

¹ HierarchyCraft: A Benchmark Builder for Hierarchical Reasoning

³ **Mathis Fédérico**  ^{1,2}, **Shang Wang**  ¹, **Yuxuan Li**  ^{1,3}, and **Matthew E. Taylor**  ^{1,3}

⁵ 1 Department of Computing Science, University of Alberta, Canada 2 CentraleSupélec, University of Paris-Saclay, France 3 Alberta Machine Intelligence Institute (Amii), Canada

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Tristan Miller](#)  

Reviewers:

- [@lwu9](#)
- [@Christopher-Henry-UM](#)
- [@inpefess](#)

Submitted: 14 January 2024

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

⁷ Summary

⁸ Hierarchical reasoning poses a fundamental challenge in the field of artificial intelligence ([Botvinick & Weinstein, 2014](#)). Existing methods may struggle when confronted with ⁹ hierarchical tasks ([Bacon et al., 2017; Heess et al., 2015; Nachum et al., 2018](#)), yet despite ¹⁰ the importance of understanding how the structure of an underlying hierarchy affects task ¹¹ difficulty, there is a lack of suitable environments or benchmarks to facilitate this exploration. ¹²

¹³ We introduce **HierarchyCraft**, a software package that allows researchers to create custom ¹⁴ environments based on their hierarchical structures, enabling the study of hierarchical reasoning.

¹⁵ To isolate hierarchical behavior and ensure compatibility with classical planning algorithms, we ¹⁶ exclude unstructured data like images and text, avoiding the complexity of feature extraction, ¹⁷ and allowing comparisons between classical planning and reinforcement learning.

¹⁸ HierarchyCraft simplifies the creation of diverse hierarchical environments from a list of [a single building block](#) as showcased by the [set of pre-defined environments](#).

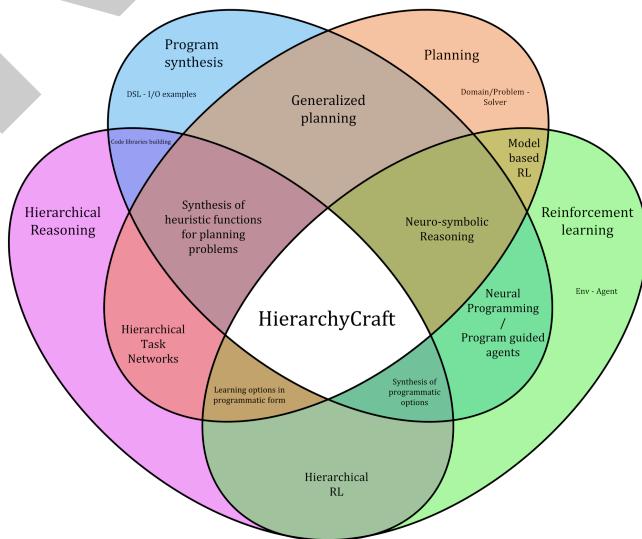


Figure 1: HierarchyCraft is at the intersection of reinforcement learning, planning, hierarchical reasoning and program synthesis.

20 Statement of need

21 HierarchyCraft is designed as a user-friendly Python library for constructing environments
 22 tailored to the study of hierarchical reasoning in the contexts of reinforcement learning, classical
 23 planning, and program synthesis as displayed in [Figure 1](#).

24 Analysis and quantification of the impacts of diverse hierarchical structures on learning agents is
 25 essential for advancing hierarchical reasoning. However, current hierarchical benchmarks often
 26 limit themselves to a single hierarchical structure per benchmark, and present challenges not
 27 only due to this inherent hierarchical structure but also because of the necessary representation
 28 learning to interpret the inputs.

29 HierarchyCraft is a benchmark builder (not a benchmark) designed to study how different
 30 hierarchical structures impact the performance of classical planners and reinforcement learning
 31 algorithms. It includes several [examples](#), detailed in the [documentation](#), that can serve as
 32 initial benchmarks. These examples include basic parametrised hierarchical structures (Random,
 33 Recursive, Tower) and fixed structures imitating other environments (MineHCraft for Minecraft
 34 tasks, MiniHCraft for Minigrid tasks). Researchers are encouraged to create custom hierarchical
 35 environments to explore structures of their choice.

