Deontic Dynamic Logic: a Retrospective

John-Jules Meyer

Abstract

In this paper a retrospective is given on the development of deontic dynamic logic. It first reviews the basic system PDeL as introduced in 1988, with emphasis on conceptual issues and technical choices and properties. It then continues with later developments and applications by ourselves and related work by others. Thus we will see how contrary-to-duties and free choice permissions are treated, and how violations can be handled more expressively, including a way of dealing with red/green states and transitions.

1 The basic system PDeL of Deontic Dynamic Logic

In [2] Alan Ross Anderson proposed a way of *reducing* deontic logic to alethic modal logic by using an additional 'propositional constant' V, with the interpretation of indicating a 'bad' state-of-affairs (or *violation* of some set of rules). The terse treatment of this idea in [2] is followed by several other papers, and in particular much more elaborated and explained in [3]. Basically the idea is that obligation to ϕ is taken to be equivalent with the statement that the non-fulfillment of ϕ implies the bad state-of-affairs V, with respect to some kind of conditional (in [3] Anderson argues for relevant implication, but in the modern literature on deontic logic often strict implication in the sense of necessary implication in a modal logic is taken to be the conditional employed in Anderson's reduction.) By adhering to the standard equivalences / definitions for prohibition (forbidden is equivalent with obligated to not) and permission (permitted is not forbidden), we obtain also a similar reducing expression for prohibition: ϕ is forbidden if and only if the truth of ϕ implies the bad state V (with respect to the same choice of conditional).

Inspired by this as well as the fact that there were a lot of problems with Anderson's reduction in the sense of undesirable consequences (theorems in the logic), mostly referred to by the term 'paradoxes', cf. [28], in [29] a reduction to dynamic logic instead of to alethic modal logic is proposed, resulting in the system PDeL, which will be described below. The idea is very simple. First of all, it is made explicit now that we are talking about (performing) *actions*. (In much of the literature including the seminal work by Von Wright ([44]) and Anderson this is not really clear and a bit confused. One uses propositions, but often chooses examples involving the performance of actions or 'acts'.) An action is taken to be forbidden if and only if performance of the action leads to violation (the bad state). In (Propositional) Dynamic Logic this is written as:

$$F\alpha \;=\; [\alpha]V$$

The expression $[\alpha]\phi$ means informally that after execution of the action α it holds that ϕ. Then permission is, as usual, defined as 'not forbidden' and obligation as 'forbidden not'. The former just is expressed as

$$P\alpha \;=\; \neg F\alpha$$

The latter presents us with a problem since now we need the negation of an action (here denoted by an overbar):

$$O\alpha \;=\; F\overline{\alpha}$$

Action negation in the context of dynamic logic is by no means a trivial matter. As pointed out by Broersen [9] there are various different possibilities to give semantics of action negation, the most straight-forward being taking the complement of the relation associated with performing the action. However, for various reasons, Meyer chose a more complicated interpretation of action negation in his original paper [29], the main reasons being twofold: firstly, Meyer wanted to allow the negation of an atomic action a to contain / involve a different action b with the same effect as a. So the complement is taken with respect to a set of action *names* rather than actions viewed as state transformations. This view may be controversial. The second reason for not using the straight-forward complement has to do with sequences of actions. We use the operator ';' for concatenation of (sequences of) actions: $\alpha_1;\alpha_2$ meaning the action α_1 followed by the action α_2. To get intuitive properties for the deontic operators regarding sequences $\alpha_1;\alpha_2$, at least as argued in [29], action negation should satisfy the property: $\overline{\alpha_1;\alpha_2} = \overline{\alpha_1}+(\alpha_1;\overline{\alpha_2})$, where '+' stands for the nondeterministic choice. This ensured the desirable property $O(\alpha_1;\alpha_2) \leftrightarrow O\alpha_1 \wedge [\alpha_1]O(\alpha_2)$. The formal treatment of action negation in this fashion yields an already complicated semantics involving infinite traces of sets of atomic actions (so-called s-sets or synchronicity sets, indicating that the elements of these sets are performed at the same time (in parallel)). In the approach also a parallel operator & is used with intersection as interpretation.

The way this semantics can be best viewed is as action formulas yielding a specification of possible ways of performing actions. So, an atomic action a states that in the course of action that follows, the next step contains at least the action a being brought about. A parallel action specifies that the parallel components are to be brought about together (in parallel). A choice states that one of the components is to be performed. And a negation says that whatever is about to happen (be performed), it is not the action specified. This is captured by the following formal semantics. We use the following semantical ingredients:

Firstly, we have steps (or s-sets), which are non-empty sets of atomic actions (informally denoting a collection of atomic actions to be performed together in one step). Furthermore we have s-traces or just traces, which are infinite sequences of steps. These denote possible behaviors that occur by performing actions. Since we have nondeterministic actions in our language, we need sets of traces as denotations of actions, see below.

