Deontic Paradoxes in ASP with Weak Constraints*

Christian Hatschka Agata Ciabattoni Thomas Eiter
Institute of Logic and Computation, TU Wien, Vienna, Austria
firstname.lastname@tuwien.ac.at

The rise of powerful AI technology for a range of applications that are sensitive to legal, social, and ethical norms demands decision-making support in presence of norms and regulations. Normative reasoning is the realm of deontic logics, that are challenged by well-known benchmark problems (deontic paradoxes), and lack efficient computational tools. In this paper, we use Answer Set Programming (ASP) for addressing these shortcomings and showcase how to encode and resolve several well-known deontic paradoxes utilizing weak constraints. By abstracting and generalizing this encoding, we present a methodology for translating normative systems in ASP with weak constraints. This methodology is applied to "ethical" versions of Pac-man, where we obtain a comparable performance with related works, but ethically preferable results.

1 Introduction

Norms, which involve concepts such as obligation and permission, are an integral part of human society. They are enormously important in a variety of fields – from law and ethics to artificial intelligence (AI). In particular, imposing norms – be they ethical, legal or social – on AI systems is crucial, as these systems have become ubiquitous in our daily routines.

A main difference between norms and other constraints lies in the fact that norms typically allow for the possibility of violation. Reasoning with and about norms (*normative reasoning*) requires deontic logic, the branch of logic that deal with obligation and related concepts. Normative reasoning comes with a variety of idiosyncratic challenges, which are often exemplified by benchmark examples (so called deontic paradoxes). A crucial challenge is reasoning about sub-ideal situations, such as *contrary-to-duty (CTD) obligations*, which are obligations only triggered by a violation. Other challenges are associated, e.g., with defeasibility issues (norms having different priorities, exceptions, etc.).

The first deontic system introduced – Standard Deontic Logic [27] – was failing most of the benchmark examples, (un)deriving formulas which are counterintuitive in a common-sense reading. This has motivated the introduction of a plethora of deontic logics, see, e.g. [10]. These logics have been investigated mainly in connection with philosophy and legal reasoning, and with the exception of Defeasible Deontic Logic DDL [14, 15], they lack defeasibility and efficient provers. Defeasibility and efficient reasoning methods are instead offered by Answer Set Programming (ASP), which is one of the most successful paradigms of knowledge representation and reasoning for declarative problem solving [7]. Indeed, in a long and systematic effort of the knowledge representation community, efficient solvers for fast evaluation of ASP programs have been developed, see, e.g., [18]. Defeasibility is also inherent in ASP, due to its default-negation and also weak constraints. This paper introduces a method for using weak constraints for encoding norms in ASP. We first translate desired basic properties of deontic operators in a common core that will be used in all further encodings. These properties are established by analyzing multiple well-known deontic paradoxes (e.g., *Ross Paradox*, the *Fence Scenario*,...). By abstracting and generalising

*Work partially supported by the WWTF project TAIGER (ICT22-026).

To appear in EPTCS.

the encodings of the specific paradoxes, we provide a methodology for encoding normative systems in ASP with weak constraints. The methodology is put to work on a case study from [23, 22] that involves a reinforcement learning agent playing a variant of the Pac-man video game with additional "ethical" rules. Our encoding is used as a "shield" to filter out the non compliant actions of Pac-man and the outcome is compared with [22], that uses DDL. For space reason, we must omit details, for which we refer to [16].

2 Preliminaries

Answer Set Programming. We consider extended logic programs with disjunction [11, 20], which are finite sets of rules r (henceforth referred to as programs)

$$H_1 \vee \ldots \vee H_l :- A_1, \ldots A_n,\ not\ B_1, \ldots,\ not\ B_m. \quad l,m,n \geq 0, \tag{1}$$

where all H_i, A_j, and B_k are literals in a first-order language. Here *not* denotes weak (default) negation and $\neg$ (also written $-$) strong negation. Informally, r can be read as: "If all A_i are true and for no B_j there is evidence that it is true, at least one of $H_1, \ldots, H_l$ must be true".

The answer sets of a ground (variable-free) program Π are given in terms of consistent sets S of ground literals as follows. Let $S \models r$ denote that $S \models B(r)$ implies $S \models H(r)$, where $S \models B(r)$ denotes that $\{A_1, \ldots, A_n\} \subseteq S$ and $\{B_1, \ldots, B_m\} \cap S = \emptyset$, and $S \models H(r)$ denotes that $\{H_1 \ldots, H_l\} \cap S \neq \emptyset$. S is an answer set of Π if S satisfies all rules r in $\Pi(S) = \{r \in \Pi \mid S \models B(r)\}$, and no $S' \subset S$ satisfies $\Pi(S)$. The answer sets of a general program Π are those of its grounding $grd(\Pi)$ that consists of all ground instances of its rules. For modeling defeasibility of obligations, we use weak constraints of the form

$$:\sim A_1, \ldots, A_n.\ [w : l]$$

(as in the system DLV[20]) where $A_1, \ldots, A_n$ are literals (that may be weakly negated) and $w, l \geq 0$ are the weight and the (integer) level of the weak constraint, respectively. Informally, weak constraints single out optimal answer sets that have minimal total weights of violated weak constraints with higher levels being more important; see [20] for formal definitions and more background.

SDL. Standard Deontic Logic [27], is the best known system of deontic logic.

SDL formulas are constructed by the following grammar ($\mathscr{A}$ is the set of atomic propositions):

$$\varphi := p \in \mathscr{A} \mid \neg\varphi \mid (\varphi) \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid O\varphi \mid P\varphi \mid F\varphi$$

SDL is a monadic deontic logic, as the operators O (obligation), P (permission) and F (prohibition) apply to single formulas; they are read as "it is obligatory that φ", "it is permissible that φ", and "it is forbidden that φ", resp., and inter-definable, e.g., $P\varphi := \neg O(\neg\varphi)$, and $F\varphi := O\neg\varphi$.

