Thesis Proposal: Towards Any-Horizon Autonomy via Closed-Loop Natural Language HTN Planning

Anonymous ACL submission

Abstract

The crucial focus of this proposed thesis is to explore (useful) 'any-horizon' autonomy (which I provisionally define). The work will begin by laying a generalized theoretical foundation for what 'any-horizon' autonomy is. Then, it will explore what it might take to achieve such with language agents. This practical exploration will occur through the lens of a Minecraft language agent that handles tasks of 'any horizon' by using natural language to generate, revise, and execute an open-ended, any-depth Hierarchical Task Network (HTN) plan.

1 Introduction

1.1 Any-Horizon Autonomy

By (my) definition, a truly 'any-horizon' autonomous agent is one that is mechanistically equally suited to perform tasks of any duration scope. That is, its propensity for success should *only* correlate with the task's difficulty that is irrespective of time (i.e., difficulty that is not covariant with how long the task might take). Conversely, agents that are *not* 'any-horizon' have inductive bias for some nonuniform distribution of task time horizons.

Given this definition, building a perfect 'any-horizon' agent—for whom (all other things being equal) infinite-horizon tasks and finite-horizon tasks are of the same difficulty—is impossible in a finite-resource world. Nevertheless, an achievable version of 'any-horizon' autonomy does exist—that is, we have an existence proof of (virtually-) 'any-horizon' autonomy in humans.

Characterizing their 'any-horizon' capabilities, humans do the following:

1. **Selectively remember in decreasing fidelity/increasing abstraction:** To accomplish long-horizon tasks, humans remember and reason over their past experiences. However, they do not remember their full-fidelity sensory experience across their entire lifespan. Instead, human memories are higher-level representations (abstractions) that are selectively stored, presumably for future utility. Furthermore, with the passing of time, memories "fade". That is, memories become increasingly vague (abstract) or lost altogether. Through these mechanisms, humans are innately capable of leveraging useful information from the very distant past, without needing infinite storage.

2. **Decomposing tasks with as-needed decreasing abstraction:** Similar to how humans retain memories with varying degrees of conceptual and temporal abstraction, they also reason about the future with abstraction. Crucially, when humans plan the future, they do so by breaking down high level tasks into increasingly granular subtasks, specifying only as much detail as is needed to guide their immediate actions. Given the unpredictable nature of the world, they trust that they'll be able to adequately specify further details as they become relevant and, as needed, adapt strategies altogether. By thus thinking about the future through hierarchical abstraction, humans are able to align their actions with 'any-horizon' plans without needing infinite storage or processing.

This leads to the research question: Can machines also work around their finite resource limitations and achieve similar levels of 'any-horizon' autonomy by leveraging similar processing patterns? The proposed thesis will seek to explore and provide answers to this question.

1.2 Minecraft Language Agent

The proposed thesis will develop a novel Minecraft language agent that, unlike previously proposed

Minecraft language agents, will be developed with the *specific* goal of achieving time-horizon-agnostic aptitude using the two aforementioned human-like processing patterns.

Why Choose Minecraft? Minecraft is relatively diverse and open-ended, lending itself well to being a sandbox for the creation and execution of very lengthy, non-repetitive tasks. Furthermore, simple hierarchical planning with LLMs has already demonstrated promising success for tasks with complexity up to, for example, "mining diamond". I speculate that this success is a result of two main factors:

1. The gameplay does not involve atomic actions that affect many interdependent goal outcomes at once (like how one spin of a Rubix cube changes the position of twelve squares across four sides). Such action-by-action mental juggling can be difficult for LLMs, since the proverbial "chain of thought" necessary for deciding conducive actions is inherently acyclic. This idea will be discussed further in the related work section.

2. Given the ubiquity of Minecraft content on the internet, LLMs tend to be good "databases" of knowledge for recommending sequences of subtasks, removing the need for world-model-simulated rollouts or domain-model optimization to verify plan quality.

