Fixing symbolic plans with reinforcement learning in object-based action spaces

Christopher Thierauf[1] and Matthias Scheutz[2]

***Abstract*— Reinforcement learning techniques are widely used when robots have to learn new tasks but they typically operate on action spaces defined by the joints of the robot. We present a contrasting approach where actions spaces are the trajectories of objects in the environment, requiring robots to discover events such as object changes and behaviors that must occur to accomplish the task. We show that this allows robots to learn faster, to learn semantic representations that can be communicated to humans, and to learn in a manner that does not depend on the robot itself, enabling low-cost policy transfer between different types of robots. Our demonstrations can be replicated using provided source code[1].**

I. Introduction

Reinforcement learning (RL) is generally an effective strategy for allowing robots to learn novel skills, but they generally struggle with long-horizon tasks when compared to symbolic approaches. Conversely, symbolic approaches generally do not obtain the same degree of flexibility and robustness as learned policy-based approaches. Recently, there has been an increased interest in combining the two approaches synergistically, with symbolic reasoning and planning producing a high-level structure and reinforcement learning being used to learn policies that are mapped to symbolic planning operators (e.g., [1], [2]).

In this paper, we propose a method for reconceptualizing the action and observation spaces in RL to better supports this potential synergy. The idea is to allow the RL to *directly manipulate the environment* instead of doing so via the robot's effectors. By constructing action spaces that are based in what must happen to the objects in the environment and not the motor manipulation of the robot itself, we can learn explainable policies that remain symbolically grounded while still producing solutions that symbolic solvers may struggle with. Additionally, these reinforcement learning problems can be automatically constructed from symbolic plans, and learned policies can be applied to different robots.

To describe our approach, we will first provide context with the broader literature, and then describe the proposed method of constructing a reinforcement learning problem that uses object manipulations for the action and observation space, and how such a problem can be autonomously generated using a symbolic planning approach. We then demonstrate in simulation as well as on two real-world robot platforms that our proposed methods indeed provides several key benefits: a substantial improvement in training time, the ability to communicate the results of policies, and low-cost policy transfer between heterogenous robot platforms.

*This work was in part supported by ONR grant N00014-24-1-2024.

[1]Department of Computer Science, Tufts University, Medford, MA 02155 christopher.thierauf@tufts.edu

[2]Department of Computer Science, Tufts University, Medford, MA 02155 matthias.scheutz@tufts.edu

[1]https://github.com/cst0/object_space_solver

II. Background

Utilizing reinforcement learning and symbolic AI approaches to find creative robot behaviors has been explored as the problem of "self-discovery of motion primitives" [3]. More recently, robot creative problem solving has been addressed using experimentation-based planners [4], hierarchical reinforcement learning [5], or action babbling-based approaches [6], [7] (for a summary, see [8]). In particular, creative tool use has been explored as "the MacGyver Problem" [9], with related solutions incorporating the discovery of potential tool use [10], [11], [12], [13], [14].

Our proposed approach is closer to [1], [2], where reinforcement learning is used as a mechanism to enable exploration beyond the limitations of the pre-configured symbolic framework. Here, an agent in a simulation environment is shown to be capable of performing a long-horizon task using the structured planning and reasoning of a symbolic approach. However, when the agent encounters a novelty preventing successful completion of the existing plan, reinforcement learning mechanisms provide a fall-back strategy of exploration until a state is achieved from which the agent may continue planning.

Thus, our work is related to repairing symbolic plans with RL. As relevant work in this space, consider [15], where the authors demonstrate statistical-based approaches provide an effective strategy for these more "open-world" cases, in which symbolic knowledge must be expanded to accomodate novelty. Within RL specifically, this set of cases relates in part to cases where the domain model itself will be incorrect. RL as a strategy to repair these cases has been explored in [16] [17] and [18], each independently finding that models of the world can be adapted to address cases outside the initial domain constructed by the designer. These approaches suggest that RL has capacity for symbolic plan repair, which is demonstrated in simulated environments in [19] (similar to [1]). These works additionally show these hybrid approaches can outperform either approach independently.

In [20], the authors identify the same tradeoff between symbolic and reinforcement learning approaches that we now explore. In their work, symbolic planning is employed for long-term planning while RL is used to provide implementations of behavior. Similarly, we have seen symbolic planning

used to provide a form of scaffolding for reinforcement learning strategies (a technique we will also adopt) [21], [22].

