μ-calculus-based Deontic Logic for Regular Actions

Jan Broersen* Roel Wieringa† John-Jules Meyer‡

Abstract

This paper introduces deontic logic of regular actions as a fragment of the modal μ-calculus. Semantic characterizations of deontic notions for regular actions are given in terms of conditions on μ-calculus structures, and μ-calculus formulas capturing this semantics are constructed.

Keywords: Deontic Logic of actions, Dynamic Logic, μ-calculus, finite automatons, regular actions, free choice permission, contrary to duty norms

1 Introduction

This paper proposes a deontic logic for *regular actions*, which are actions built from atomic actions and the action connectives *non-deterministic choice*, *sequence* and *iteration*. The deontic logic notions for non-deterministic choice have been studied extensively in the literature. The notions for sequence have been studied much less (e.g. [14]), and for iteration we are aware of only two studies: one by Van der Meyden [13], who studies the notions of permission and prohibition but does not study obligation, and one by Mc Carty [12], who studies the notions in a logic programming context. We think deontic notions for sequence and iteration are of big relevance to the application of deontic logic to, for example, the specification of soft integrity constraints and exception handling in triggering systems for databases and the specification of fault tolerant systems. In all these areas, we need the ability to specify obligation properties of non-atomic processes and properties that express what must be done if such an obligation is violated. Generally, we claim that if we pursue normative logical specifications of systems in terms of non-atomic, complex actions, possibly in the form of an automaton or program, we must have a clear and complete understanding of deontic notions for sequence and iteration of actions. This paper studies these notions in

*Faculty of Mathematics and Computer Science, *Vrije Universiteit*, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands, email: `broersen@cs.vu.nl`. This work is sponsored by VU-USF as part of the SINS project, and partially supported by the Esprit Working group Aspire, contract nr. 22704

†Faculty of Computer Science, *University of Twente*, P.O. Box 217, 7500 AE Enschede, The Netherlands, email: `roelw@cs.utwente.nl`

‡Intelligent Systems group Department of Computer Science, *Universiteit Utrecht*, Centrumgebouw Noord, office A123 Padualaan 14, De Uithof 3584 CH Utrecht, The Netherlands, email: `jj@cs.uu.nl`

the context of the modal μ-calculus, which is a temporal and process logic that, due to its fixed point semantics, is typically suited for reasoning with iterated actions.

In section 2 we give semantic characterizations of deontic notions for regular actions in terms of conditions on labeled Kripke structures. In section 3 we introduce the modal μ-calculus and the μ-calculus translation of propositional dynamic logic. In section 4 we define compositional μ-calculus expressions that capture the semantic characterizations for permission and prohibition of section 2. Obligation can not be characterized in μ-calculus formulas in the same way. Therefore, in section 5, we define *weakly* compositional μ-calculus characterizations. These characterizations apply to permission, prohibition *and* obligation. For both prohibition and obligation, we also very shortly discuss the contrary to duty notions. Section 6 concludes with a discussion, and related and future work.

2 Semantic characterizations

It seems appropriate first to say something about the language and interpretation of regular actions themselves.

Definition 1 *Given a set $\mathcal{A}$ of atomic action symbols, the syntax of a regular action (α) is defined as follows:*

$$\alpha \quad ::= \quad a \mid skip \mid \alpha \cup \alpha' \mid \alpha; \alpha' \mid \alpha^*$$

The semantics of *regular actions* α is defined in terms of sets of *traces*, which are finite concatenations of atomic actions. Atomic actions a constitute elementary traces in themselves, and *skip* refers to the empty trace. The action connective $\cup$ is associated with union of traces, and ; and * with concatenation and union of all finitely repeated self-concatenations, respectively.

In this section we characterize the deontic notions $P(\alpha)$, $F(\alpha)$ and $O(\alpha)$ for regular actions α. The intended meaning is respectively: 'Permission to perform α', 'Prohibition to perform α' and 'Obligation to perform α'. Meyer [14] incorporates the deontic realm in his dynamic logic with the help of a predicate V that denotes 'violation states'. Van der Meyden's approach [13] can be seen as assigning violation predicates to possible transitions instead of states. Here we take yet another approach, that is conceptually strongly appealing, and that simplifies definitions considerably. We take atomic propositions $P(a)$ (it is permitted to perform atomic action a) as primitive, and introduce valuations of these propositions in labeled Kripke structures.

Definition 2 *Given a finite set $\mathcal{A}$ of action symbols, a structure $\mathcal{S} = (S, \delta, R_{\mathcal{A}})$ is defined as follows:*

- *S is a nonempty set of possible states*
- *δ is a total function $\mathcal{A} \to 2^S$ (if $s \in \delta(a)$, we say that $P(a)$ holds in s)*
- *$R_{\mathcal{A}}$ is an $\mathcal{A}$-indexed collection of (reachability) relations over $S \times S$*

Until section 3, where we introduce the modal μ^a-calculus, we assume a standard modal action logic over these labeled Kripke structures. In these logics $[a]\phi$ has the interpretation 'all performances of action a lead to a state where ϕ holds'. We need this in the next two subsections, where we discuss how to define deontic notions for regular actions as conditions on traces through the structures.

2.1 Permission and Prohibition

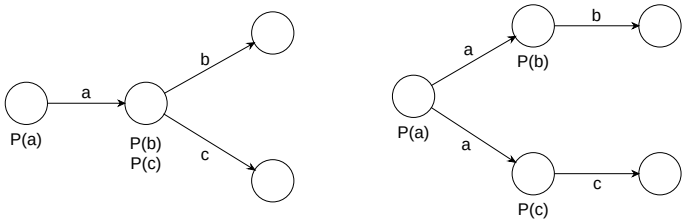
We start by formulating a first rough intuition concerning a possible semantics for the notions of permission and prohibition of regular actions.

> $P(\alpha)$ / $F(\alpha) \equiv$ **it is permitted / forbidden to choose and perform any trace of** α

In an attempt to define a logic based on this intuition, we relate it to the structures of definition 1. Traces of α are now seen as traces from a state s in a structure $\mathcal{S} = (S, \delta, R_{\mathcal{A}})$ where the property $P(\alpha)$, respectively $F(\alpha)$ is thought to hold. We use the following notations concerning traces: $\sigma(\alpha)$ is a trace that interprets the regular action α; a trace σ (of length n, denoted $length(\sigma) = n$, where the length refers to the number of traversed states) is a sequence $s_1^\sigma, a_1^\sigma, \ldots, s_{n-1}^\sigma, a_{n-1}^\sigma, s_n^\sigma$ of states s_i^σ and actions a_i^σ. Now if we define $F(a)$ to be $\neg P(a)$, we may try to reformulate the above intuition in terms of properties of individual traces from a state s in $\mathcal{S}$ as:

> $P(\alpha) \equiv$ **in each** s_i^σ **on each** $\sigma(\alpha)$, $P(a_i^\sigma)$ **holds**
>
> $F(\alpha) \equiv$ **there is an** s_i^σ **on a** $\sigma(\alpha)$ **for which** $F(a_i^\sigma)$

These statements attempt to express the intuitions for permission and prohibition of a regular action by making explicit the relation of more complex expressions concerning permission and prohibition with the primitive notion $P(a)$. But the characterization is incomplete: in terms of modal (action) logic, it leaves room for more than one interpretation, as can be seen from the following example structures.



