

HierarchyCraft: A Benchmark Builder for Hierarchical Reasoning

Mathis F  d  rico ^{1,2}, Shang Wang ¹, Yuxuan Li ^{1,3}, and Matthew E. Taylor ^{1,3}

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](#)
- [@inpeff](#)

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Published: unpublished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](#)).

Summary

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

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

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

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**.

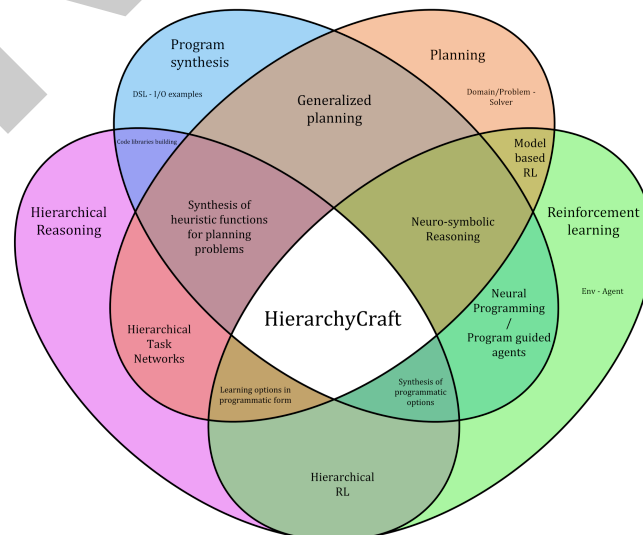


Figure 1: HierarchyCraft is at the intersection of reinforcement learning, planning, hierarchical reasoning and program synthesis.

Statement of need

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

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

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

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

GridWorld

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

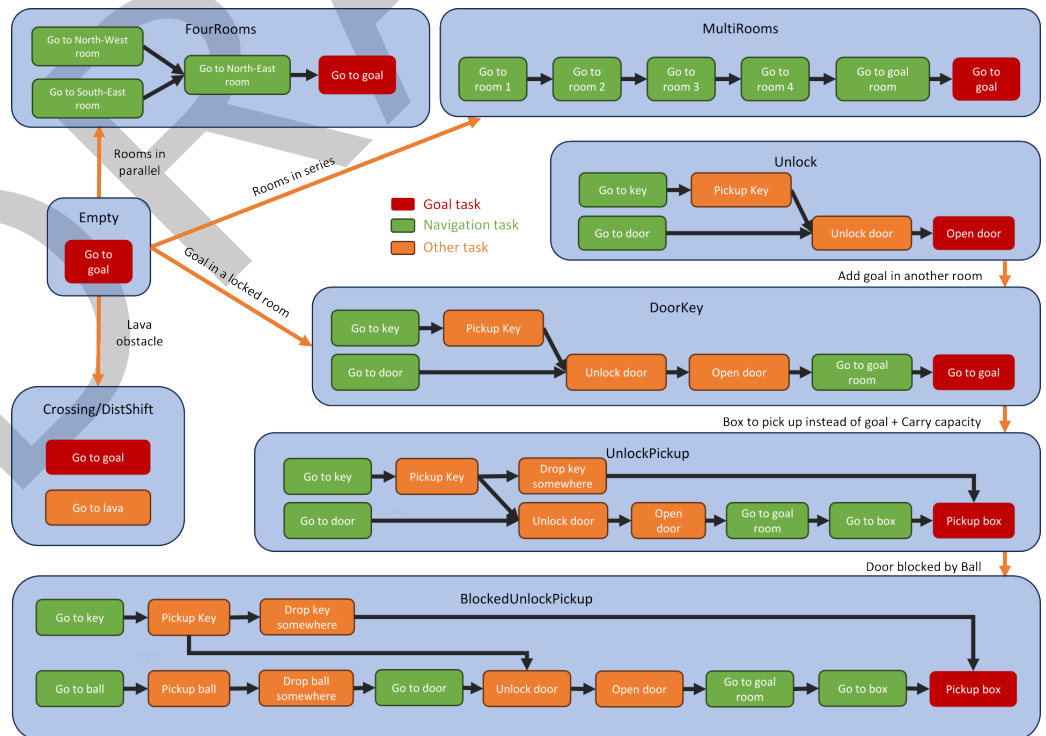


Figure 2: Example of Minigrid environments hierarchical structures and their relationships. There are only a few possible subtasks and most of them are navigation tasks (in green).

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

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

Minecraft

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

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

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

Crafter

Crafter (Hafner, 2022) presents a lightweight grid-based 2D environment, with game mechanics akin to Minecraft and poses similar challenges (exploration, representation learning, rewards 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 **PDDL Gym**

85 PDDL Gym ([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, PDDL Gym 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.

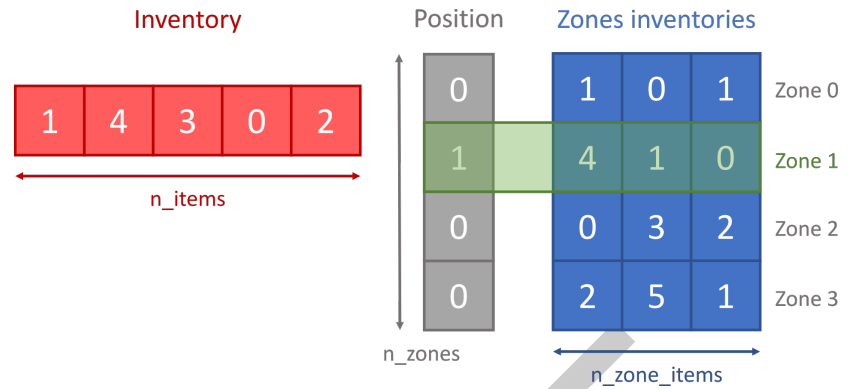


Figure 5: HierarchyCraft compact state representation.

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.

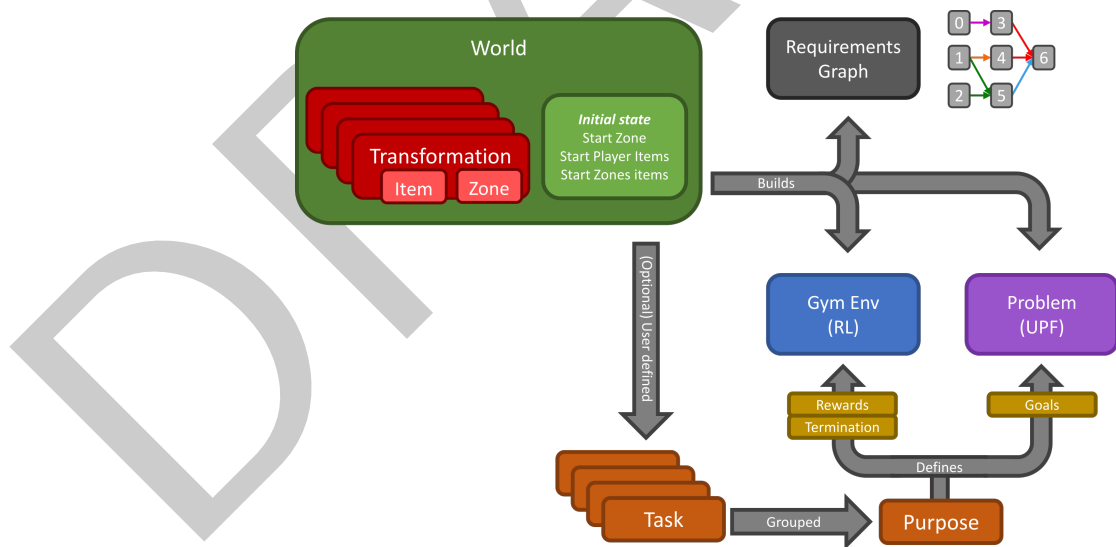


Figure 6: HierarchyCraft pipeline into different representations.

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.

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

References

- Bacon, P.-L., Harb, J., & Precup, D. (2017). The option-critic architecture. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 1726–1734.
- Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2015). The Arcade Learning Environment: An evaluation platform for general agents. *Proceedings of the 24th International Conference on Artificial Intelligence*, 4148–4152. <https://doi.org/10.1613/jair.3912>
- Botvinick, M., & Weinstein, A. (2014). Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655), 20130480. <https://doi.org/10.1098/rstb.2013.0480>
- Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., & Terry, J. (2023). Minigrid & Miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems* (Vol. 36, pp. 73383–73394). Curran Associates, Inc. <https://doi.org/10.48550/arXiv.2306.13831>
- Chollet, F. (2019). On the measure of intelligence. *ArXiv*, abs/1911.01547. <https://doi.org/10.48550/arXiv.1911.01547>
- Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). *PDDL – the Planning Domain Definition Language, version 1.2* (Tech Report CVC TR-98-003/DCS TR-1165). Yale Center for Computational Vision and Control.
- Guss, W. H., Castro, M. Y., Devlin, S., Houghton, B., Kuno, N. S., Loomis, C., Milani, S., Mohanty, S. P., Nakata, K., Salakhutdinov, R., Schulman, J., Shiroshita, S., Topin, N., Ummadisingu, A., & Vinyals, O. (2021). The MineRL 2020 competition on sample efficient reinforcement learning using human priors. *CoRR*, abs/2101.11071. <https://doi.org/10.48550/arXiv.2101.11071>
- Hafner, D. (2022). Benchmarking the spectrum of agent capabilities. *International Conference on Learning Representations*. <https://doi.org/10.48550/arXiv.2109.06780>
- Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering diverse domains through world models. *arXiv Preprint arXiv:2301.04104*. <https://doi.org/10.48550/arXiv.2301.04104>
- Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., & Tassa, Y. (2015). Learning continuous control policies by stochastic value gradients. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28, pp. 2944–2952). Curran Associates, Inc. ISBN: 978-1-5108-2502-4
- Micheli, A., Bit-Monnot, A., Röger, G., Scala, E., Valentini, A., Framba, L., Rovetta, A., Trapasso, A., Bonassi, L., Gerevini, A. E., Iocchi, L., Ingrand, F., Köckemann, U., Patrizi, F., Saetti, A., Serina, I., & Stock, S. (2025). Unified Planning: Modeling, manipulating and solving AI planning problems in Python. *SoftwareX*, 29, 102012. <https://doi.org/10.1016/j.softx.2024.102012>
- Milani, S., Topin, N., Houghton, B., Guss, W. H., Mohanty, S. P., Nakata, K., Vinyals, O., & Kuno, N. S. (2020). Retrospective analysis of the 2019 MineRL competition on sample efficient reinforcement learning. In H. J. Escalante & R. Hadsell (Eds.), *Proceedings 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). PDDL Gym: 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.](https://doi.org/10.48550/arXiv.2407.17032)
212 [2407.17032](https://doi.org/10.48550/arXiv.2407.17032)