Deontic Equilibrium Logic with eXplicit negation

Pedro Cabalar[1], Agata Ciabattoni[2], and Leendert van der Torre[3]

[1] University of A Coruña, Spain, cabalar@udc.es
[2] TU Wien, Vienna, Austria, agata@logic.at
[3] University of Luxembourg, Luxembourg, leon.vandertorre@uni.lu

Abstract. Equilibrium logic is a logical characterization of Answer Set Programming (ASP). We introduce *Deontic Equilibrium Logic with eXplicit negation* (DELX), its extension for normative reasoning. In contrast to modal approaches, DELX utilizes a normal form that restricts deontic operators solely to atoms. We establish that any theories in DELX can be reduced to ASP, and demonstrate the efficacy of this minimalist approach in addressing key challenges from the defeasible deontic logic literature.

1 Introduction

Before deploying AI systems in real-world settings, it is imperative that they satisfy legal and ethical requirements. Regulators, researchers and practitioners from various disciplines are providing such requirements, which are typically expressed as norms involving obligations and related concepts. To assess whether AI systems comply with them, we need formal languages to represent norms, and automated reasoning tools to derive conclusions from their representation.

Normative reasoning is the realm of deontic logic, which formalizes obligation and related concepts. While there is consensus on the importance of defeasibility in dealing with norms or on the fact that obligations cannot be defeated by their violations, the specific properties of deontic operators vary depending on the application. This has resulted in the emergence of numerous deontic systems (refer to the handbooks [16,15]) as advancements over the "standard" deontic logic KD [45], which proved inadequate in tackling various scenarios commonly referred to as deontic 'paradoxes'; in particular the necessity operator $\mathbf{O}\varphi$ in KD (read as "φ is obligatory") could not deal with secondary obligations (aka *contrary-to-duty*) or defeasible reasoning, as in the following well-known scenario

Example 1 (Cottage Fence [41]). The scenario consists of the norms

(i) There must be no fence (f).
(ii) If there is a fence, it must be a white (w) fence.
(iii) If the cottage is by the sea (s), there must be a fence.

If we build a fence f, we violate the norm $\mathbf{O}\neg f$ from (i) but then we are subject to the secondary obligation of a white fence $\mathbf{O}w$, that implies $\mathbf{O}f$ since a white fence is a fence. Thus, we may have to accept situations in which both $\mathbf{O}\neg f$ and $\mathbf{O}f$ coexist, something impossible in KD whose main axiom D: $\neg(\mathbf{O}f \wedge \mathbf{O}\neg f)$ states that this is inconsistent. Furthermore a cottage by the sea (iii) is usually read as an exception to the prohibition (i) when we understand the latter as a default, something that cannot be represented using (a monotonic logic like) KD.

To represent and reason about norms, deontic logic is commonly used in combination with techniques from nonmonotonic reasoning (e.g. [29,34]). Few tools exist e.g. [24,21,5], but flexible computational techniques and standardization are still lacking. Standardization and flexibility are among the key features of Answer Set Programming (ASP)—one of the most prominent paradigms of knowledge representation and reasoning for problem solving [6]. ASP's success can be attributed to its wide range of applications [14], efficient tools like `clingo` [17] and `DLV` [32], and others used in ASP competitions [18], but also to its solid theoretical foundations. The logical characterization of ASP based on *Equilibrium Logic* [39] allows the treatment of the ASP connectives, including both default and explicit negation [2]. It has been extended to deal with quantifiers [40], functions, sets and aggregates [9] and has also facilitated the homogeneous extension of ASP with temporal [1] and epistemic [8] modalities. A hybrid combination with the logic KD has been introduced in [3], and called Deontic Equilibrium Logic (DEL). Syntactically, DEL builds upon Equilibrium Logic and replaces the role of atoms by KD modal formulas that use a distinct set of Boolean connectives. This orientation is less integrated than other modal extensions of equilibrium logic, e.g. [1,8], in the sense that modal and non-modal operators cannot be freely combined. Besides, instead of collapsing to regular ASP, the non-modal fragment of DEL can capture Reiter's Default Logic [42]. More importantly, being based on KD, DEL considers the simultaneous obligation and prohibition of the same fact as inconsistent, which may need to be relaxed to deal with contrary-to-duties.

In this paper we present a novel deontic extension of Equilibrium Logic that, instead of dealing with a modal language, focuses on reasoning about literals built with *explicit negation*, originally known in ASP as "classical" negation [19]. To this aim, in Section 3, we introduce *deontic logic programs* that minimally extend ASP with two new propositions representing obligation and prohibition of atoms. This framework can be straightforwardly encoded in ASP, maintaining the same computational complexity. To overcome the syntactic limitations of logic programs, we propose *Deontic Equilibrium Logic with eXplicit negation* (DELX) in Section 4. DELX is a full logical language that extends equilibrium logic (with explicit negation) by incorporating obligations and prohibitions as new connectives. We demonstrate that any DELX theory can be reduced to a deontic logic program, enabling the use of ASP to compute its (deontic) equilibrium models. To assess the adequacy of the proposed formalism, we use our framework to tackle the most salient challenges of normative reasoning, which we formalize and discuss in Section 5 as variations of Example 1. Through various DELX expressions, we capture nuanced interpretations of norms in a formal manner, showcasing a high degree of elaboration tolerance [33].

2 ASP in a nutshell

We recall the definition of answer sets for propositional logic programs with explicit negation; we extend here the original definition in [19] by allowing default negation in the head, something familiar in modern ASP. We start from a propositional *signature*, a set of atoms At, and define an *explicit literal* as any $p \in At$ or its explicit negation $\neg p$. A *default literal* is any explicit literal L or its default negation *not* L. A *rule* is an implication of the form:

$$H_1 \vee \cdots \vee H_n \leftarrow B_1 \wedge \cdots \wedge B_m \tag{1}$$

where $n, m \geq 0$ and all H_i and B_j are default literals. The disjunction $H_1 \vee \cdots \vee H_n$ in (1) is called the rule *head*. When $n = 0$, the head is the empty disjunction $\bot$, and the rule is said to be a *constraint*. The conjunction $B_1 \wedge \cdots \wedge B_m$ in (1) is called the rule *body*. When $m = 0$, it corresponds to the empty conjunction $\top$ and, when this happens, we normally omit both the body $\top$ and the $\leftarrow$ symbol. Moreover, if $m = 0$, $n = 1$, and the head consists of a unique explicit literal H_1 (no default negation), we say that the rule is a *fact*. A *logic program* is a set of rules. For simplicity, in this paper we deal with finite programs and we sometimes represent them as the conjunction of their rules. Logic programs may contain variables, but they are understood as an abbreviation of all their possible ground instances (for simplicity, we do not allow function symbols).

A *propositional interpretation* T for a signature At is any set of explicit literals that is *consistent*, i.e., it contains no pair of literals p and $\neg p$ for a same atom $p \in At$. Given any rule r like (1) containing no default negation, we say that an interpretation *satisfies* r if there is some head explicit literal $H_i \in T$ whenever all body literals $B_j \in T$. The *reduct* of a logic program Π with respect to an interpretation T, written Π^T, is the result of: (1) removing all rules with a default literal *not* L in the body such that $L \in T$; (2) removing all rules with a default literal *not* L in the head such that $L \notin T$; and (3) removing the rest of default literals. An interpretation T is an *answer set* of a logic program Π if it is $\subseteq$-minimal among all the interpretations satisfying all rules of Π^T.