(1) free choice **(2) imposed choice**

Both structures, in their left most state, obey the semantic characterization when applied to $P(a;(b \cup c))$. But from a modal point of view, the structures are not equivalent since they do not bisimulate: model 1 satisfies $P(a) \wedge [a](P(b) \wedge P(c))$, and model 2 only satisfies the weaker $P(a) \wedge [a](P(b) \vee P(c))$.

If we think of the notion of permission at hand as part of the logic governing an agent, we can say that model 1 corresponds to the logic of an agent that reasons about a future course of events where it has full control over the choices between actions. The agent reasons

'if initially I am permitted to perform either ab or ac, then after I have done a, I can choose either b or c, which implies that after I have performed a, I am permitted both b and c'. This is exactly what is reflected by the formula $P(a) \wedge [a](P(b) \wedge P(c))$. We refer to this interpretation of permission as the (total) free choice semantics.

Model 2 corresponds to the logic of an agent that reasons about a future course of events where it has no (or only partial) control over choices: initially it is permitted to perform either the trace ab or ac, but after it has performed a, it is only permitted (by its environment) to perform either b or c. The agent reasons 'if initially I am permitted to perform either ab or ac, but if I am not permitted to choose between b and c after I have done a, then after I have performed an a, I will either be permitted b or c'. This is exactly what is reflected by the formula $P(a) \wedge [a](P(b) \vee P(c))$. We refer to this interpretation of permission as the (partially) imposed choice semantics. This terminology is inspired by the analogous distinction between internal and external choice for process algebras. The connection between external choice and (partially) imposed choice is that the lack of choice of an agent that is performing a trace, can be viewed as choice that is externally forced upon the agent. Internal choice corresponds to free choice, because all choices can be thought of as internal to, or 'under control' of the agent. Clearly imposed choice permission is a *weaker* form a permission than free choice permission.

Imposed choice is in general only partial. An agent might be deprived of (internal) choice only at certain points in the future course of events. We can easily outline more extensive structures in which this is the case[1]. Each of these structures would have a separate modal formula characterizing it. Now if we want to define the notion $P(\alpha)$, it seems that we will have to decide which permission semantics we persue: free choice, or one of the many possibilities for partially imposed choice. But since we have adopted a trace semantics for regular actions, we implicitly have committed ourselves to a free choice semantics for permission. Allready on the level of action expressions α, the trace semantics is just not able to distinguish between the many different possibilities for partially imposed choice[2]. So, given our choice concerning the action semantics, we are led naturally to the total free choice interpretation as the semantics that makes most sense for regular action expressions α. Adopting the total free choice criterion also corresponds to the semantic choice implicitly made in other systems that define deontic notions of regular actions in a modal logic setting [13, 12].

To characterize the total free choice semantics as a condition on traces through a structure, we will have to look at properties of *sets* of traces. The total free choice semantics imposes that in any point s in a trace $\sigma(\alpha)$ we are not only permitted to do the next action in the trace σ, but we are also permitted to do an action that is the next action in a trace $\sigma'(\alpha)$ with the same prefix as σ. In the following, a prefix $pre_i(\sigma)$ of a trace σ with $length(\sigma) = n$ is a trace $s_1^\sigma, a_1^\sigma, \ldots, s_{i-1}^\sigma, a_{i-1}^\sigma, s_i^\sigma$, with $i \leq n$.

Definition 3 *Semantic characterization of the notions $P(\alpha)$ and $F(\alpha)$, both in words and in a more formal notation:*

> $P(\alpha) \equiv$ ***traversing a structure, in each state all actions are permitted that extend a prefix of a trace of α to a prefix of a trace of α***

[1] Each point in a model where a non-deterministic atomic action appears represents a point where choice is forced upon the agent from the outside.

[2] We could define such a semantics, if we would for instance use (tree) automata for our action language.

$P(\alpha) \equiv$ ***for all*** s_i^σ ***on all*** $\sigma(\alpha)$ ***holds*** $\bigwedge_{a \in \{a_i^{\sigma'} \mid pre_i(\sigma'(\alpha)) = pre_i(\sigma(\alpha))\}} P(a)$

$F(\alpha) \equiv$ ***traversing a structure, there is a state where one of the actions that extends a prefix of a trace of*** α ***to a prefix of a trace of*** α***, is forbidden***

$F(\alpha) \equiv$ ***there is an*** s_i^σ ***on a*** $\sigma(\alpha)$ ***for which*** $\bigvee_{a \in \{a_i^{\sigma'} \mid pre_i(\sigma'(\alpha)) = pre_i(\sigma(\alpha))\}} F(a)$

Note that this characterization is a specialization of the earlier one.

2.2 Obligation

In this section we are concerned with the characterization of *instantaneous* obligations. This notion contrasts with the notion of obligation in the work of Maibaum [11]. Maibaum argues: "We say that obligations are *incurred* or *discharged*, which indicates that they only hold *during* execution". He observes that this contrasts with permissions and prohibitions, which are thought to last after a permitted execution has terminated. However, nothing is said about *at what point* during an execution obligations have to be discharged. Following Maibaum's line of thought, the only answer to this question seems 'at some unspecified time in the future'. But we believe that this slightly undermines the meaning of what it is to have an obligation. A 'some time in future' obligation can not really be an obligation as long as the term 'some time' is not made concrete. Indeed if we do not quantify the temporal aspect of the assertion, the obligation is void, since at any moment in any future course of events we can postpone the obligation to yet another future point. The notion of obligation we formalize is concerned with reasoning about *instantaneous* obligations: obligations that have to be discharged immediately by performing a sequence of actions, beginning *now*[3].

We start with the rough intuition:

$O(\alpha) \equiv$ **it is obliged to choose from the traces of** α **and perform one**

But again we have to become more precise. As for permission and prohibition, we give a precise characterization in words, and in more formal terms. The characterization also reveals the connection with prohibition.

Definition 4 *Semantic characterization of the notion* $O(\alpha)$*, both in words and in a more formal notation:*

$O(\alpha) \equiv$ ***traversing a structure, in each state it is forbidden to perform actions that extend a strict prefix of traces of*** α ***to a trace that is not a prefix of a trace of*** α

$O(\alpha) \equiv$ ***for all*** s_i^σ ***on all*** $\sigma(\alpha)$***, such that*** $s_i^\sigma \neq s_n^{\sigma'}$ ***for some*** $\sigma'(\alpha)$ ***with*** $length(\sigma') = n$***, holds*** $\bigwedge_{a \in \mathcal{A} \setminus \{a_i^{\sigma''} \mid pre_i(\sigma''(\alpha)) = pre_i(\sigma(\alpha))\}} F(a)$

[3]The notion of 'obligation' concerned by Maibaum can actually be expressed using our notion together with the temporal logic expresssivity of the μ-calculus (see section 6).

