A Computational Framework for Integrating Task Planning and Norm Aware Reasoning for Social Robots

Vigneshram Krishnamoorthy*, Wenhao Luo*, Michael Lewis†, Katia Sycara*

Abstract— **Autonomous robots are envisioned to increasingly become part of our lives in the house, restaurants, hospitals and offices. Additionally, self-driving cars will be soon appearing in city streets and highways and they will have to interact with cars driven by humans as well as other self-driving cars. In these settings the robots not only need to efficiently perform their tasks but also be able to interact with humans in socially appropriate ways. To accomplish this, robots must be able to reason not only on how to perform their tasks, but also incorporate societal values, social norms and legal rules so they can gain human acceptability and trust. Moreover, interactions with these robots will be long term. Long-term human interaction with robots as well as robot combined reasoning about both tasks and social norms generate multiple modeling and computational challenges. In this paper, we address one of the most important of these challenges, namely what is an *appropriate and scalable computational framework* that enables simultaneous task and normative reasoning. In particular, we report on our work on a novel computational framework, Modular Normative Markov Decision Processes (MNMDP) that integrates reasoning for domain tasks and normative reasoning for long-term autonomy. The MNMDP framework applies normative reasoning considering only the norms that are activated in appropriate contexts, rather than considering the full set of norms, thus significantly reducing computational complexity. The model modularity is also advantageous for long-term human-robot interaction. We present computational experiments that show significant computational improvements as compared with a base Normative Markov Decision Process (MDP) framework that includes the full set of norms.**

I. Introduction

Autonomous robots will soon leave university labs and enter everyday life, interacting with humans as assistants and co-workers. For this interaction to be effective, robots need to not only plan and perform domain tasks but also to incorporate societal norms and legal rules into their decision making. The requirements for these long term interactions and reasoning on domain tasks and social norms present multiple challenges such as integrating social norms into a robot so as to evaluate and reason on both task and norm related consequences, norm reasoning and norm violation handling in dynamic environments and triggering them based on context. In order to handle these challenges there is a need to create integrated and scalable computational frameworks. In this paper, we focus on developing a *scalable framework that integrates task level and normative reasoning*.

*The authors are with the Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Email: {vigneshk, wenhao, katia}@cs.cmu.edu. †The authors are with the School of Information Sciences, University of Pittsburgh, Pittsburgh, Pennsylvania, USA. Email: ml@sis.pitt.edu.

There are very few works in the literature that can shed light on these questions. There is a large literature on normative reasoning e.g., [?], [?], [?], [?], [?] focusing solely on normative agent reasoning and does not perform or integrate any task-based reasoning. Recently, there has been work [?], [?] where robot navigation and motion planning trajectories consider human presence and social conventions, such as maintaining certain distance from humans etc. Additionally, there has been sparse work considering integrated task and normative reasoning [?], [?], [?]. However, these works are not computationally scalable and therefore are extremely limited in application to realistic problems.

In contrast, we report on a scalable, computationally tractable framework, *Modular Normative Markov Decision Process* (MNMDP) that integrates task reasoning and normative reasoning in a scalable way. In particular, we assume that norm activation is context dependent which modularizes the problem. This assumption has been supported by recent human experiments [?] where it is shown via human experiments that norm activations show remarkable differentiation depending on context. Our MNMDP model has the following advantages. First, it allows for efficient computation of integrated task and normative reasoning at *deliberation and run time execution*. Second, MNMDP allows for efficient addition of norms and tasks as the robot engages in *long term autonomy operation and human interaction*. Third, our scheme allows reasoning about and evaluation of *tradeoffs* between performance requirements of executing domain level tasks while complying (or not) with normative decisions, taking into consideration the agent's preferences and the environmental context.

