal logic (see e.g. [22, 23]). These systems do contain some form of action description, but lack in a separate semantics for actions as given in dynamic logic. Therefore it is not possible to express relations between actions or the non-performance of a certain action. The latter is of crucial importance to model the violation of an obligation to perform an action (as will be shown later in section 2). Although this point will not be very prominent in describing the deadlines it forms the main motivation for making a "real" combination of temporal and dynamic logic instead of the reduction of one to the other.

We do not give an axiomatization for this logic or even a complete 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 introduce a combination of dynamic deontic logic and temporal logic without explicit time. In section 3 we will describe the examples given above in this new logic together with some other more elaborate examples. In section 4 we will discuss the choices that are involved when dynamic deontic logic is combined with a logic with explicit time. Section 5 contains some conclusions and directions for further research. We also briefly indicate how deontic temporal constraints are used in Agent Specification.

2 Combining dynamic deontic logic and temporal logic

In this section we will describe a combination of dynamic deontic logic and temporal logic without explicit time. That is, the temporal logic only contains operators to order states in time. It will be shown that a combination of such a temporal logic with the dynamic deontic logic can be made easily and in a natural way. The obtained logic is powerful enough to describe the deadlines from the examples in the introduction. In section 4 we describe the issues that arise when an application necessitates reasoning with explicit (possibly continuous) time.

2.1 Dynamic deontic logic

We now proceed with the definition of a set of *formulas* with which we can describe the behaviour of (interpreted) (trans)actions. This language is a variant of *dynamic logic* ([14]), and was first used for this purpose in [19]. In the present paper we add two "new" operators to this language. The formulas defined in (5) define the "classical" deontic formulas as introduced by v.Wright in [26, 1]. The formulas defined in (6) introduce a preference relation between actions. They state which action is preferred to be performed at a certain time.

In this paper we will not include transactions in the dynamic logic, because they are not essential for the points we want to make in this paper and the definition of the semantics of transactions contains many subtle difficulties would mainly be confusing in the present paper. For a full formal account of dynamic deontic logic with transactions we refer to [8].

We assume a fixed set *Prop* of atomic propositions and a set *Act* action expressions (see below). The set *Form* of formulas is then the smallest set closed under:

(1). $Prop \subseteq Form$

(2). $\phi_1, \phi_2 \in Form \Longrightarrow \phi_1 \wedge \phi_2 \in Form$

(3). $\phi \in Form \Longrightarrow \neg\phi \in Form$

(4). $\alpha \in Act, \phi \in Form \Longrightarrow [\alpha]\phi \in Form$

(5). $\phi \in Form \Longrightarrow O(\phi) \in Form$

(6). $\alpha_1, \alpha_2 \in Act \Longrightarrow PREFER(\alpha_1, \alpha_2) \in Form$

Note: Other propositional connectives such as $\vee$ and $\rightarrow$ are assumed to be introduced as the usual abbreviations. Also the special proposition $false$ is introduced as the abbreviation of $p \wedge \neg p$ for some $p \in Prop$. The informal meaning of $[\alpha]\phi$ is "doing α necessarily leads to a state where ϕ holds". The informal meaning of $O(\phi)$ is that ϕ should be the case in the present state. We introduce the other deontic operators using the usual abbreviations:

- $F(\phi)$ abbreviates $O(\neg\phi)$
- $P(\phi)$ abbreviates $\neg F(\phi)$

The last type of formulas defined above (in (6)) indicate that a certain action α_1 is preferred to performing α_2. This extension is used to model the obligation to perform some action "as soon as possible". It does not interfere with our main aim of combining this logic with a temporal logic.

The semantics of the formulas in *Form* is given in two stages. First we will give the syntax and semantics of the action expressions, which we will subsequently use in section 2.1.2 to define the semantics of the formulas.

2.1.1 Action expressions and their semantics

We start out by giving a definition of *action expressions*, which we shall typically denote α, possibly with subscripts. To this end we assume a set At of *atomic action expressions* that are typically denoted by $\underline{a}, \underline{b}, \ldots$. Furthermore, we assume special action expressions **any** and **fail** denoting "don't care what happens" and "failure", respectively. The action expressions are formed using the + for choice, the & for parallel execution and the $^-$ for negation.

Definition 2.1 The set Act of action expressions is given by the following BNF:

$$\alpha ::- \ \underline{a}|\mathbf{any}|\mathbf{fail}|\alpha_1 + \alpha_2|\alpha_1 \& \alpha_2|\overline{\alpha}$$

The semantics of action expressions has two components. The first component is an algebra of uninterpreted actions (called a *uniform* semantics elsewhere [3]), which allows us to interpret equalities between action expressions without taking their effect into account. Due to lack of space we refer to [7] for a formal account of this part. The following intuitions should suffice here. In the algebraic semantics, each action expression will be interpreted as a choice over possible steps, where a step is a set of events that can be performed simultaneously.
Notation: We write $[\![\alpha]\!]$ to denote the set of steps that form the uniform semantics of α.