Two differences with the characterization for permission in section 2.1 attract attention. First, in each state of a trace, there is no reference to the actions that keep us within the trace set of α, but to the actions that would bring us outside it: these actions are forbidden. Given both characterizations, it is not too difficult to see that if $O(\alpha)$ holds, for any program α'[4] other than α, the property $F(\alpha')$ holds. So, also on the program level we obey the deontic principle that 'being obliged' equals 'being forbidden not to'[5]. Second, this condition is not imposed on all states of traces, but only on states that are *not* the final state of a trace of α (alternatively: 'which are on strict prefixes of traces of α'). The reason is that otherwise the obligation to perform actions could possibly never be discharged: in some cases (whenever * is the final connective in a regular action), we would be *obliged to go on forever*. Take as an example $O(b;a^*)$. The trace set is $b, ba, baa, baaa, \ldots$. Now if in states that are the final state of one of these traces we would be obliged to do an action that keeps us within this set, we would always be forced to 'jump' to longer traces by doing an extra a. This obligation would hold forever, and we would be caught in an a-loop[6]. By demanding that only on pre-final states we are forbidden to do actions that bring us out of the trace set, we allow for the possibility to do an 'escaping' action in the final state. The justifying intuition is that in that case we have fulfilled our obligation, since we have performed one of the traces of α. And if we have fulfilled the obligation, we should not any longer be constrained in the performance of actions[7]. This semantic choice implies that $O(a^*) = \top$ for any *atomic* action a, and $O(skip) = \top$, since we can comply with these obligations by performing nothing, which is always possible.

Having characterized permission, prohibition and obligation as conditions on labeled Kripke structures, in sections 4 and 5 we look for μ-calculus formulas that enforce equivalent conditions on this type of structures. But first we introduce the modal μ-calculus and propositional dynamic logic.

3 The modal μ^a-calculus

Informally the modal μ-calculus can be described as modal logic (the weakest form, without reflexivity, symmetry or transitivity) extended with a minimal and a maximal fixpoint operator. The fixpoint operators, in combination with the standard modal logic operators $\langle\,\rangle\phi$ and $[\,]\phi$[8], enable the expression of a wide range of temporal and dynamic properties. The μ-calculus is known to subsume, among others, the temporal logics CTL, LTL, CTL*, ACTL, ECTL* [10], and propositional dynamic logic (PDL) [7, 8]. When we talk of 'the μ-calculus', we refer to the non-parameterized, temporal calculus. In this section the modal

[4]Note that we do not speak of α' as the negation of α. This is because α' is just one of the actions unequal to α, not ***the*** action unequal to α.

[5]It would not be possible to satisfy this principle if we would have chosen a weaker notion of permission. Consequently the notion of prohibition would have become stronger, too strong to relate the notion of obligation to it.

[6]For permission it poses no problem to be permitted to perform an a infinitely often. If, in case of a permission, we are in the final state of a permitted trace, and it is possible to extend it to another permitted trace, we actually *should* be permitted to go on.

[7]Note that the fulfillment of an obligation does not 'cause' an 'obligation to *stop*'.

[8]These operators should not be understood as deontic oparators. Throughout the paper, they will be parameterized with respect to atomic actions, and should be understood as action operators.

μ^a-calculus [9][18] is introduced, whose modalities are parameterized with respect to actions a. We also occasionally talk about the modal μ^m-calculus [16][2], which is the calculus whose modalities are parameterized with respect to sets of actions m.

3.1 Syntax and Semantics

Definition 5 *Given a set $\mathcal{A}$ of action symbols and $a \in \mathcal{A}$, a set $\mathcal{P}$ of proposition symbols and $p \in \mathcal{P}$, and a set $\mathcal{Z}$ of state-set variables*[9] *and $Z \in \mathcal{Z}$, a* ***well formed formula*** *ϕ of the language $\mathcal{L}$ is defined through the following BNF:*

$$\phi \quad ::= \quad p \mid Z \mid \top \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid \langle a \rangle \phi \mid \mu Z.\, \phi$$

The following syntactic abbreviations are applied: $\neg(\neg\phi \vee \neg\psi)$ to $\phi \wedge \psi$, $\neg\langle a\rangle\neg\phi$ to $[a]\phi$, $\neg\phi \vee \psi$ to $\phi \rightarrow \psi$, $(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$ to $\phi \leftrightarrow \psi$. A state-set variable $Z \in \mathcal{Z}$ is a *syntactical aid* for constructing formulas that are interpreted through fixed points. A variable Z is a symbol that will not be given an independent domain interpretation (like p), nor will it be given a logical interpretation (like $\wedge$). State-set variables from $\mathcal{Z}$ are introduced as subformulas of general well formed formulas ϕ to be able to view a formula ϕ as a function $\phi(Z) : 2^S \rightarrow 2^S$ on sets of states, which in turn makes it possible to define the semantics of $\mu Z.\ \phi(Z)$ as a minimal fixpoint (a minimal set of states Z such that $\phi(Z) = Z$) of this function. We apply the usual restriction that the state variable Z only appears within the scope of an even number of negations in bounded formulas. This might seem a severe restriction on the syntax. But the restriction is only on negations that are in the scope of fixpoint operators, and is actually required for having solutions to the fixpoint semantics that is responsible for the expressive power of the μ-calculus. The restriction guarantees monotonicity of functions $\phi(Z) : 2^S \rightarrow 2^S$, which in turn guarantees a solution to the fixed point equation $\phi(Z) = Z$. After this short 'look ahead' into the semantics of the modal μ^a-calculus, we now give the formal description of it. After that, we elaborate on how a fixed point μ^a-calculus formula can best be read.

To define the semantics of arbitrary well formed formulas we use the structures of definition 1, where, for the moment, the deontic valuation function δ is replaced by a general valuation function π that interprets general proposition symbols $\mathcal{P}$. The semantics is defined by extending the valuation function $\pi : \mathcal{P} \rightarrow 2^S$ to the interpretation function $||\phi||_\varepsilon : \mathcal{L} \rightarrow 2^S$, which depends on a function $\varepsilon : \mathcal{Z} \rightarrow 2^S$ that interprets state-set variables possibly present in ϕ. From this it follows that we can view an interpretation $||\phi||_\varepsilon$ of a formula ϕ containing Z as a subformula, as a function from states $\varepsilon(Z)$ to states $||\phi||_\varepsilon$. If we want to make this function explicit we write: $\lambda Z.\ ||\phi||_\varepsilon$[10].

Definition 6 *Given a structure $\mathcal{S} = (S, \pi, R_\mathcal{A})$, the interpretation $||\phi||_\varepsilon$ of a wff ϕ on a structure $\mathcal{S}$ (relative to an interpretation of state-variables ε, not fixed for a structure) is defined by:*

[9] Usually these are called 'state variables'. But we feel that 'state-set variable' is more appropriate, since the variables range over sets of states.

[10] This is just a more accurate notation for the above used $\phi(Z)$.

$$
\begin{array}{lcl}
||\top||_\varepsilon & = & S \\
||\bot||_\varepsilon & = & \emptyset \\
||p||_\varepsilon & = & \pi(p) \\
||Z||_\varepsilon & = & \varepsilon(Z) \\
||\phi \vee \phi'||_\varepsilon & = & ||\phi||_\varepsilon \cup ||\phi'||_\varepsilon \\
||\neg\phi||_\varepsilon & = & S \setminus ||\phi||_\varepsilon \\
||\langle a\rangle\phi||_\varepsilon & = & \{s \in S \mid \exists s' \in S \textit{ such that } (s,s') \in \mathcal{R}_a \textit{ and } s' \in ||\phi||_\varepsilon\} \\
||\mu Z.\phi||_\varepsilon & = & \textit{the least fixed point of the function } \lambda Z.\ ||\phi||_\varepsilon \\
||\nu Z.\phi||_\varepsilon & = & \textit{the greatest fixed point of the function } \lambda Z.\ ||\phi||_\varepsilon
\end{array}
$$

A superscript $\mathcal{S}$ in the interpretation function $||.||_\varepsilon$ is omitted, leaving implicit that interpretations of formulas depend on the structure they are evaluated on.[11] A formula ϕ is defined to be *valid on a structure* $\mathcal{S} = (S, \pi, R_\mathcal{A})$ if and only if $||\phi||_\varepsilon = S$, and a formula is *valid* if it is valid on all possible structures.