The semantics of SDL – also known as the modal logic KD – is based on possible worlds, where the accessibility relation is serial. A Hilbert system for SDL is obtained by adding the following axiom-schemata and rules to any axiomatization of classical propositional logic:

If φ is a theorem, $O\varphi$ is a theorem	(RND)		
$O(\varphi \rightarrow \psi) \rightarrow (O\varphi \rightarrow O\psi)$	(KD)	$O\varphi \rightarrow \neg O\neg\varphi$	(DD)

In the following we will also use the derived axiom and rule:

If $\varphi \rightarrow \psi$ is a theorem, $O\varphi \rightarrow O\psi$ is a theorem	(RMD)	$\neg O\bot$	(OD)

For more details about SDL and other deontic logics, see, e.g., [10]

3 Deontic Paradoxes

As the foundation for our work, we examine several deontic paradoxes. These consist of (un)derivable formulas which are counter-intuitive when viewed from a common-sense perspective. While referred to in the literature as paradoxes, many of them are not paradoxes *per se*, but rather puzzles or dilemmas. Deontic paradoxes play an important role in deontic logic and normative reasoning; they serve as sanity checks for existing systems, and as driving force for defining new systems. In particular, they exemplify that SDL fails to capture the nuances of normative reasoning expected in certain scenarios. There are many such paradoxes. We categorise them according to the reason for their failure, see e.g. [17, 10], and analyze one example for each class:

1. Paradoxes centering around RMD: *Ross's Paradox*, *Good Samaritan Paradox*, *Åqvist's Paradox of Epistemic Obligation*
2. Puzzles centering around DD and OD: *Sartre's Dilemma*, *Plato's Dilemma*
3. Puzzles centering around deontic conditionals: *Broome's Counterexample*, *Chisholm's Contrary-to-Duty Paradox*, *Forrester's Paradox*, *Considerate Assassin Paradox*, *Asparagus Paradox*, *Fence Paradox*, *Alternative Service Paradox*

Deontic conditionals refer to obligations that arise situationally; written as $O(A/B)$ (to be read as "A is obligatory if B") they have been introduced to cope with contrary-to-duty obligations, i.e., obligations which come into force when another obligation is violated.

Paradoxes centering around RMD show, in general, that SDL is too strong as it derives unwanted consequences. An example is *Ross's Paradox*, which consists of the following two sentences:

It is obligatory that the letter is mailed. (R1)

It is obligatory that the letter is mailed or burned. (R2)

Let $O(m)$ and $O(m \vee b)$ formalize (R1) and (R2), respectively. As $m \to (m \vee b)$ is a theorem in SDL, $O(m \vee b)$ follows from $O(m)$ by RMD and modus ponens. But it seems counterintuitive to derive an obligation that is satisfied by burning the letter, when failing to mail the letter.

Puzzles centering around DD and OD involve conflicting obligations that cannot be obeyed. An example is *Plato's Dilemma*:

It is obligatory that I meet my friend for dinner. (P1)

It is obligatory that I rush my child to the hospital. (P2)

Clearly, it is not possible to satisfy both obligations at the same time. Using common sense reasoning, P2 should override P1, but in SDL the presence of two contradictory obligations would make everything obligatory (deontic explosion).

Puzzles centering around deontic conditionals have as a prominent example the *Fence Scenario* [24], which combines two different weaknesses of SDL regarding CTD obligations and exceptions:

There must be no fence. (F1)

If there is a fence then it must be a white fence. (F2)

If the cottage is by the sea, there may be a fence. (F3)

Figure 1: The common core of our encodings

$$O(X) \vee -O(X) :- act(X). \quad (1)$$
$$F(X) \vee -F(X) :- act(X). \quad (2)$$
$$:- O(X), -Dia(X). \quad (3)$$
$$-Dia(X) :- -Do(X), act(X). \quad (4)$$
$$:- O(X), F(X). \quad (5)$$
$$Do(X) \vee -Do(X) :- act(X). \quad (6)$$
$$:- F(X), Do(X). \quad (7)$$
$$Happens(X) :- Do(X). \quad (8)$$
$$:- Do(X), -Dia(X). \quad (9)$$
$$:\sim O(X).[1:1] \quad (10)$$
$$:\sim F(X).[1:1] \quad (11)$$

Here (F2) serves as a CTD obligation that is active when (F1) is violated, while (F3) serves as an exception to (F1). Note that under this interpretation, if the cottage is by the sea the fence need not be white. The contrary-to-duty obligation (F2) cannot be formalised in SDL: having a white fence implies having a fence, and by (RND) the obligation to have a fence; this contradicts (F1). Moreover, as SDL lacks expressing defeasibility, (F3) cannot be properly formalized.

4 Encoding the Paradoxes

We now proceed to encode the paradoxes from above All encodings share the same common core, shown in Figure 1, that encodes properties of SDL, using the following predicates:

- $O(X)$ resp. $F(X)$ denotes that X is obligatory resp. forbidden;
- $act(X)$ denotes that X is eligible for reasoning about whether X is obligatory or not. While an action by default, in some cases X may not materialize but viewed as such. An example from above would be owning a white fence, as we reason about whether it is obligatory.
- $Do(X)$ denotes that the agent has chosen to take the action X, and $-Do(X)$ denotes that the agent will definitely not take the action X.
- $Dia(X)$ is an auxiliary predicate to denote that an action X is an option resp. possible (in the sense on modal logic). Thus, $-Dia(X)$ can either mean that the agent cannot take the action or that the agent has chosen not to take the action.
- $Happens(X)$ is an auxiliary predicate that denotes an event X happening. It is sometimes used in encodings to denote events happening which are usually outside the agents control.

