Using Simple Deontic Constraints for Fast Norm-Conforming Reinforcement Learning

Matthias Scheutz and Daniel Little

Department of Computer Science, Tufts University, Medford, MA 02155, USA

Abstract

Standard reinforcement learning (RL) methods discover policies that maximize a reward signal but cannot learn normative behavior quickly. We propose a novel approach that uses expert demonstrations to generate simple constraints using deontic operators, guiding the agent's decision-making process. The agent uses those demonstrations to learn which actions may be obligated or permitted in certain states. By forcing the agent to take actions it learns as obligated, we significantly reduce the state space complexity of the learning problem. Furthermore, we show how after learning low-level obligated actions, the agent can cluster the received demonstrations and analyze commonly-occurring subsequences, allowing the agent to learn higher-level obligations. We demonstrate how our method learns faster and commits no norm violations in a hybrid-planning supermarket shopping task.

Keywords: reinforcement learning, simple norms, norm violations,

1 Introduction

Human actions and interactions are governed by complex moral and social norms that every member in a human society is expected to follow. Not following a norm leads to blame and possible sanctions by members of the community. There is increasing evidence that artificial agents in general, and robots in particular, are expected to follow the same norms as humans when embedded in human contexts (e.g., [12]. Hence, it is critical for those agents to abide by the same normative principles if they are to be accepted and used by humans.

There are currently two main approaches for achieving norm compliance in autonomous agents (e.g., [7]) when their task environment is known: (1) learning norm-compliant behavior from observing (perfect) teachers without explicitly learning norm representations, and (2) learning and using explicit norms to determine what to do. The first *implicit approach* typically uses a version of inverse reinforcement learning (IRL) to infer a "reward function" or human preferences that implicitly reflect the domain norms (e.g., [4]), while the *explicit approach* attempts to learn explicit norm representations (e.g., from instructions or from observations of normative behavior). Both approaches can use some version of reinforcement learning (RL) to find (close to) optimal policies for norm-compliant task performance. The explicit approach, however,

can also use other techniques such as planning or logical reasoning to determine the best action to perform which in some cases can be advantageous (e.g., when policy training time is limited). Moreover, the explicit approach has the advantage of being *explainable* in a strong sense: explicit norm-following agents can point to their explicit norm representations when justifying their actions, even in cases of norm conflicts (e.g., [11]), while notions of norm conflicts make no sense with implicit approaches as they cannot represent conflicts, but only action choices, and thus can also not generate genuine explanations for why they acted the way they did.

There are additional challenges with the implicit approach related to assumptions about the domain (e.g., that it is "Markovian", or that a certain set of environmental features is sufficient for inferring reward functions that can be used to learn norm-compliant behavior) which then limits the kinds normative properties for which they can learn appropriate policies (e.g., they cannot learn how to act properly for domains with non-Markovian norms, see [3]). On the other hand, the explicit approach faces the challenge of how to learn normative principles when the principle is not explicitly given as it would, for example, be through an instruction like "do not steal" and how to subsequently apply it (i.e., how is "do not steal" linked to behavior?).

In this paper, we propose a novel learning approach that first extracts simple normative constraints from observed behavior of norm-following agents before subsequently learning how to perform its task. There are two main advantages to this approach: (1) the agent needs to consider only a smaller action space of permitted and obligatory actions it extracts from observation, and (2) it will not perform any norm violations during learning which implicit norm learners typically do (as they do not know yet what is allowed or forbidden).

2 Related Work

Most implicit approaches to developing norm-conforming agents essentially build on learning techniques that allow agents to learn norm-conforming behavior without ever learning the norms themselves. When learning objectives can be easily specified in terms of reward functions, RL is typically the method of choice allowing the agent to determine the value of different states, including norm-violating states (which usually have a high associated cost, thus allowing the agent to learn to avoid them). If a reward function cannot be easily defined, but teachers are available to demonstrate normative task performance, inverse RL (IRL) is often applied to obtain a reward function from the observed normative behavior, which can then be used with RL to learn how to perform the task without violating norms. The challenge with this approach is that it is computationally expensive, and often times the agent never generates an appropriate reward function from which appropriate normative behavior can be learned. The most important downside of any form of RL, however, is that RL agents never abstract any normative principles, hence cannot reason with them, cannot trade them off, and cannot make recourse to them in justifications (as is required when humans, for example, violate norms). As such, RL,

IRL (and other variants such as "cooperative IRL", e.g., [4]) are not suited for application contexts where humans expect explainable behavior from artificial agents.

On the other hand, the explicit approach to developing norm-conforming agents uses some formal logic to represent norms, typically a deontic logic with modal operator for obligations **O** and permissions **P** (which can be used to define other derived concepts such as prohibitions and options). Some formal frameworks do not explicitly include deontic operators, but define them in terms of other operators, e.g., temporal operators "next" **X**, "for all future times" **G**, and "in the future" **F**, operators for agent intent or actions such as the "see-to-it-that" (STIT) operator, or notions of control and attempt [5]. Others have introduced a utilitarian deontic STIT logic which can solve various deontic paradoxes of standard deontic logic and consists of weak and strong preference operators that allow one to explicitly formulate context comparisons like every context with ϕ true is weakly better than any context in which ψ is true [18]. A very rich formal framework is provided by [1] by adding epistemic and doxastic operators, thus allowing for the formalization of so-called "doxastic oughts" which can be interpreted as the effects of actions that maximize expected (deontic) utility. Importantly, all these formal frameworks are decidable with model checking algorithms of varying complexity. However, none of these logical approaches provide methods for learning normative principles.

