"Deontic Logic"

By Thomas Harms

March, 20th 2020

Contents

1 Abstract **2**

2 Introduction **3**
2.1 Standard Deontic Logic 3
2.2 Weaknesses of SDL 4

3 Notions of Normative Ideality and Sub-Ideality **5**
3.1 DL 5
3.2 DL* 6

4 DEON+ **7**
4.1 ALP 7
4.2 CLP 8
4.3 DEON+ 8
4.3.1 Syntax 8
4.3.2 Conditional Obligations and Deadlines 10

5 Conclusion **11**

6 Bibliography **12**

1 Abstract

Laws and norms are implemented to specify desired behaviour of a member being part of a contract. This includes human as well as articial beings of a society as agents or group of agents. Therefor deontic concepts and operators have been widely used in order to represent norms in legal reasoning and normative multi-agents systems. Since basic deontic formalization such as *Standard Deontic Logic (SDL)* is easy to implement as well as expressive enough to represent simple normative cases, SDL suffers from a lack of expressiveness to tackle several weaknesses of logic paradoxes.
In order to tackle these repurcussions this paper summarizes two notions dealing with "*contrary-to-duty*" (**CTD**) norms as well as well understood logic paradoxes. The first approach focuses on distinguishing normative ideality and sub-ideality. Two extensions of SDL, namely *DL* and *DL**, will be introduced in order to provide more expressiveness dealing with CTD norms and paradoxes while not becoming too expressive inviting different problems.
On top of SDL, the modal structure of its operators fits smoothly into abductive semantics and abductive reasoning.
This paper builds upon former works, where mappings of the most common deontic operators (obligation, prohibition, permission) to the abductive expectations of an ALP framework for agent societies has been proposed. This mapping was supported by showing a correspondence between declarative semantics of abductive expectations and Kripke semantics for deontic operators. Building upon these concepts, *DEON+* as proposed in [2] offers a language where the two basic deontic operators are enriched with quantification over time, by means of ALP and Constraint Logic Programming (CLP).

2 Introduction

Norms represent desireable behavior of agents of a society. These define expectations of human interaction. Recently concerns have been risen to extend the existing norm set of human societies towards frameworks of human and aritficial behaviors. A normative system for that matter is a set of norms combined with mechanisms to reason about, modify and apply them [2]. One of the major premises is the focus of providing agents autonomy. Normative systems have been advocated as a tool to regulate interaction in multi-agent systems, so agents might comply norms or not [11]. Norms are often encoded as formulas in some logic language build around notions of obligation, permission, prohibition, optionality, omissibility. One possible way to implement these notions would be by the traditional means of *Deontic Logic* [10] [1]. One of the main issues of choosing a proper framework in order to express legal and normative concerns is finding a good balance between the expressivity of logical languages and the efficiency of theorem provers. The implied concern of choosing a reasonable expressive logic language is the overhead of problems being shown which need to be handled becoming too expressive or not expressive enough to represent the legal concerns. One has to look for some logical framework that is expressive enough to capture the fundamental aspects of the problems of legl reasoning to be addressed and that behaves within the expectations of the chosen system of automation [3]. A very simple logical system dealing with normative concepts by its deontic modalities is *Standard Deontic Logic* **SDL** [1]. This system can be easily encoded in theorem provers but is affected by seemingly unsurmountable difficulties in representing very common normative scenarios, such as those in which a contrary-to-duty norm applies [10].

2.1 Standard Deontic Logic

Standard Deontic Logic **SDL** is a very easy to use and implement logic language expressing legal and/or notions. SDL is a monadic modal language build around the concepts of obligation, permission, option, impermissibility and omission. The basic deontic axioms are K, D and TAUT. There are some contraversies about including different axioms. Most commonly SDL is often axiomatized as follows [1]:

- A1. All tautologous wffs of the language (TAUT)
- A2. $\mathbf{OB}(p \rightarrow q) \rightarrow (\mathbf{OB}p \rightarrow \mathbf{OB}q)$ (**OB**-K)
- A3. $\mathbf{OB}p \rightarrow \neg\mathbf{OB}\neg p$ (**OB**-D)
- R1. If $\vdash p$ and $\vdash q \rightarrow \vdash q$ then $\vdash q$ (MP)
- R2. If $\vdash p$ then $\vdash \mathbf{OB}p$ (**OB**-NEC)

