Agent Hunt: Bounty Based Collaborative Autoformalization With LLM Agents

Chad E. Brown[1], Cezary Kaliszyk[2] and Josef Urban[3,*]

[1]AI4REASON
[2]University of Melbourne, Australia and University of Innsbruck, Austria
[3]AI4REASON, University of Gothenburg and Chalmers University of Technology

Abstract

We describe an experiment in large-scale autoformalization of algebraic topology in an Interactive Theorem Proving (ITP) environment, where the workload is distributed among multiple LLM-based coding agents. Rather than relying on static central planning, we implement a simulated bounty-based marketplace in which agents dynamically propose new lemmas (formal statements), attach bounties to them, and compete to discharge these proof obligations and claim the bounties. The agents interact directly with the interactive proof system: they can invoke tactics, inspect proof states and goals, analyze tactic successes and failures, and iteratively refine their proof scripts. In addition to constructing proofs, agents may introduce new formal definitions and intermediate lemmas to structure the development. All accepted proofs are ultimately checked and verified by the underlying proof assistant. This setting explores collaborative, decentralized proof search and theory building, and the use of market-inspired mechanisms to scale autoformalization in ITP.

1. Motivation: Fast Parallelized LLM Autoformalization

LLM agents [1, 2] have recently shown promising results in autoformalization of large portions of mathematical textbooks [3]. While this is encouraging, the general topology project reported in [3] was as of February 18, 2026, still ongoing.[1] This means that two months after its major launch, it was not yet finished, even though is has reached over 350k lines. The prior general-topology experiment is important here because it provides a concrete baseline for a current LLM-driven formalization workflow: a single agent working for many weeks in one large Megalodon development. Our question is not whether the resulting formalization is already a finished mathematical library, but whether the same style of cheap, LLM-driven proof production can be parallelized without immediately collapsing under merge conflicts, duplicated work, or inconsistent proof state. Thus the experiment should be read as a systems and methodology study for automated reasoning, rather than as a completed formalization of algebraic topology.

In this work we therefore explore how multiple LLM agents can collaborate on such large projects, parallelize the work efficiently, and progress in such large projects much faster than a single agent. The main idea is to use a bounty-based setting introduced by Hales in the Flyspeck project [4, 5] and then let agents compete and collaborate in proving the theorems and collecting the bounties. Our hope is that such decentralized market setting will be easier and more flexible, than trying to plan upfront, centrally and manually the detailed division of labor between many agents. This is because large-scale formalizations may have unpredictable aspects to them, such as, e.g., gaps in the proofs, forward references, etc.

Practical Aspects of Automated Reasoning (PAAR) 2026, as part of the Federated Logic Conference (FLoC) 2026, Lisbon, Portugal, July 25, 2026
*Corresponding author.
ckaliszyk@unimelb.edu.au (C. Kaliszyk); josef.urban@gmail.com (J. Urban)
https://ckaliszyk.github.io/ (C. Kaliszyk)
0000-0002-8273-6059 (C. Kaliszyk)

[1]We used publicly available data from the recent commit https://github.com/mgwiki/mgw_test/commit/a7033f85cd73201e423e76864c19d89e88d35327 of the above mentioned general topology project.

Research questions. We evaluate the bounty-based setting against three concrete questions. First, can multiple untrusted LLM agents make sustained progress in the same proof-assistant repository while preserving statement immutability and proof-checking consistency? Second, does the market mechanism lead to observable division of labor, reuse of other agents' lemmas, and completion of nontrivial proof obligations, rather than only producing independent bloated code? Third, what failure modes are introduced by competition itself, such as proof sniping, duplicated effort, bad initial statements, or incorrect bounty accounting? We do not claim that this experiment proves optimality of the market mechanism; rather, it is an exploratory ablation target for future comparisons with purely cooperative multi-agent prompting.

2. Target: Autoformalization of Algebraic Topology in Megalodon

Since we are interested in building on the autoformalization framework developed in [3] and limited to the part I of the book, we use a very similar setting, using the Megalodon higher-order set theory proof checker [6]. Since that project has already autoformalized the main general topology definitions and theorems in Munkres (part I, about 250 pages), our natural target is part II (Chapter 9 to 14, Sections 51 to 85) of Munkres, i.e., the remaining about 200 pages of algebraic topology [7].