We have previously proposed a hybrid approach that combines explicit norm representations in linear temporal logic (LTL) with RL to learn how to obey them [10,8]. [1] Importantly, our proposed method was the first to deal with norm conflicts in that for each norm $\mathcal{N}_i$ expressed in LTL that the agent was supposed to obey we associated a weight w_i and minimized the summed weights of the violated norms when norms were inconsistent, thus suspending the smallest subset of norms for the shortest time possible obeying all others (see [10] for details). Based on this norm following approach, we also developed techniques for finding the best norm tradeoffs from observed behavior, i.e., a method for determining weights w_i associated with given norms $\mathcal{N}_i$ based on a finite set of "exemplary" observed behavioral trajectories $s_0, \ldots, s_n$ (from a norm-following teacher) that "best fit the observed behavior" (see [9] for details). And we also developed methods for finding an ideally minimal set of norms such that when the best weights are inferred for it the agent exhibits behavior as close as possible to the observed trajectories $s_0, \ldots, s_n$ (see [8] for details). While these methods had formal guarantees, they did not explicitly represent obligations and prohibitions and were only feasible in small domains.

Other approaches combining explicit norm specifications with reinforcement learning to allow agents to learn policies that maximally obey the norms add a "normative supervisor" to an RL agent's architecture as a continual check that the agent's actions follow all norms [15]. Other work modifies the *restraining*

[1] Cp. to Littman et al. [13] who also used a formal LTL specification as the reward function for an RL algorithm to learn how to perform the task.

bolt mechanism from safe RL to minimize the negative rewards accumulated after the agent commits norm violations [16]. However, in all these cases the norms are given to the agent, and are not learned from observations.

There is currently no approach to our knowledge that can learn simple and more complex normative deontic principles from observations of actions of norm-following agents alone. Moreover, there is no work that directly compares the learning performance of such explicit norm learners to implicit RL-based norm learners.

3 Learning Simple Action-Level Obligations and Permissions

Our main aim is to provide an incremental norm learning algorithm that is based on observing the actions of norm-conforming agents and using those observations to build up representations of simple obligations and permissions: $\mathbf{O}(a, s, t)$ and $\mathbf{P}(a, s, t)$ meaning that in state s action a is *obligated* and *permitted*, respectively, when performing task t. Note that this representation is not yet that of standard deontic logic with unary deontic operators $\mathbf{O}$ and $\mathbf{P}$ applied to formulas ϕ. However, in the course of observing multiple agents perform the task and under the assumption that all observed agents follow the same set of norms, the learner will be able to refine the learned norm representation and will eventually be able to arrive at normative principles in terms of these standard deontic modalities.

Our proposed norm learning agent L is *maximally conservative* in how it uses observed behavior to infer normative principles: (1) an action that is observed in a given state is always taken to be obligatory, unless there are examples of different actions being taken by the same or different agents in that state in which case all action alternatives in that state are considered permissible; and (2) all unobserved actions are always taken to be forbidden unless there is explicit instruction or evidence to the contrary. It is important to point out that this restrictive assumption does not necessarily reflect the actual norms that might apply as an what the agent believes to be obligatory actions might simply be "the best actions" in that state or what they agent has not observed and takes to be forbidden might just be "suboptimal" actions not performed by non-conforming agents. For the following, we assume that the environment is deterministic, i.e., that all attempted actions are successful (we will return to this assumption in Section 6).

Initially, L starts by observing a normative agent A perform a task t (possibly with parameters p, which we will omit to simplify the notation). For each state s in which A performs action a, L adds $\mathbf{O}(a, s, t)$ to its set of task-based obligations *Obl* (going forward, we will also omit the task parameter when we only consider a single task). L also keeps track of A's trajectory $(s_0, a_0), (s_1, a_1), \ldots, (s_k, a_k)$ (for some reasonable k depending on t). In addition, L maintains an initially empty permission set *Per* which will be populated as L obtains more observations that show how norm following agents differ with respect to the action they perform in a given state. I.e., if L observes another

normative agent A' perform T (possibly with different parameters P'), L checks again for each state-action pair (s', a') in A's trajectory whether $(s', a') \in Obl$; if so, then A and A' agree on the obligation (s', a') and nothing needs to be changed. Otherwise, if there is no $(s', a) \in Obl$ with $a \neq a'$, then (s', a') is added to Obl, indicating that a new obligation was observed. However, if there is $(s', a) \in Obl$ with $a \neq a'$, then A and A' disagreed on what action to perform in state s, hence neither a nor a' can be obligatory in s (given that both agents follow the same norms). Hence, (s', a) is removed from Obl and added to Per, and (s', a') is also added to Per to indicate that both actions are *permissible* in s.[2]

The main aim of this "normative constraint learning phase" (realized by Alg. 1 in the Appendix) is to acquire the normative action boundaries for the task, i.e., what is and is not permitted, and what must and may not be done (this is akin to *learning* a "shield" in safe RL [2]). A second important benefit of the approach is to reduce the large state space so that a subsequent task learner will only need to learn what to do in cases of permissible states that present alternatives (of which it should ideally pick the best). I.e., after the normative constraint learning phase, all observed states either have a single, obligated action, or several permitted actions associated with them and the agent only needs to learn a policy to achieve the task goal in the reduced action space where it, essentially, only needs to consider states with choices. Most importantly, it can *safely* learn the policy because in the reduced action space no action can violate any norms (in the deterministic case considered here). Hence the agent could not possibly select a "bad action", compared to the situation of a regular RL learner in the full action space where some actions will violate collision norms, for example.

For the "task learning phase" in the constrained action space, we can use reinforcement learning, e.g., one-step Q-learning with ϵ-greedy exploration (but note that we could use other learning algorithms as well, e.g., planning algorithms). In states where there is a single obligated action, the agent is forced to take that action. For states where multiple permitted actions are recorded, the learner can then immediately use inference and learning methods to determine which of the permitted actions to select or prioritize if the learner learns action distributions (e.g., it might use RL to learn the best action in s or action sequence between obligated actions); to do this efficiently, the agent only needs to consider all subtrajectories that start in a state with an obligated action and end up in another such state: $\mathbf{O}(s_0, a_0)$,$\mathbf{P}(s_1, \{a_1, a'_1, \ldots, \})$,$\ldots$,$\mathbf{O}(s_*, a_*)$. The reduced learning problem then is to start in s_0 and learn how to get to s_* as efficiently as possible. We can illustrate this idea with a simple example from a shopping task in a supermarket.