In the following, U stands for the powerset of steps (excluding the empty s-set), representing a completely arbitrary step, $\cdot$ stands for concatenation on sequences (lifted to *sets* of sequences in the usual manner, cf. [6]), and S^ω stands for an infinite sequence of sets S. We now obtain the following interpretation that we will call trace semantics.

* $[\![\mathrm{a}]\!]_{traces} = \{S \mid a \in S\} \cdot U^{\omega}$
* $[\![\alpha + \beta]\!]_{traces} = [\![\alpha]\!]_{traces} \cup [\![\beta]\!]_{traces}$
* $[\![\alpha \& \beta]\!]_{traces} = [\![\alpha]\!]_{traces} \cap [\![\beta]\!]_{traces}$

The first of these clauses expresses that the trace semantics of an action a consists of the sequences that start with a step containing a, followed by an infinite number of arbitrary steps. The idea being that this trace semantics of a denotes that we start with at least doing a followed by something completely arbitrary into infinity. The second and third clause express that '+' should be interpreted as a choice (represented mathematically by a set-theoretical union) and '&' as a 'parallel' operator, represented by set-theoretical intersection.

However, this semantics yields only infinite traces, where from some point onwards anything may happen, i.e. the action(s) to be performed in those steps is/are not specified. Although the use of such infinite traces renders the above semantics relatively easy, it is not the end of the story. First of all, we need also the semantics of sequential composition $\alpha;\beta$. If we would just define $[\![\alpha;\beta]\!]_{traces} = [\![\alpha]\!]_{traces} \cdot [\![\beta]\!]_{traces}$, we would in effect get that $[\![\alpha;\beta]\!]_{traces} = [\![\alpha]\!]_{traces}$, since all traces in $[\![\alpha]\!]_{traces}$ are infinite, and thus concatenating something behind it will have no effect. This is obviously not what we want!

The semantics of negation is given by the following algebraic equalities:

* $\overline{\overline{\alpha}} = \alpha$
* $\overline{\alpha_1;\alpha_2} = \overline{\alpha_1} + (\alpha_1;\overline{\alpha_2})$
* $\overline{\alpha_1 + \alpha_2} = \overline{\alpha_1} \& \overline{\alpha_2}$
* $\overline{\alpha_1 \& \alpha_2} = \overline{\alpha_1} + \overline{\alpha_2}$
* $(a;b)\&(c;d) = (a\&c);(b\&d)$, for atomic (or negated atomic) actions a, b, c, d.

together with some distributive laws (as familiar in process algebra [5]) we omit here.

By employing these algebraic equations as a term rewriting system [8] we can see that we can write any term with any number of negations in a normal form, consisting of unions of sequences of (negated) action atoms and ending with U^{ω}. These normal forms will be used to specify whether dynamic logic properties hold.

To this end, first, we need to determine where we will test logical conditions. In standard PDL this is easy: at the end of a trace. However, here all traces are infinite, so this will not work. Moreover, we want of course to test properties at a point where it is significant. For example if we specify behaviour by considering an action $a;b$, where a and b are atomic actions, in the semantics we get sets of infinite traces where the first two steps contain executions of a and b, respectively. So we want to know what is true after these two initial steps. To this end we define the notion of significant parts of trace sets. Let *Traces* stand for the set of all (s-) traces.

Given a set T of s-traces, we define a prefix t' of $t \in T$ *significant* iff

1. $t' \cdot t \in T$ for all $t \in \mathit{Traces}$

2. no proper prefix of t' enjoys the property mentioned in 1.

We denote the set of significant (finite) traces in T as $Sig(T)$.

We now define the semantics of actions:

$$[\![\alpha]\!]_{Sig}(s) = Sig([\![\alpha]\!]_{traces}(s))$$

for all states s. (Here we assume that on the righthandside α is in normal form.) Informally, this says that the 'significant' semantics of an action is the significant part of its trace semantics. So, for example, for an atomic action a, $[\![a]\!]_{Sig}(s) = \{S \mid a \in S\}$.

Perhaps one wonders why this semantics is state-dependent. This is defined in this general way so as to accommodate conditional actions of the kind

$$\text{if condition then } \alpha \text{ else } \beta$$

as well. Obviously, in general for conditional actions of this kind, the traces produced will depend on the state. Here we will not go into this further.