We now try to give an intuitive reading of μ-calculus formulas. This reading facilitates the understanding of μ-calculus formulas in general, and in particular the definition of deontic notions for regular actions in section 5. We call this reading of μ-calculus formulas '*repetitive reading*'. As an example, take the formula $\mu Z.\ \phi \vee \langle a\rangle Z$. Now 'repetitive reading' refers to looking at this formula as if it reads $\phi \vee \langle a\rangle(\phi \vee \langle a\rangle(\phi \vee \langle a\rangle(\ldots)))$. So we read the formula as if at the place where a variable Z appears, the subformula that is bounded by μZ is repeated. But we have to be more specific about the meaning of the three dots. In case of binding by a μ, the dots mean 'arbitrary but finite repetition', and satisfying a μ-formula means satisfying *at least one* of the finitely repeated readings of the formula[12]. In case of binding by a ν, the dots mean 'infinite repetition', and satisfying a ν-formula means satisfying the complete infinitely expanded formula[13]. In general, μ-formulas can be used to describe properties of terminating (finite) behavior and ν-formulas can be used to describe properties of nonterminating (infinite) behavior.

3.2 The μ^a-calculus translation of Propositional Dynamic Logic

We now turn to the definition of propositional dynamic logic, as a translation to the μ^a-calculus. In the next section we need this translation in the compositional μ^a-calculus characterizations for permission and prohibition.

Definition 7 *Given a set $\mathcal{P}$ of proposition symbols, a well formed formula (ϕ) of PDL, with α a regular action and $p \in \mathcal{P}$ is defined as follows:*

[11] Due to Tarski [17] the definition of the least fixed point $\mu Z.\ \phi$ of the monotonic function $\lambda Z.\ ||\phi||_\varepsilon$ can be written as $\bigcap\{Z \subseteq S \mid \lambda Z.\ ||\phi||_\varepsilon \subseteq Z\}$. In the definition of the semantics we prefer just to write "the least fixed point of the function $\lambda Z.\ ||\phi||_\varepsilon$", because the characterization due to Tarski is less intuitive.

[12] This reading is justified by the μ-calculus property: $\mu Z.\ \phi(Z) = \phi(\bot) \vee \phi(\phi(\bot)) \vee \ldots$. This property assumes $\vee$-continuity of $\phi(Z)$, which is always obeyed on structures where states have a finite number of successors.

[13] This reading is justified by the μ-calculus properties: $\nu Z.\ \phi(Z) = \phi(\top) \wedge \phi(\phi(\top)) \wedge \ldots$ (assuming $\wedge$-continuity) and $\top \leftarrow \phi(\top) \leftarrow \phi(\phi(\top)) \ldots$ (from the monotonicity of $\phi(Z)$) that together imply $\nu Z.\ \phi(Z) = \phi(\phi(\ldots\phi(\top)\ldots))$, where the dots are thought to represent non-finiteness. Monotonicity also implies $\bot \rightarrow \phi(\bot) \rightarrow \phi(\phi(\bot)) \ldots$, but this is of no use in the simplification of the reading of $\mu Z.\ \phi$.

$$\phi \;::=\; p \mid \top \mid \bot \mid \neg\phi \mid \phi \wedge \psi \mid [\alpha]\phi$$

The semantics of $[\alpha]\phi$ is that ϕ holds at the end of all traces that interpret α. The semantics of the dual $\langle\alpha\rangle\phi$ is that there is trace in the interpretation of α that ends in a state where ϕ holds. This semantics is easily defined through a syntactic translation f to the μ^a-calculus ([3]):

Definition 8 *The μ^a-calculus translation of PDL:*

$$\begin{array}{lcllcl}
f(P) & \equiv & P & f(Z) & \equiv & Z \\
f(\phi \wedge \phi') & \equiv & f(\phi) \wedge f(\phi') & f(\neg\phi) & \equiv & \neg f(\phi) \\
f([a]\phi) & \equiv & [a]f(\phi) & f([\alpha \cup \beta]\phi) & \equiv & f([\alpha]\phi) \wedge f([\beta]\phi) \\
f([\alpha^*]\phi) & \equiv & \nu Z.\ f(\phi) \wedge f([\alpha]Z) & f([\alpha;\beta]\phi) & \equiv & f([\alpha][\beta]\phi) \\
f(\langle skip\rangle\phi) & \equiv & f(\phi) & f([skip]\phi) & \equiv & f(\phi)
\end{array}$$

Note that also state-set variables occur in the translation. This is because they may appear as the result of translating formulas like $[a^*]\phi$. With the help of repetitive reading it is easy to check that the formula $\langle a^*\rangle\phi$, which translates the formula $\mu Z.\ \phi \vee \langle a\rangle Z$ refers to all states in a model where by a finite number of executions of a, we reach a state where ϕ holds, and that $[a^*]\phi$, which translates to $\nu Z.\ \phi \wedge [a]Z$ means that ϕ has to hold after *any* number of executions of a.

4 Compositional characterizations in the μ^a-calculus

From now on we assume that the μ^a-calculus syntax of definition 5 is extended to include regular actions (definition 1) and formulas $P(\alpha)$, $F(\alpha)$ and $O(\alpha)$.

4.1 Permission and Prohibition

We now define μ^a-calculus[14] validities that are sound and complete with respect to the semantic characterization given in section 2.1.

Proposition 1 *The following are validities sound and complete with respect to the semantic characterization of definition 3:*

$$\begin{array}{lcllcl}
P(skip) & \leftrightarrow & \top & F(skip) & \leftrightarrow & \bot \\
P(\alpha;\alpha') & \leftrightarrow & P(\alpha) \wedge [\alpha]P(\alpha') & F(\alpha;\alpha') & \leftrightarrow & F(\alpha) \vee \langle\alpha\rangle F(\alpha') \\
P(\alpha \cup \alpha') & \leftrightarrow & P(\alpha) \wedge P(\alpha') & F(\alpha \cup \alpha') & \leftrightarrow & F(\alpha) \vee F(\alpha') \\
P(\alpha^*) & \leftrightarrow & \nu Z.\ P(\alpha) \wedge [\alpha]Z & F(\alpha^*) & \leftrightarrow & \mu Z.\ F(\alpha) \vee \langle\alpha\rangle Z
\end{array}$$

[14] This semantics can not be captured in CTL [4], because CTL lacks actions, and the possibility to assert what holds 'along the way'.

Note that fixpoint formulas only appear in the decomposition of $P(\alpha^*)$ and $F(\alpha^*)$. In several decompositions formulas of the form $[\alpha]\phi$ and $\langle\alpha\rangle\phi$ appear, which for a further decomposition rely on the μ^a-calculus translation of PDL from section 3.2.