II. Related Work

In machine ethics, the majority of works have considered agents whose *primary and only task/goal* is to reason ethically (e.g., [?], [?]). There have been several proposals for enabling cognitive agents to reason about norms. Due to lack of space, we present the most relevant (see [?] for a review and references therein). [?], [?] reports on normative agents that help humans reduce unintended norm violations by detecting and predicting violations, and suggesting alternative plans that conform to the norms. Recently, [?], [?] proposed a similar framework as [?], where Markov Decision Process(MDP) states of the agent's domain planning are associated with suitable norms and where norm conflicts and resolution, as well as decisions of whether to follow the norm or not were presented. The work in [?] has similar aims as ours but considers the full set of norms in the state space, which has various limitations: (a) considering the full set of norms leads to combinatorial explosion of the state space. (b) adding or subtracting norms requires very costly re-calculation of the full model, (c) if any norm is violated, the agent goes to a predefined fixed output state, thus could create a loop in the policy calculation and never reach the goal; their work does not present any scheme for recovering from loops. To address these issues, we propose a novel modular MDP computational framework for normative reasoning and via experimentation we show significant computational savings.

III. Norm Characterization

A norm has *modalities*, namely obligations, $\mathcal{O}$, permissions $\mathcal{F}$ and prohibitions $\mathcal{P}$ for the normative agent who is the *addressee of the norm. Sanctions* that are imposed on the addressee, if the norm is violated are also associated with a norm. Another norm component is the issuing *authority*. A *beneficiary* of a norm is the set of agents (including the addressee and including humans) that may receive the consequences of compliance or violation of a norm. Norms also have a *spatial and temporal extent*, and apply to different groups of agents. For example, a curfew can be imposed to the whole population, or some selected sub population, within a city or a country which could also be revoked at some future time. A norm has a *life-cycle* while it is kept in the knowledge base. The *status* of a norm can be *activated*, *violated*, *contradicted*, *deactivated/expired*, and *obsolete/revoked*. A norm gets activated if its activation conditions fit the current state. Additionally a norm has a *utility* and a *priority*. *Let E be the set of all possible well-formed formulae* comprising first-order predicates over terms (constants, variables and the operators $\wedge$, $\vee$, and $\neg$). Following conventional notation from the normative and MDP literature, we define a norm as follows:

Definition 1 (Norm Representation) A norm N is represented by a tuple $\langle \nu, \Sigma, \phi_n, \phi_a, \phi_d, \sigma, \delta, \pi \rangle$ where $\nu \in \{\mathcal{O}, \mathcal{P}, \mathcal{F}\}$ denotes the deontic modality and Σ is the set of states where the norm applies. The normative context ϕ_n is the set of states in Σ where ν applies, depending on the norm modality. Conditions ϕ_a, ϕ_d denote the activation and deactivation condition respectively, and the sanction σ for violating it, where σ is a tuple $\langle \beta, \phi_s \rangle$ with $\beta : \Sigma \times A \times \Sigma \rightarrow \mathbb{R}^-$ as the (monetary) penalties, A is the set of actions, and $\phi_s \in E$ is the constraint on actions imposed as a sanction. For example, a moving violation in the traffic domain may incur a monetary penalty and loss of driver licence that restricts future actions of the agent. Lastly, δ represents the authority that issued the norm and $\pi \in \mathbb{Z}^+$ represents norm priority describing the relative importance among norms where the smallest value 1 has the highest priority [1].

The normative context $\phi_n \in E$ describes the state that, depending on the deontic modality ν, some activity for α should be obligated, prohibited, or permitted. The context could be represented by a single predicate or conjunction/disjunction of multiple predicates. For example, suppose we have an instantiated normative context $\phi_n = location(\alpha, bedroom) \wedge location(\alpha, bathroom)$. If the modality is $\nu = \mathcal{P}$, the robot α is prohibited to be either in the bedroom or the bathroom. In a self-driving car example, $\phi_n = driving\_direction(\alpha, Highway1, one\_way\_north)$. If the modality is $\nu = \mathcal{O}$, the robot is obliged to drive only in the north direction on Highway1.

Definition 2 (Action-Norm Conflict) An action-norm conflict occurs if an action $a \in A$ of the addressee α contradicts one or more activated norms.

For example, if an action of the robot is $go\_to\_bedroom$ and bedroom prohibition is activated, the resulting state satisfies $location(\alpha, bedroom)$ which violates the prohibition.