By the way, we are now also in a position to solve the problem with the semantics of sequential composition, since now we define:

$$[\![\alpha;\beta]\!]_{traces} = [\![\alpha]\!]_{Sig} \cdot [\![\beta]\!]_{traces}$$

(Note that this has as a consequence that we obtain: $[\![\alpha;\beta]\!]_{Sig} = [\![\alpha]\!]_{Sig} \cdot [\![\beta]\!]_{Sig}$.)

Next, we have to look at the changes actions bring about. To this end we consider a state transition semantics for actions, which we base on the trace semantics on the one hand and the effect that the execution of an s-set has on the other hand. We assume a function $R : U \to (W \to W)$, where U is the set of all steps again, and W is the set of states (worlds), yielding for each step and each state the state the step leads to (so we assume steps to be deterministic, but this is not an essential assumption). To be able to give a state transition semantics to traces and sets of traces we lift the function R as follows: for $t_1, t_2 \in Traces$,

$$R(t_1 \cdot t_2) = R(t_2) \circ R(t_1)$$

where $\circ$ stands for function composition, and, for trace set T and state s,

$$R(T)(s) = \{s' \mid s' = R(t)(s) \text{ for some } t \in T\}$$

Finally, we obtain our definite semantics of actions by collecting the states resulting after significant parts of traces: for all states s,

$$[\![\alpha]\!](s) = R([\![\alpha]\!]_{Sig}(s))(s)$$

We are now ready to give an appropriate interpretation of formulas in our logic. Apart from the clauses for the atoms and propositional connectives we have the following (non-standard) interpretation of the box operator:

$$M, s \models [\alpha]\phi \;\Leftrightarrow\; \forall s' \in [\![\alpha]\!](s) : M, s' \models \phi$$

With this semantics the following formulas become valid with respect to the deontic operators (where sometimes the actions α_1 and α_2 have to satisfy certain constraints, having to do with significant parts, which we leave out here, for convenience, but cf. [29]): (here $\langle\alpha\rangle\phi = \neg[\alpha]\neg\phi$ is the dual of the box operator.)

$$F(\alpha_1;\alpha_2) \leftrightarrow [\alpha_1]F\alpha_2$$
$$F(\alpha_1+\alpha_2) \leftrightarrow F\alpha_1 \wedge F\alpha_2$$
$$F\alpha_1 \vee F\alpha_2 \rightarrow F(\alpha_1 \& \alpha_2)$$
$$O(\alpha_1;\alpha_2) \leftrightarrow O\alpha_1 \wedge [\alpha_1]O(\alpha_2)$$
$$O\alpha_1 \vee O\alpha_2 \rightarrow O(\alpha_1+\alpha_2)$$
$$O(\alpha_1 \& \alpha_2) \leftrightarrow O\alpha_1 \wedge O\alpha_2$$
$$P(\alpha_1;\alpha_2) \leftrightarrow \langle\alpha_1\rangle P\alpha_2$$
$$P(\alpha_1+\alpha_2) \leftrightarrow P\alpha_1 \vee P\alpha_2$$
$$P(\alpha_1 \& \alpha_2) \rightarrow P\alpha_1 \wedge P\alpha_2$$
$$P\alpha \leftrightarrow \langle\alpha\rangle\neg V$$
$$P\alpha \leftrightarrow \neg O\overline{\alpha}$$

To get an intuitive grasp of these formulas, I explain the first three: the first one says that a sequence of two actions is forbidden if and only if doing the first action of the sequence leads to the fact that the second is forbidden. The second says that a choice of actions is forbidden iff both choices are forbidden. The third says that a joint / parallel execution of two actions is forbidden if at least one of the actions is forbidden.

Remark. Note that the incorporation of the function *Sig* renders the logic non-standard, and less elegant and familiar than standard dynamic logic. So, for example, even a standard validity in PDL such as $[\alpha+\beta]\phi \leftrightarrow [\alpha]\phi \wedge [\beta]\phi$ does not hold any more without special constraints on the action α and β, as was already noted in [29]. This has to do with the aspect of significant parts. For example take $\alpha = a$ and $\beta = \overline{a}$, for some atomic action a. Then $[\![\alpha+\beta]\!]_{traces} = U^\omega$, of which the significant part is the empty sequence ϵ of s-sets, so $[\![\alpha+\beta]\!]_{Sig} = \{\epsilon\}$, and $[\alpha+\beta]\phi \leftrightarrow \phi$. On the other hand, in general $[a]\phi \wedge [\overline{a}]\phi$ is not equivalent with ϕ, as can be seen by considering an action a for which it holds that time increases with 1 unit for both executing a and $\overline{a}$. Then both, under the condition that $t = t_0$, $[a](t = t_0 + 1)$ and $[\overline{a}](t = t_0 + 1)$ but of course not $t = t_0 + 1$ before the actions are executed.

