A Computational Framework for Norm-Aware Reasoning in Autonomous Systems

Vigneshram Krishnamoorthy

CMU-RI-TR-19-14

May 2019

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Katia Sycara, Chair
Stephen Smith
Wenhao Luo

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics

Keywords: Normative Reasoning, Markov Decision Process, Knowledge Representation, Context Modeling, Propositional Logic, Decision Trees

For my parents and grandparents

Abstract

Autonomous agents are increasingly deployed in complex social environments where they not only have to reason about their domain goals but also about the norms that can impose constraints on task performance. 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. The intelligent trade-offs that these systems must make between domain and normative constraints is the key to any system being socially responsible and acceptable. Integrating task planning with norm aware reasoning is a challenging problem due to the curse of dimensionality associated with product spaces of the domain state variables and norm-related variables. In this work, we propose a Modular Normative Markov Decision Process (MNMDP) framework that is shown to have orders of magnitude increase in performance compared to previous approaches. 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.

Since norms are both context-dependent as well as context-sensitive, we must model context in an expressive, scalable and compact manner in order to find the activation and deactivation conditions for norms. To this end, we propose a generalizable context modeling approach to understand norm activations in social environments combining the expressiveness of propositional logic with the compactness of decision trees. We show how we can combine our context model with our MNMDP framework to support norm understanding as well as norm enforcement for real systems. We discuss the inferences obtained from the human experiments data that we conducted confirming the complexity of the relationship between contexts and norms. We demonstrate the effectiveness of our approach through scenarios in simulated social environments in which agents using the framework display norm-aware behavior. We also show the significant computational improvements that we obtain when using our proposed approach for computationally modeling social interactions.

Acknowledgments

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

Contents

List of Figures

Chapter 1

Introduction

As autonomous systems become more involved in our daily life, their responsibilities have expanded from industry and military to more personal scenarios like childcare, personal assistant or housekeeping. Instead of only concentrating on the accomplishment of the assignment, autonomous agents have to consider human's expectation and preference in those tasks for them to be accepted and trusted. However, constraints in social interactions are usually fuzzy and soft; thus cannot be explicitly programmed as hand-crafted functions/rules in a unified fashion. One important factor that informs the appropriateness of actions is the set of norms within a particular community. Norms, such as prohibitions, permissions, obligations, are the socially agreed upon guidelines of behavior which are acknowledged by most of the members of a community [3]. With the competence of reasoning about norms, intelligent agents can align their behaviors with human values to better collaborate and communicate with people in a socially desirable way. Thus it is critical for the robots to incorporate social norms in their decision-making process in performing their various tasks.

We are interested in the reasoning of robots engaged in long term autonomy, where they perform a set of tasks in one or multiple domains and when appropriate, engage in normative reasoning to ascertain the consequences of their actions so as to determine their best course of action. There exist few works in the community which aim to tackle this important problem [9, 14]. However, the existing literature suffers heavily from the curse of dimensionality problem and are hence infeasible in real systems. We also tackle the more challenging problem of computational models for reasoning that can be well suited for long-term autonomy applications, where the set of norms that the agent has to follow and the environments that the agent is deployed in, change over time.

Currently, there is little research about generalizable representations and models of environmental context for modeling social interactions. The social science literature [1] [26] have reported that norm activation is dependent on environmental and social context. Additionally, we claim that the determination of priorities among different norms or even between instances of the same norm depends on the environment where the agent operates. However, the specific representation of context-dependent norm activation framework in terms of perceiving context, norm enforcement and policy execution is an open problem which this work addresses in detail. Since it is impossible to elicit all possible situations and their norms that the robot will operate under, these systems must be able to learn the mapping between environmental context and norm

activations directly from human interaction and also be able to generalize to unseen contexts. On top of the above factors, our work also focuses on scalability, which is the most crucial factor with the choice of the context model in terms of deployment in autonomous systems.

A robot operating openly in society is likely to encounter an immense number of situations that require adherence to differing human norms. If a representation such as an MDP is used to guide behavior this leads to a computationally infeasible solution in which every norm must be considered at every step. In this work we propose a novel solution in which smaller MDPs incorporating only those norms active in particular contexts are indexed and accessed through a context tree.

1.1 Related Work

BOID [4] and NoA [15, 16] present architectures illustrating the potential for norms to be part of the agent behavior. Some authors have studied norm-aware planning algorithms [20, 22, 23], but not from a utilitarian and probabilistic perspective. Further, recent papers [8, 9] propose a framework similar to another work [21], where Markov Decision Processes (MDPs) are used to combine the agent's domain planning with normative constraints and sanctions, but fail to address the curse of dimensionality problem associated with the large joint state spaces when combining the domain state variables with the normative variables. [5] provides an MDP formulation for scheduling in randomized traffic patrols in a real domain using a game-theoretic approach by modeling the trade-off as a bi-objective optimization problem, but only consider a specific set of interactions between the agents. [28] provide mechanisms for detection and resolution of normative conflicts. A review of approaches for detection and resolution of normative conflicts is presented in [25]. More recently, a computationally scalable framework, called Modular Normative Markov Decision Processes (MNMDPs), was proposed for integrating domain goal and norm reasoning [17].

In order to integrate normative constraints into the domain task planning, there is a need to model and represent social interactions effectively. However, very few works in the literature shed light on scalable, practical, and generalizable systems for modeling social interactions for norm inference. Recent work [26] explores the relationship between environmental context and norms but their formulation doesn't have any explicit context representation and lacks generalizability, and feasibility for real systems. [27] represent norms using deontic logic and learn them under uncertainty from human data but again don't use any explicit context representation for modeling social interactions. In their work, norms are only differentiated based on location and not other potentially important contextual factors (such as the assigned task of the agent), context-sensitive norm activation is ignored, and the computational complexity of the norm-learning algorithm is not scalable. [13] represent normative constraints using Linear Temporal Logic (LTL) and present an approach to determine norm priorities from behavior, which is limited by restrictive assumptions such as considering that the violation cost for norms is only conditioned on the duration of the violation. Additionally, their algorithm computes a product MDP which runs in time exponential to the number of norms, rendering it impractical for real systems.

1.2 Contributions

Scalable Policy Computation and Execution: A core problem in norm learning and enforcement is to be able to compute robust policies for executing the underlying normative behavior. Norms can be viewed as soft constraints that the agent can violate at the cost of being sanctioned. For example, the optimal execution of the task of going from point A to point B would be to go as fast as one's system can perform. Speed-limits (norm) impose constraints on this 'optimal' agent task execution. Markov Decision Processes (MDP) are a natural way to model the soft trade-offs to be reasoned about by the agent such as whether to be norm compliant and possibly sub-optimal w.r.t the domain task versus executing domain tasks optimally but ignoring norms. Further, MDPs can be extended to stochastic and partially observable domains as well. However, modeling the decision making process directly using MDPs is challenging since it suffers from the curse of dimensionality problem when dealing with large state spaces that come up when we try to add an increasing number of normative variables into the domain MDP that reasons only about reaching the goal. Our Modular Normative Decision Process framework [18] which will be discussed in detail in Chapter 2 provides a modular, scalable way to compute and execute general normative policies by using the properties of the norms themselves. Its performance both in terms of memory consumption and runtime is orders of magnitudes better than previous approaches as will be discussed in Section 2.3.