Definition 3 (Normative Conflict) A normative conflict occurs if two or more activated norms contradict one another. In other words, a conflict arises when a state is simultaneously prohibited and permitted/obliged, and its variables have overlapping values.

For example a norm conflict arises when an activated norm obliges the robot to be in the bathroom while another activated norm prohibits the robot from being in the bathroom.

IV. Resolving Norm Conflicts

There are a variety of ways of resolving action-norm conflicts. Given that the agent is in state s_t, it examines the next possible states in the MDP to determine whether a norm activation conflicts with the agent's next set of actions and possibly move to a conflict-free state. If no such conflict-free state can be found, the agent has a number of options: (1) the agent may be able to *curtail* the conflicting norm [**?**]. (2) The robot, based on utility calculations, may choose to violate the norm, and incur sanctions.

To resolve normative conflicts, the most common way, which we also adopt, is to define priorities [2] over norms so, in case of norm conflict, the highest priority norm is considered for compliance whereas conflicting lower priority norms get violated as done in [**?**]. For a given set of activated norms $\mathcal{N}$, then after the norm deconfliction, we assume the overall penalties over the states (S) and actions (A) become $\mathcal{B} : \mathcal{N} \times S \times A \times S \rightarrow \mathbb{R}^-$.

V. Task Planning with Normative Reasoning for Long Term Autonomy

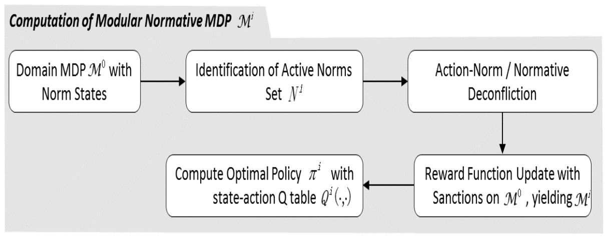
Autonomous agents performing domain tasks in dynamic environments not only must decide the relative utility of one plan over another, but also be aware of norms and decide whether to comply with those norms, given the current context. One way to combine MDP and normative reasoning

[1]All norm elements may not need to be explicitly modeled for each domain.

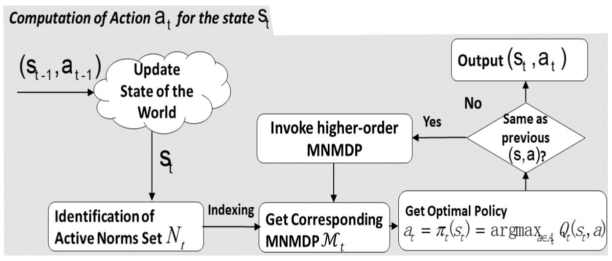
[2]We realize that it is potentially problematic to determine norm priorities since they may depend on context, culture etc. We plan to design human experiments to determine norm priorities in robot service and traffic domains. We will also relax the assumption of fixed priorities in the future.

could be via embedding the whole norm set [3] into the states of the task/domain MDP, as in [?], [?].

Definition 4 (Markov Decision Process with Norms) An MDP is represented as a tuple $\mathcal{M} = \langle S, A, R, T, \gamma, M \rangle$ where S denotes the finite set of states, A denotes the finite set of actions, $R : S \times A \times S \to \mathbb{R}$ is a reward function, $T : S \times A \times S \to [0, 1]$ is a state-transition function, and $\gamma \in [0, 1]$ is the discount factor. The normative knowledge base M is a set of norms with representations defined in Definition 1.


Computation of Modular Normative MDP $\mathcal{M}^i$
Domain MDP $\mathcal{M}^0$ with Norm States
Identification of Active Norms Set N^i
Action-Norm / Normative Deconfliction
Reward Function Update with Sanctions on $\mathcal{M}^0$, yielding $\mathcal{M}^i$
Compute Optimal Policy π^i with state-action Q table $Q^i(\cdot,\cdot)$

(a) Computation of modular normative MDPs (MNMDPs)


Computation of Action a_t for the state s_t
(s_{t-1}, a_{t-1})
Update State of the World
s_t
Identification of Active Norms Set N_t
Indexing
Get Corresponding MNMDP $\mathcal{M}_t$
Get Optimal Policy $a_t = \pi_t(s_t) = \arg\max_{a \in A} Q_t(s_t, a)$
Same as previous (s,a)?
Yes
No
Invoke higher-order MNMDP
Output (s_t, a_t)

(b) Computation of action a_t for the state s_t during execution

Fig. 1: Computation of modular normative MDPs and online action execution.