On the other hand, leaving out the aspects of significance from the above semantics, thus rendering a more traditional form of dynamic logic, is - in general - not adequate either since it has some very undesirable properties as has been shown in [30] and more recently in [4]. Because the full logic PDeL is so intricate, it is therefore no surprise that in later work often (e.g. in [21]) restrictions to the language are adhered to, where action negation is less problematic (e.g. only allowing for action literals, i.e. atomic actions and their negations).

From a logical view, it would be very elegant and tempting to consider the whole action language in the logic and stipulate that terms that are equivalent in the algebra

can be substituted for each other into logical formulas, salva veritate. But, indeed, this leads to very undesirable things, such as the ones shown in [4], where this approach is taken (and erroneously assumed to be also the approach taken in PDeL!). For example, as was shown in [30], since it holds that $a = (a + (a; b))$ in the algebra, using the 'naive rule' of substitution, we get that $Fa \leftrightarrow (Fa \wedge F(a; b))$, which implies $Fa \rightarrow F(a; b)$, which is only allowable under a very strict reading of forbidden: if it is forbidden to do anything beginning with atomic action a, then it also forbidden to do anything starting with a (and then something else starting with a b). But, in general this should not be a validity, we think. (It depends on the action b whether a violation occurring after performance of a is maintained after b.)

The complication in the logic thus stems from a form of significance (called 'relevance' in [29]) that should be taken into account. In the algebra a stands for any sequence of actions that at least begins with the execution of an s-set containing a, followed by arbitrary s-sets. As we explained above, this is done to give a manageable account of negation. So naturally the denotation of $a; b$ is included in that of a, but as to the issue of signaling a violation after doing action a, the point of when exactly the violation occurs becomes significant, which is not yet captured by the action algebra per se. So this must be added separately in the semantics of the logic. So, in this semantics both the action algebra and the significance function *Sig* w.r.t. the moment of when a formula (violation) becomes true must be taken into account.

In short, [4] thus shows what happens if the algebra of actions (used for dealing with action negation) is used as a basis for PDeL without taking the notion of significance into account, and in particular the price to be paid for this simplification becomes apparent: apart from a new paradox mentioned above (which perhaps may be lived with in view of a particular application), it is shown that one cannot treat contrary-to-duties (CTD) properly anymore, and the addition of a dynamic variant of the D axiom yields that no possible action is forbidden! In a way it also shows that this approach resembles classical SDL more closely, which is quite interesting in itself.

2 Later developments

After the publication of [29], in which the basic dynamic deontic logic PDeL was proposed, Meyer and colleagues have worked on various extensions of dynamic deontic logic as well as applications in computer science, especially the theory of databases and information systems, more in general.

We discuss here a number of these further developments.

In [1] the dynamic deontic logic PDeL is combined with Anderson's reduction so as to obtain a logic for both ought-to-do and ought-to-be constraints. After making plausible that ought-to-be constraints are not reducible to ought-to-do ones, in the paper it is shown that the two separate frameworks actually merge well, and that the SDL part, although perhaps (too) simplistic, and not capable of representing difficult scenarios such as contrary-to-duties (like the Chisholm set) in an adequate way, it does provide a simple and useful logic to reason about obligations viewed as what happens in ideal worlds. In the integrated model it also becomes possible to prove the earlier informal attempts of reducing ought-to-be to ought-to-do to be false.

Speaking about contrary-to-duties and the Chisholm set in particular. This has obtained much attention in the deontic logic literature. This is about the following scenario:

* You ought to go to the party
* If you go to the party, you ought to tell you are coming
* If you do not go, you ought not to tell you are coming
* You do not go to the party

In [32, 34] a possible solution is offered in the framework of PDeL. In fact, since in PDeL one can express actions more properly including their ordering in time (do a first, followed by b), one can now distinguish between three versions of the Chisholm set, which are called the 'forward', the 'parallel', and the 'backward' version.

Forward version of Chisholm set

* it is obligatory to do α
* if you do α, you have to do β afterwards
* if you do not do α, you have to refrain from doing β

Parallel version of Chisholm set

* it is obligatory to do α
* you have to do β while α is being done
* if you do not do α, you have to refrain from doing β

Backward version of Chisholm set

* it is obligatory to do α
* if you do α, you have to do β first (i.e., before α)
* if you do not do α, you have to refrain from doing β (first)

In fact, what is here called the parallel version of the Chisholm set, is very much related to the Forrester set [24]:

* it is obligatory not to kill
* if you kill, you should do it gently (of course, at the same time as the killing)

It is shown in [32, 34] that all of these can be represented in PDeL, if one allows for multiple distinct violation atoms signaling violations of particular norms. One might object, though, that the fourth premise of the Chisholm set cannot be represented in PDeL. That is because dynamic logic comes down to hypothetical reasoning in the sense that if an action takes place such and so is the result. It cannot express that an action is actually performed.