Intuitively, the common core guesses whether something is obligatory (1), forbidden (2) and whether the agent takes the action (6). The remaining rules then encode connections between predicates and exclude answer sets that we deem inconsistent, e.g., something being obligatory and forbidden or something being obligatory and that action not taken. The weak constraints (10) and (11) are used to eliminate answer sets that derive obligations/prohibitions with no need.

We note that the common core is not a faithful encoding of full SDL and its axioms: while theoretically possible, this would be undesired as SDL axioms lead to multiple paradoxes. Instead, the common core captures some of the SDL axioms, while satisfactory handling the deontic paradoxes. For instance

Proposition 1. *The SDL axiom DD holds in the common core.*

Proof. *DD* is formalized in rule (5), that forbids an action from being both forbidden and obligatory, as $O\varphi \rightarrow \neg O\neg\varphi$ which is equivalent to $\neg O\varphi \vee \neg F\varphi$, and to $\neg(O\varphi \wedge F\varphi)$. □

The idea behind the common core is to generate all maximal sets of non-auxiliary predicates that are consistent and filter out suboptimal answer sets using weak constraints. We require that in a consistent set an action is not obligatory and forbidden at the same time and any obligatory action is taken (resp. any forbidden action is not taken). For instance, disregarding the auxiliary predicates *Dia*, $Happens$ and *act*, the following are all maximal consistent sets of deontic predicates:

$$\{O(action), -F(action), Do(action)\}, \{F(action), -O(action), -Do(action)\},$$
$$\{-F(action), -O(action), -Do(action)\}, \{-F(action), -O(action), Do(action)\}$$

The next proposition can be seen as a soundness and completeness result for the common core.

Proposition 2. *The rules* (1)–(9) *from Figure 1 allow for all and only the maximal consistent sets of deontic predicates as answer sets.*

This can be seen by inspecting the answers sets output by a solver. Intuitively, soundness of the rules (1) to (9) is achieved, as inconsistent answer sets are excluded. Completeness on the other hand is given, as all answer sets that are considered consistent for an action are generated.

Note that in Prop. 2 the weak constraints (10) and (11) are excluded: as every answer set for them would represent an optimal way to handle given norms, they would eliminate every answer set containing obligations or prohibitions, and completeness would be lost. By including them, preference will be given to unrestricted behaviour, in interplay with possible further rules.

4.1 Paradox encodings

We describe how to extend the common core to encode selected paradoxes from above.

Ross's Paradox. In contrast with what happens in SDL, we do not want to derive (R2) in the ASP encoding. To achieve this, we add the following rule and facts to the core:

$$:\sim -O(mail).\ [1:2] \quad (12) \qquad act(mail). \quad (13) \qquad act(burn). \quad (14)$$

Note that a disjunction over obligations is represented by two different answer sets that each contain one possible way to satisfy the obligation over the disjunction.

The obligation $(R1)$ is created using the weak constraint (12), while the facts (13) and (14) declare *mail* and *burn* as actions to reason about. For *mail* (resp. *burn*) the core encoding guesses it as obligatory or not; (12) may penalise the guess at the highest level if the program does not entail mailing the letter as obligatory. As constraint violation at the highest level is minimised, each optimal answer set includes the obligation to mail the letter, should such an answer set exist.

The program has two answer sets; none of them derive the obligation to burn the letter, and they only differ for the choice of the agent to burn the letter or not. By adding a rule specifying that it is not possible to perform both acions: burn the letter and mail it, the answer set where the agent chooses to burn the letter would not be derived.

Plato's Dilemma. Recall that the desired outcome of this dilemma would be that the agent takes her child to the hospital, thereby violating the obligation of meeting her friend for dinner. The encoding presents two interesting aspects: prioritisation of the obligations and the impossibility of taking both actions. This can be encoded as follows:

$:\sim -O(help), Happens(emergency).[1:3]$	(20)	$act(help).$	(23)
$:\sim -O(meet).[1:2]$	(21)	$:- Do(help), Do(meet).$	(24)
$act(meet).$	(22)	$Happens(emergency).$	(25)

The weak constraint (20) is at level 3, the highest in this encoding, and penalises answer sets in which $Happens(emergency)$ is true but the obligation to help is not derived. In other words, it derives the obligation $(P2)$ to help the child in case of an emergency. The weak constraint (21) encodes the obligation $(P1)$ to meet the friend for dinner, but at a lower level (viz. 2), which gives priority to $(P2)$. The constraint (24) encodes the impossibility of taking both actions. With the assertions that an emergency occurs and *meet* and *help* are the possible actions, as desired, a single answer set exists containing the obligation to help the child.

Fence Paradox One might think that CTD obligations could be handled like exceptions to obligations. While one could accordingly state "There may be a fence if it is white", it would not have the same meaning as in the paradox. Handling a CTD obligation as an exception results in a loss the original obligation to a certain degree; it could in this case be seen as *the least thing to do to set things right*. While having a white fence improves the situation, the presence of the fence itself remains undesirable [24].

The important fact to consider is that should the cottage be by the sea, then as (F1) is not active due to (F3), the fence need not be white. To this end, we add the following to the common core:

$$:\sim -F(have\_fence), not\ Location(sea).\ [1:2] \quad (30)$$

$$:\sim Do(have\_fence), not\ Location(sea), \quad (31)$$
$$-O(have\_white\_fence).\ [1:2]$$

$$act(have\_fence). \quad (32)$$

$$act(have\_white\_fence). \quad (33)$$

Here (30) caters for (F1) and (31) for (F2); notably, (F3) also affects the CTD obligation (F2). This is needed as the fence has to be white only when the cottage is not by the sea. Otherwise the obligation for the fence to be white would also be derived if the cottage was by the sea.

To check whether the obligation for the fence to be white is deduced when the cottage is by the sea (and we have a fence), we further add:

$$Location(sea). \qquad Do(have\_fence).$$

Then two answer sets exist; both do not derive the obligation for the fence to be white. When testing other constellations, the answer sets obtained also represent the expected results.