The solution for an MDP is the optimal policy $\pi^* : S \to A$ that selects the best action for each state so that the expected cumulative discounted reward for all states is maximized. As the robot accumulates experiences about norms, and as norms and the context of different norm activation changes over time, the normative MDP leads to a vast number of states as well as different reward functions incorporating sanctions due to activated norms and possible norm violations on the states.

We construct a knowledge base for normative models indexed by each of the task domains of the robot. Each norm has a priority. For each task domain, the approach is depicted in Figure 1(a). We construct an MDP only for the agent's domain tasks, i.e. not including norms in the states. Let us call this MDP, the *domain MDP* $\mathcal{M}^0$. Each time the agent transitions to a new state in the domain MDP, we determine whether the set of most promising next states contains a state where one or more norms may be activated. If there is no such state, the MDP process continues as usual. If one of the most high reward states contains a norm, say N^1, then the agent retrieves from memory a pre-computed modular MDP that consists of the domain MDP but also contains norm N^1 in its state space. Let us call this the Modular Normative MDP $\mathcal{M}^1$.

This modular normative MDP is much smaller than one that would contain the whole norm set. The reasoning procedure now follows the reward structure that includes rewards and sanctions for norm N^1 (see next section of how the normative MDP is computed). If some other state contains the possible activation of a (small set) of norms, say $N^{1,2} = N^1 \cup N^2$ where N^2 is another activated norm, then the agent retrieves a precomputed MNMDP $\mathcal{M}^{1,2}$ i.e. the Modular Normative MDP consisting of the domain MDP further modified by norms N^1 and N^2 in the state space and follows the reward (and sanctions) dictated by this new modular normative MDP $\mathcal{M}^{1,2}$.

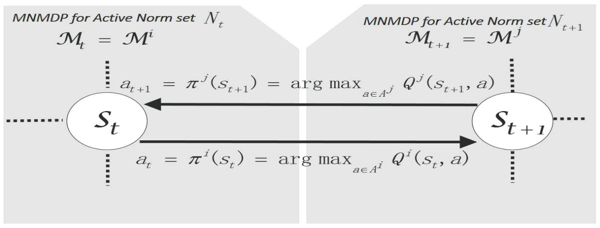

MNMDP for Active Norm set N_t
$\mathcal{M}_t = \mathcal{M}^i$
MNMDP for Active Norm set N_{t+1}
$\mathcal{M}_{t+1} = \mathcal{M}^j$
$a_{t+1} = \pi^j(s_{t+1}) = \arg\max_{a \in A^j} Q^j(s_{t+1}, a)$
s_t
s_{t+1}
$a_t = \pi^i(s_t) = \arg\max_{a \in A^i} Q^i(s_t, a)$

Fig. 2: Example of looping behavior

Once the knowledge base of the modular normative MDPs and their policies has been computed, the robot will start the execution process as shown in Figure 1(b). For state $s_t \in S$ at time t, $N_t \subseteq M$ represents the set of activated norms associated with s_t. After transitioning to the current state s_t from the previous state s_{t-1} with action a_{t-1}, the robot with active norms set N_t will select the best action a_t according to the optimal policy π_t for the corresponding MNMDP $\mathcal{M}_t$, so that the robot can achieve domain-specific goals while satisfying the normative constraints. If the norm states change because of the dynamic environment, the robot will choose its new action based on optimal policy for another modular normative MDP that reflects the changed norm states. In this way, the robot will only consider the normative constraints in the states where those constraints apply, avoiding the unnecessary incorporation of all possible norms in its policy.

