# Extended Abstract: Grounding Language to Reward Functions With Abstract Markov Decision Processes

Dilip Arumugam Sidd Karamcheti James MacGlashan Nakul Gopalan Stefanie Tellex

***Abstract*—The abstract goes here.**

## I. Introduction

When dealing with the issue of efficiently solving large-scale sequential decision making problems defined by complex state-action spaces, the ability to identify and leverage multiple layers of abstraction is key. Developing good methods of abstraction represents a logical way to accelerate planning time by breaking apart large, complex tasks into smaller sub-tasks. Moreover, this notion of decomposing tasks into their smaller constituent pieces more closely resembles our own human methodology for solving problems by identifying and achieving sub-goals. Our goal in this work is to analyze an alternative method of abstraction for sequential decision making problems that utilize multiple layers of abstraction over all components of the Markov Decision Process (MPD) in order to efficiently learn in complex domains. In order to convey the strength of the abstraction imposed by the abstract Markov decision process (AMDP) formalism, we extend prior work on grounding natural language commands to reward functions and, in particular, evaluate the abstraction based on the success of these groundings.

## II. Prior Work

Typically, methods of abstraction have been applied to only a single element of the standard Markov decision process (MDP) formalism in the hopes of improving overall problem tractability. Specifically, there have been some approaches that attempt to perform state abstraction [5] by compressing similar states into single units or intelligently pruning states that are irrelevant to the overall task from consideration. Other approaches examine abstraction over the action space via macro-actions [2] or options [7] that compose sequences of atomic actions into a single "action". Unfortunately, these methods suffer from potentially increasing the size of the state or action space thereby increasing the time needed for planning [3].

## III. Abstract Markov Decision Processes

Our approach in tackling this problem of abstraction is to utilize abstract Markov decision processes (AMDPs) which represent a decomposition of a regular MDP resulting in a tiered hierarchy where each level represents an abstraction over the previous one. To briefly summarize, MDPs are represented by a five-tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of possible states in the world; $\mathcal{A}$ is a set of all available actions that can be executed; $\mathcal{T}$ is a function that computes the probability of transitioning from one state to another by executing an action; $\mathcal{R}$ is a function specifying the agent's reward for taking an action within a particular state ; and $\gamma$ is a discount factor.

Building on top of this, AMDPs take a layered approach by maintaining a stack of MDPs, each of which represents a different layer of abstraction for the input problem. The base (lowest) level MDP is always the MDP representing the original problem while higher level MDPs are hard coded over smaller state and action spaces (with corresponding transition dynamics and reward functions). Each state and action in a higher level MDP is implemented as an abstraction over multiple states and actions in the lower level creating a many-to-one mapping (as we move up the stack) resulting in smaller, more tractable state-action spaces. Intuitively, moving up the stack of MDPs captures the notion of identifying smaller, easier sub-goals within the problem. In terms of execution, a learning agent always begins at the base level MDP; whenever an action must be selected at a lower level, the current state information is fed upward and mapped to the next highest level of the AMDP effectively forcing the agent to think at one higher level of abstraction. This process repeats until the top of the MDP stack is reached, at which point, standard reinforcement learning algorithms are applied in order to compute a stationary policy for the entire level (which is then stored and reused so as to avoid redundant computation). Given this policy, the action required at highest level of abstraction becomes known and a series of surjective state mappings are maintained from higher to lower level MDPs allowing for a sort of backpropagation to occur once a higher level sub-task has been solved. This process of mapping backwards, computing a policy, and taking the specified action then repeats all the way down back to the base level MDP. It is important to note that since policies computed at each level of the AMDP are stored, the true cost of planning in an AMDP is only equivalent to the cost of planning and solving all higher level MDPs each of which is decreasing in size and complexity as we move up the AMDP stack.