Where we differ from all of the above approaches which merge symbolic and RL approaches, however, is in our fundamental construction of the problem action and observation spaces, and the method in which we integrate these for robot behavior. In short, where it is common to see reinforcement learning as subsymbolic, we instead remain symbolic at both the decision-making and execution levels, making use of RL to provide new solutions to the symbolic solver. We are therefore able to grow the symbolic agent's skillset to include new behaviors which can be used in planning, retaining the advantages of symbolic approaches without remaining constrained by them.

In demonstration-based approaches [23], [24], [25], the authors demonstrate how training can be accelerated by presenting examples of successful task completion. It is easy to imagine presenting examples of successful environment manipulations, and in the same way, we expect to remain fully compatible with this approach. The same cannot be said of transfer-based approaches [26], however: we will specifically demonstrate how transfer between heterogenous robot platforms is made fundamentally different with our approach (with unique pros and cons to be discussed). We expect heuristics-based approaches such as [27] or optimzation approaches [28] to remain broadly compatible for reasons we will later discuss in detail, but providing comprehensive analysis of the full spectrum of RL-acceleration techniques is well beyond the scope of this paper and so we do not make this claim.

III. Proposed Method

At a high-level, our approach is to first identify all possible degrees of freedom in the environment. Each of these degrees of freedom (for example, the full Cartesian motion of a block on a table, or the one linear motion of a sliding drawer) is provided to the RL agent directly. By allowing the agent to learn the task directly, rather than needing to learn the details of environment and robot interactions before approaching the task, training time improves. In doing so, we empower an agent to understand the sequence of object interactions that must occur to solve the actual task, and leave motion planning as an already-solved problem that does not need to be addressed by the agent here. Further, by using symbolic planning as a framework to construct a reward function, the produced policy remains useful for a symbolic plan.

Thus, in order, we:

1) Construct a simulation environment in which an RL policy can train, but do not place a robot in this environment;
2) Provide the agent with the ability to manipulate positions, velocities, or forces applied to objects in the environment directly;
3) Provide the ability to manipulate each joint in the environment (e.g., hinges) to the agent's action space;
4) Train on this space to produce a policy which describes how the environment should be manipulated;
5) Convert this understanding to a series of actions that can be performed on the robot, using an object-position to robot action mapping.

We start with a description of requirements and system inputs and then construct the "Object–Centric Planning Problem", followed by demonstrations.

A. Preliminaries

Our approach aims to construct a reinforcement learning problem which can integrate into a symbolic framework by creating an "object-first" approach to RL. We thus must first define an object o as some element of the robot's scene which can be impacted by forces in the environment (perhaps it can be slid, lifted, etc); the set of all possible objects in the environment will be O. These degrees of freedom will include linear or rotational velocities and accelerations, but may be constrained (for example, a block can perhaps be lifted to any orientation while a drawer is constrained by its hinges). We can then define an *object mobility function* $\zeta(o)$ which, given an object, returns the n-tuple describing the degrees of freedom of that particular object. It is this n-tuple which can produce some action a describing which degrees of freedom will be impacted, and to what extent (e.g., "0.1 meters/second in the x direction" or "0.2 radians/second in roll and 0.3 meters/second in z"). Similarly, we can then define the *object state transition function* $\sigma(o, a)$ which, given an object, returns the 18-tuple describing the full pose of the object after applying the action[2]. In simulation environments this is straightforward to obtain, in real-world environments this is more challenging, requiring, for example, vision processing or prior knowledge.

From this, we now have the tools necessary to construct our reinforcement learning problem as an MDP $M = \langle S, A, \sigma, r \rangle$ where S is the set of states (produced by applying the object state transition function ζ to all objects); A is the set of actions (each of which will meet the constraints provided by the state transition function σ); σ is used as the state transition function; and r is the reward function where $r(s)$ for some $s \in S$ provides a reward for task completion.

For integration into a symbolic framework, we will focus on the construction of the reward function r. From previous literature we know that this reward function can be constructed from symbolic representations (as we will later discuss). While existing approaches generally focus on using these symbolic representations as a form of scaffolding to better constrain the RL agent for efficiency, we observe that a symbolic representation of the reward function also enables the integration of a reinforcement learning policy into a broader symbolic planning problem.