For example, $\underline{a}\&\underline{b}$ is interpreted as the set of steps where each step contains at least$\{a, b\}$ whereas $\underline{a} + \underline{b}$ is interpreted as the choice from possible steps containing at least $\{a\}$ or containing at least $\{b\}$. The extension of the sets implied in the clause "at least" in the above concerns a technical issue related to open versus closed systems which is elaborated on in [7] and not further explained here.

It is also important that the uniform semantics provide a formal semantics for the negation of actions.

The second component of the semantics is a state-transition semantics of action expressions where we define the effect of steps on the state of the world. Most notable here is that the effect of any action can always be a set of new states. (One for each step in its semantics)

2.1.2 Semantics of formulas

Having defined the semantics of the action expressions within the formulas, we can now give the semantics of formulas in *Form* by means of the notion of a Kripke structure

$\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, \leq, R_O)$.
Σ is a set of states (worlds).
$\mathcal{A}$ is a finite set of events.
π is a truth assignment function to the atomic propositions relative to a state: π is a function $\Sigma \rightarrow (Prop \rightarrow \{tt, ff\})$, where tt and ff denote truth and falsehood, respectively. Thus, for $p \in Prop$, $\pi(\sigma)(p) = tt$ means that the atomic proposition p is true in state σ.

The accessibility relation $R_{\mathcal{A}}$ specifies how actions can change states. The relation $R_{\mathcal{A}}$ is defined as follows: $R_{\mathcal{A}} = \{R_t | t \subseteq A\}$, reflecting that R_t is the relevant entity. We write, slightly abusing notation,

$$R_{\mathcal{A}}(\sigma, \sigma') \Longleftrightarrow \exists t : R_t(\sigma, \sigma')$$

$\leq$ is a function $\Sigma \times \Sigma \rightarrow \{tt, ff\}$. The function indicates for two states which of the two is preferred .
In this paper we only use the preference relation to indicate a preference relation between actions. We do not give a logic for the preference relation itself. However, one might intuitively think that an action α is preferred over an action β if it leads to states in which less constraints are violated or the violations are considered less harmful (i.e. which are more ideal in a deontic sense). For a thorough treatment of this type of logic we refer to [4]. Here we take the preference relation to be primitive.

The relation R_O relates states σ to states σ' that are deontically ideal with respect to σ. Intuitively in the states σ' all the obligations are fulfilled and no prohibitions have been violated.
We assume the relation R_O to be serial. I.e. for every world σ there exists at least one world σ' such that $R_O(\sigma, \sigma')$ holds.

We now give the interpretation of formulas in *Form* in Kripke structures. We interpret formulas with respect to a structure $\mathcal{M}$ and a state $\sigma \in \Sigma$

Definition 2.2 Given $\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, R_O, \leq)$ as above and $\sigma \in \Sigma$, we define:

1. $(\mathcal{M}, \sigma) \models p \Longleftrightarrow \pi(\sigma)(p) = tt$ (for $p \in Prop$)
2. $(\mathcal{M}, \sigma) \models \phi_1 \wedge \phi_2 \Longleftrightarrow$ $(\mathcal{M}, \sigma) \models \phi_1$ and $(\mathcal{M}, \sigma) \models \phi_2$
3. $(\mathcal{M}, \sigma) \models \neg\phi \Longleftrightarrow$ not $(\mathcal{M}, \sigma) \models \phi$
4. $(\mathcal{M}, \sigma) \models [\alpha]\phi \Longleftrightarrow$ $\forall t \in [\![\alpha]\!] \forall \sigma' \in \Sigma [R_t(\sigma, \sigma') \Rightarrow (\mathcal{M}, \sigma') \models \phi]$
5. $(\mathcal{M}, \sigma) \models O(\phi) \Longleftrightarrow$ $\forall \sigma' \in \Sigma [R_O(\sigma, \sigma') \Rightarrow (\mathcal{M}, \sigma') \models \phi]$

6. $(\mathcal{M}, \sigma) \models PREFER(\alpha_1, \alpha_2) \Longleftrightarrow$
$\forall t \in [\![\alpha_2]\!](R_t(\sigma, \sigma_2) \longrightarrow$
$(\exists t' \in [\![\alpha_1]\!](R_{t'}(\sigma, \sigma_1) \wedge (\sigma_1 \leq \sigma_2)))$

7. ϕ is *valid* w.r.t. model $\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, R_O, \leq)$, notation $\mathcal{M} \models \phi$,
if $(\mathcal{M}, \sigma) \models \phi$ for all $\sigma \in \Sigma$.