Context Representation and Modeling: For the MNMDP framework to be successful, it needs the knowledge of which norms are active at any given time. Norms are often sparse in the state-space, but finding when they get activated is crucial for successful norm enforcement. Norms are not only context dependent but are also very context sensitive, making it a challenging problem to predict norm activations just from the environmental context. For example, given the same task in the same location, different norms can get activated depending on other factors such as the characteristic of the people present there (e.g. guest, stranger, owner). Moreover, in many situations, multiple norms can be active at the same time, with some of them being in conflict with one another. In order to resolve these norm conflicts, norm priority/importance must be determined. This calls for a principled way to handle context based activation of a variable number of norms as well as feasible ways to determine norm priorities. However, modeling environmental context in a scalable, generalizable fashion and finding a model that can map it to a set of active norms is an open research problem in the community, which we address in detail. (1) We employ propositional logic to expressively represent environmental context, including context with temporal components; (2) We model the set of environmental contexts in the knowledge base using learnable decision trees, which provides significant computational benefits and also helps create a hierarchy of relevant context factors; (3) We propose a principled approach for integrating social norms into task planning by combining our context model with the Modular Normative Markov decision process (MNMDP) framework [17]; (4) We propose extensions to support generalization to unseen scenarios using SimRank [11].

Chapter 2

Scalable Policy Computation and Execution

2.1 Norm Characterization

Ethical principles embody societal values and form the basis of social norms and laws. Norms are complex entities with many different attributes [10]. Norms have *modalities* [29], namely obligations, $\mathcal{O}$, permissions $\mathcal{F}$ and prohibitions $\mathcal{P}$ and they apply to the normative agent, who is termed the *addressee of the norm*. *Sanctions* are imposed on the addressee by the issuing *authority* (e.g. the government), if the norm is violated. A *beneficiary* of a norm is the set of agents, including the addressee that benefit (or suffer) the consequences of norm compliance or violation. Norms also have a *spatial and temporal extent* that could be enforced to a group of addressee agents. An example of this would be time-constrained special permissions that apply to specific regions given to agents in case of a regional emergency.

2.1.1 Norm Modalities

Norms create explicit *modalities*, namely obligations, $\mathcal{O}$, permissions $\mathcal{F}$ and prohibitions $\mathcal{P}$ for the normative agent, the robot in our case, who is the *addressee of the norm*. The norm also has associated *sanctions*, e.g. monetary fines or imprisonment, that are imposed on the addressee, if the norm is violated. Sanctions provide an explicit way of reasoning about the cost of violating 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. For example, if breaking the speed limit results in an accident that kills another driver, the other driver is the beneficiary of the norm violation by the addressee. 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. The *status* of a norm can be *activated*, *violated*, *deactivated/expired*, and *obsolete/revoked*. A norm gets activated if its activation conditions fit the current state. An activated norm influences the state resulting from an action of the agent. A norm is violated if a conflict between the resulting state of an action and the norm (*action-norm conflict*) is not

resolved while the norm is activated. Additionally a norm has a *utility* and a *priority*.

2.1.2 Norm life-cycle

Norms have a *life-cycle* associated with them while they belong to 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. For example, in a self-driving car scenario, $\phi_n = direction(\alpha, Highway1, one\_way\_north)$. If the modality is $\nu = \mathcal{O}$, the robot is obliged to drive only in the north direction on Highway1. 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.

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 α is $go\_to\_bedroom$ and bedroom privacy norm which entails robot entry 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. The most common way of resolving normative conflicts is by defining priorities. We have conducted human experiments [19] to identify relationships between norms and contexts in domestic environments and determine priorities among these norms. To resolve norm conflict, the highest priority norm is considered for compliance whereas conflicting lower priority norms get violated.

2.1.3 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 [28]. (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 [1] 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 [9]. 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 $\mathcal{B} : \mathcal{N} \times S \times A \times S \rightarrow \mathbb{R}^{-}$.

2.2 Task Planning with Normative Reasoning for Long Term Autonomy

We are interested in designing agents that are *norm-autonomous*, namely agents that reason as to whether to violate or comply with norms. We believe that such agents are more realistic in the context of long-term autonomy, as opposed to agents whose norms are automatically enforced by hard-wiring them into the agent. In one of the most relevant works using MDPs [9], it is assumed that the norm set is invariant and determined at design time. Since norms are invariant, [9] can do the normative reasoning for all states before runtime. Once norm reasoning is done, the norms are excluded from the framework by encoding only the sanctions from norm violations in the states. This has two limitations: (a) considering all norms (full normative model) at once is extremely computationally intensive, and (b) considering norms to be invariant is unrealistic. Our Modular Normative MDP (MNMDP) framework proposed in [17] alleviates these core problems by (1) allowing for efficient computation of integrated task and normative reasoning by modularizing the full-normative MDP into much smaller MNMDPs, (2) MNMDP allows for efficient addition/removal of norms as the robot engages in *long term autonomy operation and human interaction*.

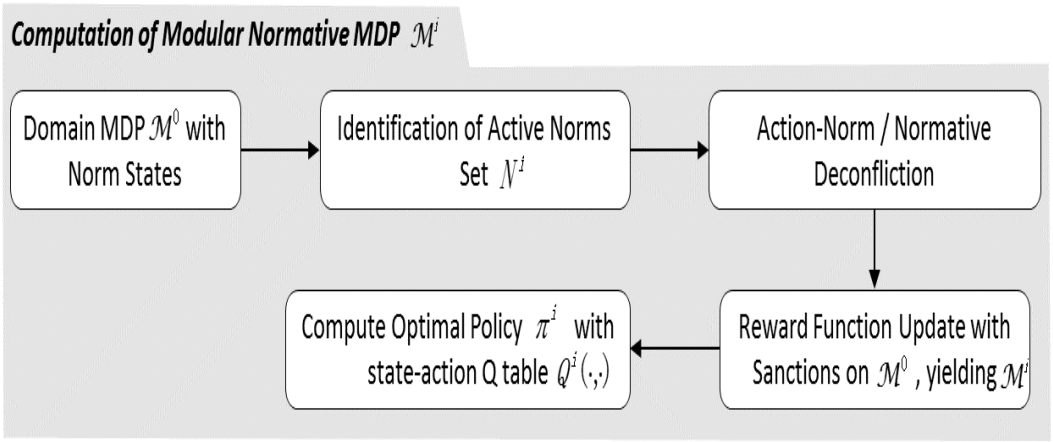
2.2.1 Modular Normative MDP

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 could be via embedding the whole norm set [2] into the states of the task/domain MDP, as in [8], [9].

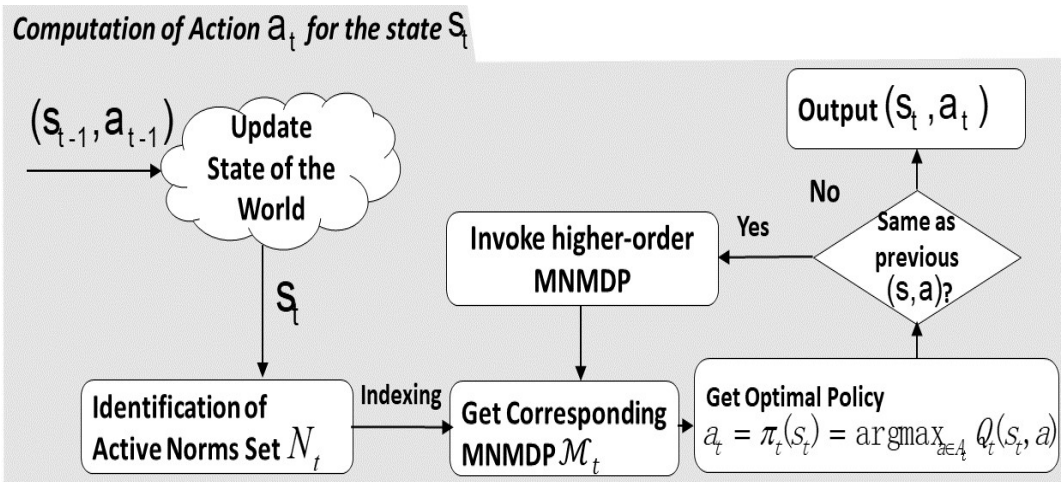
Definition 4 (Markov Decision Process with Norms) An MDP is represented as a tuple $\mathcal{M} = \langle S, A, R, T, \gamma, M \rangle$ where S denotes the finite set of states, A denotes the finite set of actions, $R : S \times A \times S \rightarrow \mathbb{R}$ is a reward function, $T : S \times A \times S \rightarrow [0, 1]$ 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.