[2] While we explicitly separate out learned obligations and permissions here for easy access and analysis, it is possible to only track permitted actions in state s defined as $P_s := \{a|(\text{s,a}) \text{ occurs in some of the observed non-conforming behavior traces}\}$ when there is only one such action in P_s it is obligatory.

Supermarket example: Suppose there are multiple paths from the entrance of a supermarket to the location of the shopping baskets $s_0, s_1, \ldots, s^*$ and $s_0, s'_1, \ldots, s_*$. An untrained agent observes these trajectories and runs algorithm Alg. 1, hypothesizing that s_0 has two permitted actions, those being the actions that led to two different paths to the basket. During learning, the agent explores the permitted actions in s_0 and learns which one is optimal. Since all other actions in $s_1, \ldots, s_*$ and $s'_1, \ldots, s_*$ are obligated, the agent has no need to learn anything else and can simply take the obligated actions until it later encounters another state with permitted actions.

Note that in the full shopping task, in which the agent must complete multiple subtasks such as navigating to the basket, getting milk off the shelf, paying at checkout, etc. a set of permissions and obligations is maintained for each subtask.

4 Learning Higher-Level Obligations

In addition to updating the obligated and permitted action for states, the agent can start to abstract single state-based action obligations and permissions by comparing the observed trajectories to arrive at higher-level obligations that are characterized by propositional context descriptions (e.g., the agent might have access to a labelled MDP that provides all propositions that are true in a given state, e.g., see [10,8], or it could learn them from observations, e.g., [19]). Suppose L compares two agents that start in the same state s_0 but subsequently differ in the trajectories they take. L then attempts to find the first state s_* after s_0 in *Obl* such that the action a in s_* is performed in both trajectories leading to the new state s_*', in short, the first shared state with an obligatory action in both trajectories.

Supermarket example: Suppose two agents start at the store's entrance, but take different routes to the shopping baskets where they both pick up a basket, i.e., $s_0, s_1, \ldots, s_*$ and $s_0, s'_1, \ldots, s_*$ (possibly of different lengths). L then considers the details of the state change after s_* attempting to find a state property that was false up until s_* and true in s_*'. In this case the formula is $\neg has(basket, agent)$ which is true up to s_* and false afterwards due to the pickup action in s_*. L can then use the formula to abstract a more complex norm that is state-independent: there is a point in time when in the shopping task the agent must have a basket (here we use the temporal logic **F** operator to indicate a future time):

$$\mathbf{FO}has(agent, basket)$$

The agent then continues the above process to determine another common norm that agents must obey for shopped items (we assume for simplicity that all observed agents actually put items in their baskets):

$$\mathbf{FO}paid(agent, shopped\_items)$$

Since the second norm happens after the first norm for both agents, the learner can even retain the order of the norm applicability:

$\mathbf{F}(\mathbf{O}has(agent, basket) \wedge \mathbf{FO}paid(agent, shopped\_items))$

Overall, multiple different observations allow the learner to switch from very specialized norms that prescribe actions in particular states to more general norms that prescribe certain state properties, i.e., partial state descriptions in some formal language irrespective of the action or action sequence needed to arrive in a state of which the description is true.

We thus developed a second algorithm that can be used to learn higher-level, more general norms than Alg. 1, again based only on observations of norm-confirming behavior of other agents. In Alg. 2, the agent uses the given set of expert demonstrations to find places where the expert demonstration trajectories "intersect". It first identifies the *common states* from all trajectories in the order in which they appear. Then, it determines which propositions in the state descriptions have changed between one common state s and the subsequent common state s'. For each proposition p that has changed, the agent stores the new truth-value in ΔP, e.g. if p was false in state s and true in s', it adds p to ΔP. Note that this procedure only builds higher-level obligations from the differences between states. If the agent encounters two common states with no differences in propositional descriptions, it will consider the next two common states to handle the case where multiple common states appear in sequence in a set of trajectories (e.g., if in all trajectories the agents pay at the checkout counter and then immediately go north towards the exit). New higher-level obligations are only hypothesized when a difference is detected. Ideally, the agent could then represent a sequence of obligations in terms of regular expressions instead of nested temporal formulas which would make the representation more compact that the above formulas as it would allow for alternatives and repetitions. For example, if initially getting a basket is obligated, and paying is obligated at the end, but in between different alternative permitted sequences for getting various food item $f_1, f_2, \ldots, f_m$ are observed, then the agent could abstract the following regular expression (where ";" denotes concatenation, braces denote alternatives, and "*" denotes iteration):

in(basket,food);
{in(basket,f_1),in(basket,f_2,)...,in(basket,f_m)}*;
paid(agent,shopped_items)

and note that the agent here makes a more general inference than that strictly warranted by the observations about alternative actions being arbitrarily repeatable (e.g., one can get any number of apples).

5 Experimental Evaluations

The goal of the experimental evaluation was twofold: (1) to demonstrate the feasibility of our proposed approach and to verify that it allows the learner to quickly learn both the normative boundaries in the task as well as how to perform the task without violating any norms; and (2) to compare its performance to typical RL and IRL algorithms. We used our *Proper Shopper*