3 Deontic Logic Programs

Following a minimalist approach, we extend ASP with two new types of propositions that talk about atomic *obligations* $\mathbf{O}p$ (read as "p is obligatory") and atomic *prohibitions* $\mathbf{F}p$ ("p is forbidden"), for any atom $p \in At$. In many deontic logics (e.g. KD [45]) the prohibition $\mathbf{F}p$ can be defined as the obligation $\mathbf{O}\neg p$. However, at this point, we refrain from reading $\mathbf{O}$ and $\mathbf{F}$ as real operators, and see them as prefixes for new ASP atoms called "$\mathbf{O}p$" and "$\mathbf{F}p$" in the signature. Keeping p, $\mathbf{O}p$ and $\mathbf{F}p$ separated as three independent propositions makes sense since, for instance, there is no established connection between $\mathbf{O}p$ and p, as one may have the obligation of p but yet, p may not hold (i.e., the obligation is not fulfilled), and similarly for prohibitions. In addition, under certain conditions we will allow $\mathbf{O}p$ and $\mathbf{F}p$ to hold together, as discussed in the introduction.

A *deontic atom* is either $p \in At$ or any of the expressions $\mathbf{O}p$ or $\mathbf{F}p$. The *deontic signature* At' is defined as $At' := At \cup \{\mathbf{O}p \mid p \in At\} \cup \{\mathbf{F}p \mid p \in At\}$. We may now form explicit literals for At'. Intuitively, p (and $\neg p$) means that p is true (false, resp.) in a factual sense, so that when none of the two hold, there is no evidence that p or $\neg p$ hold or have happened. E.g., if p means "pay taxes", when p holds we can read it as "the payment can be checked", and when $\neg p$ holds as "we can prove that the payment was not done". The explicit literals $\mathbf{O}p$ and $\neg\mathbf{O}p$ stand for "the obligation of p is true" and "is explicitly false", or "$\neg p$ is explicitly permitted" (we will see below that permissions can also be expressed in a weaker way by using default negation). Again, we may also have that none of the two hold. We permit having at the same time both the literal $\neg p$ in the real world and an obligation $\mathbf{O}p$, meaning that the latter is violated. Finally, the prohibition $\mathbf{F}p$ is dual to the obligation. Its explicit negation $\neg\mathbf{F}p$ can be read as "p is explicitly permitted" whereas a violation happens when both $\mathbf{F}p$ and p hold simultaneously. By introducing default negation, for any atom $p \in At$ we can form 12 default literals corresponding to (atomic) *normative positions* (cf. [43]):

$$p, \neg p, \mathbf{O}p, \neg\mathbf{O}p, \mathbf{F}p, \neg\mathbf{F}p, \mathit{not}\ p, \mathit{not}\,\neg p, \mathit{not}\ \mathbf{O}p, \mathit{not}\,\neg\mathbf{O}p, \mathit{not}\ \mathbf{F}p, \mathit{not}\,\neg\mathbf{F}p$$

For instance, the reading of *not* $\mathbf{O}p$ is "there is no evidence about $\mathbf{O}p$" as opposed to $\neg\mathbf{O}p$ that provides evidence for $\mathbf{O}p$ to be explicitly false. In fact, we can also see *not* $\mathbf{O}p$ as an implicit permission for $\neg p$, and something similar happens with *not* $\mathbf{F}p$, that becomes an implicit permission for p (see C1 in Sec. 5). A literal like *not* $\neg\mathbf{F}p$ reads as "there is no reason to conclude the explicit permission of p".

An interpretation containing both the obligation $\mathbf{O}p$ and the prohibition $\mathbf{F}p$ is a dilemma and should be inconsistent. This corresponds to the Deontic axiom D, present in most deontic logics. Let us encode (i)-(iii) from Example 1 as:

$$\mathbf{F}f \qquad\qquad \mathbf{O}w \leftarrow f \wedge \mathbf{F}f \qquad\qquad \mathbf{O}f \leftarrow s \tag{2}$$

and assume we add the fact s ("the cottage is by the sea"). The only answer set is $\{\mathbf{F}f, \mathbf{O}f, s\}$ and so, we have a specification demanding both the presence and absence of a fence simultaneously. This specification should be considered inconsistent because, somehow, we have *contradicting indications on how to proceed*. To achieve the inconsistency of the program $(2) \cup \{s\}$ we could define the *deontic answer sets* as those in which for no atom p, $\mathbf{O}p, \mathbf{F}p$ occurs. However, to deal with secondary obligations, we may require that both $\mathbf{O}p$ and $\mathbf{F}p$ hold, if one of them has been violated. As a white fence is also a fence, we add:

$$f \leftarrow w \qquad\qquad \mathbf{O}f \leftarrow \mathbf{O}w \tag{3}$$

and if we take the extended program $(2) \cup (3) \cup \{f\}$ we obtain the answer set $\{f, \mathbf{F}f, \mathbf{O}w, \mathbf{O}f\}$ so, we conclude both $\mathbf{F}f$ and $\mathbf{O}f$. These two deontic atoms however *do not provide indications on how to proceed*, as the decision to put a fence has been already taken, forcing the violation of $\mathbf{F}f$ and the fulfillment of $\mathbf{O}f$, derived from $\mathbf{O}w$. The conclusion is that $\mathbf{O}f$ and $\mathbf{F}f$ may coexist, provided that one of the two has been violated. This leads us to the following definition.

Definition 1. *A* deontic interpretation T *is a propositional interpretation for* At' *satisfying:* $\{\mathbf{O}p, \mathbf{F}p\} \subseteq T$ *implies* $\{p, \neg p\} \cap T \neq \emptyset$*, for any* $p \in At$*.*

T is consistent by definition, that is, T cannot contain literals A and $\neg A$ for a same deontic atom A. To be a deontic interpretation, we additionally require that the atoms $\mathbf{O}p$ and $\mathbf{F}p$ can only hold together when T contains information about p, i.e., either p or its explicit negation $\neg p$ are in T. Note that the mere presence of $\mathbf{O}p$ and $\mathbf{F}p$ together will not permit to derive p or $\neg p$, as the derivation can only be achieved by application of rules in the logic program. We call *deontic answer sets* to the answer sets of a deontic logic program that are also deontic interpretations. To obtain them, we can use the axiom schema (for any $p \in At$):

$$\bot \leftarrow \mathbf{O}p \wedge \mathbf{F}p \wedge \mathit{not}\ p \wedge \mathit{not}\ \neg p \qquad (\mathbf{wD})$$

that is a weaker version of the Deontic axiom D, and states that the conjunction of $\mathbf{O}p$ and $\mathbf{F}p$ is inconsistent only if none of the two has been violated.

Proposition 1. *T is a deontic answer set of a deontic logic program Π iff T is an answer set of $\Pi \cup (\mathbf{wD})$.*

To see the effect of ($\mathbf{wD}$), consider again the program $\Pi = (2) \cup \{s\}$. As mentioned before, the only answer set of this program would be $T = \{\mathbf{F}f, \mathbf{O}f, s\}$ but this is ruled out by the ($\mathbf{wD}$) constraint. In fact, T is not a deontic answer set since we have both the obligation and the prohibition of f but we did not provide any information about f or $\neg f$. This means we face a dilemma, because we have two contradictory norms and none of them has been violated. If we take program $\Pi' = \Pi \cup \{f\}$ (that is, we decide to put a fence) then $\mathbf{F}f$ is violated and consistency is restored, obtaining the answer set $T' = \{\mathbf{F}f, \mathbf{O}f, s, f, \mathbf{O}w\}$. Note how we derive the obligation of a white fence $\mathbf{O}w$, and that the prohibition of $\mathbf{F}f$ *has not been retracted*, but has been violated instead. If, instead of f, we are said that no fence will be built $\neg f$ (i.e. we have evidence that there is no fence), then the program $\Pi'' = \Pi \cup \{\neg f\}$ also becomes consistent, leading this time to answer set $T'' = \{\mathbf{F}f, \mathbf{O}f, s, \neg f\}$ where $\mathbf{O}f$ is violated.

