

Deontic Logic, Contrary to Duty Reasoning and Fault Tolerance

Tom Maibaum

McMaster University

(in collaboration with Pablo Castro)

Talk Outline

- Brief introduction to Deontic Logics.
- Our Deontic Logic.
- Extension to Temporal Logic.
- Contrary-to-Duty Reasoning.
- An Example.
- Conclusions and further work.

Deontic Logics

- Created to reason about ethics and laws.
- Usually these logics have two modalities: P (permission) and O (obligation).
- Ernst Mally was the first to try of formalize Deontic notions. But, in his system: $\vdash p \equiv O(p)$.
- Different kinds of deontic logics have been developed, many of them are modal logics.

System KD (Ought-to-be)

Some researchers pointed out that obligation could be seen as a variation of modal necessity (\Box). This has the following benefits:

- Straightforward semantics (Kripke Models).
- Easily Axiomatizable.
- Well-Known proof properties.
- We can define: $P(\varphi) \equiv \neg O(\neg\varphi)$.

but...

Drawbacks of KD

It inherits some features from Modal Logics:

- We can nest obligations: $O(O(\varphi))$.
- $P(\varphi \vee \psi) \equiv P(\varphi) \vee P(\psi)$ (Weak Permission).

If you are allowed to drive, then you are allowed to drive or to rob a bank.

- $O(\varphi) \rightarrow O(\varphi \vee \psi)$ (Ross' paradox).

If you are obliged to send a letter, then you are obliged to send a letter or burn it.

Ought-to-do Systems

Computer scientists may prefer ought-to-do logics, because they allow us to predicate about action behavior. J.J.Meyer and Maibaum proposed dynamic deontic logics or deontic action logics, where:

- The deontic predicates are applied to actions: $P(\alpha)$ (α is allowed), $O(\alpha)$ (α is obliged)
- We have different operators for actions, some of them are well-known: $\alpha; \beta$ (composition), α^* (iteration), $\alpha \sqcup \beta$ (choice).
- Different authors consider different combinators on actions.

Description vs Prescription

Deontic predicates allow designers to distinguish between *prescription* and *description* of systems.

- Description: What the system does, stated with pre and post-conditions.
- Prescription: What the system should do, stated using the deontic predicates

If the two are 'mixed', i.e., reducing deontic predicates to modal connectives, this can lead to paradoxes.

Deontic Logics and Fault-Tolerance

Deontic logics allow us to distinguish between normal and abnormal situations: using this we can characterize what may go wrong and what to do about it. Some benefits of deontic logics are:

- A language to express normative reasoning (permission, obligation, forbidden).
- A natural level of abstraction in semantic structures (states are divided into “good or “bad” ones).
- It is easy to mix it with temporal logics, and therefore to gain the good properties of temporal frameworks to verify systems.

Contrary-to-Duty Paradoxes

Scenarios which intuitively are consistent, but their formalization in deontic logic gives an inconsistent set of sentences. For example (the gentle killer paradox):

- You ought not to kill.
- If you kill, you ought to kill gently.
- You kill.

This set of sentences is inconsistent in KD (the “standard” deontic logic).

- Contrary-to-Duty statements are usual in fault-tolerance, where an obligation to perform an action arises after another obligation was not fulfilled.

Our Ought-to-do deontic logic

We use two versions of permission:

- $P(\alpha)$, strong permission. α is executable in all (local) contexts.
- $P_W(\alpha)$, weak permission. α can only be executed in some (local) contexts.
- $O(\alpha) \equiv P(\alpha) \wedge \neg P_W(\bar{\alpha})$.

The definition of obligation avoids some paradoxes such as Ross' paradox. In addition, there exists a strong connection between the two versions of permission:

- $P(\alpha) \wedge \alpha \neq \emptyset \rightarrow P_W(\alpha)$.
- $\neg P(\alpha)$ implies that for some $\alpha' \sqsubseteq \alpha$: $\neg P_W(\alpha')$