In summary, Minecraft is ideal for studying 'any-horizon' autonomy because: (1) individual action choices do not require the mental juggling of many outcomes concurrently; (2) LLMs tend to already be well-knowledged with regards to the game's mechanics; and (3) perhaps most crucially, the space of possible highly composite actions is very diverse.

2 Related Work

2.1 Planning With LLMs

Generally, planning with LLMs is a well-researched topic. One common trend is research that explores using LLMs to generate single "high-level" plans that then get further decomposed to low-level robotic actions (Lin et al., 2023; Huang et al., 2022). Another trend is work that looks at using LLMs or "LLM-modulo systems" to generate action plans that are interpreted as programs for a "Blocksworld", "Logistics", "AlfWorld", or other simulated domain, in which overall task success is binary and programmatically verifiable (Valmeekam et al., 2023; Verma et al., 2024; Kambhampati et al., 2024). Another interesting case in which LLM-based systems are developed and tested for planning is that of TravelPlanner. In TravelPlanner, language agents are tasked with generating travel plans (for a human) that satisfy multiple criteria simultaneously (Xie et al., 2024).

Revisiting the previous notion of tasks where individual actions affect many interdependent outcomes simultaneously, TravelPlanner serves as a notable example. This is because singular actions—such as selecting a specific hotel—can impact multiple criteria at once, including budget, transportation feasibility, or pet accommodation (Xie et al., 2024). This planning challenge can be thought of as requiring acyclic iteration to eventually arrive at a criteria-fulfilling plan.

In contrast, in Minecraft, reasoning about the effects of actions can be nicely siloed within the scope of a parent task. For example, if I am focused on "collecting wood planks", I do not necessarily need to consider how the action of "breaking an oak log" might also effect my future ability to "get a golden apple". Even if doing so could lead to extra efficiencies (after all, an oak tree whose logs have been broken could drop the soon-needed apple), not incorporating this information into current action would not preclude my ability to acquire an apple later on.

2.2 Minecraft Language Agents

Many existing Minecraft agents operationalize LLMs as planners. Most instances of LLM-based Minecraft planning involve creating only a 2-layer deep hierarchical plan, leveraging an LLM-generated "high-level" plan to inform the sequencing of primitive actions (Fan et al., 2022; Wang et al., 2023; Feng et al., 2023; Qin et al., 2024; Yuan et al., 2023). However, Zhu et al. propose an agent that recursively decomposes tasks to create an any-depth hierarchical plan (Zhu et al., 2023). Despite doing this (and, in this way, being somewhat similar to my proposed agent), this work made few empirical comparisons to previous works and only focused on obtaining items.

2.3 HTN Planning

The field of classical planning has, for a long time, leveraged top-down task decomposition for classical planning tasks like, for example, determining behavior in video game AI. In this field, such planning is categorized as hierarchical task network (HTN) or hierarchical goal network (HGN) planning. Classical HTN and HGN planning involves searching through possible hierarchical combinations of tasks from a pre-defined task ontology, aiming to find an optimal plan given a domain model (Georgievski and Aiello, 2014).

Since, for my proposed agent, tasks will be open-endedly generated in natural language, the "HTN planning" will not be constrained to a pre-defined ontology of tasks[1]. Furthermore, the "planning" does not involve any sort of "search" for optimality within a domain model. This latter distinction is perhaps especially interesting, since it contrasts with the current wave of research that looks to plan action trajectories with world-model-simulated roll-outs (Ha and Schmidhuber, 2018; Hafner et al., 2024; Yang et al., 2024).

2.4 Combining HTN Planning With LLMs

One existing suggestion to combine LLMs and classical planning is to use LLMs to generate Planning Domain Definition Language models (PDDL models—i.e., in classical planning, the ontology of tasks and other necessary logic associated with the domain model) (Kambhampati et al., 2023). Then, once the PDDL is generated, one would use classical planning algorithms like HTN planning, perhaps even with LLM-suggested search heuristics to guide the process. Interestingly, such a suggestion is reminiscent of François Chollet's advocacy of *learning* domain specific language(s) (DSLs) for neural-network-informed combinatorial "reasoning" (search), instead of manually building generally-capable propositional systems (Chollet et al., 2025).