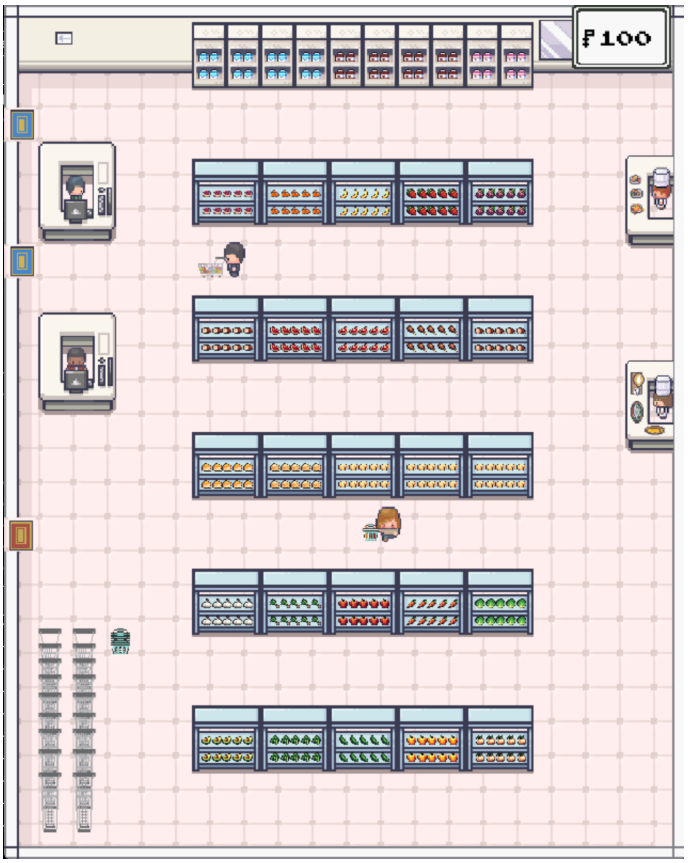

Fig. 1. The ProperShopper supermarket simulation environment with two players (one with a cart, one with a basket), two cash registers, a meat and a fish counter, and various isles with food items on shelves.

simulation ProperShopper which emulates typical shopping tasks in a supermarket for all experimental evaluations (see Fig. 1). The simulation was built using the Pygame library and has been provided as a Gymnasium "wrapper" to enable easy sharing with other researchers.

Human and/or artificial agents play shoppers in a supermarket with the goal to purchase all items on their shopping list from the store. Shoppers can pick up carts or shopping baskets, navigate the aisles of the store, retrieve food from shelves or prepared food from counters, return unwanted food to shelves, and purchase the contents of their carts/baskets at a checkout counter. I.e., the action space consists of *nop* (a no-op action); *north*, *south*, *east*, and *west*, (to navigate around the supermarket); *interact* (to interact with another object); *toggleCart* (to pick up or release a currently held cart or basket); *cancel* (to cancel ongoing interactions); and *remove* $\langle n \rangle$, for some $n \in \mathbb{Z}$ (to remove a particular item from a cart, basket, or checkout counter). The supermarket

has 27 different food items distributed over 30 shelves and two food counters, supporting up to nine different players that can simultaneously navigate and interact with objects. Each player starts with a set amount of money to spend on items (e.g., $100). The simulation also consists of several *norm violation monitors* which run before and after each action to determine whether an agent has violated a norm (see the Appendix).

Norm violation monitors thus provide information about any new norm violations that arise as a result of agent actions (such as which agent caused the violation, other agents/objects involved in the violation, etc). In particular, they can track temporally complex (non-Markovian) norms, such as those specified in temporal logic [3]. Each monitor is responsible for determining what constitutes a unique violation of its corresponding norm.

For the first set of evaluations, we considered the subtask of navigating from any place in the store to the basket location. The agent's state consists of its (x, y) position, both of which are integer values. When the continuous state space is discretized in this way, the *Proper Shopper* environment has dimensions 19×23, for a total of 437 states. To generate expert trajectories automatically in this "find and fetch the basket" subtask of the shopping task, we trained an RL agent using one-step Q-learning for 2500 episodes of 100 time steps each (see the Appendix for details on Q-learning). The agent's initial position was randomized. The agent received -1 reward each time step except when it reached the goal states, in which case it received +1000. The agent was also penalized -50 times the number of norm violations every time it caused any. In other words, we utilized the norm monitors to provide violation feedback to the learner to help it avoid committing violations. After training, the agent was run for 100 more episodes and it was able to reach the basket every time. The important side-effect of generating expert trajectories this way was that we were also able to obtain a baseline agent for regular RL training that allowed us to determine (1) how long it would take an RL with explicit norm violation feedback to this task and, and (2) how many norms that agent violated at different times during learning and after.

For the second set of evaluations, we trained a one-step Q-learning agent on the full shopping task, where it had to purchase all items on a shopping list. In each episode, a random shopping list was generated from which a plan was created for the various subtasks: where the agent needed to go to pick up a basket, to pick up all items on the shopping list, to pay at checkout, and to exit. The agent was trained by maintaining a separate Q-table for each subtask (e.g., for navigating to the basket, to the chicken shelf, picking up an item from the chicken shelf, etc.). For navigation tasks, the agent received +100 for reaching the navigation goal. For get and pay tasks, the agent was given +100 for getting the required item, or paying at checkout. The agent received -1 otherwise. The expert agent was trained for 75000 episodes, and given 300 time steps to complete each subtask. The agent could only progress to the next subtask in the plan by completing the current subtask; if it failed at a subtask, the environment was reset and a new training episode started.

5.1 Extracting low-level normative task structure

We generated 1000 demonstration trajectories from the first baseline agent in the fetch-the-basket task to learn which actions are obligated and which are permitted in each state. Figure 2 shows the result where each grid square is a possible location for the agent. Black squares are inaccessible forbidden states, and green squares are goal states (they are adjacent to the basket). Obligated actions are represented by a black arrow, permitted actions by gray arrows. Squares with no arrows were not visited by the trained agent in any trajectory. As shown, the algorithm significantly reduces the number of states an agent needs to consider: Out of the 437 states, 88 are inaccessible to the agent or are the goal state. From the 349 remaining states, the agent learns an obligated action for 278 of them, representing a reduction of 80%.