2.1. Initial Autoformalization of Statements and Setting of Bounties

A major change from the one-agent setting of [3] is that we want to set up the formal definitions and theorems (*statements* below) upfront and attach adequate bounties on them. This to some extent corresponds to Hales's "Flyspeck blueprint" setting and book, with one expert mathematician making this high-level formal sketch and price estimation. We are aware that this can lead to problems if some statements are autoformalized wrongly (which did happen in [3]). While in [3], the single agent can recover and correct the statements, here we decide to fix them, so that the agents cannot game the system and collect bounties easily. We therefore invest substantial effort into double-checking the initial autoformalized statements by repeated LLM checking prompts and high-level manual inspection. We are also ready to step in as admins if we see a major statement/definition problems later.

In particular, we initially gave the 200 LaTeX pages (together with the background theory) to a single LLM agent (Claude Opus 4.6) with stringent rules summarized as follows:

- The background file already contains a lot of foundational material (set theory, topology, etc.) to be re-used, and only the new algebraic topology section is allowed to be edited.
- The task is to formalize every definition and theorem from algtop.tex (the LaTeX representation of part II of Munkres' book), in order, but without proofs (everything admitted for now). However, definitions must be mathematically meaningful (no dummy or stub definitions) and repeatedly double-checked.
- The process must be extremely disciplined: no duplicates, no adding extra lemmas not in algtop.tex, no modifying earlier parts of the file, frequent compilation checks with megalodon, regular backups, and careful progress tracking.
- There are strong quality-control rules: definitions must be substantive, infrastructure must be built properly, previous work must never be lost, and each theorem must include an effort/cost estimate. In particlular, for every stated lemma/theorem, the agent must estimate (1) the number of lines of a textbook proof, (2) the formalization difficulty on a 1–10 scale, and (3) the approximate simulated USD cost assuming $100/hour. This effort estimate assumes all previous algtop.tex results are already proved.

This prompt is repeatedly given to the agent until it converges in about 8 hours (producing 32 backups), announcing (after several debugging and double-checking runs) that it is not aware of any possible issues. The resulting file with the background theory and the newly created 230 definitions and 393 toplevel theorems (with the effort-based bounties) is then used as the start for our multi-agent autoformalization.

3. Agent and Bounty Based Setup

We ultimately used four LLM agents (named Alice, Bob, Charlie and Dave), using two ChatGPT Pro Codex 5.3 models and two Claude Code (Opus 4.6 and Sonnet 4.6) models. Our setup is based on the rules of work and other settings described in [3]. In particular, we largely follow the CLAUDE.md rules file published there with similar agent workflow for a single agent, but we modify it for the agentic and bounty setting. The summary of these modified rules (234 lines) is as follows:

- Competitive–collaborative bounty system: Four agents compete for the theorem bounties (45k "simulated USD" tokens total) but are incentivized to collaborate to finish early and earn bonuses. Agents can issue sub-bounties, solve others' bounties, and strategically choose between cooperation and competition to maximize total and personal earnings.
- Locking and earning mechanics: Agents can lock a theorem by paying 10% of its bounty (max 10 locks, 24h each), reserving the right to collect the full bounty if completed. If someone else proves a locked theorem, the bounty still goes to the locker; expired locks must be removed, and balances can never go negative.
- Strict commit and ownership discipline: Agents must not modify others' locks, partial proofs, or sandboxes, and cannot change existing definitions/theorem statements. Frequent pull-commit-push cycles are mandatory, locks must be pushed immediately, and guard tools must confirm no rule violations before committing.
- Strategic focus and safety rules: Prioritize major theorems over exercises (better financial and project impact), avoid reverting or losing work, and never edit outside the AlgTop section. Progress should be continuous, incremental, and carefully merged to prevent destructive conflicts or wasted effort.

The four agents were intentionally not assigned fixed mathematical roles. All agents received the same repository, rules file, and list of available bounties; their specialization emerged only from their own theorem selection and the locking mechanism. The two ChatGPT agents and the two Claude agents were run as independent command-line coding agents, each interacting with Megalodon through compilation, local editing, Git commits, and the guard scripts described below. Human intervention was limited to starting and stopping agents, resolving repository-level conflicts, tightening global rules after observed failures, and manually correcting administrative inconsistencies such as the Charlie balance reset described below.