8. ϕ is *valid*, notation $\models \phi$, if ϕ is valid w.r.t. all models $\mathcal{M}$ of the form considered above.

The definitions are quite standard and we will not explain them any further here. It should be noted that using the semantic definition of [**any**]ϕ we can express the usual temporal operators over static formulas as given in e.g. [9]. In the next section, however, we show that we can do this more elegantly and have more expressive power by adding real temporal operators and their semantics.

2.2 Adding Temporal Logic

We extend the dynamic deontic logic framework introduced so far by the most simple temporal logic framework: propositional linear time temporal logic (PLTLB). (The B stands for a temporal logic having Both future and past tense). Intuitively, an $R_{\mathcal{A}}$ step, corresponding to an action expression in the dynamic logic framework, is now also interpreted as making a step forward in time. As yet, no quantative notion of time is associated with such a step, neither in the sense that a step has an explicit duration nor in the sense that steps could be counted explicitly; so, only ordering in time is added. If from a state another one is reached in one or more steps, this just means that it occurs at some later moment. For the moment, we only consider sequential time, i.e., we only use the traces through the Kripke model and do not consider its branching structure. Note that, as to be expected, this involves only the accessibility relation $R_{\mathcal{A}}$ of the Kripke model and does not address the relation R_O. At this stage of our investigations, we leave the possibility open that if a state is connected by R_O to an ideal state, time in the ideal state need not necessarily be the same or later than in the original one.

For an introduction to PLTLB in the context of computer science, but starting from a modal logicians point of view, see [11]; for a thorough treatment of its role in computer science see [17, 18].

2.2.1 Syntax of dynamic deontic linear time temporal logic (DDLTLB)

The monadic next operator, $\mathcal{X}$, and the binary until operator $\mathcal{U}$ are used as temporal operators for the future. The meaning of $\mathcal{X}$ is the obvious one, the until states that some property is preserved until the first time that some other property is fulfilled. We will also define an unless operator in terms of the until operator which basically means the same but does not require that this fulfilment need ever occur. For the past time we introduce the mirror operators $\mathcal{P}$ and $\mathcal{S}$, indicating that a certain formula held in the previous state and that some property held since the first time some other property was fulfilled.
We also introduce the operators DO and DONE over actions, indicating that a certain action will be performed next or was performed last.
We do not define PLTLB separately, but immediately define DDLTLB as an extension of the dynamic deontic logic defined above.

Thus, the definition in section 2.1 is extended with the following six production rules.

TL1a. $\phi \in Form \Rightarrow \mathcal{X}\phi \in Form$

TL1b. $\phi \in Form \Rightarrow \mathcal{P}\phi \in Form$

TL2a. $\phi_1, \phi_2 \in Form \Rightarrow \phi_1 \mathcal{U} \phi_2 \in Form.$

TL2b. $\phi_1, \phi_2 \in Form \Rightarrow \phi_1 \mathcal{S} \phi_2 \in Form.$

TL3a. $\alpha \in Act \Rightarrow DO(\alpha) \in Form.$

TL3b. $\alpha \in Act \Rightarrow DONE(\alpha) \in Form.$

The derived operators $\Diamond$ (eventually), $\Diamond^-$ (some time in the past), $\Box$ (always from now), $\Box^-$ (always up till now), $\bigsqcup$ (unless) and Σ (since) are defined as follows:

- $\Diamond\phi := true\mathcal{U}\phi$; $\Diamond^-\phi := true\mathcal{S}\phi$;
- $\Box\phi := \neg\Diamond\neg\phi$; $\Box^-\phi := \neg\Diamond^-\neg\phi$.
- $\phi \bigsqcup \psi := \phi\mathcal{U}\psi \wedge \Box\phi$
- $\phi\Sigma\psi := \phi\mathcal{S}\psi \wedge \Box^-\phi$

The last "since" operator is seldom used but is given here for reasons of symmetry.