TAUT is standard for normal modal systems including all tautologous well formed formulas of the language, which intuitively makes sense. OB-K is the K axiom present in all normal modal logics, tells us that if a material conditional is obligatory, and its antecedent is obligatory, then so is its consequent [1]. OB-D states that p is obligatory only if its negation isn't. MP is just Modus Ponens, expressing that if a material conditional and its antecedent are theorems, then so is the consequent. TAUT combined with MP gives us the full inferential power of the Propositional Calculus [1]. OB-NEC describes that if anything is a theorem, then the claim that that thing is obligatory is also a theorem [1].

2.2 Weaknesses of SDL

As described SDL is an easy to implement logical framework with some expressive power. However, SDL is suffers from several well understood and formulated weaknesses which one has to consider if SDL in its basic form is to be applied to a set of norms. First to mention are **CTD** norms as described in [9]. The problem arises if one obligation is contrary to another obligation of the normative framework or a situation requires agent(s) to break at least one obligation. One of the most famous representatives of CTD norms would be **Chisholm's Paradox** as shown in [8]. The argument runs as follows:

1. it ought to be that Alice helps her neighbors
2. it ought to be that if Alice helps her neighbours, she tells them that she is coming
3. if Alice does not help, she ought not to tell them that she is coming
4. Alice does not help her neighbours

Obviously 1 is contradicting 4. Any plausible formalization in SDL turns out to be either inconsistent or not logically independent. This state of affairs is very undesirable.
Another problem of SDL arises by runing into scenarios spawning "Ross's Paradox". Since $(OB)A \rightarrow (OB)(A \vee B)$ is a provable schema in SDL the following example would be expressable in SDL:

- $(OB)A$: Bob ought to talk to his wife.
- $(OB)(A \vee B)$: Bob ought to either talk to his wife or kill his wife.

In a formalization of a normative system the inference is at least difficult to justify.
Among many more the last weakness of SDL to be mentioned shall be Paradox of Kant´s Law. The premise is focused upon the notion that obligation implies ability. Anything an agent is supposed to do, it must be able to do. The following example sketch the paradox:

1. If you borrow my car, you are obligated to give it back.
2. You crashed my car.

SDL is not expressive enough to handle the difference between a obligation of an agent to do something and dealing with the inablity to oblige.
On a side note worth mentioning is the Free Choice Permission Paradox, which is also a weakness of SDL´s lack of expressiveness.

3 Notions of Normative Ideality and Sub-Ideality

As [12] is claim, many issues arise from the interpretation of the operator **O**. The semantic intuition associated with a formula of any kind $O\ A$ in SDL is that A is true in all *normatively ideal circumstances* (or possible worlds). Namely in all those circumstances, in which *every* prescription is observed. The majority of scentences describing propositions true in all normatively ideal circumstances or not particularly normatively relevant. Especially focusing on tautologous wffs such as "it either rains or does not rain" has no normative relevance. Thus, in order to formally capture the meaning of 'ought'-sentences, one has to take into account "*criteria of normative relevance for sentences*".
Jones and Pörn made in [12] a proposal in requiring an 'ought'-sentence to describe a proposition which not only holds in all normatively ideal circumstances, but also fails in some normatively *sub-ideal circumstance*. Those circumstances are these, in which *not every* prescription is observed. Concerning before pictured Chisholm´s paradox, if Alice ought to help her neighbors, then this is supposed to happen in all normatively ideal scenatios, but fails to be the case in some normatively sub-ideal scenario.

3.1 DL

On the practical side, Jones and Pörn propose to extend the language SDL with an operator *O'* in order to distinguish between normative ideality and sub-ideality. The formula *O' A* means that A is true in all normatively sub-ideal worlds. Then, they propose the following formalization of 'ought'-sentences:
$Ought(\mathrm{A}) =_{\mathrm{def}} OA \wedge \neg O'A$. The resulting system by adding O' is called **DL** and represents a bimodal version of SDL supplemented with the axiom-schema (O A $\wedge$ O' A) $\rightarrow$ A.