For example, the knowledge that "the block is on the table" can be used to define a sparse binary reward:

$$r = \begin{cases} 1 & \text{if } on(Table, Block) \\ 0 & \text{otherwise} \end{cases}$$

[2]Composed of x, y, z, roll, pitch, and yaw positions; plus the x, y, z, roll, pitch, and yaw velocities, plus the accelerations.

It is also possible to imagine more complex reward functions using first-order logic to provide increasingly specific constraints: the block must be on the table, *and* the drawer it comes from must be closed, *and* no other objects must have been knocked over, etc.

Finally, we define some mapping between object manipulations and actions which can be employed by the robot. We state that for every every *object mobility function* $\zeta(o)$, there exists a corresponding *object manipulation function* $\eta(o)$ which the robot can employ to actually produce the motion described by ζ. If it should be the case that η cannot provide some mobility described by the appropriate ζ, then ζ must be reduced to correctly reflect the capabilities of the platform.

B. Modules enabling approach

In operation, several key components work in conjunction to produce our results (Figure 1). First, a symbolic agent (Figure 1, highlighted in green) is assumed to be functioning as expected: it can be provided with goals which produce an action planning and execution problem. As extensive prior work has demonstrated, these symbolic domains can be solved symbolically, producing a series of symbolic actions (such as "go to pose (x, y)"). These symbolic actions can then be executed (e.g., a kinematic solver can construct a series of motor movements which accomplishes the symbolic goal).

Inevitably, however, there will come a point where the symbolic domain is missing some key information to solve the problem, e.g, an operator to bridge the gap between two symbolic states. In these cases, we can use RL to learn a policy that effects that mapping (e.g., using the techniques described in [1], [2]).

C. Approach in Practice

Here, we will utilize the stable-baselines implementation of PPO to follow state-of-the-art standards, as our approach does not introduce any new RL algorithms. It is thus also reasonable to assume that other RL approaches should work.

We also a custom PyBullet [29] environment for each environment in which the agent must train. Again, we have not introduced any novel requirement to the RL problem, and so a variety of environments could be used in the place of PyBullet: MuJoCo [30], for example, also provides the ability to manipulate the position and velocity of objects in a scene.

On the symbolic side of our implementation, we have kept the very broad requirement of representation within a first-order logic. This enables substantially streamlined implementation into other architectures: note that the widely-used "Planning Domain Definition Language" (PDDL) meets this definition, although we will not extensively explore an integration here. These behaviors, being symbolic, have a symbolic representation which can be executed. More specifically, we make use of the MoveIt [31] framework to implement an "open drawer" and "grab object" behavior (though importantly, for the sake of our experimentation, the agent will lack knowledge of how these actions may be used in a broader planning problem).

IV. Analysis of benefits and drawbacks of the proposed approach

In extensive experimentation with our proposed system, we have observed both key advantages as well as potential costs which we discuss next.

A. Training time improves

Reformulating the reinforcement learning problem to focus on the objects in a space offers substantial reduction in training time. This is conceptually reasonable: practitioners in this space will anecdotally agree that much of the time spent training reinforcement learning agents on challenging tasks comes from the agent needing to learn first how it can manipulate objects in its environment before it can then learn how to optimally solve the task. This is particularly true for the sparse binary rewards we employ. By removing the need to learn environment interactions, we offer the agent the potential to learn substantially more quickly.

Additionally, recall that we make use of RL not to produce the specific motor control policy, but to discover what must occur in the environment. This provides another venue in which training time will be accelerated: we no longer require the most efficient motor control policy, because the generated converted sequence will remain the same regardless of any minor optimizations. This is in stark contrast to policies which must also learn to make their motor controls more efficient when attempting to complete the task. The consequence of these is a control policy that is substantially simpler to learn, and as a consequence, it is reasonable to expect it to generally be faster.

B. Policies remain within symbolic frameworks

Using symbolic logics to construct a reward function allows us to ensure that the resulting reinforcement learning policy remains symbolically grounded. This, in turn, offers a number of substantial benefits. First, symbolic plans generally outperform reinforcement learning plans when attempting to provide solutions to long-horizion tasks (at the cost of their brittleness, which we aim to address here). By prioritizing the use of RL to make symbolic plans more robust to deviations from expectation, we produce a hybrid approach that allows the reinforcement learning agent to make use of the strengths of symbolic approaches.