Guard Scripts: To enforce correct handling of balances, bounties, and locks, we use local guard scripts that agents must run before committing. The initial lightweight version checked core invariants (non-negative balances, positive bounties, at most 10 locks per agent, lock expirations within 24 hours, and basic bounty collection rules), but relied on relatively naïve line-based parsing, which allowed certain edge-case violations. A later stream-based version tokenizes the entire file, enforces immutability of definition and theorem statements (while allowing proof modifications), precisely tracks `Qed` vs. `Admitted` (completed proofs vs incomplete ones), validates lock persistence and expiry more robustly, checks balance transitions, and prevents misuse of keywords inside comments. The script is intentionally run locally rather than as a blocking Git hook, permitting coordinated structural changes when needed. After resolving minor time-zone-related lock inconsistencies, the improved script has been functioning reliably.

Preserving the trusted core. Agents were only allowed to edit the algebraic-topology region of the single Megalodon file. The guard script enforces this by comparing the token stream of the current file against the previous committed version: definitions and theorem statements must remain syntactically identical, whereas proof bodies may change from `Admitted` to `Qed`. This is supplemented by Megalodon's own recursive dependency check: a proof can be closed with `Qed` only when all dependencies are already closed or are among the explicitly allowed imported axioms. Consequently, the main consistency invariant is not social trust in the agents, but proof checking plus statement immutability.

4. Formalization Growth and Collaborative Aspects

The following metrics should be interpreted cautiously. Line count is not a measure of mathematical value, and bounty balance is not a measure of semantic difficulty. We nevertheless report them because they measure operational properties central to this experiment: whether four agents can continuously contribute to one repository, whether bounties are actually claimed rather than ignored, whether agents create subgoals (new unproved lemmas stated in the file) for others, and whether the development grows without frequent destructive rollbacks. For mathematical progress, we separately report completed non-recursive `Qed` theorems and case studies in Sect. 6.

The formalization of the proofs with multiple agents started at about 8pm on Feb 16, with about 19k normalized lines[2] of the previous library (mostly set theory and topology). By Feb 19, 11am, (2 days and 15 hours later) this number of lines has reached 121k. I.e. the four agents jointly produced about 39k lines per day. We have compared these numbers with the publicly available numbers from the General Topology project as of Feb 19, 2026[3] The general topology project has reportedly been then running for about 60 days, reaching about 406k normalized lines, i.e. about 7k lines per day on average (using however only a single agent). This is only a rough comparison, one should also be looking at the capability to prove major theorems (Sect 6), etc. Still, this speed is encouraging.

Fig. 1 shows the formalization size over the run of the experiment.[4] The formalization grows mostly linearly with only very small local dips, mostly corresponding to refactoring done by the agents. In Fig. 2, the we see that the agents' balances rise overall. We have added Dave (4th agent) later, and had to reset Charlie's balance manually after it used a wrong Megalodon version, committing wrong theorems and collecting bounties incorrectly (this led us to tightening the framework). This was prompted by high-level manual checking of the overall formalization status performed several times a day. Cumulative bounties collected increased steadily across the agents, all agents locked some theorems throughout and most agents placed bounties on sub-lemmas they create.

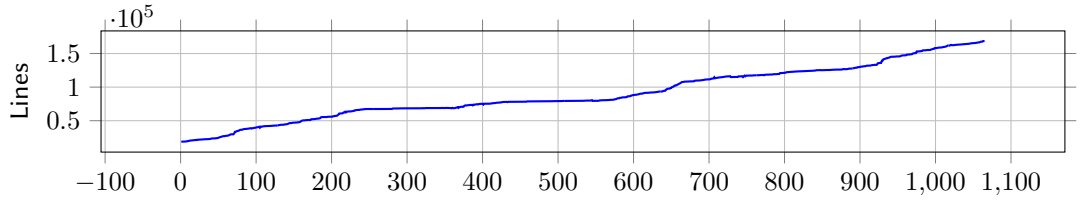

Figure 1: Growth of the formalization over history (commit numbers).