However, due to unforeseen policy conflicts between different MNMDPs that apply to different states, the robot could be trapped in a loop where the actions from different MNMDPs at the imposed states lead the robot to periodically visit the same states, namely, fall into the same $(state, action)$ pair. For example, consider the two consecutive states s_t and s_{t+1} with different active norm states N_t and N_{t+1} in Figure 2. If one state's active norm set does not apply to the other, then by Definition 2 and 3 there is no action-norm conflict nor normative conflict, and hence the robot will simply execute the actions dictated by the policies of the particular MNMDPs for the respective states. As in the figure, at state s_t the optimal action from MNMDP $\mathcal{M}_t = \mathcal{M}^i$ for norm set N_t leads to the state s_{t+1}, and afterwards, assuming the active norm set changes

[3]In the worst case, i.e if all norms are interacting with one another, MNMDP will result in the full normative MDP.

to N_{t+1}, then at s_{t+1} the action from $\mathcal{M}_{t+1} = \mathcal{M}^j$ will navigate back to the state s_t. With the norm set changing to N_t, the robot will execute the same sequence of the two actions and get trapped between the two states. As shown in Figure 1(b), in order to identify a loop, the robot checks whether the subsequent $(state, action)$ pair is same as the current one. If a loop is identified, instead of taking the actions from the corresponding MNMDPs $\mathcal{M}^i$, $\mathcal{M}^j$ etc., we take the 'second-best action' which is defined as the optimal policy from the pre-computed higher-order MNMDP $\mathcal{M}^{i,j,\cdots}$ comprising of multiple norms that are involved in the states in the loop, namely, $\mathcal{M}^{i,j,\cdots}$ is the MNMDP encoding the norms $N^{i,j,\cdots} = (N^i \cup N^j \cup ...)$. Since this pre-computed MNMDP has an optimal policy over the whole state sequence $(s_t, s_{t+1}, ..., s_{t+i})$ leading towards the goal, it is guaranteed that the agent will not fall into this state-action loop again. Likewise, in the future where loops may occur on other states, the same process can be applied to eliminate the loops.

The MNMDP framework has multiple advantages: First, it avoids computing a huge MDP that would include the whole norm set. Second, each time a new norm is added/changed or deleted, as would often be the case in a lifelong autonomous agent, only a much smaller MDP needs to be re-computed, namely the domain MDP plus the new/changed norm or minus the obsolete norm and any other norms that the new/changed norm may be mutually active with. Third, the appropriate MNMDP policy gets retrieved at run time when the agent has determined which state it finds itself in and which norms are likely to be active in the next set of states.

VI. Modular Normative MDP Computation

In this section we describe the calculation for a Modular Normative MDP $\mathcal{M}^i$ for $i \in \mathbb{Z}^+$ whose active norm set is $N^i \subseteq M$ with M as the full norm set. We refer to the domain MDP without normative constraint as $\mathcal{M}^0$. For state $s_t \in S$ at time t, $N_t \subseteq M$ represents the set of activated norms associated with s_t. If $N_t \sim N^i$ (the activated norms are equivalent after action-norm/normative deconfliction) then as shown in Figure 1 the action to take for s_t is selected based on the MNMDP $\mathcal{M}_t = \mathcal{M}^i$. We assume that state transition is stochastic where the probability distribution over next states $T^i(s_t, a, s_{t+1}) = P^i(s_{t+1}|s_t, a)$ is known. In each state, state variables are directly accessible to the agent so the satisfiability of the activation and deactivation condition of a norm (ϕ_a^i and ϕ_d^i) can be decided in constant time.

Algorithm 1 captures the computation of the different MNMDPs depending on the activation and deactivation conditions. Note that (1) the function Active() used in Algorithm 1 outputs the intersection of the domain of the norms given to its input i.e. Domain($N^i \cap N^j \cap N^k \cap$...). and (2) the function Combine() first deconflicts the input norm set as discussed in Section IV based on norm priorities and outputs the encoded sanctions of the combination for the reduced subset of norms which are not in conflict. In the MDP, we need to determine the status variables of N_{t+1} for every action in A and its resulting states. Each modular normative MDP is much smaller compared to the full normative MDP that contains the full set of norms since each norm has its limited domain of states where it applies. Once a state transition is done and the agent is in s_t, N_{t+1} is considered for the next set of states based on s_t, actions, and the transition probabilities of the actions. The norm reasoning process checks whether N_{t+1} has action-norm conflicts or normative conflicts and then invokes the corresponding MNMDPs. The MDP framework can then compute the optimal policy using any policy optimization algorithms such as value/policy iteration, trust region policy optimization (TRPO) [**?**], etc. . Take value iteration algorithm for example, for MNMDP $\mathcal{M}^i$ it calculates