A related issue is addressed in [20]. Here it is observed that in the PDeL approach, an action that is performed in a violation state and does not change this state is forbidden in the strict sense of the logic. Of course, one can mitigate this by a more careful representation using multiple violation atoms, so that it is clear that the reason why the action is forbidden is a violation of an earlier norm (which is not restored). But one can also give a more careful analysis by distinguishing notions of prohibition and permission taking the status of violation change into account. So, for example, an action taking a non-violation state to a violation state is called forbidden in the sense of a deontic deterioration, and such an action is called deontically undesirable. On the other hand an action that removes a violation is permitted in the sense of causing a deontic improvement, and the action is called deontically desirable. In the two other cases leaving the (non)violation state intact, the action at hand is called deontically indifferent. So formally we define:

$$F^{-}(\alpha) = \neg Violation \wedge [\alpha]Violation$$
$$P^{\sim}(\alpha) = \neg Violation \wedge [\alpha]\neg Violation$$
$$F^{\sim}(\alpha) = Violation \wedge [\alpha]Violation$$
$$P^{+}(\alpha) = Violation \wedge [\alpha]\neg Violation$$

So, in a way this is a shift from just looking at the deontic status of the end state after performing an action to a consideration of the change of deontic status that the action brings about. The paper shows that, especially if one also uses multiple violation atoms (annotated with the norm violated), one is now able to reason about contrary-to-duty scenarios such as the Chisholm set, the Forrester paradox and the Reykjavik scenario in a more refined way without resorting (yet) to such sophisticated means as nonmonotonic reasoning techniques! (cf. [38])

The approach taken in [20] modifies the simple perspective of Meyer's original PDeL [Mey88], where the deontic status of an action is determined only by the deontic status of the end state after the action has been performed. Later work has refined this perspective further. Ron van der Meyden [36] observed that it gave rise to paradoxical results when, for instance, looking at (complex) actions having intermediary states that are deontically bad but ending up eventually with states that are deontically good. Since original PDeL only looks at the end state, something like killing the president and then doing something harmless is still permitted. ("The end justifies the means".) This is reflected in the validity of

$$\langle\alpha\rangle P(\beta) \rightarrow P(\alpha;\beta)$$

"If, after killing of the president, it is permitted to open the door, it is permitted to kill the president and open the door". So Van der Meyden came up with a more sophisticated 'process-based' dynamic logic for permission to deal with these situations

properly. His models contain not only permitted (or 'green') one-step state transitions but also trajectories or sequences of state transitions that are permitted ('green'): such a trajectory is green if all one-step transitions occurring in it are designated green by the model. Next he introduces a formula $\Diamond(\alpha, p)$, stating that p holds after some green trajectory in the execution of action α, and $\pi(\alpha, p)$, stating that all trajectories in the execution of α satisfying the formula p are green, expressing the free choice of the agent to choose among these ways of executing α.

In a similar vein work by Carmo and Jones [12] distinguish both ideal/sub-ideal states and transitions, and later Sergot & Craven [35, 13] incorporate green/red states as well as green/red transitions, thus rendering a more refined treatment of deontically good and bad behavior. The latter comprises a deontic extension of the action logic C+ of Giunchiglia et al. [14], designed for specifying and reasoning about the effects of actions and the persistence of facts over time. (So this is a language for defining certain classes of theories in a *nonmonotonic* formalism called 'causal theories'.) This language is used to describe a labelled transition system and the deontic component provides a means of specifying the deontic status (permitted/acceptable/legal/green) of states and transitions. It features the so called green-green-green (ggg) constraint: a green transition in a green state always leads to a green state. Or, equivalently, any transition from a green state to a red state must itself be red!