Collaboration and Competition: We tracked the bounty lifecycles over the course of the experiment by detecting when a theorem first entered a Bounty state and when it later transitioned to a Collected state. Restricting to bounties with identifiable creators (In some cases a different agent would state a theorem and a different one would put a bounty on it), the agents have placed a total of 709 tokens in new bounties in total: 279 were proved and collected by the same agent who created them, 114 were proved/collected by a different agent, 312 remained active/unsolved at the end of the experiment, and 4 were removed or rewritten without a direct collect transition. Indeed, there is both substantial self-completion by creators, and some cross-agent collaboration and proof completion.

We have also observed some competition: At one point Bob was quite far in the proof of `ex68_3_conjugate_intersection_trivial`[5], but did not lock the theorem (admittedly having his maximum of 10 other locked theorems). When he was close to completing the last admits, Alice stepped in and replaced the 3716-line almost complete proof, by a complete proof and collected the

[2]We normalize the number of lines by translating to html and then back to text. This is because some agents adopted a peculiar proof style spanning many lines.

[3]https://github.com/mgwiki/mgw_test/commit/a7033f85cd73201e423e76864c19d89e88d35327

[4]We use unnormalized number of lines in Fig. 1 since it is easier to compute across many commits.

[5]If a group G is the free product of the subgroups G_1 and G_2, then every conjugate of G_1 intersects G_2 only in the identity element.

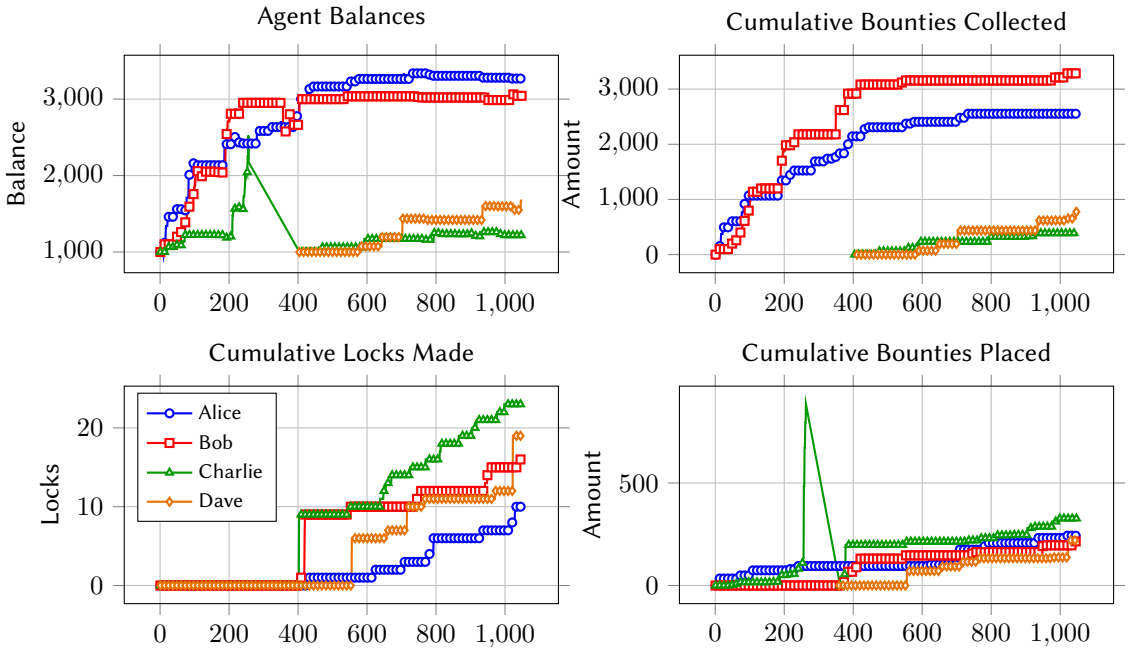

Figure 2: History of balances, locks, bounty collection and placement (over commit numbers), the line for Charlie has been reset at balance-reset point as discussed in Sect. 4.

bounty. Similarly, the agents would leave comments for themselves, such as `TODO Bob show <some property>`, and in one instance we saw a different agent, Charlie, change the proof but also change the comment still leaving it as a TODO for Bob, but slightly changing the thing that remains for Bob to do.