5 Generalisation and Methodology

We can classify the obligations that appeared in the paradoxes into the following classes:

- **Regular obligations:** These obligations should be followed as long as possible (without violating a more important obligation).
- **Conditional obligations:** These are obligations that only need to be followed given certain preconditions. E.g., the obligation to wear a suit when at a formal event.
- **Obligations over disjunctions:** obligations that are fulfilled by satisfying any disjunct that constitutes the obligation; e.g., bring dessert or salad.
- **Conjunctions of obligations that all need to be satisfied:** obligations consisting of multiple parts where satisfying all parts is necessary.
- **Obligations with exceptions:** obligations to be followed unless an exception is given.
- **Contrary-to-duty obligations:** obligations that arise as another obligation is violated.

Type of obligation	Encoding
Regular	$:\sim -O(o).\ [w:l]$
Conditional	$:\sim -O(o), condition.\ [w:l]$
Disjunction	$:\sim -O(o_1), -O(o_2), \ldots, -O(o_n).\ [w:l]$
Conjunction	$:\sim not\,Conj.\ [w:l]$
	$Conj: -O(o_1), \ldots, O(o_n).$
Exceptions	$:\sim -O(o),\ \ not\,Exception.\ [1:2]$
Contrary-to-duty	$:\sim -Do(o_1), -O(o_2).\ [1:2]$

Table 1: Encodings for different types of obligations

Note that prohibitions are viewed as regular *negative* obligations, i.e. the obligation not to do something. The different kinds of obligations are encoded as shown in Table 1. Their encoding uses weak constraints to model defeasibility. An obligation to take an action a that should always hold is encoded in the following way:

$$:\sim -O(a).\ [w:l]$$

Note that the weight w and the level l of the weak constraint depend on the importance of the obligation and conflicting obligations. In most cases, $w = 1$ and merely l is used to encode priorities among obligations. Conflicts between obligations are detected and the priority among obligations is established through weak constraints (more important obligations have higher level).

By generalising the encodings of the considered paradoxes (from Section 3), we propose the following encoding methodology that consists of the following steps:

Step 1. For each of the norms determine what kind of obligation it represents, among the six different kinds of obligations we have considered.

Step 2. Determine which actions are simultaneously incompatible. Knowing which actions are in conflict eases determining the importance of the obligations. Incompatibility needs to be determined through context. E.g., while in general it is possible to watch a movie while browsing the internet, these actions may be incompatible on an old smartphone.

Step 3. Encode the different kinds of obligations and their importance. Here weights as priorities play an important role. There are two cases we need to consider:

Case 1. An obligation (of whatever kind) is more important than the other. In this case, setting the level of one constraint higher than the other is sufficient. E.g., consider the case of two obligations o_1 and o_2 where the latter is more important. If o_1 and o_2 have no special properties, the encoding is: (with $j \geq 1$)

$$:\sim -O(o_1).\ [1:l] \qquad :\sim -O(o_2).\ [1:l+j]$$

Note that j may differ given additional obligations. We account for this as follows.

Suppose the incompatibilities between actions are given, as well as the importance of obligations by a preference $O' \succ O$ stating that the obligation O' is strictly more important than O. We generate a directed graph $G = (V,E)$ whose vertices $V = \{O_1, \ldots, O_n\}$ are the obligations having the edges $E = \{(O_i, O_j) \in V^2 \mid O_i \succ O_j\}$; note that G must be acyclic.[1] The sinks of G, i.e., vertices with no

[1]For Non-strict preference $\succeq$, we can use the supergraph of G, whose nodes cluster all equally preferable obligations.

outgoing arcs, are assigned priority $p_1 = 2$. After simultaneously removing all sinks, we iterate the process with increased priority, i.e., assign the new sinks priorities $p_2 = 3$, $p_3 = 4$ etc.; this results in a priorization by levels.

Case 2. Multiple obligations $o_1, \dots o_n$ are in conflict with an obligation o. While satisfying o is better than satisfying a single o_i, satisfying multiple o_i's may be equally good or better than satisfying o. In this case, we can use the weights of the weak constraints for $o_1, \dots, o_n$ to encode this: they must then correspond to their importance and o must have a weight that is equal or smaller than the joint weights.

For example, if o_1, o_2 and o_3 are mutually non-exclusive and equally important to o if all are satisfied, the following weights could be chosen, for a number $k \geq 1$ (the level is the same):

$$:\sim -O(o_1).\ [k:l] \qquad :\sim -O(o_2).\ [k:l] \qquad :\sim -O(o_3).\ [k:l] \qquad :\sim -O(o).\ [3k:l]$$

Step 4. Encode the exclusion of combinations of actions found incompatible. If two actions a_1 and a_2 are incompatible, this is encoded by adding the following constraint:

$$:- Do(a_1), Do(a_2).$$

Step 5. Encode additional information. This includes denoting constants as actions using the predicate *act*, and specifying dependencies between actions; e.g., if the action running entails the action moving, we add a rule

$$Do(move) :- Do(run).$$

Observation 1. *The common core simulates the SDL axioms RND and KD.*

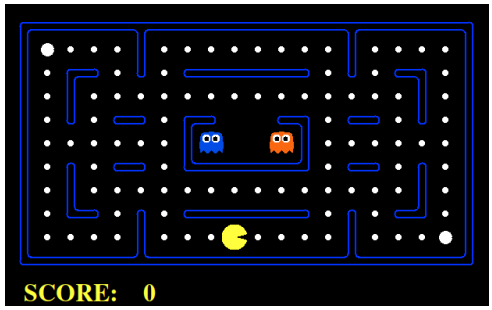
As there are no theorems in our framework, *RND* is not directly implemented. However, if we read a theorem as something than cannot be violated, *RND* becomes: "If it is impossible to violate φ, then φ is obligatory." In our semantics this obligation would not be derived; however the obligation to take the action φ is given indirectly as the agent must do φ (that cannot be violated). For *KD*, recall that if an obligation is in an answer set, the agent has to take the corresponding action. We encode $O(\varphi \to \psi)$ using a weak constraint that sanctions answer sets including $Do(\varphi)$ but not $O(\psi)$. As every answer set that contains $O(\varphi)$ also contains $Do(\varphi)$, each answer set that contains $O(\varphi)$ but not $O(\psi)$ is also sanctioned. This way we simulate *KD*, as an answer set that contains $O(\varphi)$ must also contain $O(\psi)$, unless the latter conflicts with an obligation of higher or equal importance.