36 We argue that arbitrary hierarchical complexity can emerge from simple rules. To the best of
 37 our knowledge, no general frameworks currently exist for constructing environments dedicated
 38 to studying the hierarchical structure itself. We next highlight five related benchmarks,
 39 underscoring the necessity for the development of a tool like HierarchyCraft.

40 GridWorld

41 GridWorld, a general class of 2D grid-based environments, is frequently utilised in hierarchical
 42 reinforcement learning research, notably within the options framework ([Sutton et al., 1999](#)).

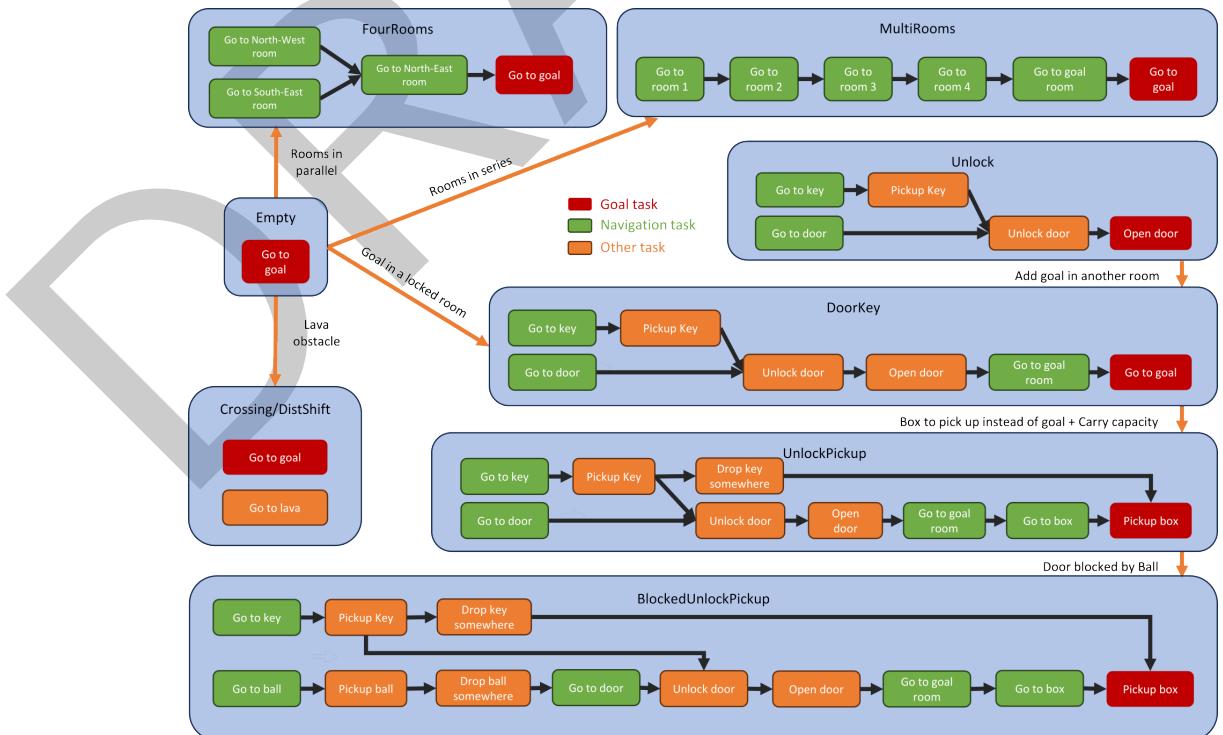


Figure 2: Example of Minigrid environments hierarchical structures and their relationships. There are only a few possible subtasks and most of them are navigation tasks (in green).

43 Minigrid (Chevalier-Boisvert et al., 2023) is a user-friendly Python library that not only
 44 implements a GridWorld engine but also expands its capabilities. This allows researchers to
 45 create more intricate scenarios by introducing additional rooms, objectives, or obstacles.