Division of Labor: We also observe that there is some division of labor across the development: While there are overlaps at interfaces (notably between homotopy, covering-space, and algebraic sections), a clear thematic ownership by specific agents can be observed, with no part developed by all four agents in any single region. In particular, Bob focused on core homotopy and fundamental-group pipeline, together with much of the technical covering-space infrastructure and later algebraic machinery; Charlie focused on geometry-driven topology, especially circle/disk and projective-space constructions, plus compactness and related topological consequences; Alice concentrated on foundational path-concatenation and group-law formalization and on key naturality/isomorphism bridge lemmas, and Dave did abstract group-theoretic support, including subgroup/quotient/commutator and recursion-based algebraic facts, with a smaller contribution to covering-map results. This division emerged naturally, we believe it is based on the locks, initial lemmas proved by the agents, and the different models.

Resources Used: We have used three $200/month LLM subscriptions (two ChatGPT Pro and one Claude Max) each for about 3-4 days. This has depleted the weekly usage of these LLMs. Based on that we estimate the cost of the experiment so far to be about $150. This is a bit more than $1 per 1k normalized lines (with comments removed).

5. Observations and Remarks

Exercises: Exercises typically do not have proofs in textbooks like Munkres. This unfortunately led to very misleading cost estimates in the initial phase that was assigning the effort to the theorems. As an extreme case, we have then witnessed agent Charlie writing 800 lines of proof and getting only 10-token bounty for it. After that, we have updated CLAUDE.md to disregard exercises as much possible, both as less relevant than the main textbook material, and because they are much less profitable. We did not restart the experiment, leaving the started parts of the exercises.

5.1. Megalodon Changes

Megalodon was originally built for checking compact, human-written formal proof developments, that would manually import required parts and manually specify which definitions should be Opaque (i.e., not automatically unfolded). To make Megalodon work more effectively with single long LLM-produced file, we modified several parts of Megalodon's core checking pipeline to be more efficient, in particular replacing many repeated linear lookups and heavy comparison paths with operations. Similarly, the trust model was tightened for untrusted agents: The Qed command is now blocked for proofs that are complete but whose dependencies were only partially proved. This means that only proofs with recursively checked dependencies can be closed with Qed, otherwise the agents must mark the proofs as Admitted. Furthermore, we modified the error messaging so that the agents are more clearly informed about their mistakes, and can more effectively fix them. All symbol and lemma names are now shown in the readable format to the LLMs rather than the previous opaque hashes. Additionally, we made some improvements to the Megalodon-hammer [8], although it is currently almost unused by the agents. Finally, as we now forbid the use of unproved Axioms, and the development relies on the general topology that is not complete, we use an *index* file of *allowed* axiom hashes passed to Megalodon as an additional argument.

6. Major Theorems Proved and Case Studies

Table 1 shows the proved theorems with a normalized proof length of over 400 lines (no admits or recursive admits) sorted according to the length of their direct Megalodon proof.[6] Note that e.g. the first one—*a group is cyclic of infinite order iff it is isomorphic to Z*–is mentioned only in passing in Munkres and without a proof. The table does not list the lemmas which are finished modulo a major admitted dependency. There are several such lemmas with very long proofs such as lemma59_1_open_cover_generates_pi1_core (length 6132), lemma68_1_extension_condition_free_product (length 5546), lemma54_1_path_lifting (length 2431), thm55_5_nonvanishing_vector_field (length 1604). In addition, to progress with some of the proofs, the agents have introduced 5 new definitions: homotopy_flip_map, ex53_1_slice_family, s55_radial_collapse_map, comm_closure_pred, commutator_closure. We suspect that shorter proofs of these lemmas are possible, but we have not asked the agents to prioritize readable or short proofs, instead we asked them to focus on completing the formalization. We discuss some of these theorems and their blockers below.

Table 1
Algebraic topology theorems with line counts.