The formalization of frames to interpret **DL** is of kind $F = \langle W, R_O, R_{O'} \rangle$, where W is a domain of worlds and R_O and $R_{O'}$ are binary relations over W satisfying the following properties:

1. $\forall w \in W$, there are $v, u \in W s.t. w R_O v$ and $w R_{O'} u$
2. $\forall w \in W, R_O(w) \cap R_{O'}(w) = \emptyset$
3. $\forall w \in W$, either $w \in R_O(w)$ or$w \in R_{O'}(w)$

Property 1 can be captured already by the axiomatic basis of bimodal SDL, property 2 requires further discussion that will be provided below and property 3 can be captured only if one extends the axiomatic basis of bimodal SDL with a schema such as $(OA \wedge O'A) \rightarrow A$. Despite its broader expressive power, DL still encounters some obstacles in dealing with contrary-to-duty obligations. It gives rise to a 'pragmatic oddity' when the sentences (1)-(4) above are formalized as suggested by Jones and Pörn, namely (taking P to be 'Alice helps her neighbors' and Q to be 'Alice tells her neighbors that she is coming'):

1. *Ought(P)*
2. $O(P \rightarrow$ *Ought(Q)*$) \wedge O'(P \rightarrow$ *Ought(Q)*$)$

3. $O(\neg P \rightarrow$ The oddity is hidden in the fact that in DL 1-4 entail both $O\ P$ and $O\neg Q$, which means that in all normatively ideal worlds Alice helps her neighbors without telling them that she is coming. Furthermore some inxtances of paradoxes of deontic reasoning still hold in DL, such as the following verion of Ross´s paradox [3]: if you neither return my car nor crash it, while you ought to return it, then you ought to return it or crash it. The underlying schema $(\neg A \wedge \neg B) \rightarrow (Ought(A) \rightarrow Ought(A \vee B))$ is provable in DL. In order to remedy the latter problem by preventing it to be provable, [13] offers a solution by extending DL into the following system.

3.2 DL*

DL* can be understood as a modification of DL by replacing the ooperator *Ought* by *Ought**, such that *Ought**(A) $=_{\text{def}} OA \wedge O'\neg A$. This way the problematic section is no longer provable. However, there is another concern which has to be analyzed: the fundamental aspect of *Ought**. Since *Ought** is true at a world only if $O'\neg A$ is true there, then the meaning of O' proposed by Jones and Pörn needs to be revised. In any other way every prescription would be violated in all sub-ideal worlds, which is implausible, since world can be calssified as normatively sub-ideal even if *some but not all* prescriptions are violated there. Therefor if one want to keep using the *Ought**, the meaning of O' A should be altered towards A holds in all *normatively aweful worlds.* This provides the opportunity to get rid of the schema $(OA \wedge O'A) \rightarrow A$, as well as frame condition 3 associated with it. A current world might not turn out to be neither

normatively ideal nor normatively awful.
DL* will be denoted as the logic resulting from DL by removing the forementioned axiom and adding the definition of *Ought**. The 'pragmatic oddity' of Chisholm´s paradox described may affect the formalization of the given example even if *Ought* is replaced by *Ought**. For instance in [13] is an argument for the absence of property 3, the formalization of sentences (2) and (3) needs the following revision in order to allow for the detachment of $Ought * \neg Q$, which is an intended consequence of the scenario (Alice ought not to tell her neighbors that she is coming, since she decided not to help them). A formalized approach is described in [3].