Second, symbolic plans are generally more straightforward to communicate through natural language. While there has been progress in explainability of RL policies that should not be discounted, the goal of converting policy to language involves converting RL to symbols. By already providing the approprate symbols, rather than needing to learn or infer them, language descriptions can be more straightforward to produce and more accurate.

Finally, it is more straightforward to reuse policies if they remain symbolically grounded. Consider a case where an agent learns that it can open a box, thus solving its reinforcement learning task. If the only framework provided to this task is its reward function, the agent will only ever produce a context-specific policy that solves the given

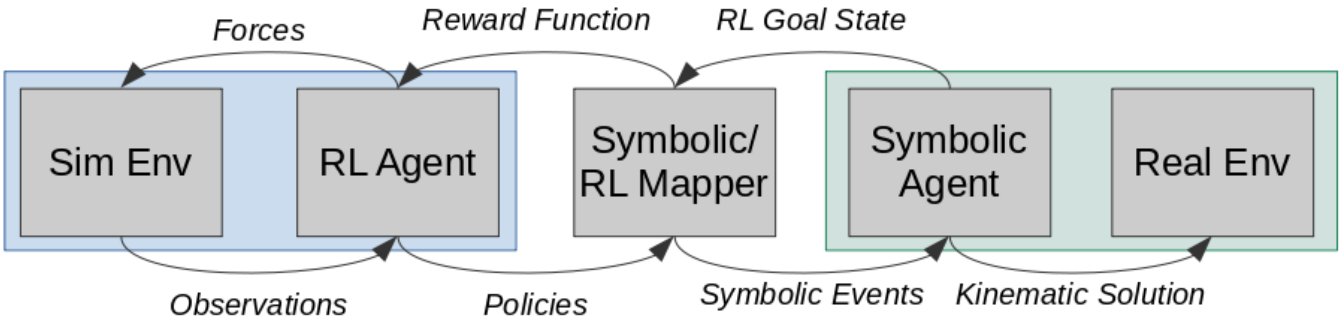


Fig. 1: High-level representation of the interaction between RL and Symbolic systems. If the symbolic system (highlighted in green) is unable to resolve a plan, it can construct a goal state which the Symbolic/RL mapper converts to a reward function. The RL Agent (highlighted in blue) finds a policy describing what must occur in the environment to solve the problem. The policy can then be mapped into a series of symbolic steps, which can be passed on to a kinematic solver for execution.

task. There may come a new problem in which the agent would benefit from the knowledge that it has learned: for example, grabbing an object within the box. An agent which exclusively makes use of RL as its strategy would be forced to relearn the knowledge that the box can be opened. While certainly this is not prohibitive to the agent (it will eventually relearn this and succeed), our hybrid-symbolic approach does not incur this cost. Instead, the fact that an open-box state can be achieved is known and can be reasoned upon: in this case, knowing a policy to open the box can be employed to obtain a needed state to grab the block.

C. Enables reduced-cost domain transfer

The nature of our approach making use of symbolic execution strategies after a policy conversion step is that the policy remains unconstrained to a specific platform. Any platform which has the same actions implemented is capable of executing these converted policies. Additionally, it is possible to envision cases where the agent will not have the same precise method of constructing an effect, but will have a method of producing the desired outcome anyway. For example, consider one agent which can lift and place a block versus another which can only push it. Because actions are defined not by the manner in which they are performed, but are instead defined by the way they impact the environment, these are two equally viable actions for a specific subset of problems.

Of course, this hinges on the assumption that a set of such actions exists. As we will next discuss, this is not a zero-cost assumption. It remains difficult to quantify the degree to which this will (or, case depending, may not) reduce the cost of transfer. For this reason we claim ‘reduced cost’ transfer rather than ‘zero cost’.

D. Requires both symbolic and RL domains

In a purely symbolic implementation of an agent, it is not necessary to provide a physics domain. Similarly, it is not necessary to provide a full symbolically represented domain to a RL domain. In this sense, we have introduced an additional cost of requiring both. Whether this additional cost is outweighed by the advantages will be application-specific. Conversely, however, it is worth noting that any approach which attempts to achieve resilient symbolic planning will also require implementation, and so this cost can be considered minimal.