Sketch of a proof

Soundness of $P(\alpha;\alpha') \leftrightarrow P(\alpha) \wedge [\alpha]P(\alpha')$: we prove that this formula holds given that for the traces of $P(\alpha;\alpha')$, $P(\alpha)$, and $P(\alpha')$ the conditions of definition 3 hold. The first observation is that the traces of $\alpha;\alpha'$ are concatenations of traces from α' to traces of α. Now recall from section 3 that $[\alpha]\phi$ means that ϕ holds after all traces in α. So as the formula $P(\alpha)$ on the right hand side imposes the right condition on the first part of concatenations, the formula $[\alpha]P(\alpha')$ imposes the right condition on the second part. The only state that deserves closer attention is the point of concatenation, since in this point we might not satisfy the condition that actions from other traces with the same prefix are also permitted. But, again recalling the semantics of the PDL-expression $[\alpha]\phi$ in terms of traces, $[\alpha]P(\alpha')$ states that traces of $\alpha;\alpha'$ that have equal prefixes in α, all have to obey **the same** permission property $P(\alpha')$ after the performance of α. So the validity actually reflects the total free choice semantics. It is illustrative to verify this for the example structures for $P(a;(b \cup c))$ given above. The validity 'decomposes' $P(a;(b \cup c))$ into $P(a) \wedge [a]P(b \cup c)$. The right hand side states that in both traces ab and ac, after a is performed, the permission property $P(b \cup c)$ must hold. This is only the case for the right model, the one that corresponds to the total free choice semantics.

Soundness of $P(\alpha \cup \alpha') \leftrightarrow P(\alpha) \wedge P(\alpha')$: similar.

Soundness of $P(\alpha^*) \leftrightarrow \nu Z.\ P(\alpha) \wedge [\alpha]Z$: similar.

Completeness: to prove completeness we have to go the other way: given the validities we have to prove that all the right conditions on traces are imposed. The validities, together with the μ^a-calculus expressions for PDL of definition 5, can be used to 'break down' any formula $P(\alpha)$ and $F(\alpha)$ into formulas with only atomic deontic formulas of the form $P(a)$ and $F(a)$. Given the soundness of the validities, this can be used to prove that on all states the conditions as stated in definition 3 are imposed.

■

4.2 Obligation

It would be nice to have similar compositional μ^a-calculus characterizations for the notion of obligation of a regular action. But the following conjecture contradicts this prospect.

Conjecture 1 *For obligation we cannot define the semantics compositionally for* $\cup$, ; *and* *.

First of all, we claim that $O(\alpha \cup \alpha')$ is not likely to be expressible in terms of $O(\alpha)$ and $O(\alpha')$. This is already seen at the atomic level. The semantic characterization of section 2.2 defines $O(a \cup b)$ as $\bigwedge_{x \in \mathcal{A} \setminus \{a,b\}} F(x)$, and $O(a)$ as $\bigwedge_{x \in \mathcal{A} \setminus \{a\}} F(x)$, and $O(b)$ as $\bigwedge_{x \in \mathcal{A} \setminus \{b\}} F(x)$. From this, it follows that: $O(a) \vee O(b) \rightarrow O(a \cup b)$, and obviously we also have $O(a) \wedge O(b) \rightarrow O(a) \vee O(b)$. It is easily seen that the semantic characterization does not allow that the implications are reversed. So the expression we look for (a decomposition $O(a \cup b)$) must be even weaker than $O(a) \vee O(b)$ but yet compositional in $O(a)$ and $O(b)$. This seems impossible.

To show the non-compositionality of $O(\alpha^*)$, we look at the example: $O(a;b)^*$. A μ^a-calculus expression[15] capturing exactly the semantic characterization of section 2.2 is $\nu Z.\,[a](O(b)\wedge [b]Z)$. This is not likely to be equivalent to a notion that is functional in $O(a;b)$, which is identified with $O(a)\wedge[a]O(b)$, because $\nu Z.\,[a](O(b)\wedge[b]Z)$ does not contain any information equivalent with $O(a)$.

Finally we argue that $O(\alpha^*;\alpha')$ is not expressible in terms of $O(\alpha^*)$ and $[\alpha^*]O(\alpha')$. Again we only look at the atomic case: $O(a^*;b)$ is not expressible in terms of $O(a^*)$ and $[a^*]O(b)$. μ^a-calculus formulas corresponding to the characterization of section 2.2 are $\nu Z.\,O(a,b)\wedge[a]Z$, $\top$ and $\nu Z.\,O(b)\wedge[a]Z$. We cannot express the first of these formulas in terms of the other two because $O(a,b)$ cannot be broken down in $O(a)$ and $O(b)$ as was just shown. Furthermore, if we could break down $O(a,b)$, the formulas would still be incomparable, because obligations concerning the atomic action a are completely absent in the second two formulas. However, this is only a counterexample that proves the non-compositionality with respect to the compound operation '*;'. Indeed, for regular actions in which the ; is never preceded by a *, we can prove that $O(\alpha;\alpha')\leftrightarrow O(\alpha)\wedge[\alpha]O(\alpha')$ holds.

Summarizing we may say that the non-compositionality with respect to the regular action syntax has two main causes. The first one is simply that obligation of (free) choice is not compositional. The second one is that obligation of iteration is non-compositional, which is caused by the fact that we can comply with an obligation concerning an iteration by doing nothing. This relates to the absence of atomic obligations in the final states of traces, as discussed before.

Note that we do not claim that obligation is not compositional. The composition is just more complex, and recursive in the structure of deterministic finite automatons corresponding to a regular action. In the next section we show how this works. The form of compositionality involved we call 'weak compositionality'.

5 Weakly compositional μ^a-calculus characterizations through DFAs

We now turn to an alternative μ^a-calculus characterization of deontic notions, that does not aim at compositionality with respect to regular action connectives, but that constitutes a recursive composition from atomic deontic notions, governed by the structure of deterministic finite automatons of a regular action. We show that the notion of obligation as defined in section 2.2 can be captured in the μ^a-calculus in this way. But first we define weakly compositional characterizations for permission and prohibition, and show that they are equivalent to the compositional characterization of section 4.1.

5.1 Permission and Prohibition

We first define the deontic notions for atomic choice actions m, which are sets of atomic actions. We call an element $m\in 2^{\mathcal{A}}$ a *choice action*, because we interpret m as a non-deterministic choice between the atomic actions in m. If $m=\{a,b\}$ we write for instance $P(m)$ as $P(a,b)$.

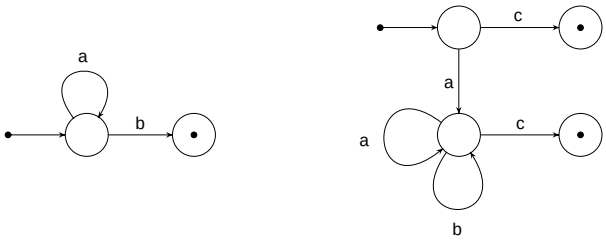
[15] Here we anticipate on the characterization in section 5.3.

Definition 9 *The definition of the deontic notions $P(m)$ and $F(m)$ for atomic choice actions m:*

$$P(m) \quad \leftrightarrow \quad \bigwedge_{a \in m} P(a) \qquad\qquad F(m) \quad \leftrightarrow \quad \neg P(m)$$

Clearly these definitions are compatible with the definition of $P(\alpha \cup \alpha')$ and $F(\alpha \cup \alpha')$ given in 2.1. The first step in the alternative μ^a-calculus definition of deontic notions of regular actions is to associate a regular action with a deterministic finite automaton (DFA) that describes the same set of traces. The second step is to express the semantics of $P(\alpha)$ and $F(\alpha)$ completely in terms of the semantics of $P(m)$ and $F(m)$ by building a μ^a-calculus formula based on the DFA. There is always more than one DFA for a given regular action, but we prove that μ^a-calculus translations of different DFAs of the same regular action are logically equivalent. To get a first impression of this approach, two initial examples are given.