$$V_{k+1}^i(s_t) = \max_{a \in A^i} \underbrace{\left[\sum_{s_{t+1} \in S} P^i(s_{t+1}|s_t, a)\big(R^i(s_t, a, s_{t+1}) + \gamma V_k^i(s_{t+1})\big) \right]}_{Q_{k+1}^i(s_t, a)} \quad (1)$$

until convergence where k is the iteration number. The reward function $R^i(s_t, a, s_{t+1})$ implies the immediate reward from s_t to s_{t+1} by taking action $a \in A^i$, which considers the original predefined domain-related reward $R^0(s_t, a, s_{t+1})$ as well as the penalty due to sanction in MNMDP $\mathcal{M}^i$ as follows.

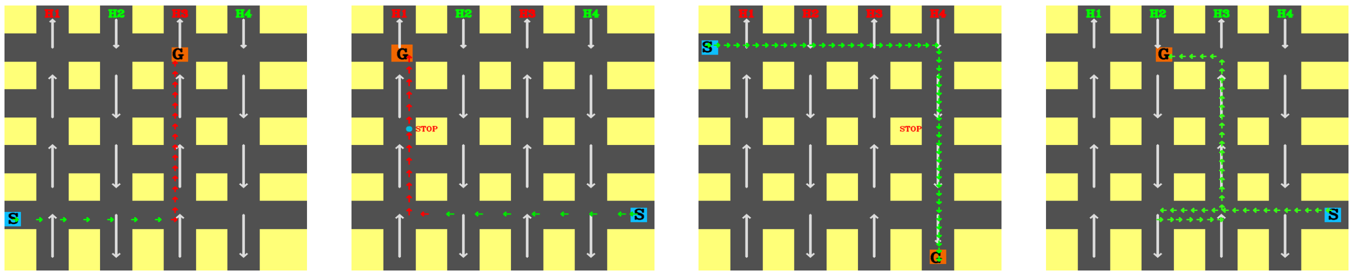
$$R^i(s_t, a, s_{t+1}) = R^0(s_t, a, s_{t+1}) + \mathcal{B}^i(s_t, a, s_{t+1}, N^i) \quad (2)$$

Equation (2) assumes that the domain reward and the norm penalty applies to each state equally, but a different calculation could be used, with weights determined by domain knowledge and other criteria, e.g. agent preferences. Imposing a sanction does not alter only the utility in each of the set of subsequent states but also the set of subsequent states themselves. We do consider these in the modular MDP formulation.

Algorithm 1 Modular Normative MDP Computation

```
Input  : $\mathcal{M} = \langle S, A, R, T, \gamma, M \rangle$
Output: $\mathcal{M}^0, \mathcal{M}^{i,j,k\cdots}$
$\mathcal{M}^0 = \langle S, A, R, T, \gamma \rangle$
forall $\{N^i, N^j, N^k, ...\}$ in Powerset($M$) do
    if Active($N^i, N^j, N^k, ...) \neq \emptyset$ then
        $N^{i,j,k} \leftarrow$ Combine($N^i, N^j, N^k, ...$)
        $R^{i,j,k,\cdots} \leftarrow R^0 + \mathcal{B}^{i,j,k,\cdots}$ as in Equation (2)
        $\mathcal{M}^{i,j,k,\cdots} \leftarrow \langle S, A, R^{N^{i,j,k,\cdots}}, T, \gamma \rangle$
    end
end
return $\mathcal{M}^0, \mathcal{M}^{i,j,k\cdots}$
```

Hence, for a given state s_t we can compute the best action a_t from (1) as described in Algorithm 2 and as shown in Figure 1(b) where we find the set of activated norms, N_t for the state s_t and map it to the corresponding MNMDP $\mathcal{M}^{i,j,k,\cdots}$. Then, we can select the action a_t for state s_t from the pre-computed policy of this MNMDP.