Semantics

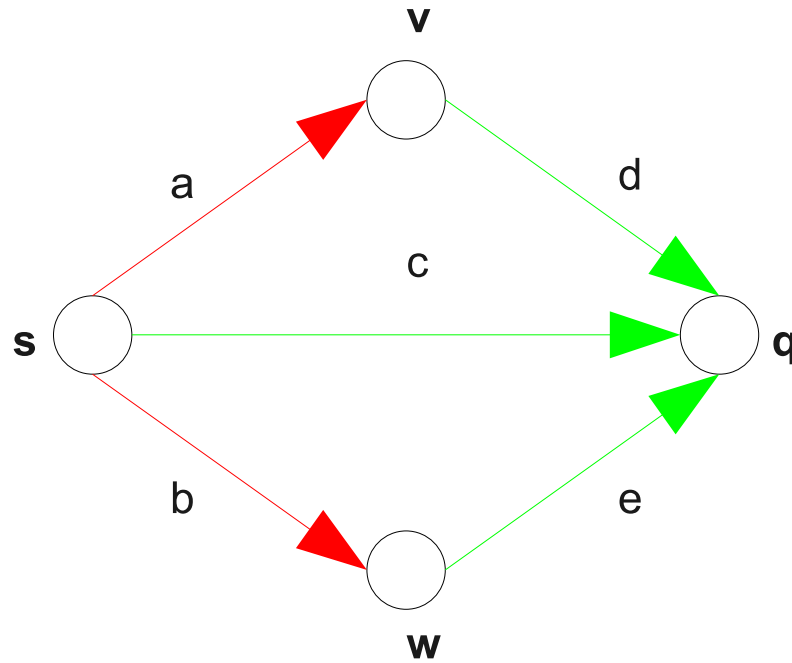
Rather than the classical approach to dynamic logic (actions interpreted as standard accessibility relations), we take the approach used by Kent-Maibaum.

We interpret each action as a set of “events”:

- $I(\alpha) = \{e_1, \dots, e_n\}$.
- $I(\alpha \sqcup \beta) = I(\alpha) \sqcup I(\beta)$
- $I(\alpha \sqcap \beta) = I(\alpha) \sqcap I(\beta)$
- $I(\bar{\alpha}) = E - I(\alpha)$

Our models are labelled transition systems, where we have “coloured” transitions.

Model Example



Semantics

- $w, M \models \langle \alpha \rangle \phi \Leftrightarrow$ there exists some $w' \in \mathcal{W}$ and $e \in \mathcal{I}(\alpha)$ such that $w \rightarrow^e w'$ and $w', M \models \phi$.
- $w, M \models P(\alpha) \Leftrightarrow$ **for all** $e \in \mathcal{I}(\alpha)$, $\mathcal{P}(w, e)$ holds.
- $w, M \models P_W(\alpha) \Leftrightarrow$ **there exists** some $e \in \mathcal{I}(\alpha)$ such that $\mathcal{P}(w, e)$

The underlying action calculus is an atomic boolean algebra.

About the complement

The complement is a powerful, but a dangerous, operator. Combined with iteration it makes the logic undecidable. Our complement is relative to the state from which a transition is taken, not the whole space of actions.

● $w, M \models [\bar{\alpha}]\varphi$ iff for every $e \in E - \mathcal{I}(\alpha)$ and $w \xrightarrow{e} w'$, then $w', M \models \varphi$

The complement does not relate states which are not related in the model; this happens when we use the relational complement: $U - R$ where U is the universal relationship.

Some Axioms

- Classical Axioms for BA

- $P(\emptyset)$

- $P(\alpha \sqcup \beta) \leftrightarrow P(\alpha) \wedge P(\beta)$

- $P(\alpha) \vee P(\beta) \rightarrow P(\alpha \sqcap \beta)$

- $\neg P_W(\emptyset)$

- $P_W(\alpha \sqcup \beta) \leftrightarrow P_W(\alpha) \vee P(\beta)$

- $P_W(\alpha \sqcap \beta) \leftrightarrow P_W(\alpha) \wedge P_W(\beta)$

- $P(\alpha) \wedge (\alpha \neq_A \emptyset) \rightarrow P_W(\alpha)$

- $(\bigwedge_{[\alpha]_{BA} \wedge \alpha \sqsubseteq \alpha'} (P_w(\alpha) \vee (\alpha =_A \emptyset))) \rightarrow P(\alpha')$
(note that this is a finite formula).