Example 1 *The properties: $P(a^*;b)$ and $P((a;b^*)^*;c)$.*
DFAs of the regular actions $a^*;b$ and $(a;b^*)^*;c$:



The property $P(a^*;b)$ expressed in terms of $P(a,b)$ by means of a μ^a-calculus formula: $\nu Z.\ P(a,b) \wedge [a]Z$. In words: "the permission to perform the action $a^*;b$ equals the permission to initially choose between the atomic actions a and b, and if a is chosen to reach this same deontic state again".

The property $P((a;b^*)^*;c)$ expressed in terms of $P(a,b,c)$ and $P(a,c)$ by means of a μ^a-calculus formula: $P(a,c) \wedge [a](\nu Z.\ P(a,b,c) \wedge [a]Z \wedge [b]Z)$[16]. In words: "the permission to perform the action $P((a;b^*)^*;c)$ equals the permission to initially choose between the atomic actions a and c, and if a is chosen to reach a state where one is permitted to choose between the atomic actions a, b and c, and where if one chooses either an a or a b one reaches the same deontic state again".

The examples show that the μ^a-calculus formula expressing the deontic notion is related directly to the automaton. We now give the precise definition of how the formula is built. First the notions of 'well founded loop' and 'return state' for DFAs are defined.

Definition 10 *A well-founded loop of a deterministic finite automaton $DFA = (S, A, N, s^i, T)$, with S a set of states, A a set of actions, $N : S \times A \to S$ the transition function, $s^i \in S$ the initial state and $T \subset S$ the set of terminal states, is a sequence of edges $e_1, e_2, \ldots, e_n$ ($e_i \in E$ and $E \subset S \times A \times S$) such that:*

[16]Note that in this example the description of the regular action does not contain the choice operator $\cup$ but that in the μ^a-calculus translation to permissions concerning atomic actions, many non-deterministic choices appear. This is the non-determinism that in the action description is part of the semantics of the iteration.

- *e_1 leaves the initial state*
- *each e_i leaves the state that is entered by e_{i-1}*
- *there is an e_i with $i < n$ such that e_i and e_n enter the same state*
- *there is no other pair of edges e_j and e_k that enter the same state*

A state that is entered by the final edge e_n of a well-founded loop is called a "return state". In the following, the set of return states of a DFA is denoted by R.

Proposition 2 *Given a deterministic finite automaton, there are finitely many well-founded loops, and each well-founded loop is of finite length.*

Proof

From negative demonstration: an infinite number of well-founded loops can only be realized with an infinite number of edges and thus with an infinite automaton, and a well-founded loop of infinite length can only be realized with an infinite number of edges in combination with an infinite number of states, since in a well-founded loop we cannot visit a state for the second time (the only exception is the return state at the end of a loop, but there the loop ends).

■

In general there are many well-founded loops, and many return states of these loops coincide. In the procedure defined next, the procedure that builds a μ^a-calculus formula from a given deterministic finite state machine, each state that is the return state of one or more well-founded loops is assigned a separate μ^a-calculus state variable.

Definition 11 *Let α be a regular action, and $M^\alpha = (S, A, N, s^i, T)$ a corresponding deterministic finite automaton with S a set of states, A a set of actions, $N : S \times A \to S$ the transition function, $s^i \in S$ the initial state, and $T \subset S$ the set of terminal states. Furthermore, let $R \subseteq S$ be the set of return states of the DFA, let Z_s be a state variable associated with a return state $s \in R$, and let $out(s)$ be the outgoing automaton actions of a state $s \in S$. Then a μ^a-calculus formula representing the semantics of the deontic notion $P(\alpha)$ is built with the help of a recursive mapping f that associates a μ^a-calculus formula to each automaton state:*

$$
\begin{array}{lllll}
\textit{if } s \in S \setminus R, & f(s) & = & P(out(s)) \wedge \bigwedge\limits_{a \in out(s)} [a] f(N(s,a)) \\
\textit{if } s \in R, & f(s) & = & \nu Z_s.\, P(out(s)) \wedge \bigwedge\limits_{a \in out(s)} [a] f(N(s,a)) \\
\textit{if } s \in R, & f'(s) & = & Z_s
\end{array}
$$

A return state $s \in R$ has two associated formulas: $f(s)$ and $f'(s)$. The value $f(s)$ is used if in a thread of recursive calls of the function, the return state s is visited for the first time, and the value $f'(s)$ is used if in this same thread the state is returned to (visited for the second time). If automaton states have no outgoing actions, their associated formula is $\top$. The semantics of $P(\alpha)$ is defined as the formula associated to the initial state of the automaton: $P(\alpha) = f(s^i)$.

The reader is invited to check that this recursive procedure, applied to the deontic notions of example 1, returns the correct μ^a-calculus formulas. For the notion $F(\alpha)$ we can define a separate recursive mapping with $\vee$ instead of $\wedge$, $\langle\ \rangle$ instead of $[\ ]$ and μ instead of ν. But this is equivalent to defining $F(\alpha)$ as $\neg P(\alpha)$.

Proposition 3 *The recursive mapping f in definition 11 always returns a finite, well-formed μ^a-calculus formula.*

Proof

Each 'thread' of recursive calls either follows a well-founded loop through the automaton or ends in an end-state of the automaton. Since the automaton contains only finitely many well-founded loops and since the loops are of finite length, the recursive calls eventually stop. The well-formedness is straightforward given the definition of the mapping f.

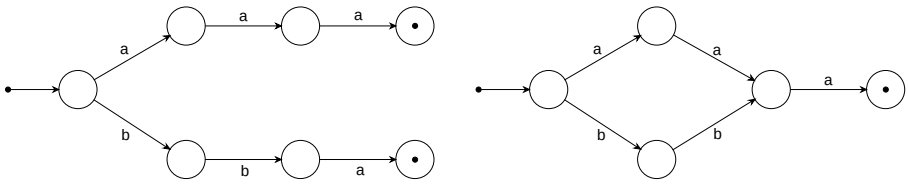
■

A regular action is equivalent with many different DFAs. The following proposition states that nevertheless the semantics as defined by definition 11 is preserving.

Proposition 4 *μ^a-calculus translations of different DFAs describing the same set of traces are logically equivalent.*

Sketch of a proof

We do not prove this formally, but reveal the intuition behind the proof. From automaton theory it is known that for any DFA (and even FA) there is a unique minimal DFA (MDFA) that describes the same trace set, and, that DFAs differ from the MDFA only in the sense that some MDFA-states have equivalent copies. The recursive function of definition 11 is not able to distinguish between DFAs that only differ in the sense that certain states have copies. This is illustrated by a small example .



Both the DFAs of the example represent the trace set $\{aaa, aba\}$. The right DFA is minimal. The left DFA contains copies of the end and before-end state of the minimal DFA. This is a very simple example of copies of automaton states. In general also return states can be copied, which may lead to DFAs that are not easily recognizable as equivalent to the minimal one. It is not difficult to see that for both DFAs the mapping f of definition 11 returns exactly the same μ^a-calculus formula. When copies of return states are involved, μ^a-calculus formulas may differ in the names of state-set variables Z_s with $s \in R$. But clearly this does not have any influence on the logical content of the formula.