4 DEON+

Several extensions to Standard Deontic Logic (SDL) have been proposed, with the aim of capturing different notions of obligatoriness or permissibility, such as conditional obligations, obligations with deadlines, etc. This motivated, for instance, the introduction of dyadic deontic operators, in dyadic deontic logic, such as: **Obl**(A|B) to be read as *it is obligatory that A, given B*. The modal nature of deontic operators smoothly fits into abductive semantics of the specified formulas, where the basic deontic operators are represented as abducible literals. Complementary to abduction, the power of constraint processing can be exploited to model time and a variety of constraints upon the chosen time structure [2].
In this section Deon+ will be introduced, a language where the already announced deontic operators are enriched with a time structure and quantification over time. This allows to state, that an action must be done once over time (e.g. "It is mandatory that John answers me", with an existential flavor for the deontic obligation operator), or it must be always done over time (e.g., "It is mandatory that John drives carefully", with a universal flavor over time for the deontic obligation operator). In Deon+, the basic deontic operators, namely obligation and prohibition, can be applied to actions as it is usual in SDL, but also taking into account different flavors for them over time, i.e., existential or universal. [2]
Deon+ is grounded on *Abductive Logic Programming* (**ALP**) and *Constraint Logic Programming* (**CLP**), by representing the basic deontic operators as *abducible literals* and by mapping time variables into CLP variables, possibly constrained in order to represent deadlines, quantifier restrictions, etc. [2] At the beginning basic notins of ALP and CLP will be quickly introduced in order to provide a foundation to build upon.

4.1 ALP

Abductive Logic Programming (**ALP**) [2] is a set of programming languages deriving from Logic Programming. In an abductive logic

program, a distinguished set of predicates, called *abducibles*, do not have a definition, but their truth value can be assumed. A set of formulae, called *Integrity Constraints* (**IC**) restrict the set of hypotheses that can be made, in order to avoid unrealistic hypotheses to focus on the wanted scenarios to be modeled [2]. ALP supports hypothetical reasoning and a simple and sound implementation of negation by failure (or *negation by default*) which is handy in many applications: e.g. by applying abductive reasoning to the famous Event Calculus, one is able to simply solve planning problems.

4.2 CLP

Constraint Logic Programming (**CLP**) is a class of logic programming languages that allows the usage of atoms and terms that are interpreted in a given theory, defined externally to the program itself. Popular instances are CLP(), on the real numbers, and CLP(FD), on the Finite Domains. Many abductive proof-procedures are integrated with CLP, as researchers discoverd that CLP was able both to extend the expressitivity of ALP in practical application and formalization of real life problems and helped reducing the search space.

4.3 DEON+

As defined in [2]: A Deon+ specification is given in terms of an *Abductive Logic Program*, enriched with *Constraint Logic Programming* variables, with explicit existential or universal quantification.

4.3.1 Syntax

The syntax is build upon an action language, where positive actions are represented by terms, e.g. *answer/2, smoke/1, use/2.* In logic programming terms can contain variables, constants, terms. For instance, the atom *answer(john,me)* stands for the action of answering to me, performed by John. Negative actions are represented with *not(Action).* For instance, the term *not(respect(john, speed limit))* represents the action, performed by John, of not complying to the speed limit.
Obligations are represented as formulas:

$$_{3.}\ obl\ (A, T)$$

where *obl/2* is an abducible predicate, A is any (positive or negative) action description, and T is a CLP variable, (possibly explicitly) existentially or universally quantified. In *Deon+* we can explicitly represent if an obligation is due once or always over time, by specifying the chosen quantification over the time variable. The variable T has to be explicitly quantified. For instance, the sentence "It is mandatory that John answers me", corresponds to:

$$\exists T obl(answer(john, me), T)$$

as any reply at any point in time complies to the obligation. As an example for universal quantification over time "It is mandatory that John respects the speed limits" would mapped into:

$$\forall T obl(respect(john, speedLimit), T)$$

since John is obliged to respect the limit at any time . While he drove carefully one minute ago does not allow him to speed at any other point in time.
Prohibitions are represented as formulas in a similar fashion:

$$forb(A, T)$$

as *forb/2* is an abducible predicate, A is any action description (possibly positive or negative) and T is a CLP variable (possibly explicitly) existentially or universally quantified.
Two examples, one of each possible quantification, run as follows. Any process cannot consume all the CPU time, because that would block the execution of concurrent processes. That means, there must exist some time where a process p does not use the CPU:

$$\exists T forb(use(p, cpu), T).$$

In *Deon+* prohibitions can be raised once or holding always over time. Therefor the sentence "It is forbidden that John smokes" corresponds to:

$$\forall T forb(smoke(john), T)$$

since John is not allowed to smoke at any point in time and the fact that he did not smoke in the past does not allow him to smoke now or in the future.
Note that in *Deon+* quantification might be explicit or implicit. In case there is no explicit quantification an implicit universal quantification takes place. Furthermore, time variables refer to the instant of performance of an action. There are notions of distinguishing between external and the internal time of a norm: the former is the interval in which a norm belongs to the legal system, and the latter is the interval that the norm is applicable to. In *Deon+*, the set of norms does not change over time, and norms are always applicable, so these notions are not relevant and are in the language presented in the section as shown in [2].
Explicit quantification (existantial or universal) can be applied to variables occurring in the action as well. Explicit universal quantification is limited to variables occurring in abducibles. Quantification can be omitted as standard in logic programming, when variables are universally quantified with scope a clause or an implication. For instance, the sentence "It is forbidden to smoke" corresponds to:

$$\forall T, \forall A, forb(smoke(X), T)$$

where X is universally quantified. The adoption of CLP variables for representing time adds expressiveness to deontic operators and easily recovers deadlines by constraints over time variables. A sentence like "It is forbidden that John leaves the meeting before 1" is therefore represented in *Deon+* as:

$$\forall T : T < 10 forb(leave(john, meeting), T)$$

and "It is forbidden that John leaves the meeting before its end" by:

$$\exists T', \forall T < T' forb(leave(john, meeting), T), end(meeting, T').$$

A comprehensive overview of deontic literals is as defined in [2]:

*Deontic Literal ::= [**not**]Deontic Atom*
Deontic Atom ::= obl(Term, Time)
Deontic Atom ::= forb(Term, Time)
Time ::= Variable | Number

Definition: *A Deon+ specification consists of an (abductive) logic programming set of clauses (P), a set of integrity constraints (IC), and a goal (G). A set of meta-level integrity constraints (M) can be added to express or vary the semantics of deontic operators.* [2]
The syntex of P and *IC* runs as [2]:

*P ::= [Clause**
Clause ::= Atom ← QConjunction
*IC ::= [Integrity Constraint]**
QConjunction ::= [ExistentialQ][UniversalQ]*Conjunction*
Conjunction ::= Literal, Conjunction | Literal
ExistentialQ ::= ∃ Variable[: Constraints]
UniversalQ ::= ∀ Variable[: Constraints]
Constraints ::= Constraint | Constraint,Constraints
Literal ::= DeonticLiteral | Constraint | DefinedLiteral
IntegrityConstraint ::= QConjunction → Disjunction.
Disjunction ::= false | QConjunction | QConjunction ∨ Disjunction
G ::= ← QConjunction

where DefinedLiteral is a literal of a defined predicate, and Constraint is a constraint in the chosen CLP language.

4.3.2 Conditional Obligations and Deadlines

Integrity constraints can be also exploited to represent conditional obligatoriness and the deontic logic of deadlines. In order to remedy the forementioned weaknesses of SDL, these concepts are quite powerful in order to balance expressivity vs complexity. For instance, integrity constraints of the kind $B \rightarrow Obl\ A$ are suitable to represent the obligatoriness of A given B, and the dyadic deontic operator *Obl(A|B)*. The power of the dyadic concept and a faithful implementation is shown in [4].
Deontic logic with deadlines and the operator $O(\rho \leq \delta)$, meaning that the action ρ ought to be brought about before or at least at the same time as another action δ happens. This sentence can be mapped into the following integrity constraint:

$$hap(\delta, T_{\delta}) \rightarrow obl(\rho,\ T_{\rho} :\ T_{\rho} \leq T_{\delta})$$

where action occurrence is represented by the first-class predicate *hap/2* relating an action to its occurrence time. The integrity constraint above therefore represents the obligatoriness of action ρ at time $T_{\rho} \leq T_{\delta}$, if δ happens at time T_{δ} [2].
Deon+ can capture of (conditional) obligation with deadline in a metric time setting with an explicit mapping of time and is further described in [2].