E. Requires underlying representations and implementations

Our approach requires underlying implentations of behaviors: In our implementation these behaviors are based in more traditional kinematic motion planning strategies, although they certainly could be implemented directly from RL strategies. While this is a limitation, we argue that it is not a prohibitive one: any deployed system would be required to have such an implementation ready for use. Therefore, when taken in combination with the key advantages of this approach, we view it as a form of “computational creativity”: when symbolic plans fail, this approach offers a method of finding new behaviors which may then become a part of a plan solution.

V. Demonstrations

To demonstrate the value to real-world robot platforms, we take a simulation environment of a real-world environment in which an agent must learn to open a drawer and remove a block. The Kinova Gen 3 (a 7-DoF arm) with a Robotiq 2f-85 (a 2-fingered gripper) is placed on a table and must complete the same task. The same environment can be modified by placing the Fetch Robotics “Fetch” (a mobile manipulator) in the same environment with the same task.

We modify the environment slightly for the aid of the robots. The handle on the filing cabinet has been replaced with one which is larger and slightly spring-loaded for compliance, to better acommodate the precision limitations of the platforms. This is an acceptable simplification because we are not interested in exploring problems of hardware precision. Similarly, the block has been prevented from sliding around when within the cabinet using a piece of foam: this is also an acceptable simplification because we are only interested in exploring cases where there is a reasonable mapping between real world and simulated environments.

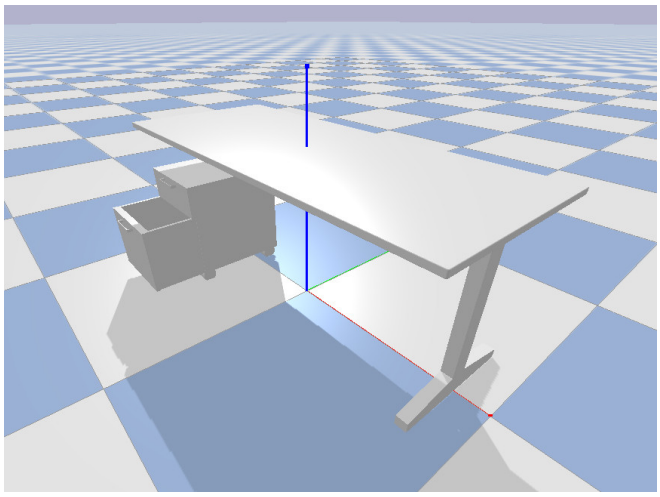


Fig. 2: PyBullet recreation of the real-world environment for the agent to explore. The filing cabinet drawers can both be opened, with a block in the top drawer that can be manipulated. The table remains static, per η, because the robot would be unable to change its position.

A. One-Shot sim–to–real within a symbolic framework.

The object manipulation function η provides the agent with the ability to perform one-shot sim–to–real. This is possible because this function is rooted in the assumption that pre-existing mappings between possible object movements and robot behaviors have been constructed. We use the same policy as trained in simulation previously, and so the agent is unaware of the critical knowledge that the drawer must be opened before the block can be accessed.

For our demonstration (Figure 3), the Kinova Gen 3 is already provided with a mapping η, primarily implemented using the ROS MoveIt framework. With this framework, the sequence of observed events – the drawer opening, moving the block, placing it on the table – can be converted to a series of robot behaviors: the arm catching and sweeping the drawer open, grabbing and lifting the block, releasing the block. In this way, despite missing the knowledge of cause and effect in a symbolic space, we show that using RL as a creative solving approach is capable of resolving the gaps in the symbolic plan.

B. One-Shot policy transfer with heterogenous agents.

In the same way we enable sim–to–real policy transfer, we enable policy transfer between heterogenous agents. We assume Robot A has some object manipulation function η^A and Robot B has some object manipulation function η^B, and that there exists a one to one mapping between η^A and η^B. These assumptions are reasonable because of the requirements of our approach, and they permit us to execute a policy on robot A in the same way we would execute a policy for robot B.

To demonstrate this, we take the existing environment and replace the Kinova robot with the Fetch mobile manipulator platform (Figure 4). Both these platforms have their own independent implementations of the central behaviors. However, the mapping between those behaviors is one-to-one: thus, any behavior that we deploy onto the Kinova, we see is deployable onto the Fetch.