(a) Direction and Speed-limit norms

(b) Direction, Speed-limit and STOP sign norms

(c) Hospital Emergency, Direction, Speed-limit and STOP sign norms

(d) Direction norms only

Fig. 3: Trajectories of the agent in different norm constrained environments in a traffic example

Algorithm 2 Selection of actions

```
Pre-compute policies $\pi^{i,j,k,\cdots}$ for each $\mathcal{M}^{i,j,k,\cdots}$
Input  : $s_t, M, \mathcal{M}^0, \mathcal{M}^{i,j,k\cdots}$
Output: $a_t$
Find $N^{i,j,k,\cdots} \sim N_t$ where $N^{i,j,k,\cdots} \in \text{Powerset}(M)$
  $\mathcal{M}_t = \mathcal{M}^{i,j,k,\cdots}$
  $a_t \leftarrow \pi^{*,i,j,k,\cdots}(s_t)$
  return $a_t$;
```

VII. Results

A. Illustrative Example

In this section, we present several examples [4] using our Modular Normative MDP framework along with experimental results comparing the performance of our MNMDPs against the full normative MDP in terms of computation time and cumulative reward. The traffic scenarios as shown in Figure 3 have four vertical highways labelled as H1-H4 with different norms sanctioned on them. The start and goal states are denoted by S and G respectively. The state representation (x, y, θ) in the domain MDP captures agent location and orientation. Also, the domain reward is sparse. The norms in this environment consist of direction norms on the vertical highways, speed-limit norm, STOP sign norm and an emergency norm. The speed limit is indicated in the color of the highway labels H1-H4 with red corresponding to low-speed and green to high-speed.

Given the set of norms and the domain goal, the proposed MNMDPs to compute consist of three classes of MDPs: 1) domain MDP $\mathcal{M}^0$ whose optimal policy only considers the reward at the goal G, 2) normative MDP $\mathcal{M}^{i>0}$ that considers both the reward at G and the sanction from any single norm N^i, referred as the one-order norm, and 3) normative MDP $\mathcal{M}^{i,j,\cdots}$ with higher-order of norms that consider domain reward as well as sanctions due to activated norms with overlapped applicable states. In Figure 3(a) for example, for direction norm N^1 and speed limit norm N^2 we have $\Sigma^1 \cup \Sigma^2 \neq \emptyset$ as both of them apply to states on lane H3, and hence we need to compute a second-order normative MDP $\mathcal{M}^{1,2}$ with domain rewards as well as sanctions from the two norms N^1 and N^2.

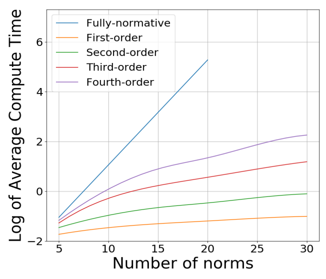
[4]The corresponding videos can be found here: http://bit.ly/2GA4HLj

In Figure 3(a), the agent operates in high-speed mode till it reaches H3 and then slows down to comply with the speed-limit norm to reach the goal state. In Figure 3(b) we explore the trajectory when an additional STOP sign is added. In Figure 3(c), we impose an additional hospital emergency context in all the states. However, in the emergency context, the norm *permitting* the agent to use the high-speed mode in all states and not stop at the STOP sign, is given a higher priority. In Figure 3(d), only the direction norms are present. As can be seen, the agent goes towards H2 and loops back to reach the goal via H3. This occurs since the modular MDP of H3 doesn't inform the agent of the direction norm in H2 and hence the agent proceeds towards H2 and then looping back to H3.