6 A Case Study: Ethical Pac-man

We put the methodology described in the previous section to work on a reinforcement learning agent playing variants of the game Pac-man. Pac-man features a closed environment with simple game mechanics and parameters which are easy to manipulate, and extend with norms that can simulate normative conflicts.

The starting position of the game is depicted in Fig. 2. Pac-man's objective is to eat all the pellets in the maze while avoiding two ghosts (orange and blue) that will kill him upon contact. Pac-man and the ghosts typically move one step at a time, with the ghosts' movements being non-deterministic. If Pac-man ate one of the larger pellets, the ghosts enter a scared state and become vulnerable, allowing Pac-man to eat them. In this state, the ghosts move at half speed. A scared ghost is instantly eaten by Pac-man when their distance is less than 1 on both axes. Points are awarded for consuming pellets and ghosts, with a discount applied based on the duration of the game (the longer the game lasts, the lower the score). Pac-man wins if he collects all the pellets, and a faster completion time results in a higher score.

Figure 2: Pac-man



Following [23, 22], we consider variants of the Pac-man game with additional "ethical norms". *Vegan Pac-man*, in which Pac-man is not allowed to eat any ghost has been introduced in [23] and implemented there using multi-objective Reinforcement Learning (RL) with policy orchestration. Vegan Pac-man and its vegetarian variant were analyzed in [22], that will serve as the benchmark for our work. *Vegetarian Pac-man* can eat the orange ghost (as it would be cheese) but not the blue ghost. The approach used in [22] combines RL with formal tools for normative reasoning. The authors implement a logic-based *normative supervisor* module, which informs the trained RL agent of the ethical requirements in force in a given situation. At each step, Pac-man chooses an action complying with the norms, and a least evil action if there is no such action. Their approach allows to deal with complex normative systems, conflicting obligations, and situations where no compliance is possible. Norms and the current state of the agent's environment are encoded in defeasible deontic logic [14, 15], which is a deontic logic with defeasible rules that specify typical correlations, such as "birds usually fly"; its theorem prover SPINdle is used to check norms compliance. Exceptions are encoded by so-called defeaters, e.g., if the bird is a penguin.

We consider here an alternative realization of the normative supervisor, based on our norms encoding and using the DLV reasoner. We experimentally compare the obtained results and apply our methodology to more intricated "ethical norms" for Pac-man.

Vegan Pac-man: To prohibit Pac-man from eating any ghost, we can state:

$$O(\neg eat(g)) \qquad \text{respectively} \qquad F(eat(g)), \quad \text{for } g \in \{blue\_ghost, blue\_ghost\}.$$

Vegetarian Pac-man: To prohibit Pac-man from eating the blue ghost, we state:

$$O(\neg eat(blue\_ghost)) \qquad \text{respectively} \qquad F(eat(blue\_ghost)).$$

We encode the norm bases by forbidding Pac-man to move in a direction if the ghosts are scared and moving there could lead to eat a ghost. Furthermore, we forbid Pac-man from stopping if a ghost could move into Pac-man (and then be eaten). Pac-man may still eat a ghost. This can happen if both a ghost and Pac-man move towards a larger pellet from perpendicular directions. In this case, Pac-man will first eat the pellet and then the ghost. Furthermore, Pac-man could be cornered between two scared ghosts, leaving him no choice but eating one of them.

The scenarios that can precede Pac-man eating a ghost are the same for both norm bases. As Pac-man and the ghosts can move at most one step at a time, we can derive that the Manhattan distance between Pac-man and a scared ghost must in this case be at least 1 and at most 2 (coordinates are integers). This leaves three possibilities for their relative locations. We encoded the norms by accounting for the locations of the ghosts relative to Pac-man and forbidding Pac-man to make moves that could lead to eating a ghost.

Experimental Results. The vegan norm base was implemented in [23] who trained two different models for Pac-man; in one model he was trained to maximize the game score, and in the other to comply with the norms using respective data. An external function enabled the agent to decide which model to use for choosing the next move. When the importance of the norm-compliance model was low, the agent in general did not comply with the norm, resulting in around 2 ghosts eaten per game. Making the

norm base	% games won	game score avg[max]	avg ghosts eaten (blue/orange)	avg time (s)
Vegan	90.7 | 91.2	1209.86[1708] | 1217[1538]	0.023/0.02 | 0.013/0.018	10.1 | 6.7
Vegetarian	94.0 | 90.6	1413.80[1742] | 1366[1751]	0.01/0.79 | 0.001/0.788	9.8 | 6.5
Weak Vegan	| 89.9	| 1204[1731]	| 0.002/0.043	| 6.7

Table 2: Results for Neufeld et al.'s normative supervisor [22] | our encoding

importance sufficiently high, the number of ghosts eaten did decrease to one (similar to our results, see Table 2). In this approach, however, it is not clear how to enforce more complex norm sets, including, e.g., contrary-to-duty obligations.

In order to compare Neufeld et al.'s approach [22] with our ASP encodings, we utilized their Java-based framework. This framework is built in Java and provides the normative reasoners with a ready-to-use interface to the Pac-man game.[2] The RL agent was trained on 250 games and the normative supervisor was evaluated on 1000 games with the same initial state.

