Deontic Logic

Introduction and Application in Computer Science

Morteza Amini

m_amini@ce.sharif.edu

Network Security Center (NSC)
Department of Computer Engineering
Sharif University of Technology, Tehran, Iran

IPM Workshop on Modal Logics & Application in Computer Science (March 2010)

Outline

1 Introduction

2 Two Types of Deontic Logic

3 Paradoxes of Standard Deontic Logic

4 Applications in Computer Science

5 Summary

Outline

1 Introduction

2 Two Types of Deontic Logic

3 Paradoxes of Standard Deontic Logic

4 Applications in Computer Science

5 Summary

History

Ernst Mally (1926) was the first to try of formalize Deontic notions. But, in his system: $\vdash p \equiv \mathtt{OB}p$

Von Wright (1951), Castaneda (1981), and Alchourron (1971) developed deontic logic by extending modal logic with operators for permission, obligation and prohibition.

Different kinds of deontic logics have been developed, many of them are modal logics.

Definition

Narrow and Wide Definitions

- [Narrow Def.] Deontic logic is the logic of obligation, permission, and prohibition.
- [Wide Def.] Deontic logic is a symbolic logic concerned with the logic of normative expressions: a systematic study of the contribution these expressions make to what follows from what.

From Application View

[Wieringa, Meyer 93] Deontic logic is the logic that deals with actual as well as ideal behavior of systems.

Definition

More Precise Definition

Deontic logic is that branch of symbolic logic that has been the most concerned with the contribution that the following notions make to what follows from what:

- permissible (permitted)
- impermissible (forbidden, prohibited)
- obligatory (duty, required)
- gratuitous (non-obligatory)
- optional
- ought

Deontic Statuses

Five (deontic) normative statuses:

OB	it is obligatory that
PE	it is permissible that
IM	it is impermissible that
GR	it is gratuitous that
OP	it is optional that

Outline

1 Introduction

2 Two Types of Deontic Logic

3 Paradoxes of Standard Deontic Logic

4 Applications in Computer Science

5 Summary

Prescription vs. Description

Prescription

- For giving norms. What the system should do.
- Example: “you may not park your car this side of the street”.
- Used prescriptively imposes a prohibition.

Description

- For Stating that a norm to such and such effect has been given (exist). What the system does.
- The same as the above example.
- Used descriptively to give information about parking regulations.

Prescription vs. Description

The Logical Acceptable View

- We have norm against norm-proposition.
- Norms, as prescriptions, have no truth-value (true or false).
- It is logical to have *descriptive view*.

In fact, deontic logic is the logic of norm-propositions.

Two Different Views

The variables *p*, *q*, etc. are representations of

1. names of human action
2. sentences describing the state of affairs which can come to obtain as the result of human action (*doable* state of affairs).

Now, what is the meaning of obligation:

1. OB means "ought to do"
2. OB means "ought to be"

Example

Von Wright's example:

1. "one ought to close the window"
2. "the window ought to be closed"

Standard Deontic Logic (SDL)

Based on the first view: "ought to be".

Here, obligation is taken 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: $\mathtt{PE}p \equiv \neg \mathtt{OB} \neg p$.

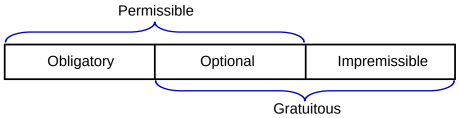
Deontic Statuses in SDL

We take $\mathtt{OB}$ as basic, and define the rest:

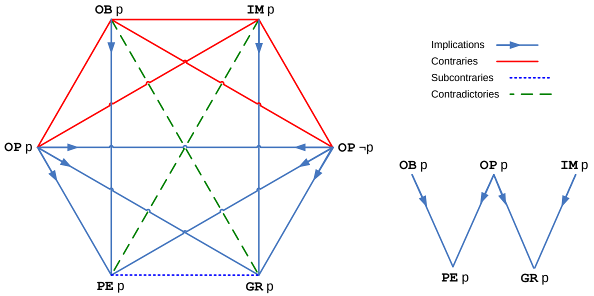
- $\mathtt{PE}p \leftrightarrow \neg\mathtt{OB}\neg p$
- $\mathtt{IM}p \leftrightarrow \mathtt{OB}\neg\mathrm{p}$
- $\mathtt{GR}p \leftrightarrow \neg\mathtt{OB}p$
- $\mathtt{OP}p \leftrightarrow (\neg\mathtt{OB}p \wedge \neg\mathtt{OB}\neg p)$