2.2.2 Semantics of DDLTLB

The first question is, how to interpret the temporal operators on the Kripke models that contain transitions from R_A and R_O. Traditionally, linear time temporal

logic formulas are interpreted over state sequences. For systems with a Kripke structure semantics a formula is then defined to hold for the system if it holds for all the paths through its Kripke structure. Because of the presence of transitions from R_O, and possible mixtures of deontic and temporal operators, we have to apply this sequentialisation of the Kripke structure into a set of paths after each occurrence of either a dynamic logic operator or the deontic O operator. This situation is similar, but not identical, to the situation in branching time temporal logic (cf. [9]).
We will interpret every formula in a triple $(\mathcal{M},\rho,\sigma)$. In this triple the state σ can be seen as an index into the path ρ, making the interpretation similar to the one for PLTLB in [9].

As said before, the relation R_O does not depend on the temporal paths. However, it seems natural that in an ideal state with respect to the present state, also the ideal temporal path is different from the present temporal path. Therefore, in the Kripke structure the definition of the R_O relation is reinterpreted to be a relation between *pairs* of state/path (σ,ρ) to relatively deontic ideal state/path pairs (σ',ρ').

Before giving the semantic interpretation of the formulas in DDLTLB we introduce some notation on paths to facilitate the definitions.
Let $\rho = \langle \ldots \sigma_{-1}, t_{-1}, \sigma_0, t_0, \sigma_1, t_1 \ldots \rangle$ an indexed state/transition path through $\mathcal{M}$, then the set of all such paths ρ is called $Path(\mathcal{M})$. The state σ_i on the path ρ is denoted by $\rho(i)$. We use $\rho^-(\sigma_i)$ for the prefix of ρ that ends at σ_i and $\rho^+(\sigma_i)$ to denote the suffix of ρ that starts with σ_i.

Definition 2.3
$Path(\mathcal{M},\sigma)=\{\rho \mid \rho \in Path(\mathcal{M}) \text{ and } \exists i\colon \sigma = \rho(i)\}$.
$Path^+(\mathcal{M},\sigma)=\{\rho \mid \rho \text{ starts with } \sigma\}$.
$Path^-(\mathcal{M},\sigma)=\{\rho \mid \rho \text{ ends with } \sigma\}$.

The following definition defines the concatenation of (sets of) paths.

Definition 2.4 Let $\rho = \langle \ldots \sigma_{-1}, t_{-1}, \sigma_0 \rangle$ and $\rho' = \langle \sigma_0, t_0, \sigma_1, t_1 \ldots \rangle$ then
$\rho \circ \rho' \equiv \langle \ldots \sigma_{-1}, t_{-1}, \sigma_0, t_0, \sigma_1, t_1 \ldots \rangle$
Let $\rho = \langle \ldots \sigma_{-1}, t_{-1}, \sigma_0 \rangle$ and $\Omega \subseteq Path^+(\mathcal{M},\sigma_0)$ then
$\rho \circ \Omega=\{\rho' \mid \rho'^-(\sigma_0) = \rho and \rho'^+(\sigma_0) \in \Omega\}$
Let $\rho = \langle \sigma_0, t_0, \sigma_1, t_1 \ldots \rangle$ and $\Omega \subseteq Path^-(\mathcal{M},\sigma_0)$ then
$\Omega \circ \rho=\{\rho' \mid \rho'^+(\sigma_0) = \rho and \rho'^-(\sigma_0) \in \Omega\}$
Let $\Omega \subseteq Path^-(\mathcal{M},\sigma_0)$ and $\Omega' \subseteq Path^+(\mathcal{M},\sigma_0)$ then
$\Omega \circ \Omega'=\{\rho' \mid \rho'^-(\sigma_0) \in \Omega and \rho'^+(\sigma_0) \in \Omega'\}$

The semantics of formulas in DDLTLB is given as follows:

Definition 2.5 Given $\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, R_O, I)$ as above and $\rho \in Path(\mathcal{M},\sigma)$, we define:

1. $(\mathcal{M},\rho,\sigma) \models p \iff \pi(\sigma)(p) = tt$ (for $p \in Prop$)

2. $(\mathcal{M},\rho,\sigma) \models \phi_1 \wedge \phi_2 \iff$
$(\mathcal{M},\rho,\sigma) \models \phi_1$ and $(\mathcal{M},\rho,\sigma) \models \phi_2$

3. $(\mathcal{M},\rho,\sigma) \models \neg\phi \iff$ not $(\mathcal{M},\rho,\sigma) \models \phi$

4. $(\mathcal{M},\rho,\sigma) \models [\alpha]\phi \iff \forall t \in [\![\alpha]\!]$
$\forall \sigma' \in \Sigma \forall \rho' \in (\rho^-(\sigma) \circ \langle \sigma, t, \sigma' \rangle \circ Path^+(\mathcal{M},\sigma'))$
$[R_t(\sigma,\sigma') \Rightarrow (\mathcal{M},\rho',\sigma') \models \phi]$,
i.e. basically it means that if α is performed then ϕ holds true afterwards. The paths that can be considered in the new state are fixed to the past (equal to ρ followed by t) and open to the future.