The results reported in [22] for their normative supervisor (using SPINdle) and for our ASP-based normative supervisor (using JDLV, the java framework for DLV) are shown in Table 2. The latter outperformed the original supervisor for both norm bases w.r.t. the ghosts eaten and speed: its average running time was less than 7 seconds, while for Neufeld et al.'s normative supervisor it was some seconds (roughly 50%) longer, using a Lenovo Y50-70 with 8GB RAM and Ubuntu 22.04 LTS. For the vegan norm base, our encoding resulted in a higher winning rate and an improved average score. (A game loss costs 500 points, resulting possibly in a negative score.) The higher maximum score in Neufeld et al.'s results is likely attributed to a game where both ghosts were eaten. In the vegetarian norm base, the lower winning rate and average score is probably due to Pac-man's preference to lose in our framework a game rather than eating a ghost.

Weak Vegan Pac-man. Consider the following more intricated variant of the vegan norm base:

O1 It is obligatory not to eat the blue ghost.

O2 It is obligatory not to eat the orange ghost, unless a ghost has already been eaten.

O3 It is obligatory to stop (for one move) after having eaten a ghost.

Thus, the behaviour may change at some point. To encode it, we follow our methodology:

Step 1: We categorise the obligations.

– O1 is a regular obligation. It has no exceptions and should be followed at all times.

– O2 is an obligation with an exception. Once the latter occurs, it holds until the game is over.

– O3 is a derived obligation. Note that it is not necessarily a CTD obligation, as eating a ghost may not violate an obligation if the ghost is blue.

Step 2: We look at pairs of obligations that cannot be fulfilled simultaneously.

– O1 and O2 could possibly be in conflict, as Pac-man may be stuck between two ghosts moving towards him. This may force him to eat one of them. In this case we prioritize O1, as we want preventing the blue ghost from being eaten to have the highest priority.

– O1 and O3 could be in conflict, as stopping after eating an orange ghost may lead to eating a blue ghost.

[2]For complete DLV code and experimental data, see `github.com/Chrisi-boop/DLV-Normative-Reasoning-`.

Figure 3: Encoding for the Pac-man norm base

$:\sim -F(eat(blue\_ghost))$. $[1:3]$
$:\sim not\ \ Exception,$
$\quad -F(eat(orange\_ghost))$. $[1:2]$
$:\sim -O(stop),$
$\quad Do(eat(orange\_ghost))$. $[1:2]$
$:\sim -O(stop),$
$\quad Do(eat(blue\_ghost))$. $[1:2]$

(a) obligations

$:- Do(stop),\ \ Do(north).$
$\quad \ldots$
$:- F(eat(blue\_ghost)),$
$\quad -F(east), Pacman(A,B),$
$\quad BlueGhost(C,D,1),$
$\quad E = C - A, E \leq 2,$
$\quad G = B - D, G \leq 1.$
$\quad \ldots$

(b) conflicting actions

$act(d)$. for $d \in D$
$act(eat(blue\_ghost))$.
$act(eat(orange\_ghost))$.
$:- \bigwedge_{d \in D} F(d)$.

(c) action specs; $D = \{stop, north, east, south, west\}$

– O2 and O3 cannot be in conflict, as after eating a ghost the exception to O2 is empowered. We give priority to O1 over O3 since the preservation of the blue ghost from being consumed has highest priority.

Summarizing the statements above, we obtain the following preferences: O1 $\succ$ O2, O1 $\succ$ O3.

Step 3: We derive the following weights and levels for the weak constraints corresponding to the obligations: O1 is assigned [1 : 3], O2 is assigned [1 : 2], and O3 is assigned [1 : 2].

We next look at the predicates used in the encoding:

- *Pacman(X,Y)* denotes the location of Pac-man. X and Y refer to his coordinates on the map.
- *BlueGhost(X,Y,B)* and *OrangeGhost(X,Y,B)* denotes the location of the blue resp. orange ghost, where X, Y are its coordinates and B is 1 if the ghost is scared and 0 otherwise.
- *Exception* means the agent has already eaten a ghost and may eat orange ghosts. Whenever *Exception* is in an answer set, the normative reasoner will inject it into any later answer set.

The actions to reason about are *stop*, *north*, *east*, *south*, *west*, *eat(blue_ghost)*, *eat(orange_ghost)*:

- *Do(stop)* means that the agent remains stationary for one action.
- *Do(d)*, $d \in \{north, east, south, west\}$, means that the agent will move in direction d.
- *Do(eat(g)*, $g \in \{blue\_ghost, orange\_ghost\}$ means that the agent eats that ghost.

To encode the norms we add the rules and facts in Fig. 3 to the common core. Following our methodology, we encode the obligations as in Fig. 3a. Note that the obligation to eat the blue and orange ghost are encoded as separate actions as two different obligations (O3).

Step 4: The conflicting actions from Step 2 are encoded in Fig. 3b. For Pac-man, we encode that he cannot take two actions at once, e.g., stop and move north. For ghosts, as a scared ghost may move both towards and away from Pac-man, we cannot encode ensuing conflicts directly. However, we can assert that permitting an action that could potentially involve consuming a ghost is inconsistent with the prohibition of consuming that ghost. Fig. 3.b shows one such rule; for space reasons we omit the other rules, as well as further action constraints for Pac-man.[2]

Step 5: We state the actions to reason about (Fig. 3c), and encode that it is not possible to prevent Pac-man from taking any action (i.e., Pac-man must either stop or move).

In fact, our actual encoding omits the actions $eat(blue\_ghost)$ and $eat(orange\_ghost)$; they are substituted by prohibitions on moving towards scared ghosts. We used them here for readability.