46 As illustrated in Figure 2, GridWorld environments only exhibit **limited and similar hierarchical**
 47 **structures** that primarily focus on navigation tasks, making these hierarchies mostly sequential.

48 Minecraft

49 A good example of a hierarchical task is the collection of diamonds in the popular video game
 50 Minecraft, as showcased in the MineRL competition (Guss et al., 2021), where hierarchical
 51 reinforcement learning agents have dominated the leaderboard (Milani et al., 2020).

52 Due to sparse rewards, the difficulty of exploration, and long time horizons in this procedurally
 53 generated sandbox environment, DreamerV3 (Hafner et al., 2023) recently became the first
 54 algorithm to successfully collect diamonds in Minecraft without prior training or knowledge.
 55 Unfortunately, DreamerV3 required training on an Nvidia V100 GPU for 17 days, gathering
 56 roughly 100 million environmental steps. Such **substantial computational resources are**
 57 **unavailable to many researchers**, impeding the overall progress of research on hierarchical
 58 reasoning.

59 Moreover, although Minecraft has an **undeniably complex hierarchical structure**, this **underlying**
 60 **hierarchical structure is fixed** and cannot be modified without modding the game, a complex
 61 task for researchers.

62 Crafter

63 Crafter (Hafner, 2022) presents a lightweight grid-based 2D environment, with game mechanics
 64 akin to Minecraft and poses similar challenges (exploration, representation learning, rewards
 65 sparsity and long-term reasoning) at much lower compute cost.

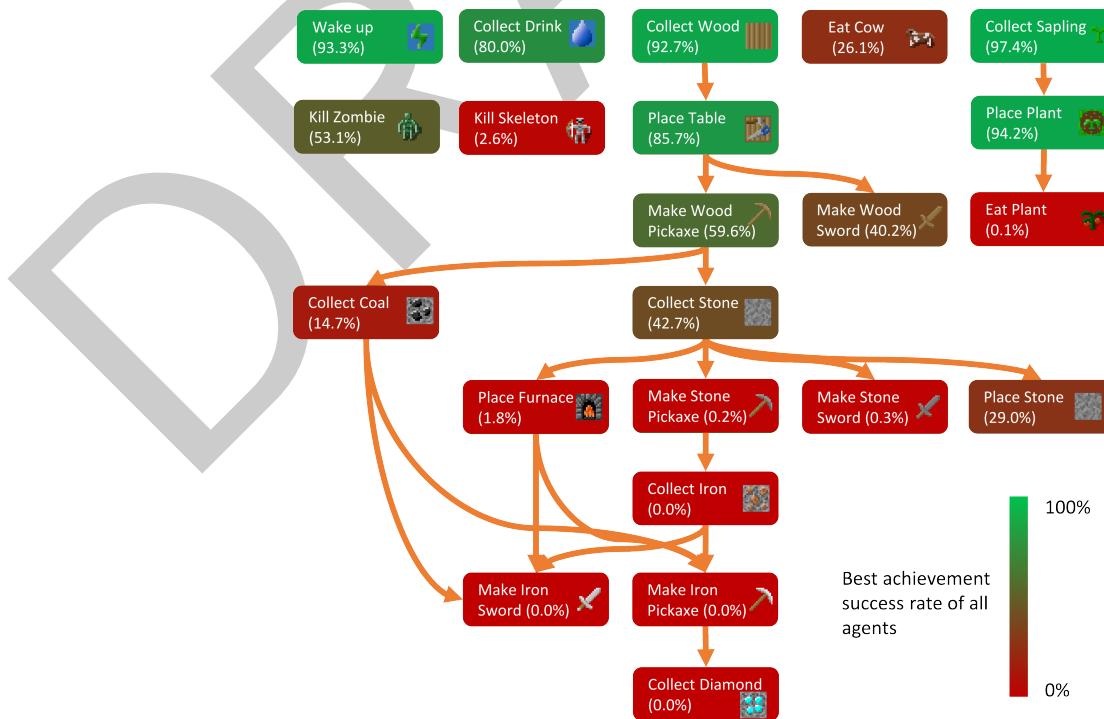


Figure 3: Hierarchical structure of the Crafter environment as presented by the authors of Crafting with their success rates. Inspired by Figure 4 of (Hafner, 2022).