[1] 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.

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



(a) Computation of modular normative MDPs (MNMDPs)



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

Figure 2.1: Computation of modular normative MDPs and online action execution.

The solution for an MDP is the optimal policy $\pi^* : S \rightarrow A$ 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 2.1(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 $\mathcal{M}^{1,2}$.

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 2.1(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.

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 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 2.1(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.

2.2.2 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 2.1 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 2.1.3 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. Take value iteration algorithm for example, for MNMDP $\mathcal{M}^i$ it calculates

$$V_{k+1}^i(s_t) = \max_{a \in A^i} \underbrace{\left[\sum_{s_{t+1} \in S} P^i(s_{t+1}|s_t, a)(R^i(s_t, a, s_{t+1}) + \gamma V_k^i(s_{t+1})) \right]}_{Q_{k+1}^i(s_t, a)} \tag{2.1}$$

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.

$$R^i(s_t, a, s_{t+1}) = R^0(s_t, a, s_{t+1}) + \mathcal{B}^i(s_t, a, s_{t+1}, N^i) \tag{2.2}$$

Equation (2.2) 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. For example, if the agent drives extremely recklessly, (and is caught), with immediate arrest and jail as sanction, a whole new branch in the MDP would result that may involve additional penalties,

and norm violations. 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: $\mathcal{M}^0, \mathcal{M}^{i,j,k...}$
$\mathcal{M}^0 = \langle S, A, R, T, \gamma \rangle$
forall $\{N^i, N^j, N^k, ...\}$ *in* $\text{Powerset}(M)$ **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 (2.2)
    $\mathcal{M}^{i,j,k,\cdots} \leftarrow \langle S, A, R^{N^{i,j,k,\cdots}}, T, \gamma \rangle$
  end
end
return $\mathcal{M}^0, \mathcal{M}^{i,j,k...}$;

Hence, for a given state s_t we can compute the best action a_t from (2.1) as described in Algorithm 2 and as shown in Figure 2.1(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.

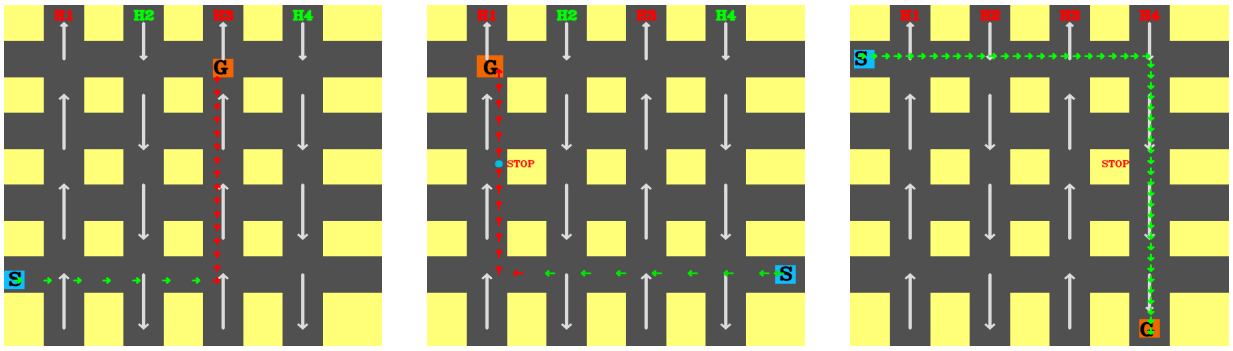
Algorithm 2: Selection of actions

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

2.3 Experimental Results

2.3.1 Roadway Simulator

We illustrate the efficacy of our MNMDP framework on a custom-designed roadway simulator encapsulating traffic rules such as speed-limits, STOP signs, and lane directions. The videos for these experiments can be found at this link: http://bit.ly/2GA4HLj. Figure 2.2 shows the output policies generated by our MNMDP framework for the four different scenarios. Lane directions to be followed are marked using arrows in the four vertical roadways H1-H4 and the speed-limit enforced is shown using red font on the highway signs. In the scenario represented in Figure 3.5(a) where the agent must follow the speed-limit norm on H1 and H3 in addition to the existing lane norms, the agent follows the speed limits imposed on H3 and slows down. In Figure 3.5(b), the agent encounters an additional STOP sign norm in addition to the norms discussed in the previous example. It follows the STOP sign placed in H1 as illustrated by the blue circle. In Figure 2.2(c), the agent encounters a hospital emergency in addition to the lane direction rules, speed-limit on all four vertical highways as well as a STOP sign norm on H4. However, the agent choose to disregard all other norms in favor of reaching the goal as quickly



(a) Direction and Speed-limit norms

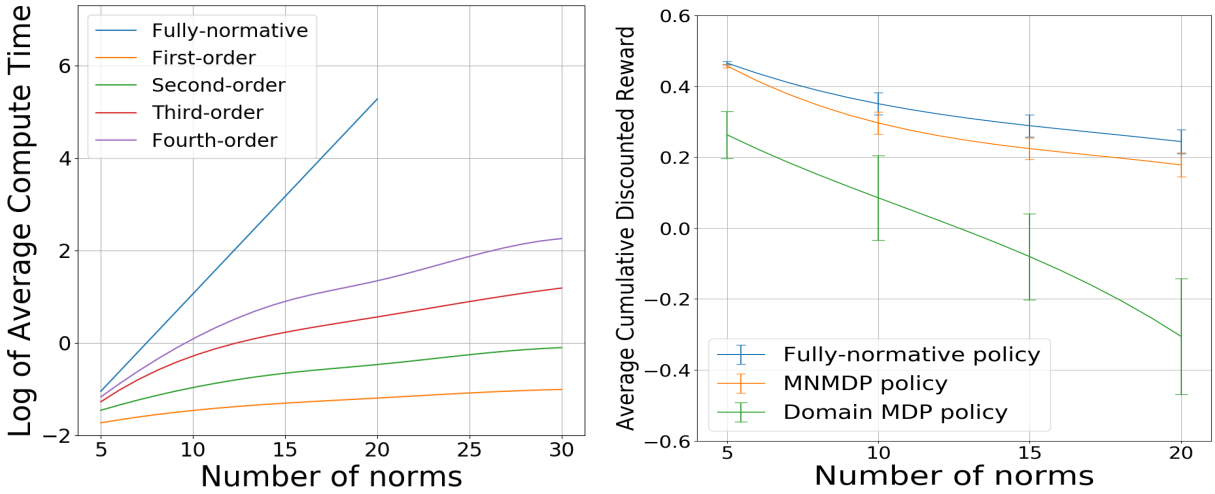
(b) Direction, Speed-limit and STOP sign norms

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

Figure 2.2: Trajectories of the agent in different norm constrained environments in a traffic example

as possible, since the hospital emergency norm (to save a human life) which conflicts with the other norms is given a higher priority.

2.3.2 Empirical Evaluation



(a) Log of computation time against number of norms

(b) Average Cumulative Discounted reward

Figure 2.3: 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.