5. $(\mathcal{M},\rho,\sigma) \models O(\phi) \iff$
$\forall \sigma' \forall \rho' \in Path(\mathcal{M},\sigma')[R_O((\sigma,\rho),(\sigma',\rho')) \Rightarrow$
$(\mathcal{M},\rho',\sigma') \models \phi]$,
i.e. ϕ is obligated if in all states and paths that are ideal with respect to the present state and path ϕ is true.

6. $(\mathcal{M},\rho,\sigma) \models PREFER(\alpha_1,\alpha_2) \iff$
$\forall t \in [\![\alpha_2]\!](R_t(\sigma,\sigma_2) \longrightarrow$
$(\exists t' \in [\![\alpha_1]\!](R_{t'}(\sigma,\sigma_1) \wedge (\sigma_1 \leq \sigma_2)))$

7. $(\mathcal{M},\rho,\sigma_i) \models \mathcal{X}\phi \iff (\mathcal{M},\rho,\sigma_{i+1}) \models \phi$

8. $(\mathcal{M},\rho,\sigma_i) \models \mathcal{P}\phi \iff (\mathcal{M},\rho,\sigma_{i-1}) \models \phi$

9. $(\mathcal{M},\rho,\sigma_0) \models \phi\mathcal{U}\psi \iff$
$\exists k((\mathcal{M},\rho,\sigma_k) \models \psi)$ and
$\forall j, 0 \leq j < k, (\mathcal{M},\rho,\sigma_j) \models \psi))$,
i.e. ϕ holds along the path ρ until ψ becomes true.

10. $(\mathcal{M},\rho,\sigma_0) \models \phi\mathcal{S}\psi \iff$
$\exists k((\mathcal{M},\rho,\sigma_k) \models \psi)$ and
$\forall j, k < j \leq 0, (\mathcal{M},\rho,\sigma_j) \models \psi))$,
i.e. ϕ holds along the path ρ since ψ became true.

11. $(\mathcal{M},\rho,\sigma_0) \models DO(\alpha) \iff$
$\rho = \langle \ldots, \sigma_0, t_0, \sigma_1, t_1 \ldots \rangle$ and $t_0 \in [\![\alpha]\!]$,
i.e. α is the action performed next.

12. $(\mathcal{M},\rho,\sigma_0) \models DONE(\alpha) \iff$
$\rho = \langle \ldots, \sigma_{-1}, t_{-1}, \sigma_0, \ldots \rangle$ and $t_{-1} \in [\![\alpha]\!]$,
i.e. α is the last action performed.

13. ϕ is *atemporal* w.r.t. model $\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, R_O, \leq)$, notation $(\mathcal{M},\sigma) \models \phi$, if $(\mathcal{M},\rho,\sigma) \models \phi$ for all $\rho \in Path(\mathcal{M},\sigma)$.

14. ϕ is *valid* w.r.t. model $\mathcal{M} = (\mathcal{A}, \Sigma, \pi, R_{\mathcal{A}}, R_O, I)$, notation $\mathcal{M} \models \phi$,
if $(\mathcal{M}, \rho, \sigma) \models \phi$ for all $\sigma \in \Sigma$ and $\rho \in Path(\mathcal{M}, \sigma)$.

15. ϕ is *valid*, notation $\models \phi$, if ϕ is valid w.r.t. all models $\mathcal{M}$ of the form considered above.

2.3 Obligations and deadlines

Using the above definitions we now introduce the deontic operators over actions. Before we give the most general type of obligation with deadlines We will introduce some simpler obligations over actions with temporal aspects.

In [19] for the first time an obligation over actions was defined using dynamic logic. The obligation had an immediate character. I.e. the action had to be performed as the next action. This type of immediate obligation can be redefined now in two ways:

Definition 2.6
$O!(\alpha) \equiv [\mathbf{any}]O(DONE(\alpha))$
or
$O!(\alpha) \equiv O(DO(\alpha))$

Although the first definition looks more complicated it fits with the intuition that a violation of an obligation to perform an action as the next one can only be checked after the next action has been performed.

Using the temporal operators we can now also denote the obligation that states that it is obligated to perform a certain action at some time in the future.

$$O(\Diamond(DO(\alpha)))$$

If we want to state that the action should be performed before a certain condition becomes true this can be done as follows:

Definition 2.7 $O(\alpha < \psi) \equiv O(\neg\psi\mathcal{U}DO(\alpha))$

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 2.8
$PO(\phi < \alpha < \psi) \equiv \Box[\phi \longrightarrow O(\neg\psi\mathcal{U}DO(\alpha))]$

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.