- $O(\alpha) \leftrightarrow P(\alpha) \wedge \neg P_W(\bar{\alpha})$

Metatheoretical Properties

The logic has some good metatheoretical properties:

- Soundness: $\vdash \varphi \Rightarrow \models \varphi$
- Completeness: $\models \varphi \Rightarrow \vdash \varphi$
- Decidability: The small model theorem allows us to have a decision procedure, though exponential.
- Strong Completeness ($\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$).

Temporalizing Deontic Logics

To allow us to reason about traces we extend the logic to a CTL temporal logic. We have the following operators:

- $AG(\varphi)$, $AF(\varphi)$, $AN(\varphi)$.
- $EG(\varphi)$, $EF(\varphi)$, $EN(\varphi)$.
- $A(\varphi U \psi)$.
- $done(\alpha)$.

The done operator is useful for several reasons, for example:

$$AN(done(\alpha))$$

the next action to be executed is α

Stratified Norms

We extend the logic with several versions of permissions. We have a (finite) set I_0 of indexes, and we introduce the following formulae.

- If α is an action and $i \in I_0$, then $P^i(\alpha)$ is a formula.
- If α is an action and $i \in I_0$, then $P_w^i(\alpha)$ is a formula.

Using these different versions of permissions we can define several versions of obligations. These formulae allow us to have stratified norms, and avoid contrary-to-duty paradoxes.

Violation Constants

We can consider a finite set of logical constants v_1, \dots, v_n . These constants allows us to state that a violation has occurred. Violations may occur since a prescription was not fulfilled, or since an error state was reached.

- $F(\alpha) \rightarrow [\alpha]v$. If it is forbidden to execute α and the system performs α , then we get a violation v .

Usually we have several violation predicates, we can consider composed violation predicates:

$$V = *v_1 \wedge *v_2 \wedge \dots \wedge *v_n$$

where $*$ is \neg or blank.

Violations II

For each composed violation predicate V , we have

- $\mathcal{U}(V) = \{v_i \mid v_i \text{ does not appear negated in } V\}$.
- $V \rightarrow [\alpha]V'$, if $\mathcal{U}(V) \subset \mathcal{U}(V')$.

*When we are in a state where the violation state V is true, executing α adds more violations into the state. (α is a **degrading** action.)*

- $V' \rightarrow [\alpha]V$, if $\mathcal{U}(V) \subset \mathcal{U}(V')$.

*When we are in a state where the violation state V' is true, executing α removes some violations from the state. (α is an **upgrading** action.)*

Example: Trains

Consider a simple train system; we have n trains and m rail segments. Rail segments have a signal which indicates if the segment is occupied or not. We have the following propositions:

- $t_i.r_j$: *Train i is in the segment j .*
- $r.j.green$: *The signal of segment j is green.*
- $t_i.stop$: *Train t_i is stopped.*
- $r_i.Rr_j$: *segments i and j are connected.*
- v_j : a violation occurs in segment v_j when two trains are in the same segment.

Trains II

We have the following actions:

- $t_i.move(j)$: train t_i moves to segment j .
- $t_i.stops$: train t_i stops.
- $r_i.ggreen$: the signal of r_i is set to green.
- $r_i.gred$: the signal of r_i is set to red.

We consider three versions of deontic predicates for each train and segment.

Examples of Axioms

$$\bullet \left(\bigvee_{1 \leq i, j \leq n \wedge i \neq j} t_i.r_k \wedge t_j.r_k \right) \leftrightarrow \mathbf{v}_k.$$

If we have two trains in the same segment, then we have a violation in this segment.

$$\bullet t_i.r_j \rightarrow r_j.\mathbf{O}^1(\text{gred}).$$

When there is a train in a segment, the signal for this segment must be red.

$$\bullet \left(\bigwedge_{1 \leq i \leq n} \neg t_i.r_j \right) \rightarrow r_j.\mathbf{O}^2(\text{ggreen})$$

If there is no train in the segment, then the signal for the segment must be green.

More axioms

- $\neg t_i.r_k \wedge \neg r_k.green \rightarrow t_i.F^1(move(k))$

If the signal for a segment is red, then any train is forbidden to move into the segment.