Deontic Statuses in SDL

Deontic 3-Fold



Deontic Hexagon



Standard Deontic Logic (SDL)

Axiomatic System

A1.	If p is a tautology, then $\vdash p$	(TAUT)
A2.	$\vdash \mathtt{OB}(p \longrightarrow q) \longrightarrow (\mathtt{OB}p \longrightarrow \mathtt{OB}q)$	(OB-K)
A3.	$\vdash \mathtt{OB}p \longrightarrow \neg\mathtt{OB}\neg p$	(OB-D)
A4.	$\vdash \mathtt{PE}p \leftrightarrow \neg\mathtt{OB}\neg p$	(PE-Def)
A5.	$\vdash \mathtt{IM}p \leftrightarrow \mathtt{OB}\neg p$	(IM-Def)
A6.	$\vdash \mathtt{GR}p \leftrightarrow \neg\mathtt{OB}p$	(GR-Def)
R1.	If $\vdash p$ and $\vdash p \longrightarrow q$, then $\vdash q$	(MP)
R2.	If $\vdash p$ then $\vdash \mathtt{OB}p$	(OB-NEC)

Standard Deontic Logic (SDL)

Useful Theorems

- $\vdash \neg \mathtt{OB}\bot$ (OB-OD)
- $\vdash \mathtt{OB}(p \wedge q) \leftrightarrow (\mathtt{OB}p \wedge \mathtt{OB}q)$ (OB-MC)
- $\vdash \mathtt{OB}p \vee \mathtt{OP}p \vee \mathtt{IM}p$ (OB-Exhaustion)
- If $\vdash p \rightarrow q$, then $\vdash \mathtt{OB}p \rightarrow \mathtt{OB}q$ (OB-RM)

Two Principles

Two principles in standard deontic logic:

- $\nvdash \mathtt{OB}p \rightarrow p$ (if it is obligatory that p, then p is true)
- $\nvdash p \rightarrow \mathtt{PE}p$ (if p is true, then it is permissible)

In a real system obligations can be violated, and impermissible things do hold.

Semantics of SDL

Kripke Semantics

$\mathcal{M} = \langle W, R, \Phi \rangle$

- W: possible worlds
- $R \subseteq (W \times W)$: ideal, deontic alternative relation on W
- Φ: maps each *formula* to a subset of possible worlds where the formula is correct.
  - $\phi = Propositions \rightarrow 2^W$
  - $\Phi(p) = \phi(p)$, if p is a proposistion
  - $\Phi(\neg\alpha) = W - \Phi(\alpha)$
  - $\Phi(\alpha \wedge \alpha') = \Phi(\alpha) \cap \Phi(\alpha')$
  - $\Phi(\mathtt{OB}\alpha) = \{w | R(w) \subseteq \Phi(\alpha)\}$
  - $\Phi(\mathtt{PE}\alpha) = \{w | R(w) \cap \Phi(\alpha) \neq \varnothing\}$
  - $\Phi(\mathtt{IM}\alpha) = \{w | R(w) \subseteq \Phi(\neg\alpha)\}$

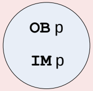
Semantics of SDL

Seriality of Worlds Relations

Definition: relation R is serial if for all $w \in W$, exists a world $w' \in W$ such that $\langle w, w' \rangle \in R$.
It is required for the OB-D axiom to be hold.

Counter Example

Suppose R is not serial.



$\mathtt{OB}p \rightarrow \mathtt{PE}p$ does not hold !

Dynamic Deontic Logic (DDL)

- Based on the second view: "ought to do".
- We need to have a logic of actions as a basis.
- The logic called actions logic or dynamic logic.
- Dynamic in contrary to state of affairs which are static.