## IV. Evaluation

### A. Experiments

In order to examine the effectiveness of the abstractions imposed by these Abstract Markov Decision Processes (AMDPs), we extend prior work by MacGlashan et. al. on grounding natural language commands to the reward function of a Markov Decision Process (MDP). The primary contribution of

this prior work is the ability to represent tasks as MDP reward functions and further combine learning from demonstration methods with language models to directly embed a user's verbal command into the reward function [6]. Specifically, given a dataset of natural language commands and their associated task demonstrations, inverse reinforcement learning (IRL) is applied to the demonstrations in order to inform a distribution over possible reward functions which is then used to train a language model in a weakly supervised fashion. The authors followed the paradigm set up in MacGlashan et. al. and used an IBM Model II Translation System for translating between natural language and reward functions (machine language), augmented by the provided demonstration. More specifically, we modify the IBM Model II decoding procedure to weight the probability of a given translation (the probability of a machine language command given the source language command) by the probability of the machine language command given the provided demonstration. With this process, the success of the resultant models in grounding reward function tasks is dependent on both the various natural language commands across the dataset as well as the provided demonstrations.

We show that grounding an arbitrary natural language command to a reward function will be most successful when the natural language command and reward function correspond to the same level of abstraction. Staying consistent with the experimental method used by MacGlashan et.al., we leverage objected-oriented MDPs [1] within the Brown-UMBC Reinforcement Learning and Planning library (BURLAP) to conduct our experiments. As with MacGlashan et. al. we use the Cleanup World Domain inspired by Sokoban [4].

We build our own AMDP over this domain, with each level corresponding to a different level of abstraction. There are three levels: **low**, **mid**, and **high**. The low-level MDP corresponds to simple directional commands, like telling the agent to move north, south, east, and west. The mid-level MDP corresponds to commands referencing the agent, the block, the doors in each of the domain's three rooms, and the rooms themselves. Finally, the high-level MDP abstracts away doors, so that these commands only reference agents, blocks, and rooms.

To test the effectiveness of using AMDPs, we collect datasets for each level, with each dataset consisting of a list of natural language commands that match the given level's abstraction and their respective demonstrations. The following are some examples of natural language commands at each level of the AMDP:

**Low:** Go three steps south, then two steps east.

**Mid:** Go to the red door, then go into the red room.

**High:** Go to the green room.

Our experiment consists of training IBM II Translation Models between every permutation of the levels of the AMDP. More precisely, we train a model to translate a natural language command to a machine language reward function for each level of the AMDP. We then evaluate the models by passing in a natural language command of a given level into each of the three models. In this "swap", we show that we get the best results for a given level of the AMDP with the corresponding natural language command, rather than any others. We use Leave-One-Out (LOO) accuracy as the evaluation metric.

### *B. Results*

The following table shows our LOO results for each swap. The rows are labeled with the level of the Natural Language (NLP) command, while the Columns designate the level of the MDP to which the language is grounded (the level-specific IBM II Translation system.)

TABLE I
LOO ACCURACY FOR EACH SWAP

| | MDP Level 0 | MDP Level 1 | MDP Level 2 |
|---|---|---|---|
| NLP Level 0 | **30.0 %** | 0.0% | 0.0% |
| NLP Level 1 | 0.0% | **0.0%** | 20.0% |
| NLP Level 2 | 0.0% | 10.0% | **10.0%** |

## V. CONCLUSION AND FUTURE WORK

Lorem ipsum

## REFERENCES


[1] Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In *ICML*, 2008.

[2] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas L. Dean, and Craig Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *UAI*, 1998.

[3] Nicholas K. Jong, Todd Hester, and Peter Stone. The utility of temporal abstraction in reinforcement learning. In *ATAL*, 2008.

[4] Andreas Junghanns and Jonathan Schaeffer. Sokoban: Enhancing general single-agent search methods using domain knowledge. *Artif. Intell.*, 129:219–251, 2001.

[5] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

[6] James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael L. Littman, Smaranda Muresan, Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. Grounding english commands to reward functions. In *RSS*, 2015.

[7] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112: 181–211, 1999.