66 Although Crafter offers 22 different tasks displayed in [Figure 3](#), the **underlying hierarchical**
67 **structure is fixed**, restricting how researchers can investigate the impacts of changes to the
68 hierarchical structure.

69 Moreover, the tasks considered by the authors do not encompass various navigation-related
70 subtasks (such as finding water, locating a cow, waiting for a plant to grow, or returning to a
71 table), nor do they include certain optional but beneficial subtasks (for example, using swords
72 or the skill of dodging arrows can make the task of defeating skeletons easier).

73 These omissions results in abrupt drops in success rates within the hierarchy, rather than a more
74 gradual progression in difficulty. This highlights that the hierarchy presented by the authors
75 is incomplete, as it fails to capture the full range of subtasks in Crafter and the necessary or
76 helpful interactions between them for successfully completing higher-level tasks.

77 **Arcade Learning Environment (Atari)**

78 The arcade learning environment ([Bellemare et al., 2015](#)) stands as a standard benchmark
79 in reinforcement learning, encompassing over 55 Atari games. However, **only a few of these**
80 **games, such as Montezuma's Revenge and Pitfall, necessitate hierarchical reasoning.**

81 Each Atari game has a **fixed hierarchy that cannot be modified** and agents demand substantial
82 computational resources to extract relevant features from pixels or memory, significantly slowing
83 down experiments.

84 **PDDLGym**

85 PDDLGym ([Silver & Chitnis, 2020](#)) is a Python library that automatically constructs Gym
86 environments from Planning Domain Definition Language (PDDL) domains and problems.
87 PDDL ([Ghallab et al., 1998](#)) functions as a problem specification language, facilitating the
88 comparison of different symbolic planners.

89 However, constructing PDDL domains and problems with a hierarchical structure is challenging
90 and time-consuming, especially for researchers unfamiliar with PDDL-like languages.

91 Additionally, PDDLGym is **compatible only with PDDL1** and does not support numeric-
92 fluents introduced in PDDL 2.1 that are required to represent quantities in the inventories of
93 HierarchyCraft environments.

94 **Abstraction and Reasoning Corpus (ARC)**

95 The Abstraction and Reasoning Corpus (ARC) ([Chollet, 2019](#)), is both hierarchical and diverse,
96 as each task exhibits its own implicit hierarchical structure. However, these hierarchical
97 structures are not explicitly provided within the dataset, such as with shorter programs for
98 each task. Making these hierarchical structures explicit would also contribute significantly to
99 the development of hierarchical reasoning like what HierarchyCraft is trying to achieve.

100 Much like Gridworld, ARC tasks require feature extraction from 2D grids, leveraging priors
101 related to their spatial nature, which bias the nature of the tasks on that specific data structure.

102 Additionally ARC tasks do not emphasise long-term reasoning as they are relatively short
103 compared to tasks within other benchmarks like Minecraft or even Gridworld; this makes
104 underlying hierarchical structures shallow for each task and more wide than deep for the whole
105 corpus.

106 Partitioning those underlying hierarchical structures and classifying them relatively to the
107 difficulty of finding a solution, independently of the solution's nature, is at the core of
108 HierarchyCraft's motivation.

¹⁰⁹ **Design goals**

¹¹⁰ HierarchyCraft aims to be a fruitful tool for investigating hierarchical reasoning, focussing on
¹¹¹ achieving the following four design goals.

¹¹² **1. Hierarchical by design**

¹¹³ The action space of HierarchyCraft environments consists of subtasks, referred to as
¹¹⁴ *Transformations*, as opposed to detailed movements and controls. But each *Transformations*
¹¹⁵ has specific requirements to be valid (e.g. have enough of an item, be in the right place), and
¹¹⁶ these requirements may necessitate the execution of other *Transformations* first, inherently
¹¹⁷ creating a hierarchical structure in HierarchyCraft environments. This concept is visually
¹¹⁸ represented by the *Requirements Graph* depicting the hierarchical relationships within each
¹¹⁹ HierarchyCraft environment. The *Requirements Graph* is directly constructed from the list of
¹²⁰ *Transformations* composing the environment, as illustrated in [Figure 4](#). Requirements Graphs
¹²¹ should be viewed as a generalisation of previously observed graphical representations from
¹²² related works, including [Figure 3](#) and [Figure 2](#).