Theorem	Lines
cyclic_infinite_order_iff_Z	1999
thm60_1_pi1_product	1474
Theorem_51_3_reparametrization	1446
ex53_4_composition_covering	1426
s55_lemma58_4_homotopy_path_continuous	1351
lemma58_4_homotopy_path	1349
thm53_3_product_covering	1339
ex58_2h_open_disk_simply_connected	1102
ex53_6b_compact_finite_fiber	1054
ex67_4b_free_abelian_no_torsion	920
Theorem_51_2_left_identity	906

Theorem	Lines
Theorem_51_2_right_identity	759
Theorem_51_2_right_inverse	705
thm53_2_subspace_covering	674
path_concat_well_defined_on_classes	581
lemma52_1_cancel_double_basepoint_change_class	562
Lemma_51_1_path_homotopy_trans	523
evenly_covered_open_subset_top	521
lemma67_1_converse	446
ex67_4a_torsion_subgroup	423
ex53_1_discrete_projection_covering	420

[6]The evolving public repository is at https://github.com/mgwiki/alg_top.

Fundamental group One of the initially unproven theorems says that the fundamental group is, in fact, a group. The original estimated difficulty for the proof was (automatically) rated to be 5 out of 10, so the bounty was declared as 100 tokens. Ultimately, Alice proved the theorem and collected the bounty, but the proof relied on several previous results. Some of the previous results also had bounties. Most of these other bounties were collected by Alice, but two were collected by Bob. Some other previous results used in the main proof were created and proven by Alice or Bob, but did not have associated bounties. The combination of these previous results in the main proof give an example of collaboration between agents happening in practice. We now consider the division of labor in slightly more detail.

Given a topological space (X, Ω) and a point $x \in X$, the elements of the fundamental group are the equivalence classes of loops on x (paths from x to x). Two paths are equivalent if they are homotopy equivalent (one can be continuously deformed into the other). Formally, the elements of the carrier of the group are sets of functions, but we will describe the group as if the elements are loops. To make the definitions and proofs formally correct, the agents must handle the details of passing between the equivalence classes and their representatives. The multiplication is given by concatenating loops. The identity loop is given by constant x function. The inverse loop is given by reversing the path.

Before the experiment began, there were a number of unproven theorems that ultimately became part of the main proof: the homotopy equivalence relation is reflexive (proven by Bob for 30 tokens), path concatenation is well-defined on equivalence classes (proven by Alice for 110 tokens), the multiplication given by path concatenation is associative (proven by Alice for 275 tokens), the proposed identity is both a left and a right identity (both proven by Alice for 150 tokens and 165 tokens) and the proposed inverse is both a left and a right inverse. Alice proved the inverse is a right inverse and collected the bounty of 150 tokens. Bob proved the inverse is a left inverse and collected the bounty of 165 tokens.

Bob also created and proved 11 theorems (without bounties) that Alice used in the main proof. These tend to be minor results that are nevertheless helpful to avoid expanding various definitions. One example of such a theorem is that every member of the carrier of the fundamental group (an equivalence class of loops) has some loop as a representative.

Brouwer Fixed Point Theorem The agents completed a proof of the Brouwer Fixed Point Theorem, though ultimately it depends on an unproven theorem that the fundamental group of the circle is isomorphic to the integers. Assuming the fundamental group of the circle is isomorphic to the integers, the agents have proven (with a 2390 line proof) that the inclusion map from the circle to the real plane is not nulhomotopic (not homotopic to a constant map). Using this result, the agents have proven (with a 3729 line proof) that for every nonvanishing vector field on the closed unit disk, there is a point on the circle that points directly inward and a point on the circle that points directly outward. From this the agents proved the Brouwer Fixed Point Theorem using a 1564 line proof.

The agents were not yet able to prove the fundamental group of the circle is isomorphic to the integers. By manually inspecting the state of the formalization, we suspect part of the difficulty comes from an unsatisfactory definition of $\cos$ and $\sin$. The two functions are defined using the epsilon choice operator as the pair of continuous real functions (γ, σ) such that $\gamma(0) = 1$, $\sigma(0) = 0$, $\forall x \in \mathbb{R}.(\gamma(x))^2 + (\sigma(x))^2 = 1$, $\forall xy \in \mathbb{R}.\gamma(x+y) = \gamma(x)\gamma(y) - \sigma(x)\sigma(y)$ and $\forall xy \in \mathbb{R}.\sigma(x+y) = \sigma(x)\gamma(y) + \gamma(x)\sigma(y)$. The agents prove no properties of these functions and would need to prove there exist such functions before even the defining properties could be extracted. Unfortunately, the pair of functions is not uniquely determined. The pair of functions $\cos(kx)$ and $\sin(kx)$ (for any integer k) would also satisfy the properties. In particular, the pair of the constant 1 function and the constant 0 function satisfy the properties. Clearly the definitions of $\cos$ and $\sin$ are faulty and trying to prove anything using them could only distract the agents. The agents would need to generate new (correct) definitions and use those.