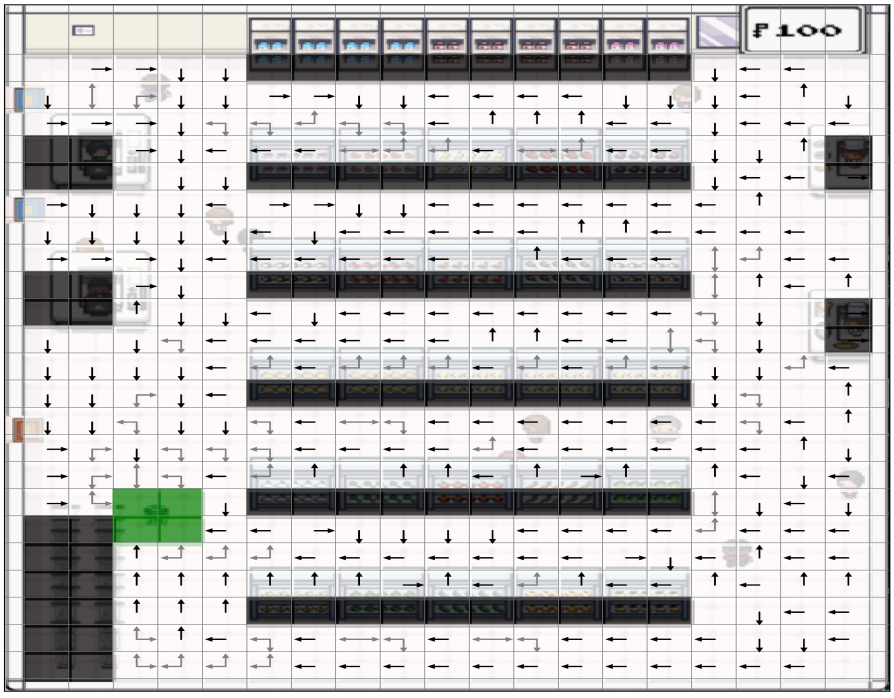

Fig. 2. Obligated and permitted actions for the basket navigation task (see text for details).

As a result, the agent can learn its task significantly faster compared to learning it in the original action space. Even when the agent needs to explore permitted actions, those actions are all "good" in the sense that a norm-abiding agent was observed selecting each of those at least once. And as Fig. 2 shows, explorations often lead the agent into a state with only one single, obligated ac-

tion. Fig. 3 shows the learning process of our proposed norm learner compared to the baseline Q-learning agent. Both agents were trained for 2500 episodes, and their results averaged over 100 experiments. The results confirm our hypothesis that applying deontic constraints results in extremely fast learning, with the agent being able to complete the task nearly all the time right from the beginning of training compared to the learner in the original action space, which ramps up of over 1000 training episodes before sufficiently high task performance is achieved at around 1500 training episodes.

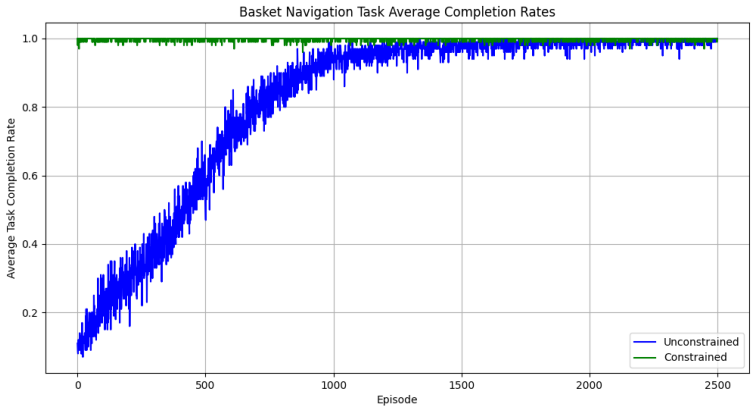

Fig. 3. Completion rates of the basket navigation task, unconstrained vs. constrained agent

We also implemented the Maximum Entropy IRL learner from [23] which was able to recover a similar reward function to that used in the baseline agent, with a sparse reward of +1000 for reaching the basket, and a step cost of -1. However, the IRL learner took extremely long to train, approximately 50 hours for this simple task which makes IRL practically infeasible for anything but similarly simple tasks, and certainly not usable for a complex task such as the shopping task. Most importantly, the recovered reward function does not provide any value for learning normative principles.

5.2 Extracting higher-level obligations

For the higher-level obligation learner, we again generated 1000 demonstration trajectories without norm violations, this time from the second baseline agent, which was trained on the entire shopping task. However, the second agent already demonstrated a major issue with standard RL algorithms like Q-learning: they tend to commit many norm violations during the exploration phase. Fig. 4 shows the average amount of norm violations committed per episode (normalized by episode length) for a Q-learning agent learning to complete a single, preset shopping list from scratch. At the beginning of training,

the agent might commit more than 50 norm violations per episode.

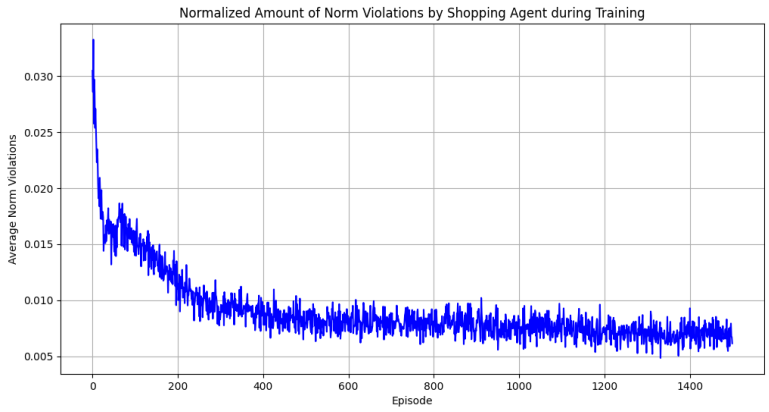

Fig. 4. Norm violations committed by Q-learning agent over 50 experiments.

Two trajectories of the full task were sampled using the trained agent, each using a different shopping list. Note that for the full shopping task, the agent has a complete state description, i.e., it includes its (x, y) position, as well as several propositions about other agent characteristics, e.g., whether the agent is holding a basket, any items the agent is holding, etc. In learning higher-level obligations, we extracted the (x, y) position from the complete state in order to determine state commonality. For example, if two trajectories visited $(2, 18)$ but in one trajectory the agent was carrying chicken, and in the other lettuce, this was considered the same state *with respect to location*. Henceforth, common states between trajectories will refer to common locations visited by both.