C. Semantic policy descriptions.

Our approach also enables some level of communication about discovered policies. We could expand our MDP M to be of the form $M = \langle S, A, \delta, r, E \rangle$, where we state that each $e \in E$ is a textual explanation for all $\{r(s) : s \in S\}$. However, this would quickly become challenging to construct and generalize. Instead, thanks to our integration in a broader symbolic architecture, a more comprehensive solution is available. Recall from our previous formalism that we associate each problem formulation with effects and preconditions in the form of a first-order logic. A number of approaches exist to convert these symbolic descriptions to text which can be communicated. As a non-comprehensive review, consider "Semiotic Schemas" [32], or approaches more rooted in training data like [33], [34], or the energy-function model based [35].

In our demonstration, however, we take advantage of the recent developments in using large-language models (LLMs) to enable textual descriptions of behavior (which again has been previously explored, consider [36], [37]). To produce our demonstration, we make use of the Mixtral open-source large language model [38]. We provide the LLM with the prompt:

"State to action conversions:
holding(mug) –>i will hold the mug
on(desk,paper) –>i will put the paper on the desk"

Observe here that we have not provided specific information about the task it will next be queried about, to help ensure generalizability. With the training completed and converted into a series of observed effects, the LLM can now be queried by providing an incomplete example of the preceding pattern:

"drawer(open) –>"

The LLM, finding the most probable next set of characters, attempts to continue the pattern:

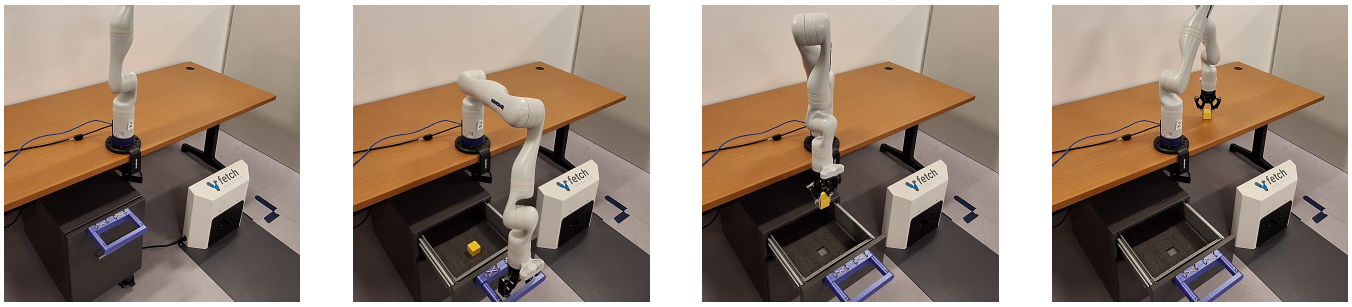
"I will open the drawer"

By repeating this for each action, a series of text responses are produced, enabling communication of the generated policy.

VI. Future Works

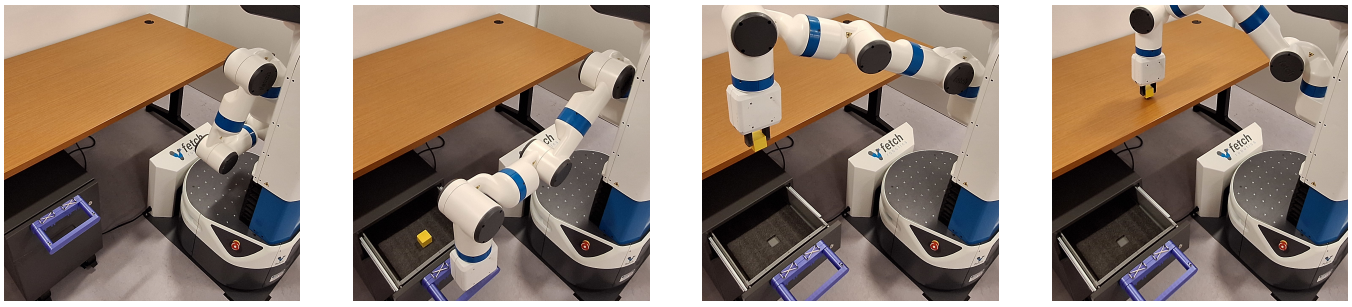
As previously discussed, this approach to RL carries valuable benefits in the research direction of creating robots which are better able to address open-world scenarios. However, there remains a variety of potential future directions woth exploring.