Proposition 1 allows a direct encoding of any deontic logic program Π into a regular ASP program Π'. To do so, a compact representation can be achieved by reifying all atoms in Π to become arguments of three predicates in Π', say `h`, `ob` and `fb` respectively standing for *holds* (in a factual sense), *obligatory* and *forbidden.* As an illustration, (2)-(3) can be represented as the ASP program[1] below where the constraint in the last line is an encoding of ($\mathbf{wD}$).

```
fb(f).                                  % The fence is forbidden
ob(w) :- h(f), fb(f).                   % CTD: if fence, it must be white
ob(f) :- h(s).                          % Obligatory fence if by the sea
h(f).                                   % We have a fence
h(f) :- h(w).                           % White fence means fence
ob(f) :- ob(w).                         % The same for obligation
:- ob(P), fb(P), not h(P), not -h(P).   % Axiom (wD)
```

This encoding can be easily automated in linear time, so the complexity results of deontic logic programs are as in the regular (disjunctive) ASP case [13]. In particular we have the following:

[1] In the ASP-core-2 input language, ←, ¬ and ∧ are represented as ‘`:-`’, ‘`-`’ and commas.

Proposition 2. *Deciding whether a deontic logic program Π has a deontic answer set is Σ_2^P-complete. If every head in Π is free from disjunction, deciding the existence of a deontic answer set is NP-complete.*

4 Extension to Equilibrium Logic

Introducing deontic atoms in logic programs provides a simple and practical approach for formalizing deontic scenarios, but falls short if we need a proper logical formalism. Note that, so far, **O** and **F** are not proper operators but just a kind of prefix for atoms: in fact, all program operators in ASP are also treated under a very restricted syntax, and their semantics relies on a syntactic transformation (the program reduct). If we are interested in arbitrary nesting of operators, defining new constructs or extensions to incorporate temporal or epistemic reasoning, we need a logical formalisation that overcomes the syntax limitations and the program reduct operation. An excellent starting point for our purposes is the logical characterization of ASP based on *Equilibrium Logic* [39] which has also been extended to deal with strong [36] or explicit negation [2]. As happens in ASP, when explicit negation is used, equilibrium models become three-valued (an atom can be true p, false $\neg p$ or none of the two). To introduce **O** and **F** in this setting, we adopt a practical approach, so that, although they will be applicable now on other operators, their expressive power is still limited to a kind of three-valued semantics. The advantage of this approach is to reduce arbitrary formulas to theories where deontic operators are only used in *explicit literals*, something that can be easily translated into ASP logic programs. Yet, when compared to a modal interpretation of **O** and **F**, the price to pay is a loss in expressiveness when dealing with obligations on compound formulas: for example, $\mathbf{O}(\varphi \vee \psi)$ will simply be $\mathbf{O}\varphi \vee \mathbf{O}\psi$. This coincides with the ASP reading of disjunction, and in fact to the natural language reading of disjunction in the free choice permission scenario [30].

Equilibrium models are defined by a selection among models from the intermediate logic called *Here-and-There* [28] (HT), or 3-valued Gödel logic. We now incorporate deontic operators in the extension $\mathcal{X}_5$ of HT with explicit negation [2], thus defining the logic of *Deontic Here-and-There with Explicit Negation* (DHTX for short). A *formula* φ of DHTX follows the grammar:

$$\varphi ::= p \in At \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \ \neg\varphi \mid \mathbf{O}\varphi \mid \mathbf{F}\varphi$$

We define the derived operators $\varphi \leftrightarrow \psi \overset{\text{def}}{=} (\varphi \to \psi) \wedge (\psi \to \varphi)$, *not* $\varphi \overset{\text{def}}{=} (\varphi \to \bot)$ and the constant $\top$ as *not* $\bot$. We assume that the conditional rule $\varphi \leftarrow \psi$ in logic programs is nothing but the reversed implication $\psi \to \varphi$. We also define the following derived deontic operators:

$$\begin{array}{rlrl}
\mathbf{O}^{\mathbf{v}}\,\varphi & \overset{\text{def}}{=} \mathbf{O}\varphi \wedge \neg\varphi & \mathbf{O}^{\mathbf{f}}\,\varphi & \overset{\text{def}}{=} \mathbf{O}\varphi \wedge \varphi \\
\mathbf{O}^{\mathbf{nf}}\,\varphi & \overset{\text{def}}{=} \mathbf{O}\varphi \wedge \mathit{not}\ \varphi & \mathbf{O}^{\mathbf{nv}}\,\varphi & \overset{\text{def}}{=} \mathbf{O}\varphi \wedge \mathit{not}\ \neg\varphi \\
\mathbf{O}^{\mathbf{u}}\,\varphi & \overset{\text{def}}{=} \mathbf{O}\varphi \wedge \mathit{not}\ (\varphi \vee \neg\varphi) & \mathbf{P}\varphi & \overset{\text{def}}{=} \neg\mathbf{F}\varphi \\
\mathbf{O}^{\mathbf{d}}\,\varphi & \overset{\text{def}}{=} (\mathit{not}\ \mathbf{P}\neg\varphi \rightarrow \mathbf{O}\varphi) & \mathbf{P}^{\mathbf{d}}\,\varphi & \overset{\text{def}}{=} (\mathit{not}\ \mathbf{F}\varphi \rightarrow \mathbf{P}\varphi) \\
\mathbf{O}(\varphi \mid \psi) & \overset{\text{def}}{=} (\psi \vee \mathbf{O}^{\mathbf{nv}}\,\psi \rightarrow \mathbf{O}\varphi) & \mathbf{F}^{x}\,\varphi & \overset{\text{def}}{=} \mathbf{O}^{x}\,\neg\varphi
\end{array}$$

$\mathbf{P}\varphi$ stands for the explicit permission for φ whereas $\mathbf{P}^{\mathbf{d}}\,\varphi$ is its default version. The superindexed variants of $\mathbf{O}$ stand for: **d**=*default*, **f**=*fulfilled*, **v**=*violated*, **nv**=*non-violated*, **nf**=*non-fulfilled* and **u**=*undetermined*. We define the same variants $\mathbf{F}^{x}$ in terms of $\mathbf{O}^{x}$ for all $x \in \{\mathbf{d}, \mathbf{f}, \mathbf{v}, \mathbf{nv}, \mathbf{nf}, \mathbf{u}\}$ having in mind that we can now replace $\mathbf{F}\varphi$ by $\mathbf{O}\neg\varphi$. The conditional obligation $\mathbf{O}(\varphi \mid \psi)$ ("φ is obligatory, given ψÂ´Â´) is explained in Section 5 (challenge C6).

A formula is said to be *deontic* if it contains deontic operators, and *non-deontic* otherwise. A *theory* is a set of formulas. Finite theories (or subtheories) are understood as the conjunction of their formulas. Notice that deontic logic programs are theories.