Other work extending PDeL concerns the so-called paradox of free choice permission [37]. In standard PDeL (as in SDL) one has a validity regarding permission read as follows:

$$P\alpha \rightarrow P(\alpha + \beta)$$

for any actions α and β. (Here + stands for the choice operator.) So this says in words, for instance, that if one is permitted to post the letter then one is also permitted to post the letter or burn it. In PDeL the choice should be read as imposed rather than free: the agent is not allowed to choose for him/herself! If one would like to express a free choice permission, we should have something like a strong permission operator P_s such that $P_s\alpha$ expresses that the agent is allowed to choose any possible way of performing the action α. Formally, $P_s\alpha = [\alpha]\neg Violation$. Such an operator would not have a property as stated above for permission, but instead enjoy a property

$$P_s(\alpha + \beta) \leftrightarrow P_s\alpha \wedge P_s\beta$$

expressing free choice permission. However, it raises other problems such as e.g. the validity of $P_s\alpha \rightarrow P_s(\alpha \& \beta)$, where & stands for the concurrent execution operator. This is counterintuitive in instances like "If one is permitted to fire a gun, one is also permitted to fire a gun and (while) aim(ing) at the president". In [21, 20] a solution is offered based on the notion of *contexts*: one has to specify precisely which context (i.e., which set of actions) one considers when talking about all possible ways of performing an action. In the above example, taking a context C excluding the action 'aiming at the president', the strong permission $P_s(fire\_a\_gun_C)$ now only states that it is permitted to fire a gun in all possible ways constrained by the context C, thus excluding doing it while aiming at the president!

As to applications: in [33] dynamic deontic logic is employed for specifying integrity constraints in knowledge bases. In particular, attention is paid to deontic con-

straints of the form

$$Balance(a,n) \wedge \neg V : update\_balance(a,m) \wedge n + m < 0 \rightarrow$$

$$[update\_balance(a,m)]V : update\_balance(a,m)$$

which expresses something like if the bank balance of *a* drops below zero (for the first time) then a violation occurs. In [41] this work is extended and it is explained how deontic integrity constraints are inherited in a taxonomic network of types. The example given in that paper is the following: if students are permitted to perform certain actions under certain preconditions, must these preconditions be repeated when specializing this action for the subtype of graduate students, or are they inherited, and if so, how? Generally (and trivially), deontic integrity constraints inherit downwards, but in the paper it is shown that, in practice, integrity constraints may not behave in a common sense way.

Also, more recently, deontic notions have started playing an important role in the area of multi-agent systems. Agents are software entities that have autonomous reasoning and decision capabilities and decide what action to choose next by themselves [43]. Of course, to keep the multi-agent system as a whole useful for performing the tasks it is devised for, the autonomous agents in the systems should be constrained in some way. This can be done by (electronic) institutions or normative elements in the system more in general (cf. [16]). So, currently there are a number of proposals for specifying normative systems by means of deontic logic or at least logics with deontic notions in them like violations and sanctions. Without having the ambition to be complete here, we mention the approaches in [23] in which a model is presented for organizational interaction in multi-agent systems and [15] in which a programming language is proposed for programming normative systems. Also there are interesting relations with (epistemic) update logics [7, 26]. The ideas behind these are directly in line with the developments on deontic reasoning discussed in this paper, where violation atoms are employed to signal violation and trigger possible sanctions or repair operations.

References

[1] P. d'Altan, J.-J. Ch. Meyer & R.J. Wieringa (1996). An Integrated Framework for Ought-to-Be and Ought-to-Do Constraints. *Artificial Intelligence and Law* 4, pp. 77–111.

[2] A.R. Anderson (1958). A Reduction of Deontic Logic to Alethic Modal Logic. *Mind, New Series*, 67 (265) pp. 100–103.

[3] A.R. Anderson (1967). Some nasty problems in the formalization of ethics. *Nous*, 1, pp. 345–560.

[4] A.J.J. Anglberger (2008). Dynamic Deontic Logic and Its Paradoxes. *Studia Logica* 89(3), pp. 427–435.

[5] J.C.M. Baeten, W.P. Weijland (1990). *Process Algebra*, volume 18 of Cambridge tracts in theoretical computer science. Cambridge, UK.

[6] J.W. de Bakker (1980). *Mathematical Theory of Program Correctness*. Series in Computer Science. Prentice-Hall International, London.

[7] A. Baltag & B. Renne (2016). Dynamic Epistemic Logic. *Stanford Encyclopedia of Philosophy*.

[8] M. Bezem, J.W. Klop, R. de Vrijer (2003). Term Rewriting Systems. Cambridge U.P., Cambridge, UK.

[9] J. Broersen (2004). Action negation and alternative reductions for dynamic deontic logics. *Journal of Applied Logic* 2(1), pp. 153−168.

[10] J. Broersen, F. Dignum, V. Dignum & J.-J. Ch. Meyer (2004). Designing a Deontic Logic of Deadlines. In: Proc. Deontic Logic in Computer Science (DEON 2004) (A. Lomuscio & D. Nute, eds.), LNAI 3065, Springer, Berlin, pp. 43−56.