In order to validate the scalability and efficiency of the proposed MNMDP, we compare the performance of our work against a fully normative MDP, which reasons about all the norms together by including all norms in the state space. For these experiments, we construct a domain MDP $\mathcal{M}^0$ with a random transition function T and a sparse, random reward function R. The number of norms $|N|$ is a parameter that can be varied during evaluation. Each of the norms can be characterized by a fixed number of norm state and action variables, which are both chosen

randomly. These settings are used so that we can mitigate any structural bias introduced by analyzing a set of hand-crafted examples, as well as to test the limits of our system with an increasing number of norms.

From Figure 2.3(a), we can see that K^{th} order interactions take x^K time. For the fully normative MDP, which considers all the norms in the norm set together, this becomes $x^{|N|}$, making it exponential in the number of norms. Figure 2.3(b) shows the average cumulative discounted rewards for the fully-normative MDP policy, our MNMDP policy, and the domain MDP policy. Note that the domain MDP policy disregards all normative constraints imposed on the system. We observe the general behavior that the cumulative reward decreases as the number of norms increases. We also observe that our MNMDP policy is only slightly sub-optimal to the fully-normative policy, which reasons in a very high-dimensional space, making it intractable for real systems.

Chapter 3

Context Representation and Modeling

3.1 Modeling Environmental Context to Determine Norm Activations

To be socially compliant, agents must determine which norms are relevant, or active, in a given scenario and leverage this information to robustly adapt their behavior. A critical challenge for creating such socially compliant agents is enabling these agents to use their low-level sensory observations of the environmental context to identify the appropriate normative behavior. This environmental context representation and the mapping from context to norms needs to be generalizable and learnable, especially for long-term autonomy applications where the agent could encounter novel context factors or changing user preferences. In such cases, the framework must scale, be robust enough to generalize to novel environmental contexts, and have the capability to continuously learn and adapt in response to changing user preferences and shifting socio-cultural norms. In this section, we detail our expressive and generalizable computational framework for modeling social context by combining logic-based approaches from artificial intelligence with data-driven, machine learning techniques.

3.1.1 Representing Environmental Context with Propositional Logic

To represent environmental context, we use propositional logic. With this approach, we model entities in the state, such as the various household objects and attributes, with atomic propositions and the interactions between the entities with logical connectives (e.g. AND, OR). Chaining connectives together enables us to build and express complex, intricate context conditions.

Propositional Logic for Determining Normative Constraints

We define our formula φ for modeling environmental context over a set of atomic propositions *AP*. Our formula has the following syntax:

$$\varphi ::= H^d s | H^d \neg s | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \neg \varphi_1 | \varphi_1 \cdot \varphi_2 | [\varphi_1]^{a,b} | X\varphi_1 | F\varphi_1 | \Box \varphi_1 | \varphi_1 U \varphi_2$$

where s is either the "true constant" T or an atomic proposition in *AP*; and $\wedge$, $\vee$, and $\neg$ respectively represent the conjunction, disjunction, and negation Boolean operators. The concatenation operator $\cdot$ specifies that first φ_1 must be satisfied and then immediately φ_2 must be satisfied. The *hold* operator H^d with $d \in Z_{\geq 0}$ specifies that $s \in AP$ should be repeated for d time units and the *within operator* $[\ ]^{[a,b]}$ with $0 \leq a \leq b$ bounds the satisfaction of ϕ to the time window $[a, b]$. The successor (next) operator $X\varphi_1$ specifies that φ_1 must hold at the next state, the sometimes operator $F\varphi_1$ is the eventually operator modeling the case where the condition φ_1 is eventually true sometime during the execution, the always operator $\Box\varphi_1$ specifies that φ_1 must be true in all states, and the until operator $\varphi_1 U \varphi_2$ means φ_1 until φ_2. The action operator $\triangledown$ represents an action φ_2 that is being taken by φ_1.

We use this logic to express more complex environmental contextual factors for activating or deactivating norms, such as:

- *Context in sequence*, which refers to when one contextual variable must occur after another one. For example, a conversation between a robot's owner and their guest can be modeled as follows:

  ((isOwner() $\triangledown$ isTalking()) $\cdot$ (isGuest() $\triangledown$ isTalking())) $\vee$ ((isGuest() $\triangledown$ isTalking()) $\cdot$ (isOwner() $\triangledown$ isTalking()))

  In other words, to be in a conversation, the owner must talk and then the guest must talk, or vice versa.

- *Context within a time window*, which refers to when a contextual variable must occur within a particular window of time according to the time counter for all agents. For example, person speaking twice within the time window of [0, 5] can be modeled as follows:

  isPerson() $\triangledown$ $[\mathrm{H}^2\mathrm{isSpeaking}()]^{[0,5]}$

  In other words, the person must speak twice within the first 6 time units according to the global time counter.

3.1.2 Constructing the Context Tree

In this section, we describe the structure of our knowledge base, which contains mappings between environmental context and norm activations. A realistic knowledge base for a domestic service robot will contain a wide range of scenarios, where each scenario is represented by a long chain of propositional logic to determine the relevant norms for that scenario. Independently searching through each of these scenarios to determine the relevant norms is highly inefficient. Therefore, we elect to structure our knowledge base as a tree, where nodes represent some context variable(s) that evaluate to either true or false. Like a decision tree, the most differentiating factors are placed near the root of the tree.

We depart from the traditional binary decision tree construction by permitting our context nodes to contain an additional attribute. This attribute corresponds to the set of activated intermediate norms given the evaluation of the evaluation of the parent nodes leading to the current node. This tree structure helps with representation and addresses two major problems that may arise in real robotics systems: (1) partially observable environments and (2) time-critical applications. In partially observable environments, some of the context cannot be evaluated, so by using

intermediate norm outputs, the robot can still determine norm activations given the context that it can observe. In time-critical applications, the agent must make a decision about which norms are activated without reaching the leaves of the tree, which is enabled by using intermediate norm outputs. Importantly, note that it is possible to learn these intermediate norm activation outputs using standard tree-learning algorithms.

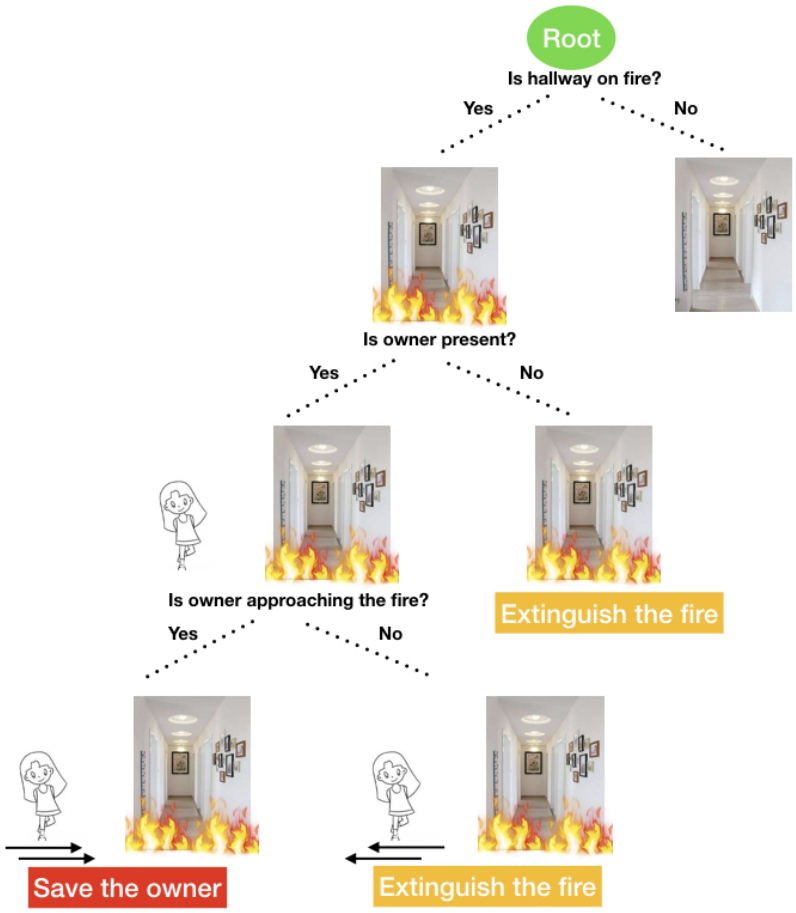


Figure 3.1: A simple example of modeling a scenario into a compact decision tree. In this example, the yellow boxes are associated with the danger norm; the red box is associated with the protect owner from harm norm.