Definition 2. *A* Deontic HT-interpretation *is a pair $\langle H, T\rangle$ of sets of explicit literals s.t. T is a deontic interpretation and $H \subseteq T$. $\langle H, T\rangle$ is* total *when $H = T$.*

Intuitively, literals in H ("here") can be considered founded or proved, literals in $T \setminus H$ are assumed but unfounded and literals not in T ("there") are not assumed and they directly do not hold. For instance, the pair $H = \{\mathbf{O}p, \mathbf{F}p\}$ and $T = \{\neg p, \mathbf{O}p, \mathbf{F}p, \neg\mathbf{F}q\}$ is a deontic HT-interpretation where $\mathbf{O}p$ and $\mathbf{F}p$ are founded whereas $\neg p$ and $\neg\mathbf{F}q$ are assumed but unfounded. The interpretation just considers the rest of literals as not assumed. Note that the potential inconsistency between $\mathbf{O}p$ and $\mathbf{F}p$ is only checked at the component T. The same effect is obtained if (**wD**) is added as an axiom instead of requiring T to be a deontic interpretation in Def. 2. In this case, the two literals can occur together because $\neg p \in T$, so we assume that $\mathbf{O}p$ is violated. On the other hand, $\neg p$ is not justified at H but we still allow $\mathbf{O}p$ and $\mathbf{F}p$ in H, since $\neg p$ is assumed at T.

We define the set of "deontic worlds" as $\{r, o, f\}$ respectively standing for *real*, *obligation* and *forbidden*. Given a world $w \in \{r, o, f\}$, its complementary world $\overline{w}$ is defined as $\overline{r} \overset{\text{def}}{=} r$, $\overline{o} \overset{\text{def}}{=} f$ and $\overline{f} \overset{\text{def}}{=} o$.

Definition 3. $M = \langle H, T \rangle$ satisfies *(resp.* falsifies*) a formula φ at a deontic world $w \in \{r, o, f\}$, written $M, w \models \varphi$ ($M, w \Dashv \varphi$), if the conditions below hold:*

φ	$M, w \models \varphi$ **when**	$M, w \Dashv \varphi$ **when**
$\top$ ($\bot$)	*always* (*never*)	*never* (*always*)
$\alpha \wedge \beta$	$M, w \models \alpha$ *and* $M, w \models \beta$	$M, w \Dashv \alpha$ *or* $M, w \Dashv \beta$
$\alpha \vee \beta$	$M, w \models \alpha$ *or* $M, w \models \beta$	$M, w \Dashv \alpha$ *and* $M, w \Dashv \beta$
$\alpha \to \beta$	$M', w \not\models \alpha$ *or* $M', w \models \beta$ *for* $M' \in \{M, \langle T, T \rangle\}$	$\langle T, T \rangle, w \models \alpha$ *and* $M, w \Dashv \beta$
$\neg\alpha$	$M, \overline{w} \Dashv \alpha$	$M, \overline{w} \models \alpha$
p	$p \in H$ *if* $w = r$ $\mathbf{O}p \in H$ *if* $w = o$ $\neg\mathbf{F}p \in H$ *if* $w = f$	$\neg p \in H$ *if* $w = r$ $\neg\mathbf{O}p \in H$ *if* $w = o$ $\mathbf{F}p \in H$ *if* $w = f$
$\mathbf{O}\alpha$	$M, o \models \alpha$	$M, o \Dashv \alpha$
$\mathbf{F}\alpha$	$M, f \Dashv \alpha$	$M, f \models \alpha$

In the definition above, if we just take the syntactic fragment for $\wedge$, $\vee$, $\neg$, $\bot$, $\top$ and atoms (we can fix $w = r$), we obtain *classical logic with strong negation* [44]. If we further extend it with the evaluation of $\to$ (still fixing $w = r$) we get *Equilibrium Logic with explicit negation* $\mathcal{X}_5$ [2]. So, the new features are the three deontic worlds and their interplay with the operators $\mathbf{O}$, $\mathbf{F}$ and $\neg$. As we can see, the interpretation of an atom $p \in At$ depends on each specific world. The real world $w = r$ works as expected whereas, in world $w = o$, satisfying (resp. falsifying) an atom p actually corresponds to requiring that the literal $\mathbf{O}p$ (resp. $\neg\mathbf{O}p$) holds in the interpretation. In the world $w = f$ the roles of literals are swapped, so satisfaction of an atom p corresponds to including the literal $\neg\mathbf{F}p$ whereas falsifying p corresponds to the literal $\mathbf{F}p$. The reason for this swapping is that a prohibition $\mathbf{F}\alpha$ is a kind of negation (we will see later that it is actually equivalent to $\mathbf{O}\neg\alpha$). The operators that permit moving to a different deontic world are $\neg$, $\mathbf{O}$ and $\mathbf{F}$. To satisfy $\mathbf{O}\alpha$ we simply check the satisfaction of α after "jumping" to world o, regardless of the world we started from, and the same happens for the falsification of $\mathbf{O}\alpha$. With $\mathbf{F}\alpha$ a similar effect is obtained for the world f, but again, it additionally swaps satisfaction to falsification and vice versa. In the case of explicit negation, satisfaction of $\neg\alpha$ becomes falsification of α but, additionally, if we are not in the real world $w \neq r$, we switch from w to $\overline{w}$. An example on how these three operators work: $M, r \models \mathbf{O}\neg p$ becomes $M, o \models \neg p$ that is interpreted as $M, f \Dashv p$ and, finally, it amounts to $\mathbf{F}p \in H$.

An HT-interpretation $\langle H, T \rangle$ is a *model* of a theory Γ, written $\langle H, T \rangle \models \Gamma$, if $\langle H, T \rangle, r \models \varphi$ for all $\varphi \in \Gamma$. A formula φ is a DHTX-*tautology* (or DHTX-*valid*), $\models \varphi$ in symbols, if any DHTX-interpretation is a model of φ. DHTX is the logic induced by all DHTX-tautologies.

The properties below are fundamental in any extension of HT.

Theorem 1 (Persistence). *For any DHTX-interpretation $\langle H,T\rangle$, any world $w \in \{r,o,f\}$ and any formula φ: (i) $\langle H,T\rangle, w \models \varphi$ implies $\langle T,T\rangle, w \models \varphi$ for any world w, and (ii) $\langle H,T\rangle, w =\!\!| \varphi$ implies $\langle T,T\rangle, w =\!\!| \varphi$ for any world w.* □

Proposition 3. *For any $\langle H,T\rangle$, world w and formula φ: (i) $\langle H,T\rangle, w \models \mathit{not}\ \varphi$ iff $\langle T,T\rangle, w \not\models \varphi$; (ii) $\langle H,T\rangle, w =\!\!| \mathit{not}\ \varphi$ iff $\langle T,T\rangle, w \models \varphi$.* □

DHTX is an extension of $\mathcal{X}_5$ in the following sense:

Proposition 4. *If φ is $\mathcal{X}_5$-valid then φ is DHTX-valid.*

For instance, the following $\mathcal{X}_5$ tautologies are also DHTX-valid:

$$\neg(\varphi \to \psi) \leftrightarrow \mathit{not}\ \mathit{not}\ \varphi \wedge \neg\psi \qquad \neg \mathit{not}\ \varphi \leftrightarrow \mathit{not}\ \mathit{not}\ \varphi \tag{4}$$

As happens in $\mathcal{X}_5$ the validity of $\varphi \leftrightarrow \psi$ does not guarantee that we can always substitute φ by ψ. To this aim, we introduce the following stronger relation (taken from [2]). Two formulas φ and ψ are DHTX-*equivalent*, written $\varphi \equiv \psi$, if for any DHTX-interpretation $M = \langle H,T\rangle$ and any world $w \in \{r,o,f\}$, we have both: (1) $M, w \models \varphi$ iff $M, w \models \psi$; and (2) $M, w =\!\!| \varphi$ iff $M, w =\!\!| \psi$.