Experiments. The implementation of this norm base yielded results that align with our expectations, as depicted in Table 2. Pac-man consumed more ghosts compared to the vegan norm base but fewer than the vegetarian norm base, and mostly orange ghosts. The obligation to halt when Pac-man eats a ghost resulted in a lower average score, as time passage leads to point deductions. This observation also explains the marginal increase in the maximum score and the slight decrease in the winning rate. Despite the added complexity of the norm base, there was no significant difference in the running time.

7 Related Work and Conclusion

Starting with an analysis of well-known deontic paradoxes, we have introduced a methodology to encode normative systems in ASP, using DLV as the system of choice. Our approach determines optimal ways to handle scenarios, using agreed upon prioritizations of obligations.

Multiple approaches to implement normative systems do exist. Some of those related to the multi-agent systems community can be found, e.g., in [2]. We will discuss below the approaches most similar to ours.

One of the earliest works on encoding normative systems is Sergot et al. [25], who did encode the British Nationality Act in logic programming. However, their work focused on determining the applicability of the British Nationality Act to specific individuals, without delving into the reasoning about obligations or seeking optimal courses of action within the framework of given norms.

[26] introduced in IMPACT syntax and semantics for reasoning about obligations and prohibitions among agents in a rule-based language under different semantics, among them stable model semantics. Although they refer to deontic logic, the proposed way of dealing with conflicting obligations is to satisfy a maximal subset of obligations, without considering possible preferences among them.

An abductive logic programming framework has been employed in [19] to encode obligations in various deontic paradoxes. Rather than trying to derive all optimal ways of fulfilling given obligations, the authors focused on finding a best model of a definite Horn logic program that satisfies given goals. When it comes to establishing model preferences, the auxiliary symbols used in the encodings played a significant role. However, their usage requires care, and no systematic way of defining model preference was considered, for which our work could provide inspiration.

Using a combination of input-output logic and game theoretic methods, [6] encoded the behaviour of (multi-)agents subjected to a normative system. In their work, agents possess the ability to reason in a manner more akin to humans, for example, determining if the consequences of violating an obligation are worth the penalty. Their work lacks however computational support.

Temporal Logic has been used to simulate normative reasoning. Temporal Logic approaches offer an advantage in enforcing norms that indirectly restrict certain actions [1, 8]. Moreover there are advanced tools that effectively combine reinforcement learning, with, e.g., LTL (see [4]), and LDLf (see [9]). However, as is well known temporal logic cannot handle all the intricacies of normative reasoning, see, e.g., the discussion in [2, 21].

Different ASP approaches have been proposed in [13, 3, 12]. In these works, logic programs were extended with the SDL modality, and [12] introduces also temporal operators. Alferes et al. [3] further provides a way to check equivalence of two deontic logic programs. These approaches however require an understanding of the embedded logic and some implementation efforts, whereas our method can be used out of the box and with an ASP solver off the shelf.

In our approach, the selection of paradoxes has proven to be of utmost importance, as failure to include certain ones resulted in overlooking crucial aspects of normative systems; e.g., the analysis of the *Fence Paradox* has enabled us to differentiate between contrary-to-duty obligations and exceptions, which is a well-known problem in the field of normative reasoning [24]. An important feature of our approach is the availability of optimized tools (ASP solvers) and the simplicity of the encoding. There is a clear cut common core that is supplemented with defined ways for encoding different kinds of obligations. The approach, described for DLV in this work, can be easily transferred to other ASP solvers, e.g. clasp (`potassco.org/clasp`). A main limitation of our approach is that encoding intricate normative systems can result in large programs. Furthermore, obligations that require the agent to maintain certain conditions might be cumbersome to encode for complex scenarios where taking particular actions might indirectly lead to a violation. Consider for instance "See to it that the child stays dry." Here it is not enough to simply not get the child wet, but one must also take measures to protect the child from getting wet through other means. This may entail preventing the child from going outside if it is about to rain. For such normative systems, approaches extending ASP with temporal logics [12], may be preferable.

Outlook. A peculiar feature of our approach is that all encoded obligations are comparable based on their associated weights and priorities. While in our encoding we did not run into problems, it is possible that certain normative systems may require optimal answer sets encoding solutions that are inherently incomparable. Future work may also look into other ASP solvers with more sophisticated means for filtering out answer sets to model normative reasoning. The clasp extension asprin (`potassco.org/asprin`) or the DLV2 system using WASP [5], could serve as examples.

The Pac-man case study highlighted the fact that even simple obligations can require a substantial number of weak constraints to accurately encode the means of fulfilling those obligations. To address this weakness, we plan to explore the integration of our framework with auxiliary software capable of reasoning about the actions neeeded to fulfill specific obligations. In the context of Pac-man, such software might have to interpret, e.g., how the obligation to not eat a ghost could be fulfilled.

References

[1] Natasha Alechina, Nils Bulling, Mehdi Dastani & Brian Logan (2015): *Practical Run-Time Norm Enforcement with Bounded Lookahead*. In Gerhard Weiss, Pinar Yolum, Rafael H. Bordini & Edith Elkind, editors: *Proceedings of AAMAS 2015*, ACM, pp. 443–451.

[2] Natasha Alechina, Mehdi Dastani & Brian Logan (2018): *Norm Specification and Verification in Multi-Agent Systems*. *J. of Applied Logics* 5(2), pp. 457–490.

[3] José Júlio Alferes, Ricardo Gonçalves & João Leite (2013): *Equivalence of defeasible normative systems*. *J. Appl. Non Class. Logics* 23(1-2), pp. 25–48, doi:10.1080/11663081.2013.798996.

[4] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum & Ufuk Topcu (2018): *Safe reinforcement learning via shielding*. In: *Proc. AAAI*, pp. 2669–2678.

[5] Mario Alviano, Carmine Dodaro, João Marques-Silva & Francesco Ricca (2020): *Optimum stable model search: algorithms and implementation*. *J. Log. Comput.* 30(4), pp. 863–897, doi:10.1093/logcom/exv061.