■

We now turn to the claim that both explored μ^a-calculus characterizations of the notions of permission and prohibition, are equivalent. In the following, we denote the initial state s^i of a deterministic finite automaton M^α corresponding to a regular action α by $s^i(M^\alpha)$.

Theorem 5 *The notions $P(\alpha)$ and $F(\alpha)$ of proposition 1 and definition 11 are equivalent.*

Sketch of a proof

We prove this by showing that this second translation into the μ^a-calculus is also sound and complete with respect to the semantic characterization of definition 3. To prove soundness, we have to prove that on structures for which the conditions for $P(\alpha)$ of definition 3 hold, the formula $f(s^i(M^\alpha))$ is valid. Now traces of α can be seen as arbitrary but finite paths from begin to end state in M^α, and different traces of α with a common prefix, initially follow the same path in M^α, because M^α is deterministic[17]. Now the conditions on traces, as stated in definition 3, demand that in each state the next action of a trace and all next actions of traces with an identical prefix, are permitted. The subformulas $P(out(s))$ and the construction of $f(s^i(M^\alpha))$ exactly guarantee this.

To prove completeness, we have to prove that the formula $f(s^i(M^\alpha))$ imposes the conditions of definition 3 on traces. This is seen directly from the construction of $f(s^i(M^\alpha))$. ■

5.2 Obligations, permissions and prohibitions, contrary to a prohibition

An important feature of deontic logics is their capacity to assert properties that say what should hold when a norm is actually violated. In this logic these *contrary to duty* notions can be dealt with very well. In case of violation of a prohibition, such properties typically have the form: $F(\alpha) \wedge [\alpha]\phi$. In this formula $F(\alpha)$ says that α is forbidden, and $[\alpha]\phi$ says that ϕ holds if this prohibition is violated. The formula ϕ can of course express a new prohibition, a permission, an obligation, or any other μ^a-calculus property.

5.3 Obligation

First we give the definitions of the characterization through automatons. Then we show that this characterization is equivalent to the semantic characterization of obligation given in section 2.2.

Definition 12 *The definition of the deontic notion $O(m)$ for atomic choice actions m:*

$$O(m) \quad \leftrightarrow \quad \bigwedge_{a \in \mathcal{A} \setminus m} \neg P(a)$$

Note that for atomic actions a and b it follows that $O(a) \vee O(b) \rightarrow O(a, b)$. This form of Ross's anomaly only resides on this atomic level, and can actually be easily avoided by demanding that $O(m)$ also means that for all actions a in m the property $P(a)$ holds.

Definition 13 *Let α be a regular action, and $M^\alpha = (S, A, N, s^i, T)$ a corresponding deterministic finite automaton with S a set of states, A a set of actions, $N : S \times A \rightarrow S$ the transition function, $s^i \in S$ the initial state, and $T \subset S$ the set of terminal states. Furthermore, let $R \subseteq S$ be the set of return states of the DFA, let Z_s be a state variable associated*

[17] This shows why we need DFAs for the μ^a-calculus translation, and not NDFAs.

with the return state $s \in R$, and let $out(s)$ be the outgoing automaton actions of a state $s \in S$. Then a μ^a-calculus formula representing the semantics of the deontic notion $O(\alpha)$ is built with the help of a recursive mapping f that associates a μ^a-calculus formula to each automaton state:

$$\begin{array}{lll}
\textit{if } s \in S \setminus (R \cup T), & f(s) & = O(out(s)) \wedge \bigwedge_{a \in out(s)} [a] f(N(s,a)) \\
\textit{if } s \in R \setminus T, & f(s) & = \nu Z_s.\, O(out(s)) \wedge \bigwedge_{a \in out(s)} [a] f(N(s,a)) \\
\textit{if } s \in T \setminus R, & f(s) & = \bigwedge_{a \in out(s)} [a] f(N(s,a)) \\
\textit{if } s \in R \cap T, & f(s) & = \nu Z_s.\, \bigwedge_{a \in out(s)} [a] f(N(s,a)) \\
\textit{if } s \in R, & f'(s) & = Z_s
\end{array}$$

The distinction between primed and unprimed values equals that in definition 11. The semantics of $O(\alpha)$ is defined as the formula associated to the initial state of the automaton: $O(\alpha) = f(s^i)$.

The differences with the definition of the mapping f for permission follow directly from the semantic choices discussed in section 2.2. First, on the level of atomic actions, we have of course $O(out(s))$ instead of $P(out(s))$. Second, the final states $s \in T$ are treated seperataly, in definitions of $f(s)$ for states $s \in T \setminus R$ and $s \in R \cap T$. The difference with the definition of $f(s)$ in the non-terminal states $s \in S \setminus (R \cup T)$ and $s \in R \setminus T$ is that the atomic obligations $O(out(s))$ are left out, which corresponds to the semantic choice that in the final states of traces (terminal states in the automatons) no *atomic* obligations should hold.

Theorem 6 *The translation $O(\alpha) = f(s^i(M^\alpha))$, is sound and complete with respect to the semantic characterization of definition 4.*

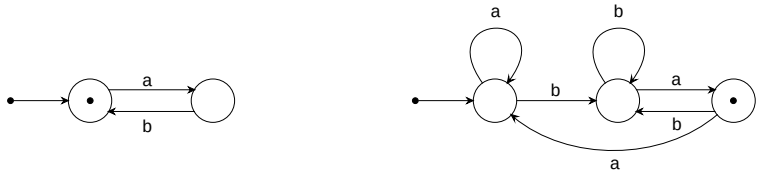
Sketch of a proof

The proof of the soundness and completeness is similar to that for $P(\alpha)$, and follows from the correspondence of the semantic characterization in terms of traces, and the structure of automatons, seen as canonical structures for $O(\alpha)$.

■

We conclude this section with two examples.

Example 2 *The properties: $O((a;b)^*)$ and $O((a \cup b)^*;b;a)$.*
DFAs of the regular actions $(a;b)^*$ and $(a \cup b)^*;b;a$:



The μ^a-calculus expression for $O((a;b)^*)$ is: $\nu Z.\, [a](O(b) \wedge [b]Z)$. Note that $O((a;b)^*) \not\to O(a)$, while for permission: $P((a;b)^*) \to P(a)$. Again this shows the difference between permission and obligation with respect to atomic notions holding in final states of traces.

The μ^a-calculus expression for $O((a \cup b)^*; b; a)$ is: $\nu Z.\ O(a,b) \wedge [a]Z \wedge [b](\nu Y.\ O(a,b) \wedge [b]Y \wedge [a]([b]Y \wedge [a]Z))$. This example shows that it is not always straightforward to recognize that a regular action and a DFA describe the same set of traces. Consequently, a μ^a-calculus expression is not always easily recognized as describing a certain deontic notion of a regular action.