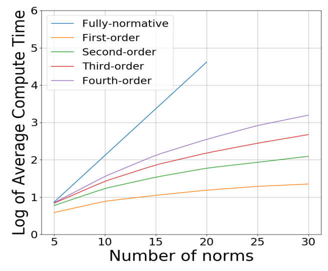
B. Quantitative Results

To compare the performance of our proposed MNMDP framework in more general cases, we provide average quantitative results shown in Figure 4 from 50 random trials. We construct a domain MDP whose transition $\boldsymbol{T}$ is random and the reward $\boldsymbol{R}$ is sparse. We model the number of norms as a parameter that can be varied during evaluation. Each of these norms can be characterized by a fixed number of norm state and action variables, both chosen randomly. We evaluate such an experimental setting in order to negate any bias in the performance comparison possibly introduced by analyzing only some specific normative structures. We use two different methods to compute the policies: (1) a traditional value based approach, Value Iteration and (2) a policy gradient based approach, Trust Region Policy Optimization (TRPO) [**?**], thus capturing the similarity in properties of the evaluation parameters between two different policy optimization techniques.

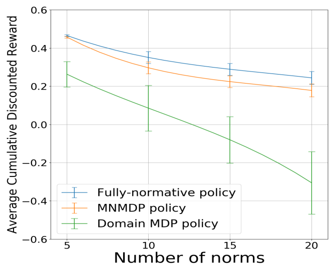
From Figure 4(a) and 4(b), we observe that the compute time (in log scale) for upto N^{th} order interactions lie on $N \log x$ curves, meaning that the compute time is of the order x^N. Hence, as we proceed to higher order interactions, we encounter higher-order polynomial times. We also observe a linear curve (in log scale) for the compute time of a full normative MDP against number of norms $|\boldsymbol{N}|$, leading to a compute time of the order of $x^{|\boldsymbol{N}|}$ which grows exponentially with the number of norms. The computation of the fully-normative MDP could not be performed with our computer



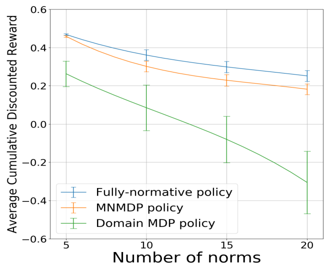
(a) Log of computation time with Value Iteration approach



(b) Log of computation time with TRPO approach



(c) Average Cumulative Discounted reward with Value Iteration approach



(d) Average Cumulative Discounted reward with TRPO approach

Fig. 4: Comparison of computation time of our MNMDP against the full normative MDP and comparison of cumulative reward of our MNMDP policy, full normative MDP policy and domain MDP policy.

resources (Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz with 128 GB RAM) for number of norms greater than 20. Figure 4(c) and 4(d) illustrate the average cumulative discounted rewards for the three different policies. We see that the general trend is that the cumulative reward decreases as the number of norms increases. For the two normative MDPs, this can be attributed to the increase in the average length of the trajectory from the start to the goal, due to the increasing normative constraints. The cumulative rewards of the MNMDP policy is somewhat lower than the full normative MDP possibly due to some looping behavior. The cumulative reward of the domain MDP policy drops significantly due to the increasing penalties resulting from norm violations.

VIII. Conclusions and Future Work

We propose a unified decision-making framework, MNMDP that incorporates normative and goal-directed reasoning under uncertainty. The agent can evaluate the utility and consequences of its actions in terms of domain goals and norms. Based on the evaluation, the agent chooses task actions with high task rewards while avoiding/reducing norm violations to minimize penalties. Our architecture takes advantage of the characteristics of norm combinatorics and shows significant computational advantages compared to the full normative MDP. This scalability makes the normative reasoning problem more tractable in a real-system and makes it especially attractive for long term autonomy, where the norms will change over time, since the framework accommodates such changes with little re-computation. In future work, we will perform human studies and experiments to determine what humans think are: 1) appropriate norms for various contexts, for example, service robots such as household robots and assistive hospital robots, and 2) norm priorities that are appropriate for different contexts so as to allow more sophisticated reasoning on the part of the robot than what we presented in this paper, in order to make decisions for resolution of norm conflicts. We also plan to extend our work to account for partial observability.

IX. Acknowledgements

The work was supported by awards NSF IIS-1724222 and AFOSR FA9550-15-1-0442.