5 Conclusion

In this paper we discussed deontic notion and a simple implementation, namely *Standard Deontic Logic* (**SDL**). Since SDL is an easy to implement deontic language, it comes with some weaknesses such as all the repurcussions of *contrary-to-duty* (**CTD**) norms, e.g. Chisholm‘s Paradox, also Ross‘s Paradox, Free Choice Permission Paradox as well as Kant‘s Paradox and more.
By thinking about ways to remedy this state of affairs, one has to consider a fine balance between increase of expressivity of the chosen language and the overhead of complexity, which might bring other more complex problems along. However, there have been two very different approaches shown in the paper:

- Distinguishing Normative Ideality and SubIdeality
- Abductive Logic Programming and Constraint Logic Programming in *Deon+*

The former notion has been explained in two different approaches, by extending SDL towards **DL** by additional deontic operator *O' A* and formalizing "*ought*" sentences with respect to normative relevance. An even further extension by adjusting the axioms of DL, **DL*** has been introduced in order to remedy some oddities of *CTD norms*.
A very different approach has been introduced by formalizing a deontic language build upon an action language with the concepts of *Abductive Logic Programming* **ALP** and *Constraint Logic Programming* **CLP**, namely *Deon+*. The language can be exploitet to draw powerful notions in order to remedy the forementioned problems of SDL: e.g. concept of deadlines and conditional obligatoriness.

6 Bibliography

1 https://plato.stanford.edu/entries/logic-deontic/, 12.01.2020

2 Marco Alberti, Marco Gavanelli, Evelina Lamma: "Deon+: Abduction and Constraints for Normative Reasoning"; Chapter out of: A. Artkis, R. Craven, N Kesim, B. Sadighi, K. Stathis: "logic programs, norms and action, Essys in honor of Marek J. Sergot on the occasion of his 60th birthday", Springer 2012, DOI 10.1007 /978 − 3 − 642 − 29414 − 3_17

3 Tomer Libal, Mateo Pascucci: "Automated Reasoning in Normative Detachment Structures with Ideal Conditions", 10/2018, http://arxiv.org/pdf/1810.09993

4 Christoph Benzmüller, Ali Farjami, Xavier Parent, "A Faithful Semantic Embedding of the Dyadic Deontic Logic in HOL", 2018, https://christoph-benzmueller.de/papers/R59.pdf

5 Giada Maggenti, Andrea Bracciali, Paolo Mancarella: "Abduction and Legal Reasoning", 2008

6 Christoph Benzmüller, Ali Farjami, Xavier Parent: "A Dyadic Deontic Logic in HOL, In Deontic Logic and Normative Systems" — 14th International Conference, DEON 2018, Utrecht, The Netherlands, 3-6 July, 2018 (Jan Broersen, Cleo Condoravdi, Shyam Nair, Gabriella Pigozzi, eds.), College Publications, volume 9706, pp. 33−50, 2018

7 J. Carmo and A.J.I. Jones. "Completeness and decidability results for a logic of contrary-to-duty conditionals." J. Log. Comput., 23(3):585−626, 2013

8 R.M. Chisholm. "Contrary-to-duty imperatives and deontic logic." Analysis, 24:33−36, 1963

9 J. Carmo and A.J.I. Jones. "Deontic logic and contrary-to-duties." In D. M. Gabbayand, F. Guenthner, editors, Handbook of Philosophical Logic: Volume 8, pages 265−343. Springer, Netherlands, Dordrecht, 2002

10 Navarro, P.E. and Rodrguez J.L. (2014) "Deontic logic and legal systems". Cambridge University Press

11 Boella, G., van der Torre, L., Verhagen, H.: "Introduction to normative multiagent systems." Computational and Mathematical Organization Theory 12, 71−79 (2006)

12 Jones, A. and Pörn, I. (1985), "Ideality, sub-ideality and deontic logic." Synthese 65(2), 275−290

13 de Boer, M., Gabbay, D.M., Parent, X. and Slavkovic, M. (2012), "Two dimensional Standard Deontic Logic [including a detailed analysis of the 1985 Jones-Pörn deontic logic system]." Synthese 187(2), 623-660