Proposition 5. *For any pair of formulas φ and ψ, if $\varphi \equiv \psi$ then $\models \varphi \leftrightarrow \psi$.*

In general, the other direction does not hold. As a counterexample (already used in [2]) take the DHTX-tautology $p \wedge \mathit{not}\ p \leftrightarrow \bot$ (which is also an $\mathcal{X}_5$-tautology). It is not difficult to see, however, that $p \wedge \mathit{not}\ p \not\equiv \bot$. In fact, we cannot replace $p \wedge \mathit{not}\ p$ inside $\neg(p \wedge \mathit{not}\ p)$ to get $\neg\bot$. Indeed, the former amounts to a rule $\neg p \leftarrow \mathit{not}\ p$ while the latter to $\top$.

Yet, we can still use $\models \varphi \leftrightarrow \psi$ to perform substitutions in some contexts:

Theorem 2. *$\models \varphi \leftrightarrow \psi$ iff φ and ψ have the same DHTX-models.*

Corollary 1. *Let $\Gamma[\varphi]$ be a theory containing a subformula φ not in the scope of $\neg$, **F** or **O** and let $\models \varphi \leftrightarrow \psi$. Then, $\Gamma[\varphi]$ and $\Gamma[\psi]$ have the same DHTX-models.*

Definition 4. *A total DHTX-interpretation $\langle T,T\rangle$ is an* equilibrium model *of a theory Γ if $\langle T,T\rangle \models \Gamma$ and there is no $H \subset T$ such that $\langle H,T\rangle \models \Gamma$.*

Deontic Equilibrium logic is the non-monotonic logic induced by equilibrium models. For deontic logic programs, answer sets and equilibrium models coincide:

Theorem 3. *Let Π be a deontic logic program. A deontic interpretation T is a deontic answer set of Π iff $\langle T,T\rangle$ is an equilibrium model of Π.* □

We show below that any deontic theory can be reduced to a deontic logic program, and so, its equilibrium models can be eventually computed via regular ASP. We start observing a group of DHTX-equivalences that also hold in $\mathcal{X}_5$:

$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi \quad \neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi \quad \neg\neg\varphi \equiv \varphi \quad \neg\bot \equiv \top \quad \neg\top \equiv \bot \tag{5}$$

We can use (4)-(5) from the outermost occurrences of $\neg$ to push this operator inside non-deontic connectives. As a result we get an *Explicit-negation Normal*

Form (XNF), where all outermost occurrences of $\neg$ are applied to atoms, to $\mathbf{O}$ or to $\mathbf{F}$. To unfold expressions inside $\mathbf{O}$ or $\mathbf{F}$ we can further apply the equivalences:

$$\mathbf{O}(\varphi \vee \psi) \equiv \mathbf{O}\varphi \vee \mathbf{O}\psi \qquad \mathbf{F}(\varphi \vee \psi) \equiv \mathbf{F}\varphi \wedge \mathbf{F}\psi \tag{6}$$

$$\mathbf{O}(\varphi \wedge \psi) \equiv \mathbf{O}\varphi \wedge \mathbf{O}\psi \qquad \mathbf{F}(\varphi \wedge \psi) \equiv \mathbf{F}\varphi \vee \mathbf{F}\psi \tag{7}$$

$$\mathbf{O}\bot \equiv \bot; \quad \mathbf{F}\bot \equiv \top \qquad \mathbf{O}\top \equiv \top; \quad \mathbf{F}\top \equiv \bot \tag{8}$$

$$\mathbf{O}(\varphi \to \psi) \equiv \mathbf{O}\varphi \to \mathbf{O}\psi \tag{9}$$

$$\mathbf{O}\mathit{not}\ \varphi \equiv \mathit{not}\ \mathbf{O}\varphi \qquad \mathbf{F}\mathit{not}\ \varphi \equiv \mathit{not}\ \mathit{not}\ \neg\mathbf{F}\varphi \tag{10}$$

$$\mathbf{O}\neg\varphi \equiv \mathbf{F}\varphi \qquad \mathbf{F}\neg\varphi \equiv \mathbf{O}\varphi \tag{11}$$

$$\mathbf{O}\mathbf{O}\varphi \equiv \mathbf{O}\varphi \qquad \mathbf{F}\mathbf{O}\varphi \equiv \neg\mathbf{O}\varphi \tag{12}$$

$$\mathbf{O}\mathbf{F}\varphi \equiv \mathbf{F}\varphi \qquad \mathbf{F}\mathbf{F}\varphi \equiv \neg\mathbf{F}\varphi \tag{13}$$

By (11), we may choose either $\mathbf{O}$ or $\mathbf{F}$ as a primitive connective, and hence the primitive DELX connectives can be reduced to five $\wedge, \vee, \to, \neg, \mathbf{O}$, together with the constant $\bot$. Equivalences (6)-(13) do not cover the case when $\mathbf{F}$ is applied to an implication: if so, we can only proceed from the outermost occurrences of this operator (as happened with explicit negation) using the valid double implications:

$$\mathbf{F}(\varphi \to \psi) \leftrightarrow \mathit{not}\ \mathit{not}\ \neg\mathbf{F}\varphi \wedge \mathbf{F}\psi \qquad \neg\mathbf{F}(\varphi \to \psi) \leftrightarrow \neg\mathbf{F}\varphi \to \neg\mathbf{F}\psi \tag{14}$$

Using these properties we can reach the syntactic form we call *Deontic-Atom Normal Form* (DANF), in which all deontic operators are applied to atoms. Once in DANF, we can then resort to $\mathcal{X}_5$ reduction to logic programs.

Theorem 4. *Any deontic theory can be reduced to a deontic logic program having the same DHTX models.*

It is not hard to see that the reduction to DANF is polynomial whereas the step from arbitrary combinations of $\wedge, \vee,$ *not* $, \to$ into a logic program may be exponential due to distributivity laws. Yet, [11] proposes an alternative polynomial reduction that avoids the combinatorial blowup by introducing auxiliary atoms.

5 DELX at work on challenging normative problems

We discuss the nuances of defeasible deontic reasoning that we aimed to capture, using variants of the cottage regulation (Ex. 1). We consider below the starting program $\Pi = (2) \cup (3)$ and analyze the challenges from [7] we refer to as C1-C6.

C1 (Explicit versus Negative permission) We want to distinguish between the existence of permission vs absence of prohibition. Suppose that a new neighbor ignores the local regulations and has in mind the practical reasoning rule:

(iv) *If it is permitted, I build a fence around my cottage*

A cautious behavior is to wait for an *explicit permission* to build the fence. A more adventurous behavior is to build it if there is no explicit prohibition (*negative or implicit permission*): without more information, she concludes to build the fence, but retracts that conclusion once she becomes aware of norm (i). Explicit and implicit permissions can be respectively captured by the rules:

$$f \leftarrow \neg \mathbf{F} f \qquad (15) \qquad\qquad f \leftarrow \mathit{not}\ \mathbf{F} f \qquad (16)$$

The program (16) alone permits to conclude f (the adventurous neighbor builds the fence), but with $\Pi \cup (16)$ this is not possible anymore, as we have $\mathbf{F} f$. On the other hand, the cautious neighbor cannot conclude f from $\Pi \cup (15)$ or even from (15) alone, since it requires the permission $\neg \mathbf{F} f$. We could get it with a cottage by the sea, but then (i) should be formalized as a default (see C4).