Figure 3.1 illustrates a simple example of modeling a scenario into a compact decision tree. The root of this tree evaluates whether the hallway is on fire and if this is True, then the robot checks if the owner is in the hallway. If the owner is present, then the robot checks if the owner is approaching the fire. If so, then the norm activation of preventing the owner from incurring harm enables the robot to save the owner from the fire. On the other hand, if the owner is not approaching the fire or is not present, then the robot focuses on extinguishing the fire. Note that the intermediate norm output at the owner-in-hallway node is the output of the tree based on all the previously traversed nodes - in this case, only the root of the tree. In the other branch of the tree, where the presence of fire evaluates to False, the robot checks for the co-existence of itself and the owner in the hallway. If this is true, the robot determines that the accommodation norm is activated; if it is False, then no norm is activated. In this case, the intermediate output does not

enforce any norm since no fire was present and the location of the owner is unknown.

Constructing the Nodes of the Tree

As previously mentioned, the nodes of our tree represent some contextual variable(s) that evaluate to either true or false. We use the scenarios from recent work [19] as the input for creating our tree-structured knowledge base by, first, mapping each of these scenarios to our chained propositional logic formulae. We split each chained propositional formula based on the occurrence frequency of each sub-part in the chain throughout the set of contexts in our knowledge base. We use this co-occurrence as a measure to derive the features of our context tree so that we can reduce redundancies and hence speed up construction and traversal times. It is intuitive to think of these re-used features as a systematic way to narrow down the search by differentiating between the scenarios where they are present/absent. We set a threshold of interaction frequency f_t, which we use to determine whether an interaction should be separated out into its own node. If an interaction frequency exceeds f_t, then it is separated out into its own node. If the current interaction does not occur frequently in the knowledge base, then it remains chained together, as its information gain is low. The threshold f_t is a hyper-parameter that can be empirically tuned for a given application. For example, if the formula "((isOwner() $\triangledown$ isTalking()) $\cdot$ (isGuest() $\triangledown$ isTalking()))" (let's call it owner-guest talking) occurs in a scenario made up of other propositions chained to the above formula, but co-occurence frequency of the above formula in the knowledge base is high, then we can split the chained propositional formula at owner-guest talking not only to promote reuse, but also to help differentiating between all the scenarios with the same owner-guest talking interaction to the ones that don't.

3.2 Learning Context Trees from human experiments data

Given a set of scenarios as chained propositional formula, we can split them into a set of interaction nodes as discussed in the previous section. Some of these nodes can be much more differentiating than the others. Although the norm output depends on each specific context that the scenario has, there exists an efficient order in which we can check the interaction nodes. For example, the robot should first check the location to see if it is in the bedroom, kitchen, and hallway etc, and then decide which context needs to be examined according to the location. It would not check the food allergy in bathroom, but would only do that in the living room or the kitchen. With a increasing number of scenarios in the knowledge base, a hierarchical structure of context nodes is an intuitive way to model the problem. For a wide variety of general scenarios, we need to be able to learn the most efficient hierarchical representation of the context nodes, such that we can search and retrieve the norms associated with our scenario as fast as possible. Hence, we draw tree learning methods from the machine learning literature that enables us to learn our context trees directly from user data collected in our human experiments [19].

We base our tree construction algorithm on the CART [2] framework. To help us place the most differentiating nodes closer to the root node of the tree, we use a multi-variate variant of the Gini impurity measure [6] as our metric. Hence, as we traverse down the tree, we iteratively refine our output norm activation by conditioning it on the current context node that is being

evaluated. Each of the leaves of this tree contains the final norm activations for the corresponding paths traversed to reach them.

We use data from human experiments [19] which contain independent priority values for each norm class, for each scenario. Thus, we pose our tree learning as a multi-variate regression problem similar to [7]. We create a dataset D containing Z scenarios with each scenario characterized by an input vector of m different propositional functions $X_1, ..., X_m$. We have $x^{(i)} = (x_1^{(i)}, ..., x_m^{(i)})$ as our features for the i^{th} training instance. Our output vectors for the d different norm classes are $Y_1, ..., Y_d$. The output vector for the i^{th} training instance is $y^{(i)} = (y_1^{(i)}, ..., y_d^{(i)})$ with each $y_l^{(i)} \in [1, 7], l \in \{1, ..., Z\}$. We redefine the impurity measure for our multi-variate learning problem by making it the sum of squared error over the multi-variate response for each node,

$$L = \sum_{i=1}^{Z} \sum_{j=1}^{d} (y_j^{(i)} - \bar{y}_j) \tag{3.1}$$

where $y_j^{(i)}$ denotes the value of the output variable Y_j for the i^{th} instance and $\bar{y}_j$ is the mean of Y_j in the node, with each split selected to minimize this sum of squared error. The computational complexity for constructing this regression tree with m different context features encapsulating Z different scenarios is $O(m\ Z log\ Z)$. We construct this tree offline before policy execution.

3.2.1 Efficiently Traversing the Context Tree

During policy execution, the robot uses its current observation as input to our context tree to efficiently find the norms that are to be enforced, so that the norm-sanctioned policies could be retrieved and executed. Starting from the root, we update the graph edge activation function α using the current observation of the agent. After that, we evaluate the current context node's logic function using α. If the current node P has no implication for the given observation O, then we use our approach for generalization using the knowledge graph K as explained in Section 3.2.3 to find the most similar substitute with similar semantics and then continue our traversal from there. Given the structure of our decision tree model, which also stores the intermediate norm outputs, we incorporate the presence of possible time-constraints and/or partial observability to output the norm N without reaching the leaf nodes. Once the branch reaches a leaf node, the search time T is more than a set threshold τ, or the current node P is not observable, we return the predicted norm priorities given the observed context. These norm priorities ρ are then used to resolve any conflicts between norms in our norm set. Note that a set of norms conflict when a state becomes simultaneously prohibited by some norm and permitted/obliged by others. After resolving norm conflicts, we use a set of priority thresholds ρ_t to estimate the value of our norm activation function ϕ_a and the deactivation function ϕ_d which is modeled as the former's complement. Hence, we find the set of activated norms from the predicted priorities. The search procedure for our context tree is executed at each time step and is $O(log\ n)$, where n is the total number of nodes in the context tree.

As explained in Section 3.1.2, we build our context tree C from the given scenarios S. Then, at each time-step, the agent's observation is used to search through the context tree to obtain the

active norms N, as explained in Section 3.2.1. Then, we use the MNMDP framework to find the pre-computed MNMDP policy π, which reasons about the set of active norms, and execute actions from π. We repeat this process until we reach a terminal state in the MDP. We show the significant computational advantages of using this approach in Section 3.3.2 compared to a tabular baseline. Note that the knowledge graph K is used to generalize our approach to scenarios outside our knowledge base, as will be explained further in Section 3.2.3.