[6] Guido Boella & Leendert W. N. van der Torre (2004): *Regulative and Constitutive Norms in Normative Multiagent Systems*. In Didier Dubois, Christopher A. Welty & Mary-Anne Williams, editors: *Proceedings of KR2004*, AAAI Press, pp. 255–266.

[7] Gerd Brewka, Thomas Eiter & Miroslaw Truszczyński (2011): *Answer Set Programming at a Glance*. *Communications of the ACM* 54(12), pp. 92–103, doi:10.1145/2043174.2043195.

[8] Nils Bulling, Mehdi Dastani & Max Knobbout (2013): *Monitoring norm violations in multi-agent systems*. In Maria L. Gini, Onn Shehory, Takayuki Ito & Catholijn M. Jonker, editors: *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13*, IFAAMAS, pp. 491–498.

[9] Giuseppe De Giacomo & Moshe Y. Vardi (2015): *Synthesis for LTL and LDL on Finite Traces*. In: *IJCAI 2015*, pp. 1558–1564.

[10] Dov Gabbay, John Horty, Xavier Parent, Ron van der Meyden & Leendert van der Torre, editors (2021): *Handbook of Deontic Logic and Normative Systems*. College Publications. Amends Volume 1 (2013).

[11] M. Gelfond & V. Lifschitz (1991): *Classical Negation in Logic Programs and Disjunctive Databases*. *New Generation Computing* 9, pp. 365–385, doi:10.1007/BF03037169.

[12] Laura Giordano, Alberto Martelli & Daniele Theseider Dupré (2013): *Temporal deontic action logic for the verification of compliance to norms in ASP*. In Enrico Francesconi & Bart Verheij, editors: *International Conference on Artificial Intelligence and Law, ICAIL '13*, ACM, pp. 53–62, doi:10.1145/2514601.2514608.

[13] Ricardo Gonçalves & José Júlio Alferes (2012): *An Embedding of Input-Output Logic in Deontic Logic Programs*. In Thomas Ågotnes, Jan M. Broersen & Dag Elgesem, editors: *Deontic Logic in Computer Science - 11th International Conference, DEON 2012, Proceedings*, *LNCS* 7393, Springer, pp. 61–75, doi:10.1007/978-3-642-31570-1_5.

[14] Guido Governatori, Francesco Olivieri, Antonino Rotolo & Simone Scannapieco (2013): *Computing Strong and Weak Permissions in Defeasible Logic*. *J. Phil. Log.* 42(6), pp. 799–829, doi:10.1007/s10992-013-9295-1.

[15] Guido Governatori & Antonino Rotolo (2008): *BIO logical agents: Norms, beliefs, intentions in defeasible logic*. *Auton. Agents Multi Agent Syst.* 17(1), pp. 36–69, doi:10.1007/s10458-008-9030-4.

[16] Christian Hatschka (2022): *Representing Normative Reasoning in Answer Set Programming Using Weak Constraints*. Master's thesis, Technische Universität Wien, doi:10.34726/hss.2022.99420.

[17] Andrew Jones & José Carmo (2002): *Deontic Logic and Contrary-to-Duties*. *Handbook of Philosophical Logic* vol.8, pp. p.265–364, doi:10.1007/978-94-010-0387-2_4.

[18] Benjamin Kaufmann, Nicola Leone, Simona Perri & Torsten Schaub (2016): *Grounding and Solving in Answer Set Programming*. *AI Mag.* 37(3), pp. 25–32, doi:10.1609/aimag.v37i3.2672.

[19] Robert A. Kowalski & Ken Satoh (2018): *Obligation as Optimal Goal Satisfaction*. *J. Phil. Log.* 47(4), pp. 579–609, doi:10.1007/s10992-017-9440-3.

[20] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri & Francesco Scarcello (2006): *The DLV System for Knowledge Representation and Reasoning*. *ACM Transactions on Computational Logic* 7(3), pp. 499–562, doi:10.1145/116825.116838.

[21] Emery Neufeld, Ezio Bartocci & Agata Ciabattoni (2022): *On Normative Reinforcement Learning via Safe Reinforcement Learning*. In: *Proceedings of PRIMA 2022*.

[22] Emery A. Neufeld, Ezio Bartocci, Agata Ciabattoni & Guido Governatori (2021): *A Normative Supervisor for Reinforcement Learning Agents*. In André Platzer & Geoff Sutcliffe, editors: *Automated Deduction - CADE 28*, *LNCS* 12699, Springer, pp. 565–576, doi:10.1007/978-3-030-79876-5_32.

[23] Ritesh Noothigattu, Djallel Bouneffouf, Nicholas Mattei, Rachita Chandra, Piyush Madan, Kush R. Varshney, Murray Campbell, Moninder Singh & Francesca Rossi (2019): *Teaching AI Agents Ethical Values Using Reinforcement Learning and Policy Orchestration*. In: *Proc of IJCAI 2019*, ijcai.org, doi:10.24963/ijcai.2019.

[24] Henry Prakken & Marek J. Sergot (1996): *Contrary-to-Duty Obligations*. *Stud Logica* 57(1), pp. 91–115, doi:10.1007/BF00370671.

[25] Marek J. Sergot, Fariba Sadri, Robert A. Kowalski, F. Kriwaczek, Peter Hammond & H. T. Cory (1986): *The British Nationality Act as a Logic Program*. *Commun. ACM* 29(5), pp. 370–386, doi:10.1145/5689.5920.

[26] V.S. Subrahmanian, Piero Bonatti, Jürgen Dix, Thomas Eiter, Sarit Kraus & Robert Ross (2000): *Heterogeneous Agent Systems*, chapter 6: Agent Programs, pp. 115–170. MIT PR, doi:10.7551/mitpress/3487.001.0001.

[27] Georg Henrik von Wright (1951): *Deontic logic*. *Mind* 60(237), pp. 1–15, doi:10.1093/mind/LX.237.1.