Alg. 2 was applied to the trajectories, and it was discovered that both trajectories visited the states shown in Fig. 5. We assign a unique integer index to each state by scanning the grid in row-major order, beginning from the top-left corner. The indexing starts at 1 and proceeds left to right across each row before continuing to the next row beneath. Thus, the top-left square is assigned index 1, the square immediately to its right is index 2, and so on until the bottom-right corner. As can be seen, there are several sequences of common states. We iterate through the common states in order until a difference in state descriptions is detected. In this case, the agent's propositions at 266 are compared to the propositions at 307. In 307, $\neg holds(agent, basket)$, indicating the agent does not have a basket. At after leaving 307, $holds(agent, basket)$, hence the learner hypothesizes that $\mathbf{FO}holds(agent, basket)$. Next we compare the agent's propositions at 307 to the propositions at the next common state at 192. We observe the difference: in 307, the agent has no food in its basket, in 192, it does (chicken for the first trajectory, and lettuce for the second). We

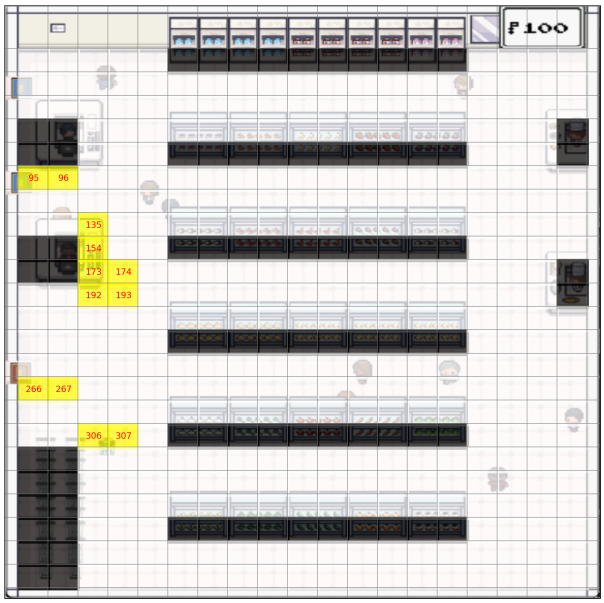

Fig. 5. States visited by both shopping trajectories

hypothesize that $\mathbf{FO}\neg empty(basket)$. Repeating this procedure for the remaining common states, we hypothesize that $\mathbf{FO}(agent(paid)$ and $\mathbf{FO}(at(exit)$. In sum, we get that that it is obligated to get a basket, pick up items, pay for those items, and leave the store.

6 Discussion

The main upshot of comparing our norm learning method from observations to regular RL and IRL was to demonstrate its superiority in two important respects: (1) it never commits norm violations, neither during learning, nor during task performance, while RL and IRL learners do even when they have been trained to achieve high task performance; and (2) it learns much faster using RL than regular RL learners and would be able to do even better with other methods such as a simple task planners that plan the best trajectory between obligated states, eliminating the need for RL altogether. Moreover, the proposed method is intrinsically incremental in that the *normative boundaries*—what is obligated and permitted—can can be refined over time as more demonstrators are seen, potentially allowing the space of permitted actions to increase and thus potentially also the performance of the agent with subsequent training on those substate with additional permitted actions. I.e., the learner can incrementally build on existing policies and refine them when new observations are added instead of re-learning the whole task from scratch; and refining will always be fast because only those areas where obligations changed to permissions need to be reconsidered.

Together, the simple obligations and permissions serve the functional role of a "shield" used in safe RL methods in that they stake out the safe space for subsequent task learning; but different from safe RL, the shield can be incrementally learned and refined over time as more observations become available. Moreover, they also go beyond defining a shield in that they serve as the building blocks for defining higher-level norm representations that can be used in many different ways, including structuring a task into subtasks that can be individually learned, modified, and adapted to other environments (e.g., how to get a basket in another store), or making the behavior of the agent explainable (e.g., "I had to pay before leaving the store"). For this purpose, "reward machines" might come in handy as they naturally allow for structuring tasks into subtasks and provide special reward functions for those subtasks [6]. Specifically, reward machines could be use to represent the *norm context* and thus allow the agent to switch between different norm contexts (that might be task-dependent) based on the states in the reward machines. Importantly, reward machines can be learned (e.g., [22]) and can represent some non-Markovian rewards as long as the state history can be represented by a regular language, which is important for norms that are non-Markovian (e.g., [7]). Finally, higher-level norms could also be used in a task planner that plans to accomplish them in sequence (the abstracted sequences would serve as plan operators in this case, e.g., see [14]).

One important upshot of the incremental method and the possibility of subsequent policy refinements is that the learner might be able to do better than any of its demonstrators. I.e, while demonstrators are by requirement norm-confirming in their task performance, they might "suboptimal" (relative to what is permissible, say). Hence, the learner will initially not be able to do any better than repeating the trajectories it has observed. But when future observations are integrated (which themselves might be suboptimal), it might be possible for the learner to do better than any of the demonstrators either by combining optimal subtasks performance or by combining subsequences in subtasks. For example, say demonstrator D_1 goes to the basket in a suboptimal manner, while D_2 does not; yet, D_2 goes to the checkout counter in a suboptimal manner, while D_1 does not, then the learner will see both deviating trajectories as opportunities to find the best possible path, in this case learning to do what D_1 with the basket, and what D_2 with the checkout counter.

Despite all of the above advantages, there are also some limitations of the current approach that we plan to address in future work. Most importantly, non-determinism in the environment (i.e., stochastic action outcomes) will introduce the possibility that an agent might end up in a state it has never encountered before (i.e., that was not seen in any of the demonstrations). In the current algorithm, this type of state would be marked as "forbidden"—it may or may not be norm-violating state—and the agent has no knowledge of how to get out it. [3] One could argue that since it has no knowledge about the

[3] Note that just because an agent has not observed another agent in a particular state and marked it as "forbidden" does not automatically imply that it is a norm-violating state.

state, it should then just try out actions until it gets back into a known "safe" state and then proceed from there. [4] The question is whether the agent could potentially do better. For example, could the agent use higher-level norms to help with non-deterministic settings? Because if an agent ends up in a state it has never seen before (and where it does not know what to do), higher-level obligations could provide ways to get out of the current state. Alternatively, if the agent could access background information about the task or task environment (e.g., in a database of common sense knowledge or by consulting a foundation model trained on such tasks and domains), it might be able to use the resource to assess whether the state is indeed a norm violating state and, if so, how to get out of it.