3.2.2 Decision Tree Illustration

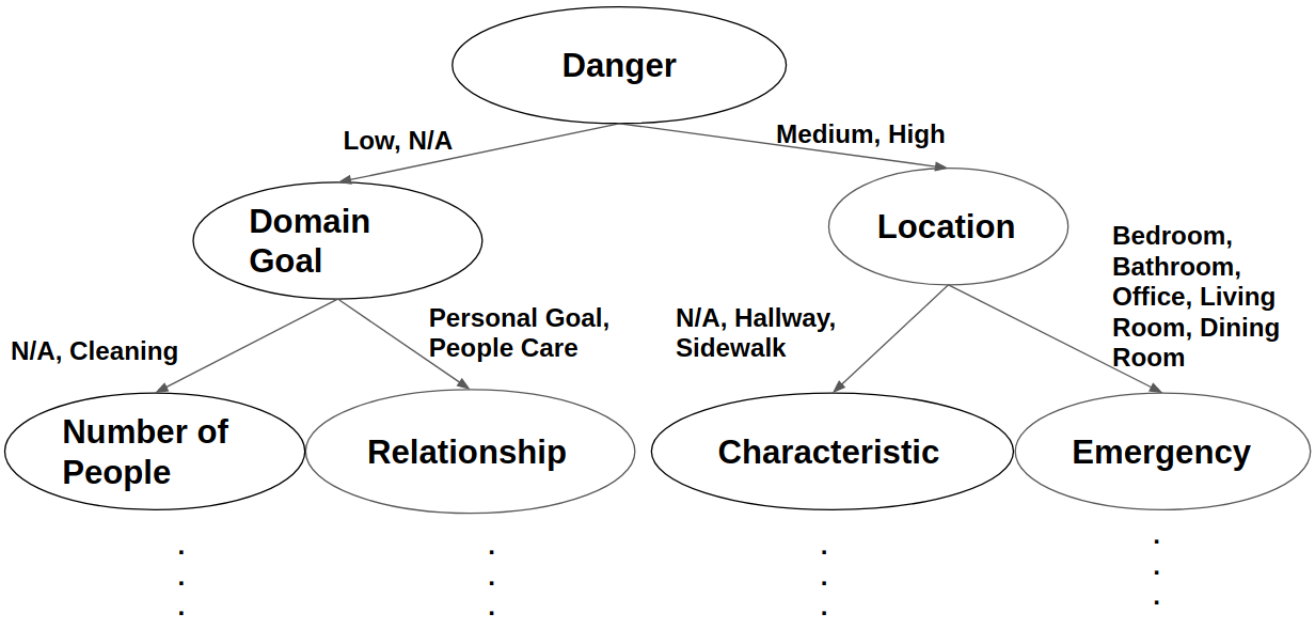


Figure 3.2: Decision Tree hierarchy illustration using the data from our human experiments with 11 different categorical context features.

Figure 3.2 shows the top 3 levels of the constructed decision trees using the categorical context features present in our human experiments data. Danger was found to be the most differentiating feature of our decision tree. The agent can then reason about its domain goal in the case where there is low or no danger. Then if the goal of the agent is to clean, then it takes into account the number of people present in the environment. For tasks such as personal goal and people care, it could start looking at relationship of the people present in the environment. In the presence of danger, the agent then looks at its location where the danger is present. For private locations inside the house such as bedroom, bathroom etc. it investigates the presence of an emergency presumably to reason about privacy related norms. For outdoor locations such as hallway, sidewalk etc., it then looks at the characteristic of the agent such as beggar, pet, elder, children etc. The tree is complex and continues on refining the scenarios further and further until it can differentiate them in terms of the norm activations they entail. The leaf nodes of this regression tree contain the output priorities of each of the norms for the given context path, thereby adapting the precedence of norms depending on the situation.

Generating explanations from context trees: By modeling high-level interactions between objects in the environment in the nodes of the context tree, each node is grounded to concepts that are easily understood by humans. Hence, our system not only provides the norm activations for the given context, but also directly generates a decision path from the given raw observations, which can be used as an explanatory model. When deployed in a real system, we could store the decisions made by the system in a computationally inexpensive way so that we could retrieve and analyze the data at a later stage. This data could be very useful for designers, vendors and investigators.

3.2.3 Generalizing to Unseen Context

Although our approach of combining propositional logic with decision trees makes for an expressive and efficient framework, it cannot directly generalize to unseen scenarios. One way to achieve generalization is to traverse down the tree to the node where no further implications are present and then use its partial norm output. However, this approach can lead to incorrect inferences of norm activations or deactivations - especially in novel situations.

A better approach is to use similarity metrics to approximate the unknown interaction to some known interaction in a larger knowledge base. To that end, we construct a k-partite interaction graph using the same nodes in the context tree and making dense connections between the nodes based on their co-existence in the scenarios present in our knowledge base. The k different partitions are determined using a heuristic on the categories present in the node (e.g. agent, action, location). By using the aforementioned high-level semantic types to partition the data into a k-partite graph, we restrict the similarity space of a given node to only its neighbours in the same category. The intuition is that nodes containing highly similar semantic information will have common connections in the interaction graph. With this approach, when the traversal procedure encounters a context variable that was previously unseen in a scenario, the knowledge graph is used to retrieve the most similar context variable. To determine the most similar context variable, we use the SimRank similarity measure.

Using the SimRank Similarity Measure for Generalization

SimRank is a well-studied, graph-theoretic structural similarity measure applicable in domains with object-to-object relationships [11]. Objects and relationships are modeled as a directed graph $G = (V, E)$, where nodes in V represent objects of the domain, and edges in E represent relationships between objects. For a node v in a graph, the set of in-neighbors and out-neighbors of v are denoted by $I(v)$ and $O(v)$, respectively. Individual in-neighbors are represented as $I_i(v)$, for $1 \leq i \leq |I(v)|$; individual out-neighbors are represented as $O_i(v)$, for $1 \leq i \leq |O(v)|$. SimRank computes similarity scores between nodes using the structural context in which they appear. The similarity $s(a, b) \in [0, 1]$ between objects a and b is 1 if $a = b$, otherwise:

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I + i(a), I_j(b)) \tag{3.2}$$

where C is a constant between 0 and 1.

We decompose the scenarios into the key features, and each scenario contains exactly one categorical value for each feature. We construct the k-partite graph by assuming the possible values (aka entities) to be nodes and put a bi-directional edge linking between two nodes if they both appear in a single scenario. Thus for each scenario, there are k edges constructed, where k is the number of features. After processing all the scenarios, some pair of nodes might be linked with more edges than other pairs, which means the entities represented by the two nodes appear more frequently together than others.

As we know for each scenario, there is only one value that can be taken for each feature. It means the edges cannot be formed between two nodes from the same feature, and the nodes from the same feature form groups. As there are k features, we have a k-partite graph. Applying the SimRank similarity metric in Equation 3.2, we have the similarity metric for each feature accordingly. The similarity metric explains for each feature, how similar each pair of entities are. This similarity is based on the intuition that similar nodes are referenced by the nodes that are also similar. Here we assume if the two entities are linked frequently by similar entities, they have a high score in similarity. Therefore, when given a new scenario unseen before, it is firstly checked through the context tree to the point that it cannot find a reference. Then for this feature that it cannot find a reference, it picks the branch that is the most similar to its entity in this feature. Finally when it is led to the most similar available scenario, our model outputs the same norm activations that are associated with the similar scenario obtained in the process.

We illustrate our approach for generalization using the following example. Suppose the robot has never observed the guest cooking in the kitchen; however, it has observed its owner cooking in the kitchen. When traversing the context tree, after encountering the location node (kitchen) and the task node (cooking), it encounters its first previously unseen context variable: the guest. Using the SimRank similarity metric in Equation 3.2 and the knowledge graph, the most similar context variable to the guest is determined - in this case, the robot's owner. The robot's owner then replaces the guest in the context tree, evaluates to true, and the search proceeds to the next context variable or corresponding active norm(s). The active norms for the owner cooking in the kitchen are then used for the scenario of the guest cooking in the kitchen.