We may be sometimes interested in generalizing this distinction into a Closed World Assumption for a given set Γ of formulas. For instance, a *Closed Explicit Permission Assumption* (CEPA) stands for "anything not explicitly forbidden is permitted" and can be simply formalized as $\mathbf{P^d}\, \varphi$ for every $\varphi \in \Gamma$. Similarly, a *Closed Negative Permission Assumption* (CNPA) rather means "anything not explicitly permitted is forbidden" and just corresponds to $\mathbf{F^d}\, \varphi$, for all $\varphi \in \Gamma$.

C2 (Contrary-to-Duty (CTD) and Compliance) A *CTD* or *secondary* obligation comes into force only when another (the primary) obligation is violated. For instance, the two sentences (i) and (ii) from Ex. 1 are primary obligation and CTD. A different, though related concept is that of compensatory obligation, as:

(v) *If you put a fence when forbidden, you should pay a fine.*

If we combine the prohibition (i) with the existence of a fence we want to derive from (v) that fences are forbidden and that a fine must be paid. This is known as *monitoring*, *compliance*, or *conformance checking*. Likewise, if obligations are fulfilled, rewards may be given. Encoding compliance in DELX is straightforward: we may just use the derived operators for violation $\mathbf{F^v}$ or fulfillment $\mathbf{F^f}$. E.g., (v) is formalized as $(\mathbf{F^v}\, f \rightarrow \mathbf{O}\mathit{pay})$ that amounts to the logic program rule:

$$\mathbf{O}\mathit{pay} \leftarrow \mathbf{F} f \wedge f \qquad (17)$$

C3 (CTD and Dilemmas) A *dilemma* is a situation where we deal with the simultaneous obligation and prohibition of a same fact. For instance having (i) together with $\mathbf{O}w$ leads to a dilemma. There is consensus in the literature that such dilemmas should be inconsistent. This is, in fact, what happens with the program $\Pi \cup \{\mathbf{O}w\}$ that has no answer set, since axiom ($\mathbf{wD}$) does not accept $\mathbf{O}f \wedge \mathbf{F} f$ without information about f. However, when a dilemma follows from a CTD, consistency should be restored. E.g., suppose we have the premises (i)-(iii) plus (v) and, additionally, there exists a fence f. By (ii), we must have a white fence, but this is in conflict with (i), that says we must have *no fence at all.* This scenario is consistent in DELX: the program $\Pi \cup (17) \cup \{f\}$ has a unique answer set $\{f, \mathbf{F} f, \mathbf{O}w, \mathbf{O}f, \mathbf{O}\mathit{pay}\}$ where $\mathbf{O}f \wedge \mathbf{F} f$ is now consistent because $\mathbf{F} f$ has been violated. Notice that some deontic approaches remove the CTD dilemma by retracting the primary prohibition (i) to have a fence. This leads to the so-called *drowning problem* [38]: we would no longer have a violation of $\mathbf{F} f$ so we cannot derive the payment of the fine $\mathbf{O}\mathit{pay}$. Note that the combination of compliance and dilemmas has become problematic for some deontic approaches (most notably Dyadic Deontic Logic [27,37]), requiring *ad hoc* representations like the introduction of so-called *violation constants.*

C4 (CTD and Defeasible Obligations) Some obligations should be read as defaults in the presence of exceptions. Let us rephrase (iii) as the permission:

(vi) *If the cottage is by the sea (s), there* **may** *be a fence.*

This was the original wording for the Cottage scenario in [41], introduced to illustrate the distinction between CTD reasoning ((i) and (ii)) and exceptions ((i) and (vi)); the two types of reasoning should be treated differently. Indeed, if we consider (i) and (ii) as instances of defeasible reasoning, we would let the primary obligation (i) be defeated by the secondary obligation (ii), which is not desirable. Premise (vi) leads to a new reading of the normative scenario: on the one hand, being by the sea provides now an *explicit permission* to build the fence; on the other hand, (i) is read now as "*There must be no fence, unless a permission is granted*" becoming a *default prohibition*. In our DELX formalization we may simply replace the first and third formulas in (2) respectively by $\mathbf{F^d}\, f$ (default prohibition) and $\mathbf{P}f \leftarrow s$ (explicit permission) leading to:

$$\mathbf{F}f \leftarrow \mathit{not}\ \neg\mathbf{F}f \qquad \mathbf{O}w \leftarrow f \wedge \mathbf{F}f \qquad \neg\mathbf{F}f \leftarrow s \tag{18}$$

If we have no information about the location, we consider the program $(18) \cup (3)$ alone, and the only answer set is $\{\mathbf{F}f\}$, we cannot put a fence by default. If we add the fact s we obtain $\{s, \neg\mathbf{F}f\}$, that is, we have the permission to put a fence, and $\mathbf{F}f$ is no longer derived. If we further know there is a fence, the program $(18) \cup (3) \cup \{s, f\}$ produces the answer set $\{s, f, \neg\mathbf{F}f\}$ so there is no CTD obligation of a white fence, because there is no violated prohibition.

C5 (Constitutive Norms) We now deal with the derivation of obligations in presence of an "is a" or a "count as" relation. Though there exist various kinds of constitutive norms of increasing complexity, in this paper we only consider simple factual rules. In the example we assume that a white fence *is a* fence. Does this also imply that the obligation for white fences implies the obligation for fences? And does the prohibition for fences imply the prohibition for white fences? In general yes, but as we see next, it is useful to allow for exceptions.

Our previous formalization was already considering a constitutive norm (3), namely, since a white fence w is a fence f, we also want to derive $\mathbf{O}f$ from $\mathbf{O}w$. In fact, contraposition for explicit negation could also be added:

$$\neg w \leftarrow \neg f \qquad\qquad \mathbf{F}w \leftarrow \mathbf{F}f \tag{19}$$

where the former means that not having a fence implies not having a white fence, and the latter, that a prohibition to put a fence is also a prohibition to put a white fence. The program $\Pi \cup (19) \cup \{f\}$, however, has no answer set. This is because we derive $\mathbf{O}w$ and $\mathbf{F}w$, whereas no evidence about the fence color is given: once w or $\neg w$ is added to the program, consistency is restored. A less rigid formalization of these implicit derived obligations is to replace (3) by:

$$f \leftarrow w \qquad \neg w \leftarrow \neg f \qquad \mathbf{O}f \leftarrow \mathbf{O^{nv}}\, w \qquad \mathbf{F}w \leftarrow \mathbf{F^{nv}}\, f \tag{20}$$

where the rules for obligations become now default rules. The condition $\mathbf{O^{nv}}\, w$ stands for $\mathbf{O}w \wedge \mathit{not}\ \neg w$, i.e., we derive the obligation of a fence if we have a non-violated obligation of a white fence. In the example, this is the case. Similarly, the condition $\mathbf{F}^{\mathbf{nv}}\, f$ stands for $\mathbf{F} f \wedge \mathit{not}\ f$ meaning that the prohibition of a white fence is derived when we had a non-violated prohibition of a fence. In our example, we do have the prohibition of a fence, but it has been violated, so the default does not apply, and we do not obtain a prohibition to put a white fence. As a result, the only equilibrium model of $(2) \cup (20) \cup \{f\}$ is again $\{f, \mathbf{F} f, \mathbf{O} w, \mathbf{O} f\}$.