Completion status. At the end of the reported run, the project was far from a complete formalization of Munkres Part II. The useful outcome is more limited: the agents discharged a substantial number

of proof obligations in the shared development, including the large closed theorems listed in Table 1, while leaving several central dependencies unproved. It might be interesting to re-run the experiment with a single agent and compare the efficiency. The most important remaining blocker is the proof that $\pi_1(S^1)$ is isomorphic to $\mathbb{Z}$, which blocks fully closed versions of downstream results such as the Brouwer fixed-point theorem. We therefore distinguish between theorems closed with recursively checked dependencies and theorems whose direct proof script is complete but still depends on a major admitted result.

7. Conclusion

The accompanying material for the paper can be found at:

https://ckaliszyk.github.io/e/26-agenthunt-paar/

This experiment shows that a number of LLM coding agents can make sustained, proof-checked progress in a single large formalization project when constrained by statement immutability, recursive dependency checking, Git discipline, and lightweight economic rules. The bounty mechanism did produce observable division of labor, lemma re-use across agents, and completion of several large proof obligations. It also introduced clear failure modes: inaccurate initial bounty estimates, duplicated effort, proof sniping, and sensitivity to faulty initial definitions such as the trigonometric infrastructure needed for $\pi_1(S^1)$.

Future work could include a comparison between the bounty-based setting and purely cooperative multi-agent prompting. Furthermore, instead of static bounties, we could consider adaptive pricing based on changing observed proof difficulty and changing and dependency structure. The experiment also highlights the need for stronger automated-reasoning backend for Megalodon.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT, Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content. Furthermore, the full formalization was done by coding agents as described in the paper.

Acknowledgments

This work was supported by the ERC project NextReason and Renaissance Philanthropy grant DEEPER, and the sponsors of the AI4REASON institute.[7]

References

[1] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: Language models can teach themselves to use tools, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

[2] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, in: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023, OpenReview.net, 2023. URL: https://openreview.net/forum?id=WE_vluYUL-X.

[7] https://ai4reason.eu/sponsors.html

[3] J. Urban, 130k lines of formal topology in two weeks: Simple and cheap autoformalization for everyone?, 2026. URL: https://arxiv.org/abs/2601.03298. arXiv:2601.03298.
[4] T. C. Hales, Dense sphere packings: a blueprint for formal proofs, volume 400, Cambridge University Press, 2012.
[5] T. Hales, M. Adams, G. Bauer, T. D. Dang, J. Harrison, L. T. Hoang, C. Kaliszyk, V. Magron, S. McLaughlin, T. T. Nguyen, et al., A formal proof of the kepler conjecture, in: Forum of mathematics, Pi, volume 5, Cambridge University Press, 2017, p. e2.
[6] C. E. Brown, K. Pak, A tale of two set theories, in: C. Kaliszyk, E. C. Brady, A. Kohlhase, C. S. Coen (Eds.), Intelligent Computer Mathematics - 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings, volume 11617 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 44–60. URL: https://doi.org/10.1007/978-3-030-23250-4_4. doi:doi:10.1007/978-3-030-23250-4_4.
[7] J. R. Munkres, Topology, second edition, reissue ed., Pearson, New York, NY, 2018.
[8] C. E. Brown, C. Kaliszyk, M. Suda, J. Urban, Hammering higher order set theory, in: V. de Paiva, P. Koepke (Eds.), Intelligent Computer Mathematics - 18th International Conference, CICM 2025, Brasilia, Brazil, October 6-10, 2025, Proceedings, volume 16136 of *Lecture Notes in Computer Science*, Springer, 2025, pp. 3–20. URL: https://doi.org/10.1007/978-3-032-07021-0_1. doi:doi:10.1007/978-3-032-07021-0_1.