First, we have chosen to employ classical kinematic solvers to execute our symbolic behaviors. This choice was motivated by the goal of integration into a broader symbolic architecture where symbolic behavior implementations can already be assumed, but an avenue of research worth



(a) (b) (c) (d)

Fig. 3: A Kinova solving the multi-step task. From the start (a), it (b) opens the drawer using a symbolic open-drawer action, then (c) grabs and (d) places the block.



(a) (b) (c) (d)

Fig. 4: Because the same symbolic actions are available to the Fetch (a), it can also (b) open the drawer and then (c) grab and (d) place the block.

considering would be to construct a hierarchical problem in which the task is addressed at a high-level using an object-based approach, and then at an execution level using a more traditional reinforcement learning approach.

Second, we suffer from the same limitation that many other reinforcement learning strategies do: it is necessary for the agent to be provided with a safe environment in which to explore potential policies. In practical terms, this requires an agent to be provided with some simulation environment in which it can train, before a policy can then be deployed. We remain subject to this limitation. For agents which attempt to use reinforcement learning in novel environments, it will be necessary for practitioners to find strategies which loosen this requirement.

VII. Conclusion

We have presented a technique which allows a robot agent to "unstick" broken symbolic plans, by employing reinforcement learning as creative problem solver to quickly find how the environment must be manipulated, to the benefit of a symbolic agent (the speed-up was accomplished by learning in object spaces as opposed to the robot's joint space). We have shown that this offers both practical and conceptual advantages. Although there remains future work to be completed (most notably, in the autonomous construction of these spaces), this hybrid approach which takes advantages of the respective strengths of RL and symbolic approaches provides a promising direction for quickly finding generalized solutions to task that are fully explainable and can be deployed to robots with different actuators without the need for retraining.

References

[1] S. Goel, Y. Shukla, V. Sarathy, M. Scheutz, and J. Sinapov, "Rapid-learn: A framework for learning to recover for handling novelties in open-world environments." in *2022 IEEE International Conference on Development and Learning (ICDL)*. IEEE, 2022, pp. 15–22.

[2] P. Lorang, S. Goel, P. Zips, J. Sinapov, and M. Scheutz, "Speeding-up continual learning through information gains in novel experiences," in *4th Planning and Reinforcement Learning (PRL) Workshop at IJCAI-2022*, 2022.

[3] E. Ugur, E. Şahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 3260–3267.

[4] V. Sarathy and M. Scheutz, "Biplex: Creative problem-solving by planning for experimentation," in *International Conference on Computational Creativity*, 2022.

[5] T. R. Colin, T. Belpaeme, A. Cangelosi, and N. Hemion, "Hierarchical reinforcement learning as creative problem solving," *Robotics and Autonomous Systems*, vol. 86, pp. 196–206, 2016.

[6] E. Gizzi, M. G. Castro, and J. Sinapov, "Creative problem solving by robots using action primitive discovery," in *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. IEEE, 2019, pp. 228–233.

[7] E. Gizzi, A. Hassan, W. W. Lin, K. Rhea, and J. Sinapov, "Toward creative problem solving agents: Action discovery through behavior babbling," in *2021 IEEE International Conference on Development and Learning (ICDL)*. IEEE, 2021, pp. 1–7.

[8] E. Gizzi, L. Nair, S. Chernova, and J. Sinapov, "Creative problem solving in artificially intelligent agents: A survey and framework," *Journal of Artificial Intelligence Research*, vol. 75, pp. 857–911, 2022.

[9] V. Sarathy and M. Scheutz, "The macgyver test-a framework for evaluating machine resourcefulness and creative problem solving," *arXiv preprint arXiv:1704.08350*, 2017.

[10] T. Fitzgerald, A. Goel, and A. Thomaz, "Modeling and learning constraints for creative tool use," *Frontiers in Robotics and AI*, vol. 8, p. 674292, 2021.

[11] S. Tuli, R. Bansal, R. Paul *et al.*, "Tooltango: Common sense generalization in predicting sequential tool interactions for robot plan synthesis," *Journal of Artificial Intelligence Research*, vol. 75, pp. 1595–1631, 2022.