[11] J. Broersen, R.J. Wieringa & J.-J. Ch. Meyer (2001). μ-calculus-based Deontic Logic for Regular Actions. In Proc. 5th Int. workshop on Deontic Logic in Computer Science (DEON00) (R. Demolombe & R. Hilpinen, eds.), Toulouse, 2000, pp. 43−61; revised version under title ?A Fixed-Point Characterization of a Deontic Logic of Regular Action?, Fundamenta Informaticae 45, pp. 1−21.

[12] J. Carmo & A.J.I. Jones (1996). Deontic Database Constraints, Violation and Recovery. *Studia Logica* 57(1), pp. 139−165.

[13] R. Craven & M.J. Sergot (2008). Agent Strands in the Action Language nC+. *J. of Applied Logic* 6, pp. 172−191.

[14] E. Ciunchiglia, J. lee, V. Lifschitz, N. McCain & H. Turner (2004). Nonmonotonic Causal Theories. *Artificial Intelligence* 153, pp. 49−104.

[15] M. Dastani, N.A.M. Tinnemeier & J.-J. Ch. Meyer (2009). A Programming Language for Normative Multi-Agent Systems. Chapter XVI of: *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models* (V. Dignum, ed.), IGI Global, pp. 397−417.

[16] F. Dignum (1999). Autonomous Agents with Norms. *Artificial Intelligence and Law* 7, pp. 69−79

[17] F. Dignum, J. Broersen, V. Dignum & J.-J. Ch. Meyer (2005). Meeting the Deadline: Why, When and How. In: Formal Approaches to Agent-Based Systems (FAABS 2004), Revised Selected Papers (M.G. Hinchey, J.L. Rash, W.F. Truszkowski, eds.). Greenbelt, MD, April 26-27, 2004, LNAI 3228, Springer, Berlin/Heidelberg, pp. 30−40.

[18] F. Dignum, R. Kuiper & J.-J. Ch. Meyer (1998). An Investigation into Deontics of Durative Actions: In Proc. DEON'98 (P. McNamara & H. Prakken, eds.), University of Bologna, pp. 179−195.

[19] F.P.M. Dignum & J.-J. Ch. Meyer (1990). Negations of Transactions and Their Use in the Specification of Dynamic and Deontic IIntegrity Constraints. VU Report IR-209, in: Semantics for Concurrency, Leicester 1990, (M.Z. Kwiatkowska, M.W. Shields & R.M. Thomas, eds.), Springer-Verlag, London/Berlin, pp. 61−80.

[20] F. Dignum, J.-J. Ch. Meyer & R.J. Wieringa (1994). A Dynamic Logic for Reasoning about Sub-Ideal States. In Proc. ECAI'94 Workshop "Artificial Normative Reasoning" (J. Breuker, ed.), Amsterdam, pp. 79−92.

[21] F.P.M. Dignum, J.-J. Ch. Meyer & R.J. Wieringa (1996). Contextual Permission: A Solution to the Free Choice Paradox. In Proc. 2nd Int. Workshop on Deontic Logic in Computer Science (DEON'94), A.J. Jones & M. Sergot (eds.), Tano A.S., Oslo, 1994, pp. 107−135; full version under title "Free Choice and Contextually Permitted Actions", Studia Logica 57(1), pp. 193−220.

[22] F. Dignum, J.-J. Ch. Meyer, R.J. Wieringa & R. Kuiper (1996). A Modal Approach to Intentions, Commitments and Obligations: Intention plus Commitment Yields Obligation. In: Deontic Logic, Agency and Normative Systems (Proc. DEON'96) (M.A. Brown & J. Carmo, eds.), Workshops in Computing, Springer, Berlin, pp. 80−97.

[23] V. Dignum (2004). *A Model for Organisational Interaction: Based on Agents, Founded in Logic*, Ph.D. Thesis, Utrecht University, Utrecht.

[24] J.W. Forrester (1984). Gentle Murder, Or The Adverbial Samaritan. *The Journal of Philosophy* 81/4, p. 194.

[25] D. Grossi, F. Dignum, L.M.M. Royakkers & J.-J. Ch. Meyer (2004). Collective Obligations and Agents: Who Gets the Blame? In: Proc. Deontic Logic in Computer Science (DEON 2004) (A. Lomuscio & D. Nute, eds.), LNAI 3065, Springer, Berlin, pp. 129−145.

[26] M. Knobbout, M. Dastani & J.-J. Ch. Meyer (2016). A Dynamic Logic of Norm Change. In: Proc. ECAI2016 (22nd European Conference on Artificial Intelligence?, The Hague, The Netherlands) (G.A. Kaminka, M. Fox, P. Bouquet, E. Hllermeier, V. Dignum , F. Dignum & F. van Harmelen, eds.), IOS Press, Amsterdam, pp. 886−894.