C6 (Defeasible Deontic Detachment) This requirement is related to the distinction between *factual* versus *deontic detachment*, that is, when a conditional obligation should sometimes be triggered by facts and sometimes by other obligations. This is typically illustrated by the notorious scenario in [12] adapted below to our running example. Assume we add the norms:

(vii) *If we put a fence, we must put a street mailbox (m).*
(viii) *If we do not put a fence, we must not put a street mailbox.*

If we have information about the presence or absence of a fence, we will respectively derive the obligation or prohibition to have a mailbox by factual detachment. However, when no information is given, by default, we still want to derive the prohibition of a mailbox from the prohibition to have a fence in (i). This corresponds to a (defeasible) deontic detachment.

A direct reading of the premises (vii) and (viii) could be formalized as:

$$\mathbf{O} m \leftarrow f \qquad\qquad \mathbf{F} m \leftarrow \neg f \tag{21}$$

The program $\mathit{\Pi} \cup (21) \cup \{f \vee \neg f\}$ as the two answer sets $\{\neg f, \mathbf{F} f, \mathbf{F} m\}$ and $\{f, \mathbf{F} f, \mathbf{O} w, \mathbf{O} f, \mathbf{O} m\}$ so the obligation about the mailbox is derived from the facts f or $\neg f$ respectively (factual detachment). The problem with (21) arises in presence of $\mathit{\Pi} \cup (21)$ without further evidence about f or $\neg f$. In that case, no obligation is derived, whereas given $\mathbf{F} f$, the mailbox would be also forbidden (deontic detachment). To strengthen our representation, we replace (21) by the conditional obligations:

$$\mathbf{O}(m \mid f) \qquad \mathbf{O}(\neg m \mid \neg f) \tag{22}$$

The derived operator $\mathbf{O}(m \mid f)$ is an abbreviation of $(\mathbf{O} m \leftarrow f \vee \mathbf{O}^{\mathbf{nv}}\, f)$ and the disjunction in the antecedent can be unfolded into the two rules $(\mathbf{O} m \leftarrow f)$ and $(\mathbf{O} m \leftarrow \mathbf{O}^{\mathbf{nv}}\, f)$. Note that $\mathbf{O}^{\mathbf{nv}}\, f$, in turn, stands for $\mathbf{O} f \wedge \mathit{not}\ \neg f$. A similar unfolding can be done for $\mathbf{O}(\neg m \mid \neg f)$ to find out that (22) amounts to the two rules (21) we had before plus the following account for deontic detachment:

$$\mathbf{O} m \leftarrow \mathbf{O} f \wedge \mathit{not}\ \neg f \qquad \mathbf{F} m \leftarrow \mathbf{F} f \wedge \mathit{not}\ f$$

As a result, for $\mathit{\Pi} \cup (22) \cup \{f \vee \neg f\}$ we get the same answer sets as before, but when we just consider $\mathit{\Pi} \cup (22)$, the only answer set is $\{\mathbf{F} f, \mathbf{F} m\}$ and we cannot put a mailbox because we have a (non-violated) prohibition to put a fence.

6 Related and Future work

Related deontic extensions of ASP are *Deontic Logic Programs* (DLP) [22,23] and *Deontic Temporal ASP* (DTASP) [21]. Both make use of the KD modality [45] (in DTASP, also temporal operators) and define answer sets in terms of the (syntactic) reduct operation on logic programs. DLP was later extended to Deontic Equilibrium Logic (DEL) [3] that avoids the reduct but, as already discussed, maintains a strict syntactic separation between logic program connectives and deontic formulas. In contrast, DELX relies on logical semantics, applicable to arbitrary combinations of operators and free from syntactic restrictions or transformations. The modal logic KD allows DLP and DTASP to deal with obligations on compound formulas, while DELX is specifically designed for obligations on literals. One final important difference is that DELX makes an homogeneous integration of explicit negation, a feature *already existing* in ASP and commonly used in its applications. This permits to deal with factual situations where no information, e.g., about *fence* nor $\neg$*fence*, is available. Representing incomplete information about the real world in DLP or DTASP, requires instead epistemic modalities or the use of ASP explicit negation, whose semantic treatment is *different* from negation inside a modality.

A computationally oriented approach for deontic logic extended with features from nonmonotonic reasoning is *Defeasible Deontic Logic* (DDL) [24] (extending *Defeasible Logic* [35]) whose syntax is similar to logic programming without the default negation. To express defeasibility, DDL relies on different types of implications in rules (strict, defeasible and defeaters) additionally subscripted with deontic modalities. This contrasts with the five primitive DELX connectives. DDL also has a more complex semantics w.r.t. DELX, that employs neighbourhood models [25] or argumentation [26], and uses a dedicated theorem prover [31].

As immediate future work, we plan to develop a deontic ASP tool that accepts both deontic logic programs and DELX expressions as input, enabling the integration of deontic knowledge into existing ASP domains or encodings. Also, we will explore the extrapolation to DELX of other ASP features, such as the temporal extension [1], the generation of explanations [10] or ASP-based policies such as [20]. As a long term goal, we plan to obtain a translation of (temporal) DELX into monitors and use them in combination with Reinforcement Learning (cf. [4]) to design autonomous agents sensitive to legal, social and ethical norms.

Acknowledgments

P.Cabalar is supported by the Spanish Ministry of Science and Innovation MCIN/AEI/10.13039/501100011033 (grant PID2020-116201GB-I00), by Xunta de Galicia, Spain and the EU (grant GPC ED431B 2022/33), and by the project LIANDA - BBVA Foundation Grants for Scientific Research Projects, Spain. A. Ciabattoni is supported by the WWTF project TAIGER (ICT22-023). L. van der Torre is supported by the FNR project Deontic Logic for Epistemic Rights (OPEN O20/14776480) and the (Horizon 2020 funded) CHIST-ERA grant CHIST-ERA19-XAI (G.A. INTER/CHIST/19/14589586)

References

1. Aguado, F., Cabalar, P., Diéguez, M., Pérez, G., Schaub, T., Schuhmann, A., Vidal, C.: Linear-time temporal answer set programming. Theory and Practice of Logic Programming **23**(1), 2–56 (2023). `https://doi.org/10.1017/S1471068421000557`
2. Aguado, F., Cabalar, P., Fandinno, J., Pearce, D., Pérez, G., Vidal, C.: Revisiting explicit negation in answer set programming. Theory and Practice of Logic Programming **19**(5-6), 908–924 (2019). `https://doi.org/10.1017/S1471068419000267`
3. Alferes, J.J., Gonçalves, R., Leite, J.: Equivalence of defeasible normative systems. J. Appl. Non Class. Logics **23**(1-2), 25–48 (2013)
4. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Proc. AAAI. pp. 2669–2678 (2018)
5. Benzmüller, C., Parent, X., van der Torre, L.W.N.: A deontic logic reasoning infrastructure. In: Manea, F., Miller, R.G., Nowotka, D. (eds.) Conference on Computability in Europe, CiE 2018, LNCS, vol. 10936, pp. 60–69. Springer (2018). `https://doi.org/10.1007/978-3-319-94418-0_6`
6. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011)
7. Broersen, J.M., van der Torre, L.W.N.: Ten problems of deontic logic and normative reasoning in computer science. In: Bezhanishvili, N., Goranko, V. (eds.) Lectures on Logic and Computation - ESSLLI 2010. LNCS, vol. 7388, pp. 55–88. Springer (2011)
8. Cabalar, P., Fandinno, J., Fariñas del Cerro, L.: Autoepistemic answer set programming. Artificial Intelligence **289**, 103382 (2020). `https://doi.org/10.1016/j.artint.2020.103382`
9. Cabalar, P., Fandinno, J., Fariñas del Cerro, L., Pearce, D.: Functional ASP with intensional sets: Application to Gelfond-Zhang aggregates. Theory and Practice of Logic Programming **18**(3-4), 390–405 (2018). `https://doi.org/10.1017/S1471068418000169`
10. Cabalar, P., Fandinno, J., Muñiz, B.: A system for explainable answer set programming. In: Ricca, F., Russo, A., Greco, S., Leone, N., Artikis, A., Friedrich, G., Fodor, P., Kimmig, A., Lisi, F.A., Maratea, M., Mileo, A., Riguzzi, F. (eds.) Proceedings of the 36th International Conference on Logic Programming (Technical Communications). EPTCS, vol. 325, pp. 124–136 (2020). `https://doi.org/10.4204/EPTCS.325.19`, `https://doi.org/10.4204/EPTCS.325.19`
11. Cabalar, P., Pearce, D., Valverde, A.: Reducing propositional theories in equilibrium logic to logic programs. In: Bento, C., Cardoso, A., Dias, G. (eds.) Progress in Artificial Intelligence, 12th Portuguese Conference on Artificial Intelligence, EPIA 2005. LNCS, vol. 3808, pp. 4–17. Springer (2005)
12. Chisholm, R.M.: Contrary-to-duty imperatives and deontic logic. Analysis **24**(2), 33–36 (1963)
13. Eiter, T., Gottlob, G.: Complexity results for disjunctive logic programming and application to nonmonotonic logics. In: Miller, D. (ed.) Logic Programming, Proceedings of the 1993 International Symposium. pp. 266–278. MIT Press (1993)
14. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine **37**(3), 53–68 (2016). `https://doi.org/10.1609/aimag.v37i3.2678`
15. Gabbay, D., Horty, J., Parent, X., van der Mayden, R., van der Torre, L. (eds.): Handbook of Deontic Logic and Normative Systems, Volume 2. College Publications (2021)

16. Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.): Handbook of Deontic Logic and Normative Systems. College Publications (2013)
17. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: Carro, M., King, A., Saeedloei, N., Vos, M.D. (eds.) Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs. OASIcs, vol. 52, pp. 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016). https://doi.org/10.4230/OASIcs.ICLP.2016.2, https://doi.org/10.4230/OASIcs.ICLP.2016.2
18. Gebser, M., Maratea, M., Ricca, F.: The seventh answer set programming competition: Design and results. Theory and Practice of Logic Programming **20**(2), 176–204 (2020). https://doi.org/10.1017/S1471068419000061, https://doi.org/10.1017/S1471068419000061
19. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**(3/4), 365–386 (1991). https://doi.org/10.1007/BF03037169
20. Gelfond, M., Lobo, J.: Authorization and obligation policies in dynamic systems. In: de la Banda, M.G., Pontelli, E. (eds.) Logic Programming, 24th International Conference, ICLP 2008. LNCS, vol. 5366, pp. 22–36. Springer (2008)
21. Giordano, L., Martelli, A., Dupré, D.T.: Temporal deontic action logic for the verification of compliance to norms in ASP. In: Francesconi, E., Verheij, B. (eds.) International Conference on Artificial Intelligence and Law, ICAIL 2013. pp. 53–62. ACM (2013)
22. Gonçalves, R., Alferes, J.J.: An embedding of input-output logic in deontic logic programs. In: Ågotnes, T., Broersen, J.M., Elgesem, D. (eds.) DEON 2012, Proceedings. LNCS, vol. 7393, pp. 61–75. Springer (2012)
23. Gonçalves, R., Alferes, J.J.: Deontic logic programs. In: Gini, M.L., Shehory, O., Ito, T., Jonker, C.M. (eds.) International conference on Autonomous Agents and Multi-Agent Systems. pp. 1333–1334. IFAAMAS (2013)
24. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Computing strong and weak permissions in defeasible logic. Journal of Phil. Logic **42**(6), 799–829 (2013). https://doi.org/10.1007/s10992-013-9295-1
25. Governatori, G., Rotolo, A., Calardo, E.: Possible world semantics for defeasible deontic logic. In: Ågotnes, T., Broersen, J.M., Elgesem, D. (eds.) DEON 2012, Proceedings. LNCS, vol. 7393, pp. 46–60. Springer (2012)
26. Governatori, G., Rotolo, A., Riveret, R.: A deontic argumentation framework based on deontic defeasible logic. In: Miller, T., Oren, N., Sakurai, Y., Noda, I., Savarimuthu, B.T.R., Son, T.C. (eds.) PRIMA 2018, Proceedings. LNCS, vol. 11224, pp. 484–492. Springer (2018)
27. Hansson, B.: An analysis of some deontic logics. Nôus **3**, 373–398 (1969)
28. Heyting, A.: Die formalen Regeln der intuitionistischen Logik. Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse pp. 42–56 (1930)
29. Horty, J.F.: Deontic logic as founded on nonmonotonic logic. Ann. Math. Artif. Intell. **9**(1-2), 69–91 (1993). https://doi.org/10.1007/BF01531262
30. Kamp, H.: Free choice permission. In: Proceedings of the Aristotelian Society, vol. 74, pp. 57–74 (1973)
31. Lam, H.P., Governatori, G.: The making of SPINdle. In: Proc. of RuleML 2009: the International Symposium of Rule Interchange and Applications. LNCS, vol. 5858, pp. 315–322. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04985-9

32. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic **7**(3), 499–562 (2006). https://doi.org/10.1145/1149114.1149117
33. McCarthy, J.: Elaboration tolerance (1998), http://www-formal.stanford.edu/jmc/elaboration.html
34. Nute, D. (ed.): Defeasible Deontic Logic. Kluwer, Dordrecht (1997)
35. Nute, D.: Defeasible logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming. Oxford University Press (1993)
36. Odintsov, S.P., Pearce, D.: Routley semantics for answer sets. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) Proc. of the 8th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning, LPNMR 2005, LNCS, vol. 3662, pp. 343–355. Springer (2005)
37. Parent, X.: Preference-based semantics for hansson-type dyadic deontic logics: A survey of results. In: Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.) Handbook of Deontic Logic and Normative Systems Volume 2. pp. 7–70 (2021)
38. Parent, X., van der Torre, L.: I/O logics with a consistency check. In: Broersen, J.M., Condoravdi, C., Shyam, N., Pigozzi, G. (eds.) Deontic Logic and Normative Systems - 14th International Conference, DEON 2018. pp. 285–299. College Publications (2018)
39. Pearce, D.: A new logical characterisation of stable models and answer sets. In: NMELP. LNCS, vol. 1216, pp. 57–70. Springer (1997)
40. Pearce, D., Valverde, A.: Quantified equilibrium logic and foundations for answer set programs. In: de la Banda, M.G., Pontelli, E. (eds.) Logic Programming, 24th International Conference, ICLP 2008, Proceedings. LNCS, vol. 5366, pp. 546–560. Springer (2008). https://doi.org/10.1007/978-3-540-89982-2_46
41. Prakken, H., Sergot, M.: Dyadic deontic logic and contrary-to-duty obligations. In: Nute, D. (ed.) Defeasible Deontic Logic. pp. 223–262. Dordrecht (1997). https://doi.org/10.1007/978-94-015-8851-5_10
42. Reiter, R.: A logic for default reasoning. Artificial Intelligence **13**(1–2), 81–132 (1980)
43. Sergot, M.: Normative positions. In: Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.) Handbook of Deontic Logic and Normative Systems Volume 1. pp. 353–406 (2013)
44. Vakarelov, D.: Notes on N-lattices and constructive logic with strong negation. Studia logica **36**(1-2), 109–125 (1977)
45. von Wright, G.H.: Deontic logic. Mind **60**(237), 1–15 (1951)