3.3 Experimental Evaluation

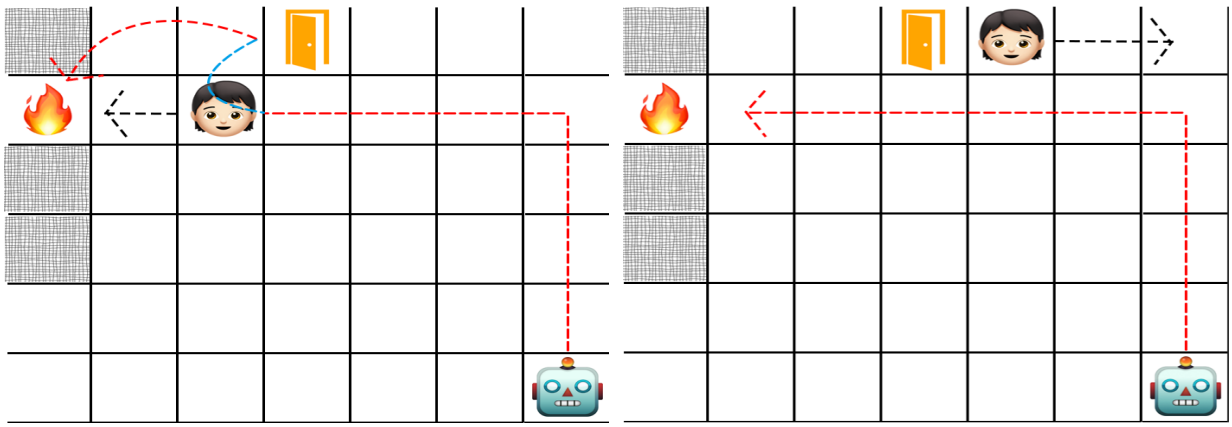
In this section, we present our results of the implementation of our model for representing environmental context and determining norm activations. First, we deploy our context model coupled with our MNMDP framework in custom-built simulation environments emulating complex social norms built on top of Minecraft. This shows the feasibility and effectiveness of our approach for usage in autonomous agents. Then, we proceed to empirical performance studies aimed at determining the computational advantages of using our framework.

3.3.1 Minecraft

We use the Project Malmo platform [12] to construct our scenarios and run our experiments. Project Malmo is a platform built on top of the open-world game Minecraft, where researchers can define many diverse, complex problems for intelligent agents to solve. Despite the large

Figure 3.3: Example stove scenario in Malmo. The robot (the male figure with black hair) is monitoring the kitchen when, suddenly, a fire starts on the stove in the upper left corner. At the same time, a child (the female figure with blonde hair) enters the kitchen. The robot determines if it should first put out the fire or move the child to a safe area, then put out the fire.

number of environments created in Malmo, to our knowledge, no domestic environments currently exist for the platform. To address this, we created four domestic scenarios for testing our context-reasoning model (two of which stem from our survey [19]), which we describe in this section. We use the scenario illustrated in Figure 3.3, as the primary scenario to illustrate our results.



(a) Child approaching the fire

(b) Child away from the fire

Figure 3.4: Symbolic view showing the output trajectories generated by using our context tree model for the stove scenario.

As illustrated in Figure 3.3, in this scenario the robot is monitoring the kitchen when, suddenly, there is a fire on the stove. At the same time, a child enters the kitchen and either approaches the fire or approaches the window to look outside. As soon as the robot sees the fire,

the norm to extinguish the fire is activated. However, when the robot observes the child entering the kitchen, it must decide between leading the child out of the kitchen and then returning to put out the fire, or ignoring the child and immediately extinguishing the fire.

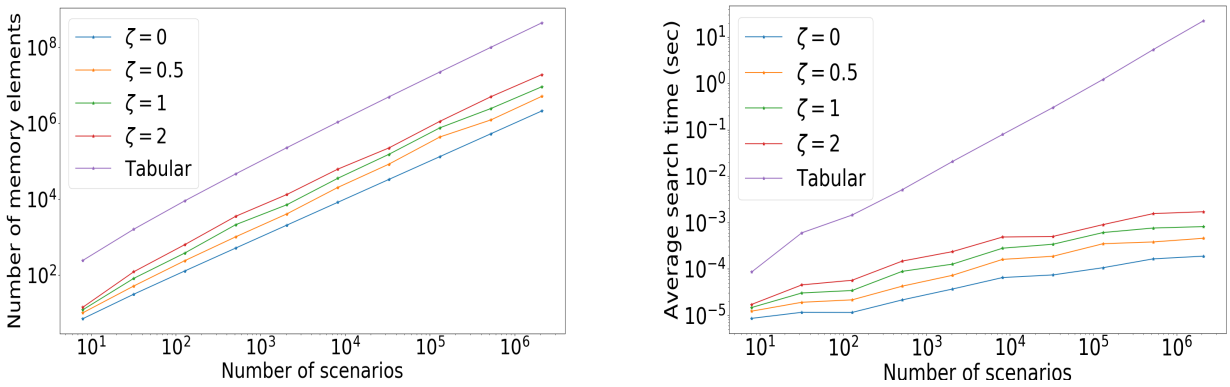
In our scenario, if the child is far away from the fire, the robot chooses to first extinguish the fire; otherwise, it chooses to first redirect the child away from the fire. After the fire is extinguished, the robot returns to its original location and continues to monitor the kitchen. If there is no fire, the robot stays still and does nothing even if the child comes in and approaches the stove. In this scenario, it seems that the two norms of extinguishing the fire and save the child have conflicts with each other, but indeed the norm of extinguishing the child has been "cached" when the norm of saving the child has been activated because the latter has the higher priority. However, when the child has been saved and the corresponding norm has been deactivated, the robot resumes the norm of extinguishing the fire based on its observation on the continuing existence of fire. Figure 3.4 shows the behavior trajectories of the robotin the different scenarios discussed above. The video of the implementation of this scenario can be found **here** or at https://vimeo.com/319540509.

Implementation details In order to obtain information about the objects and agents in the given scene, we tap into Malmo's symbolic observations. This helps us build the observation map containing all relevant interaction details required for our model. We also use the chat box in Malmo, which allows agents to globally communicate messages with each other, to simulate conversations between agents. We add this chat observation and a time counter for our sequential tasks to our observation map. We use the scikit-learn framework [24] for our tree learning. In order to augment the learning process for the scenarios implemented in Malmo, we use the additional heuristic that the two most differentiating context factors are the robot's assigned task and its location. To match the specifications of the scenario, we pre-specify the behavior of other agents (e.g. human agent walking in the hallway) and objects in the environments.

3.3.2 Performance Study

To the best of our knowledge, our work is the first to provide a framework for context modeling for norm aware reasoning, and hence there are no existing benchmarks to evaluate our model against. Hence, we consider a tabular approach with a populated list of scenarios as the baseline against which we can compare our performance against. Our experimental setup for these experiments involves randomizing chains of propositions together to form our interactions. This way we construct our set of scenarios and build our context tree as in Section 3.1.2.