[27] J. Krabbendam & J.-J. Ch. Meyer (1999). Contextual Deontic Logics. In Proc. DEON'98 (P. McNamara & H. Prakken, eds.), University of Bologna, 1998, pp. 271−290; revised version in: *Norms, Logics and Information Systems* (P. McNamara & H. Prakken, eds.), IOS Press, Amsterdam/Berlin, pp. 347−362.

[28] R.P. McArthur (1981). Anderson's Deontic Logic and Relevant Implication. *Notre Dame J. of Formal Logic* 22, pp. 145−154.

[29] J.-J. Ch. Meyer (1988). A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame J. Formal Logic* 29, pp. 109−136.

[30] J.-J. Ch. Meyer (1989). Using Programming Concepts in Deontic Reasoning. In: *Semantics and Contextual Expression* (R. Bartsch, J. van Benthem & P. van Emde Boas, eds.), FORIS Publications, Dordrecht/Riverton, pp. 117−145.

[31] J.-J. Ch. Meyer (1992). Free Choice Permissions and Ross's Paradox: Internal vs External Nondeterminism. In Proc. 8th Amsterdam Colloquium (1991) (P. Dekker & M. Stokhof, eds.), ILLC, Univ. of Amsterdam, Amsterdam, pp. 367−380.

[32] J.-J. Ch. Meyer, F.P.M. Dignum & R.J. Wieringa (1994). The Paradoxes of Deontic Logic Revisited: A Computer Science Perspective. Techn. Report UU-CS-1994-38, Utrecht University.

[33] J.-J. Ch. Meyer, H. Weigand & R.J. Wieringa (1989). A Specification Language for Static, Dynamic and Deontic Integrity Constraints. In Proc. MFDBS 89, J. Demetrovics & B. Thalheim (eds.), Visegr?d, Hungary, LNCS 364, Springer, Berlin, pp. 347−366.

[34] J.-J. Ch. Meyer, R.J. Wieringa & F.P.M. Dignum (1998). The Role of Deontic Logic in the Specification of Information Systems. In: *Logics for Databases and Information Systems* (J. Chomicki & G. Saake, eds.), Kluwer, Boston/Dordrecht, pp. 71−115.

[35] M.J. Sergot & R. Craven (2006). The Deontic Component of Action Languages nC+. In: Proc. DEON 2006 (L. Goble & J.-J. Ch. Meyer, eds.), LNAI 4048, Springer, Berlin / Heidelberg, pp. 222−237

[36] R. Van Der Meyden (1996). The Dynamic Logic of Permission. *J. of Logic and Computation* 6(3), pp. 465−479.

[37] A. Ross (1941). Imperatives and Logic. *Theoria*, 7: 53−71.

[38] L.W.N. van der Torre (1994). Violated Obligations in a Defeasible Deontic Logic. Proc. ECAI, pp. 371−375

[39] R.J. Wieringa & J.-J. Ch. Meyer (1991). Actor-Oriented Specification of Deontic Integrity Constraints. In Proc. MFDBS '91 (B. Thalheim, J. Demetrovics & H.-D. Gerhardt, eds.), Rostock, LNCS 495, Springer, pp. 89−103.

[40] R.J. Wieringa & J.-J. Ch. Meyer (1993). Actors, Actions, and Initiative in Normative System Specification. *Annals of Mathematics and Artificial Intelligence* 7, pp. 289−346.

[41] R.J. Wieringa, J.-J.Ch. Meyer & H. Weigand (1989). Specifying Dynamic and Deontic Integrity Constraints. *Data & Knowledge Engineering* 4(2), pp. 157−190.

[42] R.J. Wieringa, H. Weigand, J.-J. Ch. Meyer & F.P.M. Dignum (1991). The Inheritance of Dynamic and Deontic Integrity Constraints. *Annals of Mathematics and Artificial Intelligence* 3, pp. 393−428.

[43] M. Wooldridge. (2009). *An Introduction to MultiAgent Systems* (2nd edition). John Wiley & Sons, Chichester, UK.

[44] G.H. von Wright (1951). Deontic Logic. *Mind*, 60, pp. 1−15.

Acknowledgment The author wishes to thank the anonymous reviewer for many helpful suggestions for improvement.

Utrecht University
Institute of Information and Computing Sciences
Intelligent Systems Group
Princetonplein 5, 3508 CC Utrecht, The Netherlands
j.j.c.meyer@uu.nl

and

Emotional Brain B.V.
Louis Armstrongweg 88
1311 RL Almere, The Netherlands