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

The issue of abstraction is key to the field of reinforcement learning as it represents a logical way to accelerate planning time within sequential decision making problems by breaking apart large, complex tasks into smaller sub-tasks. Most often, methods of abstraction have been applied to only a single element of the standard Markov decision process (MDP) formalism for reinforcement learning. Specifically, there have been some approaches that attempt to perform state abstraction [4] by compressing similar states into single units or simply pruning irrelevant states 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, many of these methods fall short in that they simply add more elements to either the state or action space thereby increasing the time needed for planning [3].

Our approach in tackling this problem of abstraction is to utilize abstract Markov decision processes (AMDPs) which represent a decomposition of a regular MDPs creating multiple layers of abstraction. To briefly summarize, MDPs are represented by the five-tuple: $\langle\ \mathcal{S},\ \mathcal{A},\ \mathcal{T},\ \mathcal{R},\ \gamma\ \rangle$, where $\mathcal{S}$ is a set of states of the world; $\mathcal{A}$ is a set of actions that the agent can take; $\mathcal{T}$ is a function that computes the probability of transitioning from one state to another via applying a specified 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. AMDPs take on a slightly different form by maintaining a stack of internal MDPs, each of which represents a different layer of abstraction for the input problem. The base level MDP represents the original task as it was given, while higher level MDPs are hard coded over abstracted state and action spaces (with corresponding transition dynamics and reward functions). These higher level MDPs capture the notion of sub-goals within the problem and so whenever the agent in a lower level need take an action, the current state information is fed upward to the next highest level of the AMDP. Once the highest MDP is reached, 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. Surjective state mappings are maintained from higher to lower level MDPs allowing for a backpropagation to occur once a higher level sub-task has been solved and this process then repeats onnce the backpropagation returns all the way down 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.

In order to examine the effectiveness of the abstraction imposed by AMDPs, we extend prior work by MacGlashan et.al. on grounding natural language commands to the reward function of an MDP [5]. In particular, we take an AMDP and evaluate the effectiveness of grounding a reward function at each level of the AMDP using natural language commands taken individually from each level of abstraction. In the context of the cleanup world domain used by MacGlashan et.al., this corresponds to an AMDP with three layers of abstraction: high, mid, and low.

We take a natural language command from an arbitrary level and apply IBM Model II translation to ground it to a reward function for each level of the AMDP.

We show that grounding an arbitrary natural language command to a reward function will be most successful when the natural language command and MDP correspond to the same level of abstraction. For the purpose of staying consistent with the experimental method used by MacGlashan et.al., we will leverage objected-oriented MDPs [1] within BURLAP to conduct experiments and collect results.

II. Section

Section text here.

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

III. RSS citations

Please make sure to include `natbib.sty` and to use the `plainnat.bst` bibliography style. `natbib` provides additional citation commands, most usefully `\citet`. For example, rather than the awkward construction

```
\cite{kalman1960new} demonstrated...
```

rendered as "[**?**] demonstrated...," or the inconvenient

```
Kalman \cite{kalman1960new}
demonstrated...
```

rendered as "Kalman [**?**] demonstrated...", one can write

```
\citet{kalman1960new} demonstrated...
```

which renders as "**?**] demonstrated..." and is both easy to write and much easier to read.

A. RSS Hyperlinks

This year, we would like to use the ability of PDF viewers to interpret hyperlinks, specifically to allow each reference in the bibliography to be a link to an online version of the reference. As an example, if you were to cite "Passive Dynamic Walking" [**?**], the entry in the bibtex would read:

```
@article{McGeer01041990,
  author = {McGeer, Tad},
  title = {\href{http://ijr.sagepub.com/content/9/2/62.abstract}{Passive Dynamic Walking}},
  volume = {9},
  number = {2},
  pages = {62-82},
  year = {1990},
  doi = {10.1177/027836499000900206},
  URL = {http://ijr.sagepub.com/content/9/2/62.abstract},
  eprint = {http://ijr.sagepub.com/content/9/2/62.full.pdf+html},
  journal = {The International Journal of Robotics Research}
}
```

and the entry in the compiled PDF would look like:

[1] Tad McGeer. Passive Dynamic Walking. *The International Journal of Robotics Research*, 9(2):62–82, 1990.

where the title of the article is a link that takes you to the article on IJRR's website.

Linking cited articles will not always be possible, especially for older articles. There are also often several versions of papers online: authors are free to decide what to use as the link destination yet we strongly encourage to link to archival or publisher sites (such as IEEE Xplore or Sage Journals). We encourage all authors to use this feature to the extent possible.

IV. Conclusion

The conclusion goes here.

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.