- $t_j.F^1(move(k)) \rightarrow t_j.O^2(\overline{move(k)} \sqcup stop)$

This axiom formalizes a contrary-to-duty statement: *if you are forbidden to move to rail r_k , and you do it, you have to stop the train.*

More Axioms

$$\bullet t_i.r_j \wedge r_j.v \rightarrow t_i.O^2(stop)$$

If a train detects that another train is already in the same segment, then it ought to stop to avoid train collisions.

$$\bullet t_i.r_k \wedge \left(\bigwedge_{1 \leq j \leq m} (\langle t_i.move(j) \rangle \top \rightarrow F^1(t_i.move(j))) \right) \rightarrow t_i.O^3(stop)$$

When a train is in a segment where all the connected segments have their signals set to red, the train is obliged to stop.

Properties of the Specification

We define:

$$\Phi_1 = \{\text{AG}(t_i.\text{O}^2(\text{stop}) \rightarrow \text{ANdone}(t_i.\text{stop})) \mid 1 \leq i \leq n\}.$$

Trains fulfil the obligations of type 2.

$$\Phi_2 = \{\text{AG}(r_i.\text{O}^1(\text{gred}) \rightarrow \text{ANdone}(r_i.\text{gred})) \mid 1 \leq j \leq m\}.$$

The signals in the segments work correctly.

Using these formulae we can prove the following property:

$$\Phi_1, \Phi_2 \vdash_{\text{Train}} \neg(t_i.r_k \wedge t_j.r_k)$$

When trains fulfil their obligations of stopping at a red signal and rails fulfil the obligation of setting their signal to red, then we cannot have two trains in the same segment.

Properties II

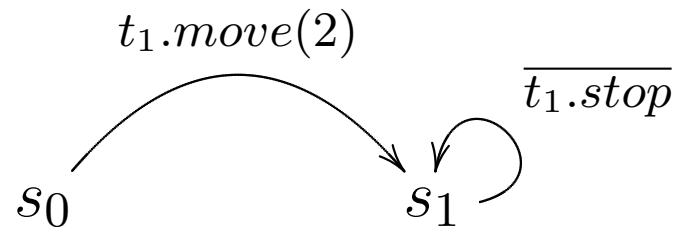
When the obligations of type 2 are fulfilled, then when we have two trains in a segment, both will stop. The property can be stated as follows:

$$t_i.O^3(stop) \rightarrow \text{ANdone}(t_i.stop) \vdash_{\text{Train}} t_i.r_k \wedge t_j.r_k \rightarrow \text{AN}(t_i.stop \wedge t_j.stop).$$

Note that trains which fulfil the contrary-to-duty statement will stop immediately after passing a red signal.

A Model

In the following model the contrary-to-duty statement is violated, and obligations of type 2 are not fulfilled; as a consequence we have a risk of collision.



We have two trains, t_1 is in segment r_1 and t_2 is in segment r_2 . Segments r_1 and r_2 are connected. Since segment r_2 is occupied, t_1 is forbidden to move to that segment, but if it moves, then it must stop. The train moves to that segment and it does not stop, i.e., the obligation of type 2 is not fulfilled. We could have a train collision.

Conclusions and further work

We note the following features of the formalism that we developed.

- We have presented a complete and sound deontic logic.
- The logic is decidable.
- We want to extend this logic with first-order operators, but this is not trivial if compactness (strong completeness) is to be preserved.
- Obligations are useful to express the concept of ideal or correct action.
- Stratified norms allow us to formalize contrary-to-duty statements which arise in fault-tolerance; we have developed some other examples: dining philosophers, byzantine generals and the Muller C-element.

References

- [1] P.F.Castro and T.S.E.Maibaum. *An Ought-to-do logic to reason about fault-tolerance: The diarrheic philosophers*. Conference in Software Engineering and Formal Methods, IEEE, London, 2007.
- [2] D.Harel, D.Kozen and J.Tiuryn. *Dynamic Logic*. MIT Press, 2000.
- [3] J.J.Meyer. *A Different Approach to Deontic Logic: Deontic Logic Viewed as a Variant of Dynamic Logic* Notre Dame Journal of Formal Logic.
- [4] T.Maibaum *Temporal Reasoning over Deontic Specifications*. Deontic Logic in Computer Science. John Wiley & Sons. 1993.
- [5] T.Maibaum, S.Khosla. *The Prescription and Description of State-Based Systems*. In B.Banieqbal, H.Barringer and A.Pnueli (eds). Temporal Logic in Computation, LNCS, Springer-Verlag. 1985.
- [6] P.BlackBurn, M.de Rijke and Y. de Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science 53, 2001.