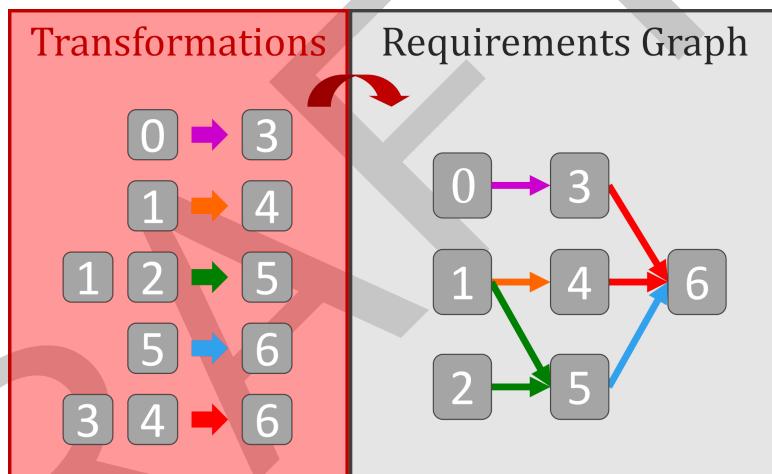


Figure 4: How subtasks build a hierarchical structure.

¹²³ **2. Easy to use and customise**

¹²⁴ HierarchyCraft is a versatile framework enabling the creation of diverse hierarchical environments.
¹²⁵ The library is designed to be simple and flexible, allowing researchers to define their own
¹²⁶ hierarchical environments with detailed guidance provided in the documentation. To showcase
¹²⁷ the range of environments possible within HierarchyCraft, multiple examples are provided.

¹²⁸ **3. No feature extraction needed**

¹²⁹ In contrast to benchmarks that yield grids, pixel arrays, text, or sound, HierarchyCraft directly
¹³⁰ provides a low-dimensional representation that does not require additional feature extraction,
¹³¹ as depicted in [Figure 5](#).

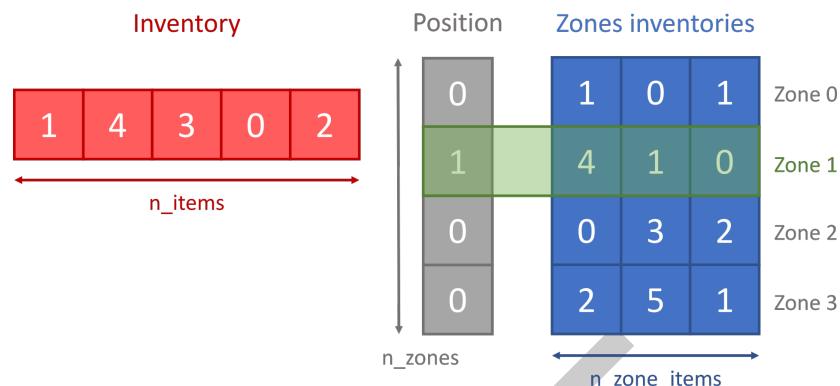


Figure 5: HierarchyCraft compact state representation.

This not only saves computational time but also enables researchers to concentrate on hierarchical reasoning, allows the use of classical planning frameworks such as PDDL (Ghallab et al., 1998) or ANML (Smith et al., 2008), and enables the creation of any arbitrary complex custom environment from a list of *Transformations*, nothing more.

4. Compatible with multiple frameworks

HierarchyCraft environments are directly compatible with both reinforcement learning through Gymnasium (Towers et al., 2024) and planning through the Unified Planning Framework (Micheli et al., 2025) (see Figure 6). This compatibility facilitates usage by both the reinforcement learning and planning communities.