[12] L. Nair and S. Chernova, "Feature guided search for creative problem solving through tool construction," *Frontiers in Robotics and AI*, vol. 7, p. 592382, 2020.

[13] L. Nair, J. Balloch, and S. Chernova, "Tool macgyvering: Tool construction using geometric reasoning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5837–5843.

[14] L. Nair, N. Shrivatsav, and S. Chernova, "Tool macgyvering: A novel framework for combining tool substitution and construction," *arXiv preprint arXiv:2008.10638*, 2020.

[15] A. Bendale and T. Boult, "Towards open world recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1893–1902.

[16] J. H. A. Ng and R. P. Petrick, "Incremental learning of planning actions in model-based reinforcement learning." in *IJCAI*, 2019, pp. 3195–3201.

[17] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised skill discovery," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[18] W. Piotrowski, R. Stern, Y. Sher, J. Le, M. Klenk, J. deKleer, and S. Mohan, "Learning to operate in open worlds by adapting planning models," *arXiv preprint arXiv:2303.14272*, 2023.

[19] P. Parashar, B. Sheneman, and A. K. Goel, "Adaptive agents in minecraft: A hybrid paradigm for combining domain knowledge with reinforcement learning," in *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Visionary Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*. Springer, 2017, pp. 86–100.

[20] D. Gordon, D. Fox, and A. Farhadi, "What should i do now? marrying reinforcement learning and symbolic planning," *arXiv preprint arXiv:1901.01492*, 2019.

[21] L. Illanes, X. Yan, R. T. Icarte, and S. A. McIlraith, "Symbolic plans as high-level instructions for reinforcement learning," in *Proceedings of the international conference on automated planning and scheduling*, vol. 30, 2020, pp. 540–550.

[22] F. Yang, D. Lyu, B. Liu, and S. Gustafson, "Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making," *arXiv preprint arXiv:1804.07779*, 2018.

[23] B. Price and C. Boutilier, "Accelerating reinforcement learning through implicit imitation," *Journal of Artificial Intelligence Research*, vol. 19, pp. 569–629, 2003.

[24] X. Zhang and H. Ma, "Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations," *arXiv preprint arXiv:1801.10459*, 2018.

[25] M. Ahmadi, M. E. Taylor, and P. Stone, "Ifsa: Incremental feature-set augmentation for reinforcement learning tasks," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 2007, pp. 1–8.

[26] L. A. Celiberto Jr, J. P. Matsuura, R. L. De Màntaras, and R. A. Bianchi, "Using transfer learning to speed-up reinforcement learning: a cased-based approach," in *2010 latin american robotics symposium and intelligent robotics meeting*. IEEE, 2010, pp. 55–60.

[27] R. A. Bianchi, C. H. Ribeiro, and A. H. Costa, "Heuristically accelerated q–learning: a new approach to speed up reinforcement learning," in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 245–254.

[28] C. Kamanchi, R. B. Diddigi, and S. Bhatnagar, "Successive over-relaxation q-learning," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 55–60, 2019.

[29] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[30] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[31] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[32] D. Roy, "Semiotic schemas: A framework for grounding language in action and perception," *Artificial Intelligence*, vol. 167, no. 1-2, pp. 170–205, 2005.

[33] J. Thomason, A. Padmakumar, J. Sinapov, N. Walker, Y. Jiang, H. Yedidsion, J. Hart, P. Stone, and R. Mooney, "Jointly improving parsing and perception for natural language commands through human-robot dialog," *Journal of Artificial Intelligence Research*, vol. 67, pp. 327–374, 2020.

[34] A. Broad, J. Arkin, N. Ratliff, T. Howard, and B. Argall, "Real-time natural language corrections for assistive robotic manipulators," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 684–698, 2017.

[35] D. K. Misra, J. Sung, K. Lee, and A. Saxena, "Tell me dave: Context-sensitive grounding of natural language to manipulation instructions," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 281–300, 2016.

[36] A. Koubaa, "Rosgpt: Next-generation human-robot interaction with chatgpt and ros," 2023.

[37] S. Wang, Z. Zhou, B. Li, Z. Li, and Z. Kan, "Multi-modal interaction with transformers: bridging robots and human with natural language," *Robotica*, vol. 42, no. 2, pp. 415–434, 2024.

[38] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.