Dynamic Deontic Logic (DDL)

J.J.Meyer and Maibaum proposed dynamic deontic logics or deontic action logics, where:

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

Axiomatic System

A1. If p is a tautology, then $\vdash p$

A2. $\vdash [\alpha](\phi_1 \rightarrow \phi_2) \rightarrow ([\alpha]\phi_1 \rightarrow [\alpha]\phi_2)$

A3. $[\alpha_1; \alpha_2]\phi \equiv [\alpha_1]([\alpha_2]\phi)$

A4. $\vdash [\alpha_1 \cup \alpha_2]\phi \equiv [\alpha_1]\phi \wedge [\alpha_2]\phi$

A5. $\vdash [\alpha_1 \& \alpha_2]\phi \leftarrow [\alpha_1]\phi \vee [\alpha_2]\phi$

A6. $\vdash [\phi_1 \rightarrow \alpha_1/\alpha_2]\phi_2 \equiv (\phi_1 \rightarrow [\alpha_1]\phi_2) \wedge (\neg\phi_1 \rightarrow [\alpha_2]\phi_2)$

A7. $\langle\alpha\rangle \equiv \neg[\alpha]\neg\phi$

A8. $[\overline{\alpha_1; \alpha_2}]\phi \equiv [\overline{\alpha_1}]\phi \wedge [\alpha_1][\overline{\alpha_2}]\phi$

A9. $[\overline{\alpha_1 \& \alpha_2}]\phi \leftarrow [\overline{\alpha_1}]\phi \vee [\overline{\alpha_2}]\phi$

A10. $[\overline{\alpha_1 \& \alpha_2}]\phi \equiv [\overline{\alpha_1}]\phi \wedge [\overline{\alpha_2}]\phi$

Axiomatic System

A11. $[\overline{\phi_1 \to \alpha_1/\alpha_2}]\phi_2 \equiv (\phi_1 \to [\overline{\alpha_1}]\phi_2) \wedge (\neg\phi_1 \to [\overline{\alpha_2}]\phi_2)$

A12. $[\overline{\overline{\alpha}}]\phi \equiv [\alpha]\phi$

A13. $[\varnothing]\phi$

R1. If $\vdash \phi$ and $\vdash \phi \to \psi$, then $\vdash \psi$

R2. If $\vdash \phi$ then $\vdash [\alpha]\phi$

Since DDL is not a kind of modal logic and also complicated, we do not talk more.

Outline

1 Introduction

2 Two Types of Deontic Logic

3 Paradoxes of Standard Deontic Logic

4 Applications in Computer Science

5 Summary

Paradoxes of Standard Deontic Logic

Meaning of Paradox

- Does not mean the inconsistency of the proposed logics.
- Mentions the difference between what is required in practice (in law, ethics, etc.) and what we obtain using a formal logic.

Different Types of Paradoxes

- Puzzles Centering Around RM
- Puzzles Centering Around NC, OD and Analogous
- Puzzles Centering Around Deontic Conditionals
- Problems Surrounding (Normative) Expressive Inadequacies
- Challenges regarding Obligation, Change and Time

Ross's Paradox (Ross 1941)

- Consider:
  1. It is obligatory that the letter is mailed.
  2. It is obligatory that the letter is mailed or the letter is burned.
- In SDL, expressible as:
  1. $\mathtt{OB}m$
  2. $\mathtt{OB}(m \vee b)$
- In SDL, $\vdash \mathtt{OB}m \rightarrow \mathtt{OB}(m \vee b)$ follows from $\vdash m \rightarrow (m \vee b)$.
- So (2) follows from (1).
- But it seems rather odd to say that
  an obligation to mail the letter entails an obligation that can be fulfilled by burning the letter.

Free Choice Permission Paradox (Ross 1941)

- Consider:
  1. You may either sleep on the sofa-bed or sleep on the guest room bed.
  2. You may sleep on the sofa-bed and you may sleep on the guest room bed.
- In SDL, expressible as:
  1. $\mathrm{PE}(s \vee g)$
  2. $\mathrm{PE}s \wedge \mathrm{PE}g$
- It is natural to say: (2) as following from (1).

Free Choice Permission Paradox (Ross 1941)

- Suppose $\vdash \mathtt{PE}(p \vee q) \rightarrow (\mathtt{PE}p \wedge \mathtt{PE}q)$ were added to a system that contained SDL.
- So, we get $\mathtt{PE}p \rightarrow (\mathtt{PE}p \wedge \mathtt{PE}q)$, and for any q, we would get $\mathtt{PE}p \rightarrow \mathtt{PE}q$
- that means if anything is permissible, then everything is,
- and thus nothing is obligatory, $\vdash \neg\mathtt{OB}p$.