We chose in this definition NOT to let the O operator range over the implication but only over the last part of the formula. This is closely related to the problems of conditional obligations. There are two ways to express conditional obligations:

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

Both have their merits and problems. We chose for the second 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 [15] 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 2.9
$O(\phi < \alpha < \psi) \equiv ([\phi \longrightarrow O(\neg\psi\mathcal{U}DO(\alpha))])\mathcal{UP}\phi$

If quantification over actions is added to the language, the general type of obligation with deadlines also enables to describe an obligation that has to be fulfilled as soon as possible. This obligation is interpreted as meaning that the action should be performed as soon as no other actions with a higher "preference" are performed. The definition is as follows:

Definition 2.10
$O?(\alpha) \equiv$
$\forall\beta O(true < \alpha < DONE(\beta) \wedge PREFER(\alpha, \beta))$

This obligation can be used when no strict deadline is given, but we want the action to be performed at some time. It resembles the "liveness" property as described in [10], except that the obligated action cannot be postponed indefinitely. 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 in several ways: ranging from counting steps and having a step represent a fixed

duration (cf. [21]), to having an explicit clock variable store the real time (cf. [13, 20]). We distinguish between relative and absolute time conditions. For the absolute time conditions we introduce a special variable *time*. The following axiom should hold for the values of this variable:

Axiom 2.11

$$\models time = k \longrightarrow \mathcal{X}(time = k + 1)$$

That is, we assume all actions to take equal time and the length of an action defines the basic unit of time. Using this axiom we define the general obligation with pure absolute temporal deadline as follows:

Definition 2.12
$O(time = temp_1 < \alpha < time = temp_2) \equiv$
$\Box[time = temp_1 \longrightarrow O(time < temp_2 \mathcal{U} DO(\alpha))]$

This definition is easier because we assume that the formula "$time = temp_1$" only is true in one state along the path.
For deadlines that are given relative to the present time the definition is as follows:

Definition 2.13
$O(now + t_1 < \alpha < (now + t_1) + t_2) \equiv$
$time = k \longrightarrow$
$\Box[time = k + t_1 \longrightarrow O(time < k + t_1 + t_2 \mathcal{U} DO(\alpha))]$

3 Modeling Deadlines (the examples)

In this section we will model the examples given in the introduction within the logical framework developed in the previous section. We will give some additional examples to show the power of our logic.

> Ernie wants to go on a midweek holiday. He has an offer to rent a cottage for relatively little money, which has to be paid within 30 days after the reservation has been made.

This example is modeled as follows:

$$DONE(reserve(Ernie, cottage)) \wedge time = k \rightarrow O(pay(Ernie, cottage) < k + 30 * day)$$

I.e. if $reserve(Ernie, cottage)$ has just been done then there is an obligation to perform the action $pay(Ernie, cottage)$ before 30 days have passed. We assume that *day* stands for an integer that indicates how many times an action should be performed to advance the absolute time with one day. Although parameterized actions were not explicitly introduced in this paper, we use them in the examples in order to get a more realistic representation. The formal introduction of parameterized actions can be found in [6].

The next example illustrates the use of periodic obligations.

> Ernie has to pay the mortgage for his house every month.(before the tenth of the month).

The above example can be modelled very simply as follows:

$$PO(md(time) = 1 < pay(Ernie, mortgage) < md(time) = 10)$$

where *md* is a function that returns the day of the month given an absolute point in time.

The next example illustrates an obligation that should be fulfilled "as soon as possible".

> Ernie borrowed some money from Bert, which he has to repay as soon as he is able to do so.

In this case it might be that Ernie does not have the money or that he first has to pay other debts (like the mortgage). Moreover, maybe before he can pay Bert the money he has to take it from the bank and drive to Bert's house. However, Ernie has to repay Bert as soon as there is no other action that is more "preferred". Here the prefer relation is taken to be something objective, which is clearly not always the case. Hence, it is obvious that this is still a very primitive way to model preferences, but we think a good handle to model this type of deadlines in a more accurate way in the future. The above example can be modeled as follows:

$$DONE(borrow(Ernie, Bert)) \longrightarrow O?(repay(Ernie, Bert))$$

The last example from the introduction is modelled straightforwardly:

> The roof of the house of Ernie started leaking. It has to be repaired before the October rains start. (It is now September).

is modelled by:

$$O(repair(Ernie, roof) < DONE(rain))$$

The last example that we will give here shows some combinations of different types of deadlines.

After the stock of computers has fallen below 10 an order should be made before the stock is less than 6. If an order has been made the delivery should follow within 5 days. If the delivery is not made in time a reminder should be sent. After the receipt of the goods payment should be effectuated within 30 days.

This example is modelled by the following formulas:

1. $O(DONE(stockdecr(Computers < 10)) < order < stock(Computers) < 6)$

2. $DONE(order) \wedge time = k \rightarrow O(delivery < time = k + 5 * day)$

3. $(time = k + 5 * day \wedge \neg DONE(delivery) \mathcal{S}(DONE(order) \wedge time = k)) \longrightarrow O!(send(reminder))$

4. $DONE(receive) \wedge time = k \rightarrow O(pay < time = k + 30 * day)$

The third formula is a typical example of how the violation of an obligation triggers another obligation. This is very natural, because the violation of an obligation should lead to some rectifying action, which is usually an obligation as well.
This is but one instance of our general desire to model violation of deadlines differently than through an inconsistency. In this way it is possible to specify desired follow-up actions in cases of violation.

4 Introducing explicit and continuous time

In the previous sections we showed how temporal logic and dynamic deontic logic can be combined. The temporal logic that was used, however, only deals with implicit time. I.e. only a (temporal) ordering between states is distinguished. The quantitative (or real-)time aspects can be modeled by introducing some clock variable that runs in parallel to the steps forming the relations between the states in the Kripke structure.

In this section we want to mention some of the issues that arise when introducing explicit time to the model. I.e. a time scale that is independent of the relations in the Kripke structure as defined up till now.

The first choice to be made is whether the actions should have different durations. If the actions have different durations the combination of actions is no longer straightforward. If $\underline{a}$ has length 2 and $\underline{b}$ has length 1, what is the length of $\underline{a} \cup \underline{b}$? In the present semantics the negation of performing $\underline{b}$ involves doing $\underline{a}$. However, already after waiting one time unit it can be checked whether $\underline{b}$ has not been performed.
It may be clear that choosing for this (intuitively natural) solution generates many problems in the (neat) algebra of action expressions. Therefore it seems advisable to opt for the second solution. I.e. giving all actions the same standard duration.

Actions with durations can then be represented in two ways. The first option is to cut the actions into smaller actions with standard length. The advantage of this approach is that every action is replaced by a transaction. These transactions can be represented in the Kripke structure as presented before.
In the second option the actions are instantaneous. Actions that have a certain duration are represented by indicating a begin-action that starts the action and an end-action that indicates when the action is finished. The advantage of this approach is that it can be extended to deal with continuous time, while that cannot be done with the first approach which needs discrete steps.
We will leave the details of this choice for further work. However, we do want to remark that the choice what is best depends on the application for which the logic is to be used.

Another issue that comes up when an explicit time is introduced is that of the synchronicity of the actions. In the framework that was used in this paper all actions that are performed in parallel start at the same time. With explicit time this does not have to be the case anymore. Actions can start at any moment in time. This does not pose a theoretical problem when we use discrete time. In this case we divide every action into the smallest possible time units. Every action will then start at a discrete point in time (possibly together with unit-actions of another longer action).
This approach does not work anymore if continuous time is added to the framework.

We finish this section with some observations about the introduction of continuous time. The notions of continuous time and dynamic logic cannot be combined as was done in the present paper. If continuous time is introduced the concepts of dynamic logic, where changes are effected in "jumps", have to be implemented in terms of the temporal logic. (Actually the reverse of what was done in the present paper). The information which actions are "active" and in which stage they are has to be recorded in the states. We leave the details of this (interesting) approach for fu-

ture work.

5 Conclusions

We have shown in this paper how deadlines can be modelled using a combination of dynamic deontic logic and temporal logic. 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. E.g. if a company delivers a product it expects a payment of the customer within a certain time.

Actions and transactions are necessary ingredients in the specification of deadlines. First of all deadlines are always specified on actions. I.e. every deadline indicates that a certain *action* is expected to take place. Secondly the deadline may be dependent on the (trans)actions that are performed. E.g. You have to pay the rent before you buy a new car. These considerations indicate that a formal specification of deadlines should involve a formal specification of actions. We have chosen a form of dynamic logic to incorporate the actions into the specification language.

A second important property of deadlines is that they are not always kept. In the case that keeping a deadline depends on an action from another system (or person) it is not possible to enforce the deadline. Therefore we should use a formalism that allows the violation of the deadline without getting in an inconsistent state. This requirement is fulfilled by the incorporation of deontic logic into the specification language.

The use of a logic as specification language enables the system to reason about the deadlines. Deadlines can be combined and inconsistent deadlines (deadlines that cannot be kept jointly) can be detected.

