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

Author Names Omitted for Anonymous Review. Paper-ID [add your ID here]

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

I. Introduction

When dealing with large-scale sequential decision making problems defined by complex state-action spaces, the ability to identify and leverage multiple layers of abstraction becomes key to efficiently solving these problems.The overall issue of developing good methods of abstraction has been key to the field of reinforcement learning as it represents a logical way to accelerate planning time breaking apart large, complex tasks into smaller sub-tasks. Moreover, this notion of decomposing task into its 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 utilizes multiple layers of abstraction over all components of the MDP 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 [4] 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 [6] 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 γ 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

The primary contribution of the grounding natural language to reward functions work done by MacGlashan et.al. 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 [5]. 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 found an IBM Model II for translating between natural language and reward functions (machine language) to be the most effective model. By this process, the success of the resultant models in grounding reward function tasks is dependent on both the representation of various natural language commands across the dataset as well as the variety in proposed tasks.

Our hypothesis is 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. For the purpose of staying consistent with the experimental method used by MacGlashan et.al., we leveraged objected-oriented MDPs [1] within the Brown-UMBC Reinforcement Learning and Planning library (BURLAP) to conduct our experiments.

B. Results

Lorem ipsum

V. Conclusion

Lorem ipsum

Acknowledgments

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

[5] 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.

[6] 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.