Another shortcoming of the low-level norm extraction method is that it is tightly coupled to the observed states of the norm following agents and thus the states of the particular task environment. As a result, the agent might consider state descriptions that are too narrow and do not transfer to other environments easily (e.g., other supermarkets with different layouts, etc.). One way to address this might be to allow the agent to make observations in other environments and the learn to generate more abstract representations that subsume states from both environments.

7 Conclusion

In this paper, we proposed a new norm learning approach that allows an artificial agent to learn explicit representations of permitted and obligated actions from *expert demonstrations* of norm-abiding agents performing a task. The agent then uses the learned normative constraints to quickly learn how to perform the task without violating any norms, different from IRL learners that from the same observations never learn normative principles, violate norms during learning, and take a much longer time to learn than our agents. Moreover, our approach also outperforms regular RL learners when they are given a reward function for the task with explicit norm violation feedback: it learns faster and never violates any norms, compared to the RL learner that even with norm violations baked into the reward function will violate them during learning and even after learning during task performance. Finally, we extended our approach to allow for the extraction of higher-level norm-based principles that can be used to learn more complex tasks and can form the basis for generating explanations and justification of agent behavior.

Future work will extend the current explicit norm learning approach to non-deterministic environments and develop methods for task transfer between different task environments, taking explicit norm-context into account. It will also compare the proposed approach to other ways of learning normative behavior from observations such as behavior cloning.

[4] Note that shielding and related approaches do not provide a solution for this situation because they simply assume safe actions for every state (the same goes for simplex approaches in safe control).

Acknowledgement

This work was in part funded by AFOSR grant #FA9550-23-1-0425. The authors would also like the thank the three anonymous reviewers for their very helpful suggestions for improving the paper.

Appendix

Preliminaries on Reinforcement Learning. We model an agent's environment as a Markov Decision Process (MDP) $M = \langle S, A, p, r, \iota, \gamma \rangle$, where S is the set of states, A is the set of actions, p is the probability distribution $p(s_{t+1} \mid s_t, a_t)$, $r : S \times A \times S \to \mathbb{R}$ is a reward function, ι is a probability distribution over initial states, and $\gamma \in (0, 1]$ [20]. A policy for M is defined as the probability distribution $\pi_M(a \mid s)$ that establishes the probability of an agent taking an action a given that it is in the current state s. We define the set of all such policies in M as Π_M. We let $S_0 = \{s \in S : \iota(s) > 0\}$. An RL problem typically consists of finding an optimal policy $\pi^*_M \in \Pi_M$ that maximizes the expected discounted future rewards obtained from $s \in S$:

$$\pi^*_M = \arg\max_{\pi_M} \sum_{s \in S} v_{\pi_M(s)},$$

where $v_{\pi_M(s)}$ is the value function and captures the expected discounted future rewards obtained when starting at state s and selecting actions according to the policy π_M:

$$v_{\pi_M(s)} = \mathbb{E}_{\pi_M} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right].$$

At each time step, the agent executes an action a and the environment returns the next state $s' \in S$ (sampled from p) and an immediate reward r. The experience is then used by the agent to learn and improve its current policy π_M.

Q-learning [21] is one learning technique in which an agent uses experiences to estimate the optimal Q-function $q^*(s, a)$ for every state $s \in S$ and $a \in A$, where $q^*(s, a)$ is the expected discounted sum of future rewards received by performing action a in state s. The Q-function is updated as follows:

$$q(s, a) \leftarrow q(s, a) + \alpha \left[\left(r + \gamma \max_{a' \in A} q(s', a') - q(s, a) \right) \right],$$

where $\alpha \in (0, 1]$ is the learning rate. The Q-learner can explore the environment, e.g., by following an ϵ-greedy policy, in which the agent selects a random action with probability ϵ and otherwise follows an action with the largest $q(s, a)$.

Inverse reinforcement learning attempts recover the reward function of expert agent from observations of that agent's task performance. In the formulation by Ng et. al [17] the recovered reward maximizes the margin between the expected cumulative return of the expert's policy and any other policy. One

issue with the original approach proposed by Ng is that many reward functions may be optimal for the expert's policy. This was resolved with the introduction of *maximum entropy* IRL by Ziebart et. al [23] which uses the principle of maximum entropy to pick a suitable reward function.

Algorithm 1 Learning Action-Level Obligations

Require: Set of trajectories $\mathcal{T}$, number of states N
Ensure: Mapping of states to their permitted and obligated transitions

$\mathcal{T}_s \leftarrow \{s \mapsto (\emptyset, \emptyset) \mid s \in [0, N-1]\}$
for each trajectory $\tau \in \mathcal{T}$ **do**
  for each transition $(s, a) \in \tau$ (excluding the last state) **do**
    $(O_s, P_s) \leftarrow \mathcal{T}_s[s]$
    if $a \notin O_s \cup P_s$ **then**
      if $|O_s| = 0$ and $|P_s| = 0$ **then**
        $O_s \leftarrow \{a\}$
      else
        $P_s \leftarrow O_s \cup P_s \cup \{a\}$
        $O_s \leftarrow \emptyset$
      end if
    end if
    Update $\mathcal{T}_s[s] \leftarrow (O_s, P_s)$
  end for
end for
Return $\mathcal{T}_s$

Algorithm 2 Learning Higher-Level Obligations

Require: Set of trajectories $\mathcal{T}$

Identify common states $\mathcal{C}$ from $\mathcal{T}$ in order of appearance
for each common state $s \in \mathcal{C}$ **do**
  Identify the next common state s'
  Add propositions that have changed value from s to s' to ΔP
  for each proposition $p \in \Delta P$ **do**
    if p is true **then**
      Hypothesize a high-level obligation: $\mathbf{FO}(p)$
    end if
    if p is false **then**
      Hypothesize a high-level obligation: $\mathbf{FO}(\neg p)$
    end if
  end for
end for

Norm violation monitors in the Propper Shopper simulation.