Deontic temporal constraints can be used in the specification of contracts between agents ([22, 25]). The specification of an agent includes the specification of the possible messages it can exchange with other agents. Certain combinations of messages, for example, a request followed by a commit, create an obligation for one agent with regard to the other. Sometimes, such obligations can be retracted again by means of cancel messages. The *contract* between two agents describes the possible messages and their effect, that is, the factual or deontic temporal constraints that are created or deleted by means of the messages.

Agents have an agenda that contains the actions they are supposed to perform at due time. The agent architecture assumes that the agenda is monitored continuously to decide what to do next. The formal meaning of the agenda is a set of deontic temporal constraints. In the context of an agent architecture, a set of deontic temporal constraints functions as a program.

The present work also opens some areas for further research. The first seems to be further research in representing explicit time in this framework. Secondly, in order to use the logic an axiomatization and inference rules have to be given. On the agent side of this work we should look into the influence of the deadlines on the (planning of) actions of the agent. A simple algorithm would be to plan the action whose deadline expires first as the first action to perform. However, from the OR research (and personal experience) we might deduce that this does not always lead to optimal (or even acceptable) plans. So, some more complicated planning algorithms are called for.

Acknowledgements
We would like to thank the anonymmous referees for their carefull and thorough reading of the article and their usefull remarks. It greatly improved the quality of the article.

References

[1] L. Åqvist. Deontic logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic II*, pages 605–714. Reidel, 1984.

[2] J.C.M. Baeten and W.P. Weijland. Process Algebra. Cambridge University Press, 1990.

[3] J.W. de Bakker, J.N Kok, J.-J.Ch. Meyer, E.-R. Olderog, and J.I. Zucker. Contrasting themes in the semantics of imperative concurrency. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Current Trends in Concurrency: Overviews and Tutorials*, pages 51–121. LNCS 224 Springer, Berlin, 1986.

[4] C. Boutilier Toward a Logic for Qualitative Decision Theory. In JonDoyle, Erik Sandewall and Pietro Torasso (eds.), *Principles of Knowledge Representation and Reasoning, proceedings of the fo urth international conference*, pages 75–86, 1994, Morgan Kaufmann Publishers, San Francisco, California.

[5] M. Broy. A theory for nondeterminism, parallelism, communication and concurrency. In *Theoretical Computer Science*, vol.45, pages 1–62, 1986.

[6] F. Dignum and J.-J.Ch. Meyer. Negations of transactions and their use in the specification of dynamic and deontic integrity constraints. In M. Kwiatkowska, M.W. Shields, and R.M. Thomas, editors, *Semantics for Concurrency, Leicester 1990*, pages 61–80, Springer, Berlin, 1990.

[7] F. Dignum, J.-J.Ch. Meyer and R. Wieringa. Contextual permission. a solution to the free choice paradox. In A. Jones and M. Sergot, editors, *Second International Workshop on Deontic Logic in Computer Science*, pages 107–135, Oslo, 1994. Tano A.S.

[8] F. Dignum, H. Weigand and E. Verharen. *A Formal Specification of Deadlines using Dynamic Deontic Logic.* CSR 96-09, Eindhoven University of Technology, 1996.

[9] E.A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 995-1072, North-Holland, Amsterdam, 1989.

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

[11] R. Goldblatt. *Logics of Time and Computation.* CSLI Lecture Notes 7, 1992.

[12] P. Haddawy and S. Hanks. Representations for Decision-Theoretic Planning: Utility Functions for Deadline Goals. In *Proc. of KR-92*, pages 71-82, Cambridge, 1992.

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

[14] D. Harel. First Order Dynamic Logic. LNCS 68 Springer, 1979.

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

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

[17] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, New York, 1991.

[18] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems.* Springer-Verlag, New York, 1996.

[19] 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.

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

[21] A. Pnueli and E. Harel. Applications of temporal logic to the specification of real-time systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 84-98, 1988.

[22] Y. Shoham Agent-oriented programming. *Artificial Intelligence 60* 1993, pp.51-92.

[23] Singh. *Multiagent systems, a theoretical framework for Intentions, Know-How, and Communication*, LNCS799, Springer-Verlag, 1994

[24] 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.

[25] H. Weigand, E. Verharen and F. Dignum. Interoperable Transactions in Business Models. In P. Constantopoulos, J. Mylopoulos and Y. Vassiliou (eds.), *Advanced Information Systems Engineering*, pages 193-209, LNCS1080, Springer-Verlag, 1996.

[26] G.H. von Wright. Deontic logic. In *Mind*, vol.60, pages 1–15, 1951.