Using this experiment, we wish to study the scalability of our approach with increasingly unbalanced trees. Consider the mean depth of a given context tree to be d_μ and the standard deviation of the tree depth to be d_σ. The coefficient of variation $\zeta = \frac{d_\sigma}{d_\mu}$, which the ratio between the standard-deviation and the mean depth of the tree, capturing the degree of imbalance of the tree for any d_μ and d_σ. We try various different ζ against the number of scenarios to show the scalability trends of our approach. Figure 3.5 shows the performance comparison between our approach and the tabular approach. Since the context trees become increasingly unbalanced with increasing ζ, the memory and search time increase correspondingly, but still remains close



(a) Memory consumption

(b) Average search time

Figure 3.5: Performance comparison between the tabular approach and our proposed context tree approach. Note that both the X and Y axes are in log-scale.

together compared to the baseline. Therefore, the proposed context tree approach is orders of magnitude better than the tabular approach, making it suitable for usage in real systems.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

In this work, we proposed a scalable, and generalizable approach towards integrated norm-aware reasoning for autonomous systems. We proposed the MNMDP framework which provides a generalizable representation for norms and integrates it into the domain planning alleviating the curse of dimensionality problem encountered by previous approaches. Our MNMDP framework is well-suited for long-term autonomy applications since it needs little re-computation for any modifications made to the norm set. We illustrated its working using a roadway simulator modeling traffic rules and also showed the computational advantages of using our MNMDP framework over existing methods. We motivated the need to model the activation function of the MNMDPs to successfully deploy the MNMDP approach in complex environments. We showed that our novel approach, which combines propositional logic and decision trees, for modeling the activation function is expressive, scalable, and interpretable. We illustrated the magnitude of the performance boost of our context model compared to a baseline with respect to both compute time and memory consumption. Using our context modeling approach, we demonstrated that we can efficiently identify which MNMDPs to use in various environmental conditions. Additionally, we showed that we can resolve normative conflicts using norm priorities, which are conditioned on the environmental context, rather than on global priorities as in previous work.

4.2 Future Work

As future work, we plan to make extensions to various parts of our approach. For the MNMDP framework, we aim to extend our approach to support partially observable environments. Additionally, we plan to learn estimates of the reward and penalty functions for the MNMDPs directly from user behavior using inverse reinforcement learning. Furthermore, we want to build a more concrete representation of the human normative reasoning process to improve our human norm network. To accomplish this goal, we plan to better capture the hierarchical structure and mutual activation of norms.

For the context modeling component, we plan to make the following extensions. First, we will extend our knowledge base with more diverse, yet realistic, social scenarios to improve

the robustness of our models and enable our generalization approach to achieve better results. Second, we will extend our context tree to support online learning, which will make it better suited for long-term autonomy applications. To accomplish this, we will use an active learning component in which people in the environment can provide the robot with feedback regarding the appropriateness of its inferred activated norms. This feedback can then be used to create new connections in the knowledge graph and the context tree. Third, we plan to investigate the use of concepts from TWTL to extend our logic to support temporal relaxation for satisfying time constraints.

Chapter 5

Bibliography

[1] Henk Aarts and Ap Dijksterhuis. The silence of the library: environment, situational norm, and social behavior. *Journal of personality and social psychology*, 84(1):18, 2003. 1

[2] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. belmont, ca: Wadsworth. *International Group*, page 432, 1984. 3.2

[3] Geoffrey Brennan, Lina Eriksson, Robert E Goodin, and Nicholas Southwood. *Explaining norms*. Oxford University Press, 2013. 1

[4] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002. 1.1

[5] Matthew Brown, Sandhya Saisubramanian, Pradeep Varakantham, and Milind Tambe. Streets: game-theoretic traffic patrolling with exploration and exploitation. In *Twenty-Sixth IAAI Conference*, 2014. 1.1

[6] Don Coppersmith, Se June Hong, and Jonathan RM Hosking. Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, 3(2):197–217, 1999. 3.2

[7] Glenn De'Ath. Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology*, 83(4):1105–1117, 2002. 3.2

[8] Moser Silva Fagundes, Sascha Ossowski, Michael Luck, and Simon Miles. Using normative markov decision processes for evaluating electronic contracts. *AI Communications*, 25 (1):1–17, 2012. 1.1, 2.2.1

[9] Moser Silva Fagundes, Sascha Ossowski, Jesús Cerquides, and Pablo Noriega. Design and evaluation of norm-aware agents based on normative markov decision processes. *International Journal of Approximate Reasoning*, 78:33–61, 2016. 1, 1.1, 2.1.3, 2.2, 2.2.1

[10] Jack P Gibbs. Norms: The problem of definition and classification. *American Journal of Sociology*, 70(5):586–594, 1965. 2.1

[11] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002. 1.2, 3.2.3

[12] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell. The malmo platform for artificial

intelligence experimentation. In *Proc. 25th International Joint Conference on Artificial Intelligence*, 2016. 3.3.1

[13] D. Kasenberg and M. Scheutz. Inverse norm conflict resolution. In *Proc. 1st AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society*, 2018. 1.1

[14] Daniel Kasenberg and Matthias Scheutz. Norm conflict resolution in stochastic domains. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 1

[15] Martin J Kollingbaum and Timothy J Norman. Noa-a normative agent architecture. In *International Joint Conference on Artificial Intelligence*, pages 1465–1466, 2003. 1.1

[16] Martin J Kollingbaum and Timothy J Norman. Norm adoption in the noa agent architecture. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1038–1039. ACM, 2003. 1.1

[17] V. Krishnamoorthy, W. Luo, M. Lewis, and K. Sycara. A computational framework for integrating task planning and norm aware reasoning for social robots. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2018. 1.1, 1.2, 2.2

[18] Vigneshram Krishnamoorthy, Wenhao Luo, Michael Lewis, and Katia Sycara. A computational framework for integrating task planning and norm aware reasoning for social robots. In *INternational Conference of Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2018. 1.2

[19] H. Li, S. Milani, V. Krishnamoorthy, M. Lewis, and K. Sycara. Perceptions of domestic robots' normative behavior across cultures. In *Proc. 2nd AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society*, 2019. 2.1.2, 3.1.2, 3.2, 3.3.1

[20] Felipe Meneguzzi, Odinaldo Rodrigues, Nir Oren, Wamberto W Vasconcelos, and Michael Luck. Bdi reasoning with normative considerations. *Engineering Applications of Artificial Intelligence*, 43:127–146, 2015. 1.1

[21] Jean Oh, Felipe Meneguzzi, Katia Sycara, and Timothy J Norman. Prognostic normative reasoning. *Engineering Applications of Artificial Intelligence*, 26(2):863–872, 2013. 1.1

[22] Nir Oren, Wamberto Vasconcelos, Felipe Meneguzzi, and Michael Luck. Acting on norm constrained plans. In *International Workshop on Computational Logic in Multi-Agent Systems*, pages 347–363. Springer, 2011. 1.1

[23] Sofia Panagiotidi and Javier Vázquez-Salceda. Norm-aware planning: Semantics and implementation. In *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 33–36, 2011. 1.1

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 3.3.1

[25] Jéssica S Santos, Jean O Zahn, Eduardo A Silvestre, Viviane T Silva, and Wamberto W Vasconcelos. Detection and resolution of normative conflicts in multi-agent systems: a literature survey. *Autonomous agents and multi-agent systems*, 31(6), 2017. 1.1

[26] V. Sarathy, M. Scheutz, Y. N. Kenett, M. Allaham, J. L. Austerweil, and B. F. Malle. Mental representations and computational modeling of context-specific human norm systems. In *CogSci*, 2017. 1, 1.1

[27] V. Sarathy, M. Scheutz, and B. Malle. Learning behavioral norms in uncertain and changing contexts. In *8th IEEE International Conference on Cognitive Infocommunications*, 2017. 1.1

[28] Wamberto W Vasconcelos, Martin J Kollingbaum, and Timothy J Norman. Normative conflict resolution in multi-agent systems. *Autonomous agents and multi-agent systems*, 19 (2):124–152, 2009. 1.1, 2.1.3

[29] Georg Henrik Von Wright. An essay in deontic logic and the general theory of action. 1968. 2.1