Sartre's Dilemma (Lemmon 1962)

- Consider the following conflict:
  1. It is obligatory that I now meet Jones (say, as promised to Jones, my friend).
  2. It is obligatory that I now do not meet Jones (say, as promised to Smith, another friend).
- In SDL, expressible as:
  1. $\mathtt{OB}j$
  2. $\mathtt{OB}\neg j$

- By NC, $\mathtt{OB}p \rightarrow \neg\mathtt{OB}\neg p$, they seems inconsistent and we have a conflict of obligations.
- In practice, people do make such conflicting promises.
- Logically it is an inconsistent situation; however, the original hardly seems logically incoherent.

Plato's Dilemma (Lemmon 1962)

- Consider:
  1. I'm obligated to meet you for a light lunch at noon.
  2. I'm obligated to rush my choking child to the hospital at noon.
- We have an indirect, non-explicit conflict of obligations, which is not recognized by SDL.

- Obligation (2) overrides obligation (1).
- Need to have conflicting obligations of different weight and the defeasibility of obligations.
- SDL does not allow for conflicts to begin.

Responses to the Paradoxes

Some Responses

- Reject OB-RM, by changing the proof theory and semantics.
- Should limit ourselves to the expressive power and interpretations considered in SDL.
- They are not acceptable in computer science and just related to law and ethics.
- Different interpretations for conditional ought.
- Proposing non-monotonic versions of deontic logic.
- Considering change and time.
- ...

Outline

1 Introduction

2 Two Types of Deontic Logic

3 Paradoxes of Standard Deontic Logic

4 Applications in Computer Science

5 Summary

Which Systems?

Which Systems?

Any system where we want to reason about ideal as well as actual behavior of the system.

Applications in Computer Science

Some applications of deontic logic in computer science:

- the specification of the normative behavior of the object system (e.g., legal automation)
- security (e.g., policy specification and authorization),
- system specification (e.g., specification of fault-tolerant systems),
- database constraints (e.g., integrity constraints)

Legal Automation

Formal logic can be used to identify ambiguities in legislation and to draw logical consequences from legal rules.
Two approaches:

- Factual approach: there is no distinction between actuality and ideality.
- Deontic approach: there is such a distinction.

Legal Automation - Factual Approach

The Approach

Legislation is viewed as a set of definitions or rules.
Disadvantages:

- Not being able to consistently express violations of these definitions,
  (e.g., $\texttt{OB p} \wedge p$ can not be stated here)
- and relations between permissions, prohibitions, and obligations are eliminated.

Legal Automation - Deontic Approach

The Approach

legislation is viewed as a set of obligations, permissions, and prohibitions issued by authorities.

Example: Language for Legal Discourse (LLD): one can specify the rule that any corporation that owns cash has an obligation to distribute cash to all its stockholders.

```
(obligate ?
        (own ?  (corporation ?X) (cash ?Y))
        (distribute-dividend ?)  (corporation ?X)))
```

Disadvantages:

- the paradoxes mentioned for deontic logic,

Security Policy Specification & Authorization

- Expression of confidentiality policies
- Access policies in emerge with obligation policies
- Analyzing consistency of security policies
- Composition of security policies

Expression of Confidentiality Policies

Deontic logic in combination with epistemic logic and temporal logic.

Confidentiality is defined as:

1. $K_{A,\tau}\phi \rightarrow R_{A,\tau}\phi$
   If A knows ϕ at time τ, then A has permission, at time τ, to know ϕ .
2. $K_{A,\tau}\phi \rightarrow O_{A,\tau}\phi$
   If ..., then A has the obligation, ...
3. $K_{A,\tau}\phi \rightarrow F_{A,\tau}\phi$
   If ..., then A is forbidden, ...

A is a subject or role.

Access & Obligation Policies

- Need to have obligations and specify them.
- Example: ((give a simple example, *begin transaction* must be ended with *commit* or *rollback*))
- The relationships between permissions, obligations, and prohibitions are important.
- How can we oblige something without getting permission.