5.4 Obligations, permissions and prohibitions, contrary to an obligation

In the logic also the *contrary to duty* notion with respect to an obligation can be dealt with very well. These properties typically have the form: $O(\alpha) \wedge [\overline{\alpha}]\phi$. In this formula $O(\alpha)$ says that α is obliged, and $[\overline{\alpha}]\phi$ says that ϕ holds (or, should be done) if this obligation is violated. But we did not yet define the notion of *regular action negation* used in this formula. The semantics of $[\overline{\alpha}]\phi$ is that ϕ holds after every atomic action that brings us outside the trace set of α. Again, we can capture this notion in a construction with finite automatons. We can simply replace all appearances of $O(out(s))$ in definition 14 with $[\overline{out(s)}]\phi$, where the notation $\overline{m}$ denotes the complement $\mathcal{A} \setminus m$ of a set of actions m. This ensures that in each automaton, after any of the actions *not* among the actions leaving that automaton state, ϕ holds. Allowing expressions like $[\overline{m}]\phi$, means that we have to generalize the μ^a-calculus we used so far to the μ^m-calculus as used by Stirling et al [16][2].

6 Discussion

We defined a deontic logic for regular actions as a mapping to the modal μ^a-calculus. Meyer [14] earlier studied the deontic notions of non-deterministic choice, sequence and parallel execution of actions. Meyer's main idea is to introduce an action algebra containing negation within the box of modal logic. Since Meyer does not consider iteration, we can only compare our present logic with his on choice and sequence. We take the example $O((a;b) \cup (c;d))$. In our present semantics this property is identified with the intuitively strongly appealing $O(a \cup c) \wedge [a]O(b) \wedge [c]O(d)$[18], whereas in Meyer's logic no simple identification is possible. This is in favor of the argument that our present semantics provides a more intuitive formalization of the combined deontic notions for sequence and choice.

Van der Meyden [13] defines two notions of permission for regular actions: one that corresponds to free choice permission and one he calls 'not forbidden'. His free choice permission $\pi(\alpha, X)$ aims at the formalization of the same notion as our permission does. The only difference is that Van der Meyden's notion of free choice permission for regular actions enforces termination of executions by demanding traces to end in states where X holds. But if we substitute $\bot$ for X in Van der Meyden's (complete) axiomatization, we see that our formalization of permission is equivalent to his[19]: the validities of proposition 1 are easily seen to be the μ^a-calculus equivalents of Van der Meyden's axioms for $\pi(\alpha, \bot)$. The reasons we redefined Van der Meyden's notion of permission here are that we wanted to reveal the connection with μ^a-calculus and finite automatons, that Van der Meyden leaves undiscussed

[18] For this simple property the strength of the μ^a-calculus is clearly superfluous.

[19] If we perform this substitution in the description of the semantics, the equivalence is less clear.

the semantic choice between partially imposed and total free choice, that we wanted to show that the semantics of this notion can be defined without labeling actions as forbidden or permitted, as deemed necessary by Van der Meyden, and that we wanted to show the differences with the μ^a-calculus definitions for obligation. Van der Meyden does not deal with obligation.

The reduction to μ^a-calculus has many advantages. First of all, the modal μ^a-calculus is decidable. Second, a complete axiomatization for the μ^a-calculus is known [18]. Third, much research on μ-calculus is already undertaken or on its way, including work on clausal resolution procedures for the modal μ-calculus. Fourth, there are several model checkers available for both the standard μ-calculus [1] and for the μ^a-calculus [15][5]. Fifth, since in the μ-calculus a wide range of temporal logics is expressible (CTL, LTL, CTL*, ACTL, ECTL* etc.[10]) we get for free a mixture of deontic and temporal notions. We can express "on all paths it globally holds that some time in the future it is obliged to perform the action $(a;b)^*$" as (using CTL* [6]) $\forall G\ F\ O((a;b)^*)$ and (using the μ^a-calculus) $\nu X.(\mu Y.(\nu Z.O(a) \wedge [a](O(b) \wedge [b]Z)) \vee [any]Y) \wedge [any]X$. This type of properties is reminiscent of the obligation properties defined by Maibaum [11], as discussed in section 2.2.

Only in section 5.3 we mentioned one of the famous anomalies that have plagued many deontic systems: Ross's anomaly. But we allready pointed out how to avoid the anomaly on the atomic level. And it is not difficult to see that on the level of regular actions, we allready avoid Ross's anomaly, so: $O(\alpha) \not\rightarrow O(\alpha \cup \alpha')$. It might be clear that we are also not subject to the free-choice anomaly, since we have formalized a notion of free choice permission.

We hope to explore extensions to concurrent actions, and how the extensions of the framework allowing such actions fair in the face of paradoxes associated with such actions, such as the penitent's anomaly, the gentle murderer anomaly, and one of the readings of the Chisholm anomaly[20]. Furthermore, we want to investigate the addition of extra program components to the action language, such as test. Finally, we plan to investigate to what extent the DFAs we associated with deontic notions can be used as canonical structures to model check properties on. This might provide a partial replacement of theorem proving by model checking for this logic.

References

[1] A. Biere. mu-cke - efficient mu-calculus model checking. In O. Grumberg, editor, *International Conference on Computer-Aided Verification (CAV'97)*, pages 468–471. Springer, 1997. Lecture Notes in Computer Science 1254.

[2] J.C. Bradfield. On the expressivity of the modal mu-calculus. In Rüdiger Reischuk Claude Puech, editor, *Proceedings STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France,*, pages 479–490. Springer, 1996. Lecture Notes in Computer Science 1046.

[3] Julian Charles Bradfield. *Verifying Temporal Properties of Systems*. Birkhäuser Boston, Mass., 1992.

[20] The temporal 'variant' of the Chisholm anomaly is actually avoided in the logic. Space limitations prevent us from elaborating on this solution to the Chisholm anomaly.

[4] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2), April 1986.

[5] R. Cleaveland and S. Sims. The NCSU concurrency workbench. In R. Alur and T. Henzinger, editors, *Computer-Aided Verification (CAV '96, New Brunswick, NJ)*, pages 394–397. Springer, 1996. Lecture Notes in Computer Science 1102.

[6] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, volume B: Formal Models and Semantics*, chapter 14, pages 996–1072. Elsevier Science Publishers B.V., 1990.

[7] M.J. Fischer and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, September 1979.

[8] D. Harel. *First Order Dynamic Logic*. Springer, 1979. Lecture Notes in Computer Science 68.

[9] Dexter Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, December 1983.

[10] Dam M. CTL* and ECTL* as fragments of the modal mu-calculus. *Theoretical Computer Science*, 126:77–96, 1994.

[11] T. Maibaum. Temporal reasoning over deontic specifications. In J.-J. Ch. Meyer and R.J. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*, pages 141–202. Wiley, 1993.

[12] L.T. McCarty. Modalities over actions I. Model theory. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principles of Knowledge Representation and Reasoning*, pages 437–448, Bonn, Germany, 1994. Morgan Kaufmann. Proceedings of the Fourth International Conference.

[13] R. van der Meyden. The dynamic logic of permission. *Journal of Logic and Computation*, 6(3):465–479, 1996.

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

[15] Bernhard Steffen, R. Cleaveland, and J. Parrow. The concurrency workbench: A semantics–based verification tool for finite state systems. *ACM Transactions on Programming Languages and Systems, TOPLAS*, 15:36–72, January 1993.

[16] Colin Stirling. Modal and temporal logics for processes. *Lecture Notes in Computer Science*, 1043:149–237, 1996.

[17] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 1955.

[18] Igor Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. In *proceedings LICS'95, San Diego, California*, pages 14–24, 1995.