- CartTheftNorm: violated whenever one player steals a cart from another

- ShopliftingNorm: violated whenever a player leaves with food that has not been purchased
- WrongShelfNorm: violated whenever a player puts food back on the wrong shelf
- PlayerCollisionNorm: violated whenever a player runs into another player
- ObjectCollisionNorm: violated whenever a player runs into a game object (checkout/food counter/shelf/cart/cart return)
- WallCollisionNorm: violated whenever a player runs into a wall
- BlockingExitNorm: violated whenever a player blocks the exit (i.e., standing too close to the exit for too long)
- EntranceOnlyNorm: violated when a player exits the store through an entrance.
- UnattendedCartNorm: violated when a player is too far from their cart for too long
- OneCartOnlyNorm: violated when a player pulls more than one cart out of the cart return
- PersonalSpaceNorm: violated when a player comes too close to another player
- InteractionCancellationNorm: violated when a player cancels the interaction with the food counters/checkouts (i.e., ends the interaction without buying food/obtaining the food)

References

[1] Abarca, A. I. R. and J. Broersen, *A deontic stit logic based on beliefs and expected utility*, in: *Theoretical Aspects of Rationality and Knowledge 2021 (TARK 2021)*, 2021, pp. 281–294.

[2] Alshiekh, M., R. Bloem, R. Ehlers, B. K—"onighofer, S. Niekum and U. Topcu, *Safe reinforcement learning via shielding*, in: *Proceedings of AAAI*, number 1 in 32, 2018.

[3] Arnold, T., D. Kasenberg and M. Scheutz, *Value alignment or misalignment – what will keep systems accountable?*, in: *AAAI Workshop on AI, Ethics, and Society*, 2017.

[4] Hadfield-Menell, D., A. Dragan, P. Abbeel and S. Russell, *Cooperative inverse reinforcement learning*, in: *30th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[5] Herzig, A., E. Lorini and E. Perrotin, *A computationally grounded logic of 'seeing-to-it-that'*, in: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, 2022.

[6] Icarte, R. T., T. Q. Klassen, R. Valenzano and S. A. McIlraith, *Reward machines: Exploiting reward function structure in reinforcement learning*, Journal of Artificial Intelligence Research (2022), pp. 173–208.

[7] Kasenberg, D., T. Arnold and M. Scheutz, *Norms, rewards, and the intentional stance: Comparing machine learning approaches to ethical training*, in: *Proceedings of the 1st AAAI/ACM Workshop on Artificial Intelligence, Ethics, and Society*, 2018.

[8] Kasenberg, D. and M. Scheutz, *Interpretable apprenticeship learning with temporal logic specifications*, in: *Proceedings of the 56th IEEE Conference on Decision and Control (CDC 2017)*, 2017.

[9] Kasenberg, D. and M. Scheutz, *Inverse norm conflict resolution*, in: *Proceedings of the 1st AAAI/ACM Workshop on Artificial Intelligence, Ethics, and Society*, 2018.
[10] Kasenberg, D. and M. Scheutz, *Norm conflict resolution in stochastic domains*, in: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[11] Kasenberg, D., R. Thielstrom and M. Scheutz, *Generating explanations for temporal logic planner decisions*, in: *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*, 2020.
[12] Lawrence, S., M. Jouaiti, J. Hoey, C. L. Nehaniv, L. and K. Dautenhahn, *The role of social norms in human-robot interaction: A systematic review*, Journal of Human-Robot Interaction (2025).
[13] Littman, M. L., U. Topcu, J. Fu, C. L. I. Jr., M. Wen and J. MacGlashan, *Environment-independent task specifications via GLTL*, in: *Proceedins of CoRR*, 2017.
[14] Lorang, P., H. Horvath, T. Kietreiber, P. Zips, C. Heitzinger and M. Scheutz, *Adapting to the "open world": The utility of hybrid hierarchical reinforcement learning and symbolic planning*, in: *Proceedings of ICRA*, 2024.
[15] Neufeld, E., E. Bartocci, A. Ciabattoni and G. Governatori, *A normative supervisor for reinforcement learning agents*, in: *Proceedings of International Conference on Automated Deduction*, 2021, pp. 565–576.
[16] Neufeld, E., A. Ciabattoni and R. F. Tulcan, *Norm compliance in reinforcement learning agents via restraining bolts*, in: J. S. et al., editor, *Legal Knowledge and Information Systems*, IOS Press, 2024 pp. 119–130.
[17] Ng, A. Y. and S. J. Russell, *Algorithms for inverse reinforcement learning*, in: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, Morgan Kaufmann Publishers Inc., 2000, pp. 663–670.
[18] Qiu, L. and X. Sun, *Deontic stit logic, from logical paradox to security policy*, Soft Computing (2018), pp. 751—757.
[19] Rodriguez, I. D., B. Bonet, J. Romero and H. Geffner, *Learning first-order representations for planning from black-box states: New results*, in: *18th International Conference on Principles of Knowledge Representation and Reasoning*, 2021.
[20] Sutton, R. S. and A. G. Barto, "Reinforcement learning: An introduction," MIT press, 2018.
[21] Watkins, C. J. and P. Dayan, *Q-learning*, Machine learning **8** (1992), pp. 279–292.
[22] Xu, Z., B. Wu, A. Ojha, D. Neider and U. Topcu, *Active finite reward automaton inference and reinforcement learning using queries and counterexamples*, in: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction* (2021), p. 115–135.
URL `https://doi.org/10.1007/978-3-030-84060-0_8`
[23] Ziebart, B. D., A. Maas, J. A. Bagnell and A. K. Dey, *Maximum entropy inverse reinforcement learning*, in: *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI'08 (2008), pp. 1433–1438.