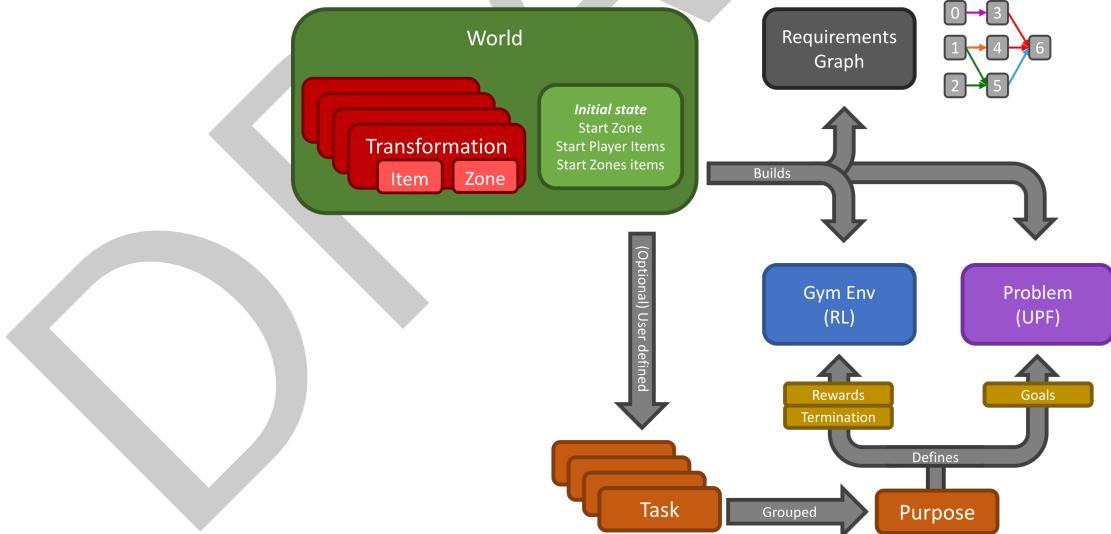


Figure 6: HierarchyCraft pipeline into different representations.

Acknowledgements

This work was made possible by the research program of the engineering cursus at CentraleSupélec, University of Paris-Saclay, France.

The research was conducted at the Intelligent Robot Learning (IRL) Lab, University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Compute Canada; Huawei; Mitacs; and NSERC.

¹⁴⁷ Special thanks to Laura Petrich and other members of the IRL Lab for their assistance in finding
¹⁴⁸ and describing related works, their critical thinking on the project, and their contributions to
¹⁴⁹ the revisions of the documentation and report.