In contrast to this suggestion, the design philosophy of my proposed agent assumes that generally, LLM-generated actions (that comprise the "HTN") will already be "good" (worth pursuing), removing the need for mathematical optimization using a domain model or world simulator. Furthermore, the ongoing closed-loop feedback will hopefully be sufficient for the LLMs to self-correct as needed.

Reinforcing my choice of Minecraft, LLMs are already knowledgeable and adept at reasoning about the game's mechanics, allowing for the study of 'any-horizon' autonomy with less preoccupation for the also important, yet orthogonal studies of incorporating model-based (combinatorial) search or expensive search-based prompting strategies like Tree-of-Thought (Yao et al., 2023). Generally, the study of 'any-horizon' autonomy is indeed largely orthogonal (and certainly complementary) to the study of learning and leveraging world models, since 'any-horizon' autonomy only deals with generalizing an agent's abilities to become as time-horizon invariant as possible.

3 Methods

The crucial distinctive behaviors of my proposed agent will be as follows. Note that these behaviors will be achieved by building algorithmic scaffolding around a number of carefully-prompted calls to instruction-tuned language models (as is typical in the field of language agents).

Task decomposition with as-needed decreasing abstraction:

1. The agent recursively decomposes a user-input task into an *any-depth* HTN plan (i.e., tree of increasingly granular subtasks described in natural language), which at any given time, is not necessarily fully fleshed-out, but rather, only fleshed-out enough to disambiguate good next atomic action(s) (deferring the fleshing out of details that are not yet necessary and better conceived later).

2. As the HTN plan's atomic actions (terminal nodes) are progressively attempted, the agent continually reflects over its progress (or lack thereof), conceiving further future plan details (and even re-hashing them) as they become imminent or otherwise pertinent in the top-down task decomposition that informs the immediate policy.

3. While by default, the HTN plan is executed with postorder traversal, the agent can abandon entire directions of approach (branches in the tree) that are not working out, reverting back to the "drawing board" of any higher-level node (goal).

Selective memory with decreasing fidelity/increasing abstraction:

1. Although it is highly unlikely that storing all details from past experiences will actually push against any storage limits, in order

[1]Hence, using the term "HTN planning" may very well be a misnomer for my work.

to (in theory) be ’any-horizon’—i.e., able to succeed if run to solve a many-decade long task—the agent will continually summarize and selectively prune information from its memory[2].

I expect my design may fall into the category of a multi-agent system (MAS), since multiple separate LLM conversations (each prompted for different "judgment calls") must be orchestrated together to achieve these behaviors. However, unlike many multi-agent systems, the orchestration of *which "specialist" to use when* will not be explicitly delegated to the judgment of an LLM. Instead, this will be handled algorithmically—albeit, conditioned upon the outputs of the "specialists" (hence, my uncertainty about the MAS categorization).

3.1 Simplified Example

To make things easier to understand, let’s walk through a simplified example of the algorithm. Let’s start with the input task, "fill this cauldron with water". The first step would be to ask, is this immediately achievable and sufficiently atomic? If not, decompose it into subtasks:

```
-> 1 [in-prog] "fill this cauldron with water"
---> 1.1 "get bucket"
---> 1.2 "fill bucket with water"
---> 1.3 "place water in cauldron"
```

Then ask, is task 1.1 immediately achievable and sufficiently atomic? If not, decompose it into subtasks:

```
-> 1 "fill this cauldron with water"
---> 1.1 [in-prog] "acquire bucket"
-----> 1.1.1 "acquire 3 iron"
-----> 1.1.2 "craft bucket"
---> 1.2 "fill bucket with water"
---> 1.3 "place water in cauldron"
```

This would recurse until the next-most action is deemed to be immediately achievable and sufficiently atomic, at which point that action would be attempted:

```
...-> 1.1.1 "acquire 3 iron"
...
...---> 1...1 [success] "walk to nearby chest"
...---> 1...2 [in-prog] "check chest for iron"
...
```