- The OB-D axiom is so important here!
  $\mathtt{OB}p \longrightarrow \mathtt{PE}p$
- Also, $\mathtt{IM}p = \neg\mathtt{PE}p$, and $\mathtt{IM}p = \mathtt{OB}\neg p$.
- Now, we can check the consistency of security policy rules.

Fault Tolerant Systems

Maibaum and Castro

- Deontic logics allow us to distinguish between normal and abnormal situations.
- we want to specify what should happen in the case of violations of normal computer behavior.
- Some benefits of deontic logics are:
  1. A language to express normative reasoning (permission, obligation, forbidden).
  2. A natural level of abstraction in semantic structures (states are divided into good or bad ones).
  3. It is easy to mix it with temporal logics, and therefore to gain the good properties of temporal frameworks to verify systems.

Fault Tolerant Systems

Contrary-to-Duty Paradoxes

Example (the gentle killer paradox):

1. You ought not to kill.
2. If you kill, you ought to kill gently.
3. You kill.

This set of sentences is inconsistent in SDL.

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

Fault Tolerant Systems

- Maibaum and Castro define an action α as a set of events.
- They use two versions of permission:
  - $\mathtt{PE}(\alpha)$, strong permission. α is executable in all (local) contexts.
  - $\mathtt{PE}_w(\alpha)$, weak permission. α can only be executed in some (local) contexts.
  - $\mathtt{OB}(\alpha) \equiv \mathtt{PE}(\alpha) \wedge \neg\mathtt{PE}_w(\overline{\alpha})$.

- The definition of obligation avoids some paradoxes such as Ross's paradox.
- There exists a strong connection between the two versions of permission:
  - $\mathtt{PE}(\alpha) \wedge \alpha \neq \varnothing \rightarrow \mathtt{PE}_w(\alpha)$.
  - $\neg\mathtt{PE}(\alpha)$ implies that for some $\alpha' \sqsubseteq \alpha : \neg\mathtt{PE}_w(\alpha')$

Integrity Constraints for Databases

Two Types of Constraints

1. Necessary constraints: the constraints cannot be violated by the real world.
   - example: the age of a person cannot be negative.
2. Deontic constraints: the constraints that could be violated, such as integrity constraints for databases.
   - example: a person cannot have an age over 150.
   - example: no one can be fired before he is hired.

Integrity Constraints for Databases

Examples

1. $borrow(p, b) \rightarrow \mathtt{OB}(return(p, b)_{(\leq 21d)})$
   After p borrows a book b, an obligation exists to return the book within 21 days.
2. $V : return(p, b) \rightarrow \mathtt{OB}(pay(p, \$2, b))$
   Whenever the violation flag V is raised, there is an obligation on p to pay two dollars.
3. $[return(p, b)]\neg V : return(p, b)$
   Returning the book lowers the violation flag.

Classification of Applications in C.S.

Classification based on the domain whose behavior is specified in deontic logic:

1. Fault-tolerant computer systems.
2. Normative user behavior.
3. Normative behavior in or of the organization.
    - Policy specification.
    - Normative organization behavior (e.g. contracting).
4. Normative behavior of the object system.
    - The specification of law.
    - The specification of legal thinking.
    - The specification of normative rules as deontic integrity constraints.
    - Other applications, not discussed above, e.g., scheduling problems.

Outline

1 Introduction

2 Two Types of Deontic Logic

3 Paradoxes of Standard Deontic Logic

4 Applications in Computer Science

5 Summary

Summary

- Deontic logic is the logic that deals with actual as well as ideal behavior of systems.
- Descriptive approach not prescriptive.
- Two Types of deontic logic:
  - based on "ought to be" approach (e.g., Standard Deontic Logic)
  - based on "ought to do" approach (e.g., Dynamic Deontic Logic)
- Paradoxes of SDL: does not spoil it usage in computer science.
- Applications in computer science: specification of fault tolerant systems, policy specification and authorization, legal automation, and deontic integrity constraints for databases.

Useful References

The Standford Encyclopedia Entry on Deontic Logic

http://www.science.uva.nl/~seop/entries/logic-deontic

ΔEON Conference: 10th International Conference on Deontic Logic in Computer Science

http://www.defeasible.org/deon2010

Deontic Logic Wiki!

https://deonticlogic.uni.lu/deon/index.php/Main_Page

Thanks for your attention ...

Questions ?