¹⁵⁰ References

- ¹⁵¹ Bacon, P.-L., Harb, J., & Precup, D. (2017). The option-critic architecture. *Proceedings of*
¹⁵² *the Thirty-First AAAI Conference on Artificial Intelligence*, 1726–1734.
- ¹⁵³ Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2015). The Arcade Learning
¹⁵⁴ Environment: An evaluation platform for general agents. *Proceedings of the 24th*
¹⁵⁵ *International Conference on Artificial Intelligence*, 4148–4152. <https://doi.org/10.1613/jair.3912>
- ¹⁵⁷ Botvinick, M., & Weinstein, A. (2014). Model-based hierarchical reinforcement learning
¹⁵⁸ and human action control. *Philosophical Transactions of the Royal Society B: Biological*
¹⁵⁹ *Sciences*, 369(1655), 20130480. <https://doi.org/10.1098/rstb.2013.0480>
- ¹⁶⁰ Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R., Willems, L., Lahou, S., Pal,
¹⁶¹ S., Castro, P. S., & Terry, J. (2023). Minigrid & Miniworld: Modular & customizable
¹⁶² reinforcement learning environments for goal-oriented tasks. In A. Oh, T. Naumann, A.
¹⁶³ Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information*
¹⁶⁴ *processing systems* (Vol. 36, pp. 73383–73394). Curran Associates, Inc. <https://doi.org/10.48550/arXiv.2306.13831>
- ¹⁶⁶ Chollet, F. (2019). On the measure of intelligence. *ArXiv*, *abs/1911.01547*. <https://doi.org/10.48550/arXiv.1911.01547>
- ¹⁶⁸ Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., &
¹⁶⁹ Wilkins, D. (1998). *PDDL – the Planning Domain Definition Language, version 1.2* (Tech
¹⁷⁰ Report CVC TR-98-003/DCS TR-1165). Yale Center for Computational Vision and Control.
- ¹⁷¹ Guss, W. H., Castro, M. Y., Devlin, S., Houghton, B., Kuno, N. S., Loomis, C., Milani, S.,
¹⁷² Mohanty, S. P., Nakata, K., Salakhutdinov, R., Schulman, J., Shiroshita, S., Topin, N.,
¹⁷³ Ummadisingu, A., & Vinyals, O. (2021). The MineRL 2020 competition on sample efficient
¹⁷⁴ reinforcement learning using human priors. *CoRR*, *abs/2101.11071*. <https://doi.org/10.48550/arXiv.2101.11071>
- ¹⁷⁶ Hafner, D. (2022). Benchmarking the spectrum of agent capabilities. *International Conference*
¹⁷⁷ *on Learning Representations*. <https://doi.org/10.48550/arXiv.2109.06780>
- ¹⁷⁸ Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering diverse domains through
¹⁷⁹ world models. *arXiv Preprint arXiv:2301.04104*. <https://doi.org/10.48550/arXiv.2301.04104>
- ¹⁸¹ Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., & Tassa, Y. (2015). Learning
¹⁸² continuous control policies by stochastic value gradients. In C. Cortes, N. Lawrence, D.
¹⁸³ Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*
¹⁸⁴ (Vol. 28, pp. 2944–2952). Curran Associates, Inc. ISBN: 978-1-5108-2502-4
- ¹⁸⁵ Micheli, A., Bit-Monnot, A., Röger, G., Scala, E., Valentini, A., Frama, L., Rovetta, A.,
¹⁸⁶ Trapasso, A., Bonassi, L., Gerevini, A. E., Iocchi, L., Ingrand, F., Köckemann, U., Patrizi,
¹⁸⁷ F., Saetti, A., Serina, I., & Stock, S. (2025). Unified Planning: Modeling, manipulating
¹⁸⁸ and solving AI planning problems in Python. *SoftwareX*, 29, 102012. <https://doi.org/10.1016/j.softx.2024.102012>
- ¹⁹⁰ Milani, S., Topin, N., Houghton, B., Guss, W. H., Mohanty, S. P., Nakata, K., Vinyals, O., &
¹⁹¹ Kuno, N. S. (2020). Retrospective analysis of the 2019 MineRL competition on sample
¹⁹² efficient reinforcement learning. In H. J. Escalante & R. Hadsell (Eds.), *Proceedings*
¹⁹³ *of the NeurIPS 2019 competition and demonstration track* (Vol. 123, pp. 203–214).

- 194 <https://proceedings.mlr.press/v123/milani20a.html>
- 195 Nachum, O., Gu, S., Lee, H., & Levine, S. (2018). Data-efficient hierarchical reinforcement
196 learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, &
197 R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31, pp.
198 3303–3313). Curran Associates, Inc. ISBN: 978-1-5108-8447-2
- 199 Silver, T., & Chitnis, R. (2020). PDDLGym: Gym environments from PDDL problems. In A.
200 Fern, V. Gómez, A. Jonsson, M. Katz, H. Palacios, & S. Sanner (Eds.), *Proceedings of
201 the 1st workshop on bridging the gap between AI planning and reinforcement learning* (pp.
202 1–6).
- 203 Smith, D. E., Frank, J., & Cushing, W. (2008, September). The ANML language. *The
204 ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling*.
- 205 Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework
206 for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2), 181–211.
207 [https://doi.org/10.1016/S0004-3702\(99\)00052-1](https://doi.org/10.1016/S0004-3702(99)00052-1)
- 208 Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M.,
209 Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.
210 J., Tan, H., & Younis, O. G. (2024). Gymnasium: A standard interface for reinforcement
211 learning environments. *arXiv Preprint arXiv:2407.17032*. <https://doi.org/10.48550/arXiv.2407.17032>

DRAFT