Now, let’s assume that the nearby chest has a water bucket in it. The agent, conscious of the higher-level goals would abandon its current branch and revert to an appropriate higher-level node. Only a summary of the salient details from the aborted branch would be retained and the subsequent plan would be re-evaluated:

```
-> 1 [in-prog] "fill this cauldron with water"
---> 1.1 [aborted] ... -> chest has water bucket
---> 1.2 "get water bucket from chest"
---> 1.3 "place water in cauldron"
```

Although simplified for the sake of explanation, this is more or less how the previously enumerated key behaviors will be achieved.

3.2 Experiments

Of course, to truly test a Minecraft agent aimed at ’any-horizon’ autonomy, one must observe how it handles tasks that far surpass the duration scope of most typically-benchmarked tasks. Indeed, this makes proper testing a non-trivial obstacle and will likely create the need to both: (1) compare empirical results (against baseline and other language agent architectures) on the typically-benchmarked array of moderate-length Minecraft tasks (which many pre-existing agents already excel at) as well as (2) perform and comparatively analyze a small number of case studies with *very* lengthy tasks.

An example of a *very* lengthy Minecraft task that would lend itself to being a valuable case study might be:

> "Build a village with as many houses as there are unique plant species in your current biome. In each house, there should be a chest containing a single unique item that the other house’s chests do not contain. For extra points, use as many distinct wood varieties as you can."

I anticipate that comparing Minecraft language agents on tasks like this will prove to be the more salient demonstration that differentiates and evidences the utility of the proposed agent.

4 Concluding Remarks

In summary, the aspiration of the proposed thesis will be to extract insight from the experience of building an agent whose design goal is to achieve useful ’any-horizon’ autonomy—i.e., an agent that, in theory, could accomplish tasks far too lengthy to test within a reasonable (simulated) time-frame. Can the human-analogous behaviors of (1) task decomposition with as-needed decreasing abstraction and (2) selective memory with decreasing fi-

[2]Aside from, in this way, staying true to the sentiment of ’any-horizon’ autonomy, this can serve the secondary purpose of increasing the signal-to-noise ratio of the information rendered into the numerous prompts.

delity/increasing abstraction be effectively implemented in an artificial agent—even if only "hacked together" with LLMs? If so, this could be a first step towards creating fully-differentiable networks that have enough mechanistic priors (pre-engineering) for these behaviors to be retained and enhanced when trained end-to-end, for example, with reinforcement learning.

References

Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. 2025. Arc prize 2024: Technical report.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge.

Yicheng Feng, Yuxuan Wang, Jiazheng Liu, Sipeng Zheng, and Zongqing Lu. 2023. Llama rider: Spurring large language models to explore the open world.

Ilche Georgievski and Marco Aiello. 2014. An overview of hierarchical task network planning.

David Ha and Jürgen Schmidhuber. 2018. World models.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2024. Mastering diverse domains through world models.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Llms can't plan, but can help planning in llm-modulo frameworks.

Subbarao. Kambhampati, Karthik. Valmeekam, Matthew. Marquez, and Lin. Guan. 2023. On the role of large language models in planning. Tutorial presented at the International Conference on Automated Planning and Scheduling (ICAPS), Prague.

Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. 2023. Text2motion: from natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365.

Yiran Qin, Enshen Zhou, Qichang Liu, Zhenfei Yin, Lu Sheng, Ruimao Zhang, Yu Qiao, and Jing Shao. 2024. Mp5: A multi-modal open-ended embodied system in minecraft via active perception.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change.

Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. 2024. On the brittle foundations of react prompting for agentic large language models.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents.

Sherry Yang, Jacob Walker, Jack Parker-Holder, Yilun Du, Jake Bruce, Andre Barreto, Pieter Abbeel, and Dale Schuurmans. 2024. Video as the new language for real-world decision making.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. 2023. Skill reinforcement learning and planning for open-world long-horizon tasks.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory.