ARE WE READY FOR PROOF WEAVING? UNDERSTANDING AGENT SWARMS THROUGH FERMAT'S LAST THEOREM

Anonymous authors
Paper under double-blind review

ABSTRACT

A collection of correct proofs is not yet a proof of a common goal. We study how agent systems can build a common proof from earlier results, a process we call *proof weaving*. The public formalization of Fermat's Last Theorem (FLT) provides a completed reference for studying this ability. We introduce FLTBENCH, ten Lean reconstruction tasks drawn from this development. Each task withholds a target proof and a specified set of supporting reference proofs, while retaining the target statement and its permitted mathematical foundation. Continuing a proof also requires recognizing what earlier work has established, so we examine both reconstruction and the interpretation of work records. In our evaluation, all four evaluated systems complete the same one of the ten tasks. Case analysis shows checked cross-agent reuse, preserved unfinished constructions, and correction of a false completion report. In a separate diagnostic, an agent assesses 100 constructed work records, including normal proof reports, partial work, and conflicting claims of success. Its judgments disagree with the assigned completion status in 20 cases. These findings highlight proof weaving as a challenge for agent research. FLTBENCH and the diagnostic cases offer the community a concrete basis for studying how local contributions can become complete mathematical proofs.

1 INTRODUCTION

The promise of an agent swarm is not that it can produce many answers. It is that it can build something no individual worker could readily produce alone. In mathematics, this promise has a precise test: do the results produced by different workers eventually support a complete proof of the same theorem? A worker may establish a useful lemma without knowing the entire argument. A later worker may use that lemma without reconstructing its derivation. What must survive between them is not the worker, or even its reasoning trace, but a mathematically usable contribution.

The recent public formalization of Fermat's Last Theorem makes this ambition concrete. Anthropic reports an approximately eleven-day, multi-agent development with around thirty thousand proved statements, and releases a machine-checkable artifact (Anthropic, 2026a;d). This is a formalization of existing mathematics, not a new discovery of FLT. Yet the achievement poses a new question for agent research: *what must an agent system do to construct, maintain, and combine a body of mathematics of this scale?* A completed proof establishes that the destination is attainable. It does not, by itself, tell us how to get there reliably with different agents, controllers, or resource budgets.

An appealing starting hypothesis is that local proving already supplies the essential ingredient. If agents can solve difficult subproblems, a shared repository should preserve their solutions, and more agents should produce more progress. The missing step is *composition*. A lemma can be correct but unavailable to the worker that needs it, expressed through an incompatible definition, or irrelevant to the route currently under construction. Even useful results may not close a parent goal until several complementary obligations have been resolved. The number of accepted statements therefore does not determine what the collective can now prove from its accumulated work.

There is a second, less visible difficulty. When the target remains unfinished, we still need to decide whether the system is advancing. A natural response is to ask an agent to summarize the project, rate its progress, and redirect the workers. But this response may assume an evaluator with a global

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

understanding that none of the workers possesses. We seek collective capability beyond a single agent while potentially relying on a single agent to recognize it. Checking a completed proof and interpreting a large collection of unfinished contributions are different tasks. The latter calls for understanding which results can support the remaining work.

FLT offers a way out of this circularity. Its released proof supplies fixed mathematical statements and an existing witness for a large, interdependent construction. We can remove a bounded region of that construction and ask a system to rebuild it. The resulting task has its own complete, independently checkable endpoint, even when reconstructing all of FLT remains beyond the tested budget. Within each run, we can also trace how intermediate contributions are used and which obligations remain open. We give up an unjustified estimate of "percent of FLT completed" in exchange for an answer to a more useful question: *which contributions can be reused, under which conditions, and which remaining obligations must be discharged to complete the shared target?*

This is the role of FLTBENCH. Its ten reconstruction tasks span an existing proof development. Its purpose is not to replace the field with one larger leaderboard, but to support an *understanding* study of proof weaving. We compare four agent systems on these tasks and trace how agents reuse checked results, preserve unfinished constructions, and correct completion claims. A separate diagnostic tests one prerequisite for continuation: whether an agent can recognize target completion from constructed work records that present candidate material and checking reports.

The precedent is methodological, not merely rhetorical. KITTI exposed limitations by changing the conditions under which existing methods were measured (Geiger et al., 2012). ELF OpenGo paired an executable reimplementation with external validation and experiments that challenged intuitive accounts of progress (Tian et al., 2019). Here, reconstruction tasks provide concrete mathematical endpoints for examining how local work becomes a common proof. Our contribution is a way to study proof weaving, a benchmark of reconstruction tasks, and empirical evidence on reuse, continuation, and assessment of target completion. Together, these give the community concrete targets and cases for studying how local contributions become a common proof. The central question is not whether the agents remain busy. It is whether their work continues to become a proof.

2 MOTIVATING OBSERVATIONS: FROM PROOF SEARCH TO PROOF WEAVING

The intuition behind scaling up a successful problem-solving swarm is straightforward: workers that can solve difficult problems should also be able to contribute useful pieces of a larger proof. Our preliminary experience gives a concrete reason to examine this intuition more carefully. We describe the successes that made it plausible, the unfinished FLT construction that left it unresolved, and the evaluation problem that arises when local results must support one continuing argument.

2.1 COMPLETE-ANSWER SUCCESSES

We began with tasks for which the desired outcome was unambiguous: the system had to return a complete solution that could be checked. Our starting system, PSSWARM, is a decentralized swarm whose workers retain their task contexts and share experience. A worker can attempt a whole target, choose an intermediate goal, or build on another worker's result; no central planner assigns a single proof plan to the whole swarm. This made it a useful starting point for asking how far local problem solving and shared work could take us toward a complete mathematical construction.

With `gpt-5.5`, PSSWARM completed the five twelve-task collections in Table 1, spanning programming, verified code generation (Ye et al., 2025; Thakur et al., 2025), and formal mathematics (Tsoukalas et al., 2024). These outcomes show that the system could deliver accepted solutions across several kinds of task. Appendix A gives the dataset details.

The same system also let us examine reuse in a mathematical construction. With `gpt-5.5`, we asked PSSWARM to search for stronger Ramsey lower bounds. Here the answer is a red/blue coloring of every edge of a complete graph. A coloring on n vertices with no red clique of size four and no blue clique of size s establishes $R(4, s) \geq n + 1$. A larger valid coloring thus gives a stronger lower bound, and the coloring itself is a certificate that can be checked. Table 2 compares the initial bounds adopted for these tasks with those obtained by the system.

Table 1: **PSSWARM completes all twelve targets in each historical collection.** Each row reports results with `gpt-5.5`; completion time is the time until all twelve targets are accepted, rounded to minutes. Programming outputs use repository-local judges and formal targets use Lean checking. These historical runs use collection-specific settings and are not matched comparisons with the later FLT reconstruction experiments reported under the common four-hour budget.

Collection	Accepted targets	Completion time (min)
ICPC World Finals 2025	12/12	29
USACO 2025 US Open	12/12	31
Verina, selected hard twelve	12/12	23
CLEVER, selected hard twelve	12/12	18
PutnamBench 2025	12/12	407

Table 2: **Recovering and extending a coloring improves the adopted Ramsey bounds.** A red/blue coloring of a complete graph on n vertices with no red four-clique or blue s-clique establishes $R(4, s) \geq n + 1$. Columns compare the published starting bounds adopted for these tasks (Google Research, 2026) with PSSWARM using `gpt-5.5`. The $s = 20$ result recovers the 251-vertex coloring of Yu (2026); adding five vertices yields the $s = 21$ bound.

Ramsey number	Initial lower bound	PSSWARM (ours)
$R(4, 20)$	237	252
$R(4, 21)$	244	257

The first construction recovered the 251-vertex coloring of Yu (2026). Every vertex follows the same cyclic rule for its red neighbors, which reduces the clique checks to neighborhoods of one vertex. Verification finds no red four-clique and a largest blue clique of size 19, establishing $R(4, 20) \geq 252$. The second construction preserves all of this coloring and adds five vertices. Three copy the red neighborhoods of selected base vertices, while the other two connect to those new vertices and to each other in red. Checking the additional cliques gives a 256-vertex coloring with no red four-clique and a largest blue clique of size 20, establishing $R(4, 21) \geq 257$. Appendix B gives the full construction and explains the clique bounds for both colorings.

The Ramsey extension is more than another solved task: it reuses an earlier solution to produce a new, complete construction. This made it plausible that shared work could also help with a larger proof. What these successes leave open is whether contributions can support different parts of the same unfinished argument, where no single contribution yet certifies the common goal.

2.2 AN UNFINISHED SHARED CONSTRUCTION

FLT gave us a setting in which to examine this further demand. It states that, for any integer $n > 2$, the equation $a^n + b^n = c^n$ has no solution in positive integers a, b, c. Workers were free to attempt the whole theorem or pursue intermediate goals. A useful local contribution now had to help others continue this common argument after its producer moved on.

Figure 1 shows a later exploratory run using `gpt-6-astra` with `max` reasoning. Many proposals reduced FLT to a smaller set of cases: proving it for exponent four and for every odd prime would suffice. Such a reduction explains how a proof could be completed, but leaves those cases to be proved. Of the 20 conditional proposals captured in one snapshot, 17 relied on this same prime-or-four core. The record also contains actual reuse: one worker continued another's candidate and used an existing proof for exponent four. The general odd-prime case nevertheless remained open after these reductions and instances of reuse (Appendix C).

The unresolved premise still contained substantial mathematics. More reductions to it did not supply the missing proof, even though workers had reused earlier work and established useful implications. The observation does not isolate a cause of failure; it shows why activity and reuse alone do not tell us how much of a common proof has been established. An evaluation must also ask which remaining claims the accumulated work can actually resolve.

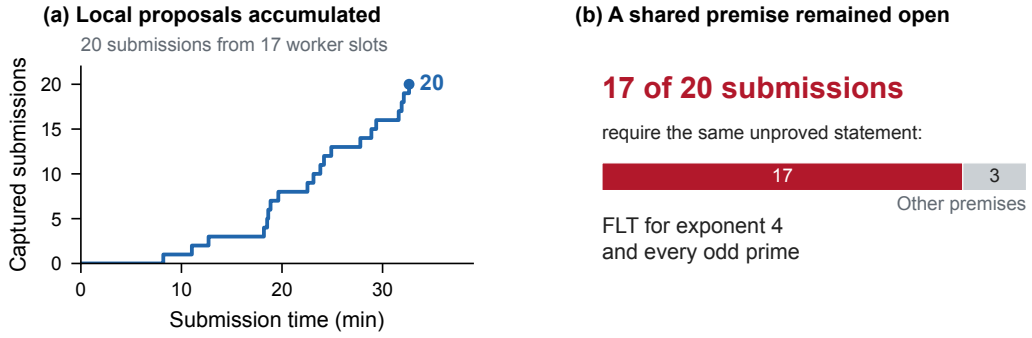

Figure 1: **More FLT proposals can leave the same mathematical premise open.** An exploratory run on Fermat's Last Theorem (FLT) uses PSSWARM and `gpt-6-astra` at `max` reasoning. (a) Cumulative count of 20 conditional proposals captured in one truncated snapshot, from 17 worker slots. Time starts at the first event in the retained observation window. (b) The same subset: 17 proposals require FLT for exponent four and every odd prime; three use other premises. These are proposed reductions, whose remaining premises still need proofs. The counts show growing proposal activity while the shared FLT obligation remains open.

2.3 WHAT EXISTING EVALUATIONS LEAVE OPEN

The FLT exploration leaves an evaluation question: how can we tell whether shared intermediate work is sufficient to complete the larger argument? Existing systems already support decomposition, reuse, and sustained formalization: LeanMarathon maintains an evolving proof blueprint, while Prove2Me assembles proofs from shared statements and results (Zhang et al., 2026; Chen et al., 2026). The remaining question is how to evaluate whether these mechanisms turn local work into an extended, complete construction. Competition benchmarks such as miniF2F and PutnamBench evaluate individual targets (Zheng et al., 2022; Tsoukalas et al., 2024); miniCTX tests proving with supplied project context, and APE-Bench evaluates focused changes extracted from library commits (Hu et al., 2024; Xin et al., 2026). These evaluations include demanding tasks and completed source artifacts, but their task boundaries differ from reconstructing an extended network of missing support. SorryDB instead collects open proof obligations from real projects, without requiring an already completed reference for each task (Letson et al., 2026). To study proof weaving, we seek both a substantial successful construction and control over what supporting work remains available. With fixed statements and foundations, withholding connected parts of such a reference lets us examine whether locally valid contributions become jointly sufficient for the target.

2.4 A REFERENCE THAT MAKES RECONSTRUCTION MEASURABLE

The public FLT formalization supplies such a successful reference. Anthropic's completed development has about 29,500 named theorem nodes in its final dependency structure (Anthropic, 2026a;b). The public Lean source[1] builds on Mathlib and adapted material from the Imperial College London FLT and `flt-regular` projects. Its campaign combined agents, Prove2Me's shared statements and proof support, and occasional high-level human guidance (Anthropic, 2026d; Chen et al., 2026). We use this completed construction as the reference from which to build reconstruction tasks.

The completed artifact gives us fixed statements, mathematical interfaces, and proofs that demonstrate how the pieces can fit together. Withholding parts of that support creates reconstruction tasks with complete, checkable endpoints. An unfinished attempt can then be studied in relation to a concrete target and the support it still needs. This lets us investigate how local work becomes a common proof without estimating a subjective percentage of the whole FLT development. The next section defines that construction process; FLTBENCH then turns its demands into reconstruction tasks with specified targets and permitted mathematical foundations.

[1] `https://github.com/anthropics/fermats-last-theorem`

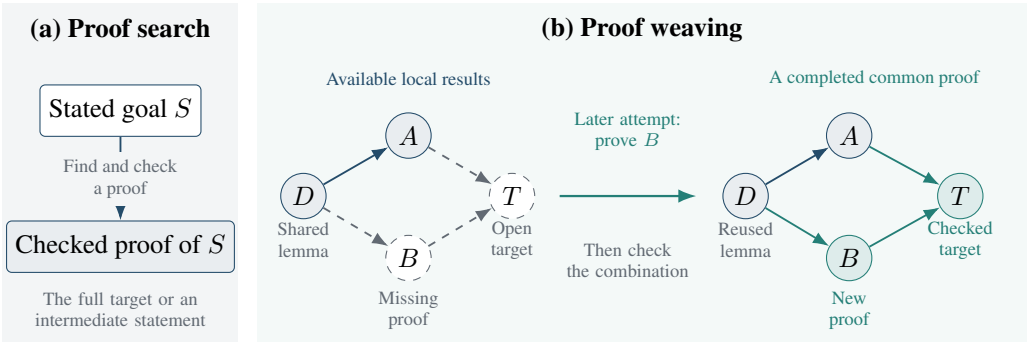

Figure 2: **Proof weaving combines complementary results into a common proof.** (a) Proof search seeks a checked proof of goal S. (b) Arrows point from supporting results to claims that use them: D supports A and B, which together prove T. Initially D and A are proved; a later attempt proves the missing B and then completes T. Dashed nodes and edges mark unfinished work; solid ones mark proved results and checked dependencies. This is a schematic example.

3 Definition of Proof Weaving

The preliminary observations ask what happens when local proofs must support one continuing argument. We make this construction process and its endpoint explicit.

Definition. We call *proof weaving* the construction of a common proof by creating, reusing, and combining contributions across branches and over time. Consider a target T obtained from A and B, both depending on a shared construction D (Figure 2b). Proving D supports both branches, but a second proof of A cannot replace the missing B. Different versions of D may require a mathematical bridge before they can be combined. A new route might instead prove T directly. The system must therefore discover suitable statements, establish their compatibility, and revise its route.

Proof search is a component of this process. Both search and weaving can involve multiple agents, and a persistent single agent with external memory can also carry out a woven construction. The distinction concerns how results support a common argument, rather than the number of workers.

Why a collective, and why an extended construction? Formal proof objects let later workers check and use an earlier contribution without reconstructing its derivation. Verification still leaves the work of finding relevant results and applying them. The potential benefit of a swarm is thus shared intermediate work beyond independent attempts at the target.

Such work can outlive its producer: a definition may be reused after other branches change and the original worker leaves. Repeated construction, reuse, and integration test whether the system preserves usable objects and resumes the right obligations. Merely spending more time on an isolated proof does not create these demands of delayed reuse and worker succession.

Completing a shared target. For a task Q with target T, Lean environment $\mathcal{E}$, and permitted foundation $\mathcal{B}$, let K_t contain the proofs obtained by time t. A proved reduction $r : A \to B \to T$ can belong to K_t while A or B remains open. Completion requires a proof of T with the supporting claims discharged under $\mathcal{B}$:

$$Y_Q(t) = \mathbf{1}\{\exists p \text{ produced by } t : \ \mathrm{Check}_{\mathcal{E},\mathcal{B}}(p : T) = \mathrm{accept}\}\,. \tag{1}$$

Here $\mathrm{Check}_{\mathcal{E},\mathcal{B}}$ denotes Lean checking under that foundation. The endpoint is one when a complete proof has been accepted and zero otherwise; an unfinished candidate may still contain useful work.

What local correctness cannot tell us. A library containing r, a proof of A, and many auxiliary results can still lack B, while a smaller library can close T. Neither lemma count nor reference overlap generally orders progress toward completion: an alternative proof may use fewer reference statements. We therefore examine both completion and the support that intermediate work develops. The next section fixes targets and foundations while allowing different proof routes.

Table 3: **The ten FLTBench targets span local interfaces and larger shared constructions.** Counts describe the modules reached through imports from each target in the completed FLT reference, excluding Mathlib and Lean's library. N: named theorems including the target; D: definition modules; H: theorem nodes on the longest dependency chain; S: theorem nodes used by at least two immediate theorem dependents. The levels group local construction, assembly within irreducibility, and connections between large theories. These are reference profiles, not solver outputs; other proofs may use fewer results, and D is not the amount of context supplied to a solver.

ID	Target	N	D	H	S
I. Focused construction					
Q01	Frey-package construction	1	1	1	0
Q02	Reducibility to a cofixed line	83	11	9	24
Q03	Local multiplicative filtration	66	11	11	16
Q04	Galois-stable prime-order quotient	100	36	12	41
II. Subtheory assembly					
Q05	Mazur's Step 3	5,303	505	74	2,699
Q06	No cofixed line for $p \geq 17$	5,427	510	75	2,734
Q07	Mazur–Frey irreducibility	5,485	511	76	2,757
III. Cross-theory synthesis					
Q08	Modularity of the Frey curve	27,820	1,405	114	12,997
Q09	Level lowering to level two	27,874	1,412	114	13,016
Q10	Fermat's Last Theorem	29,511	1,450	118	13,856

4 FLTBENCH: RECONSTRUCTION AT DIFFERENT WEAVING SCALES

FLTBENCH turns this construction problem into ten Lean reconstruction tasks, Q01–Q10, drawn from the completed FLT formalization (Anthropic, 2026a;b). A short target can hide a large construction; the reference exposes the supporting mathematics used by one successful proof.

Selecting mathematical checkpoints. The reference starts from an FLT counterexample, constructs a Frey elliptic curve, and combines properties of this curve and its representation to derive a contradiction. Our targets follow this route: Q01 constructs a Frey package of normalized counterexample data, Q02–Q07 develop and assemble the irreducibility argument, Q08 establishes modularity, Q09 lowers the level to two, and Q10 proves FLT. We select interfaces where later arguments need a particular object or property. Finer sampling within irreducibility lets us study local constructions and their assembly in the same setting. Appendix D details the mathematical targets, the reference profiles, and the supplied foundations for each reconstruction task.

What must be reconstructed. Removing only the final proof could leave a short call to an already proved result. We therefore also withhold its supporting reference proofs. For target T_Q, let C_Q contain the modules reached by following its imports in the reference. We withhold the target proof and every theorem/solution module in C_Q. The permitted foundation $\mathcal{B}_Q$ retains Mathlib and the target's definition context without reintroducing withheld proofs. Supporting statements and import relations guide search; systems choose the auxiliary definitions, lemmas, and proof route to develop. Each task runs independently and succeeds by proving its unchanged target from $\mathcal{B}_Q$ (Equation 1).

Three scales of construction. **Level I** (Q01–Q04) constructs local objects or interfaces, with Q01 providing a single-theorem anchor above its foundation. **Level II** (Q05–Q07) assembles longer chains within irreducibility. **Level III** (Q08–Q10) connects large theories. Table 3 compares the depth and shared support of these reference constructions.

Why the branches must fit together. Q07 illustrates the required assembly. Q02 turns reducibility into a cofixed line, while complementary results exclude such a line. These include Q06, whose reference uses Q03, Q04, and Mazur's step Q05. Completing one branch leaves the others open. Runs do not share solved checkpoints; each must develop the support that turns its intermediate work into a complete proof of its own target from the permitted foundation.

Table 4: **Only the local Frey-package target, Q01, is completed by all four systems.** Entries count successful Lean reconstructions out of three runs on the same FLTBench target, with the target proof and supporting reference theorem modules withheld. Success requires a complete proof from the permitted foundation within four hours. All systems use `gpt-6-astra` with `max` reasoning and at most 16 concurrent agents per run (120 runs total). Every run on Q02–Q10 remains unfinished.

System	Q01	Q02	Q03	Q04	Q05	Q06	Q07	Q08	Q09	Q10
PSSWARM	3/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
Archon	2/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
LeanMarathon	2/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
Prove2Me-Doc	3/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3

5 UNDERSTANDING WHEN WORK BECOMES A PROOF

We study how local contributions support a common proof and how agents recognize what the shared work establishes. We compare four systems on reconstruction, follow cases of reuse and correction, and then assess an agent's completion judgments on constructed work records.

5.1 EXPERIMENTAL SETUP

To compare different ways of organizing reconstruction, we hold the tasks, agent reasoning settings, and run limits fixed. Each system retains its native organization and uses `gpt-6-astra` with `max` reasoning under the common settings summarized in Table 7.

Systems. We include PSSWARM, the decentralized system introduced in Section 2.1. Archon supplies project orchestration for Lean proving (Frenzymath, 2026), while LeanMarathon supplies a shared blueprint and proof/repair workflow (Zhang et al., 2026). *Prove2Me-Doc* is our implementation of the declaration-sharing, decomposition, and reuse workflow described in Prove2Me's public documentation (Chen et al., 2026; Prove2Me, 2026). Appendix E.1 gives further details of how these systems organize and share mathematical work.

Task and evaluation. Each system receives the same target statement, permitted foundation, and supporting statement framework. It chooses its own proof route without human guidance or access to the withheld reference proofs. A run succeeds only with a complete Lean proof of the target; Table 4 reports completion under the common budget. We use the candidate histories to examine what remains unfinished and how earlier contributions support later attempts.

5.2 LOCAL CONTRIBUTIONS AND THE COMMON GOAL

Table 4 confines completion to Q01, the local Frey-package construction. The other focused targets require a subgroup or quotient with properties needed later. The cases below show what local work contributes toward these targets and what the next step still needs.

Table 5 follows three ways that work passes to another agent: a checked lemma becomes a dependency, a proposed route retains its mathematical gap, and a completion claim is corrected.

On Q03, a Prove2Me-Doc agent proves that the zero-component condition is preserved by the decomposition-group action. Another agent uses this lemma to prove subgroup stability under that action; both proofs are accepted. The stability theorem applies to a subgroup with the required torsion characterization and cardinality. Completing Q03 also requires constructing that subgroup and establishing the inertia property. The reused proof thus provides a specific part of the common argument, with a clear mathematical condition for its later use by another worker.

On Q04, a PSSWARM candidate proposes a factorization $\varphi = \psi \circ \pi$: $\pi : G \to G/L$ quotients the curve-point group G by the given subgroup L, and ψ must realize the quotient on an integral curve with the required reduction properties. A successor tries direct theorem calls and then returns the same candidate, retaining its explicit realization gap. The route and the mathematical obligation survive the handoff together, leaving a concrete construction for a later worker to complete.

Table 5: **Local contributions carry different information for continuing a common proof.** Each row follows a retained run: Prove2Me-Doc on local filtration (Q03), PSSWARM on a prime-order quotient (Q04), and Archon on a cofixed-line target (Q02). The middle column records checked reuse, preservation of a proposed route, or correction of a completion claim; the final column states the mathematics still needed by that target. Appendix E.3 details these contributions, their remaining mathematical obligations, and the recorded checking outcomes.

System and task	Local contribution	Remaining mathematical step
Prove2Me-Doc, Q03	One accepted lemma is reused by another agent to prove conditional subgroup stability.	Construct the subgroup with the required characterization and cardinality, and establish its inertia property.
PSSWARM, Q04	A successor retains a proposed quotient construction and its explicit realization gap.	Realize the quotient as an integral curve with the required reduction properties.
Archon, Q02	Review detects a congruence/equality error and corrects a completion report.	Prove the cofixed-line target with a valid argument.

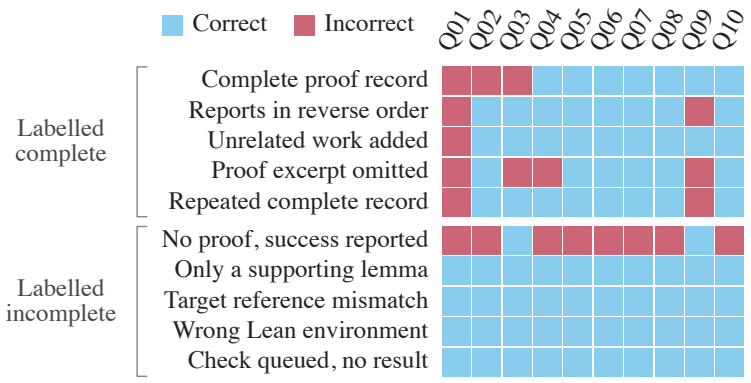

Figure 3: **All eight incorrect "complete" judgments occur when success is reported without a candidate.** Each cell is one `gpt-6-astra` (`max`) observer judgment of a constructed record pairing an FLTBench target (Q01–Q10) with candidate material and checking reports, read without tools. Rows vary the record; the upper five are labelled complete and the lower five incomplete. Blue/rose means the completion judgment agrees/disagrees with that assigned label. Omitting a proof excerpt retains its candidate reference and checking report; the unsupported-success condition supplies no candidate. Figure 4 summarizes these same 100 judgments.

On Q02, an Archon prover treats $b \equiv 0 \pmod 2$ as an equality $b = 0$ over the integers and reports completion. Its reviewer identifies the failed Lean check and the invalid inference; the next plan rejects this route. The correction restores the shared account of which claim still needs a proof.

These cases connect proof weaving to the conditions under which earlier work can be used. We next test one part of this judgment: whether an agent can substantiate a target's completion claim from the candidate material and checking reports in a constructed record.

5.3 Assessing target completion from work records

A later worker needs to connect a reported result to the claim it establishes before deciding what to reuse or prove next. We test this judgment with a separate observer agent and 100 constructed records, ten for each FLTBench target. Each record pairs candidate material with checking reports. The observer reads them without tools and identifies evidence for its completion judgment. The task

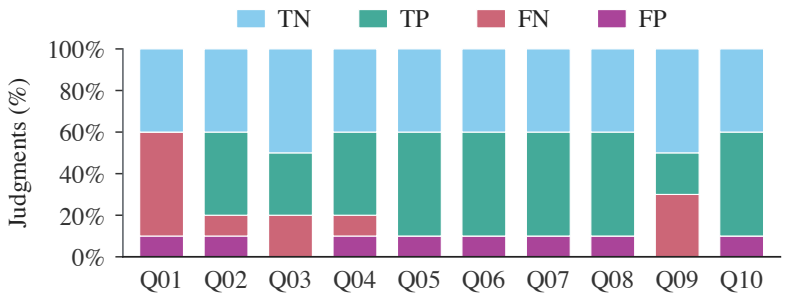

Figure 4: **The observer makes errors in both directions when judging target completion.** A `gpt-6-astra` (`max`) observer judges completion from constructed work records containing candidate material and checking reports, without tools. Each bar shows ten judgments for one FLTBench target: five records labelled complete and five incomplete. TP/TN are correct complete/incomplete judgments; FN calls a labelled complete record incomplete; FP calls an incomplete record complete. Across the same 100 judgments as Figure 3, agreement is 80% (38/50 complete and 42/50 incomplete), ranging from 40% to 90% across targets. Agreement scores the observer's judgments against the assigned completion labels.

asks what the presented record establishes about the target, without having the observer reconstruct or execute the underlying proof in its Lean environment.

Figure 3 distinguishes ten record conditions. Five represent completed results in different forms; five represent missing or mismatched support for the target. In the positive cases, a candidate reference and acceptance report remain available even when the proof excerpt is omitted. In the conflicting case, a success report accompanies an empty candidate. These variations test whether a completion judgment follows the material that is supposed to support the reported result. Appendix E.4 describes the constructed labels and record presentation used in each condition.

In Figure 3, all eight incorrect "complete" judgments occur among the ten records that report success for an empty candidate. The other forty records labelled incomplete are classified correctly. Twelve records labelled complete are judged incomplete, including four with the proof excerpt omitted. Thus the errors include both accepting a conflicting report and rejecting records that retain a candidate reference and acceptance report. Figure 4 groups these same judgments by target; its 80% agreement summarizes the observer's decisions against the assigned labels.

Both errors matter for continuation: an unsupported success can leave a needed premise open, while a missed completion can send workers back to work already done. The Q03 case in Section 5.2 shows why a completion account must preserve both sides: the stability theorem is usable under its premises, while subgroup construction and the inertia condition remain open. For proof weaving, shared records should connect each accepted contribution to the obligations still needed for the common target, keeping local success distinct from target completion.

6 Conclusion

Proof weaving asks whether local contributions can become a common mathematical proof. We introduce FLTBench to study this question through ten reconstruction tasks. The experiments reveal limited completion alongside concrete instances of reuse, preserved obligations, and corrected reports; the diagnostic examines recognition of target completion. Together, these provide the community with a basis for studying cumulative proof construction, toward an open, reproducible agent-swarm system for reconstructing FLT through shared and reusable mathematical work.

Limitation. The comparison covers four systems on ten targets from one reference using `gpt-6-astra` (`max`) under a common budget; the selected cases do not isolate causes of failure. The diagnostic measures agreement with constructed record labels without Lean replay, rather than completion in live proof development. Testing whether these observations generalize requires broader settings and records tied to checked proof artifacts.

AI USE STATEMENT

ChatGPT assisted with manuscript writing, understanding mathematical concepts, and experimental design. AI agents were used for mathematical problem solving, proof construction, and completion assessment in the experiments described in this paper. The authors take responsibility for the final content, including the mathematical claims and their interpretation.

ETHICS STATEMENT

The evaluations described here use public mathematical and programming tasks and constructed work-record diagnostics. They involve no human-subject study or sensitive personal data. We do not anticipate direct harm from these task domains.

REPRODUCIBILITY STATEMENT

The preliminary agents and datasets are listed in Appendix A. Appendix B gives the Ramsey constructions, and Appendix C explains the FLT reduction example. Appendix D describes the mathematical content of the ten FLTBENCH tasks. The anonymous supplementary archive `FLT-Bench.zip` provides the task files and their public mathematical context. Appendix D.4 describes its contents and the requirements for building the environment from source. Appendix E describes the compared systems and common experimental settings, the selected proof-construction cases, and the recognition setup. Section 5.3 presents the recognition results, and Section 5.2 examines local contributions and the obligations they leave open.

REFERENCES

Anthropic. Formalizing Fermat's Last Theorem. Research article, September 4, 2026a. URL https://www.anthropic.com/research/formalizing-fermats-last-theorem.

Anthropic. Fermat's last theorem: The route of the proof. Public repository, PROOF-PATH.md, 2026b. URL https://github.com/anthropics/fermats-last-theorem/blob/main/PROOF-PATH.md.

Anthropic. Fermat's last theorem in lean 4: Formal proof sources. Public Lean source repository, 2026c. URL https://github.com/anthropics/fermats-last-theorem.

Anthropic. Formalizing Fermat's Last Theorem in Lean: A timeline and selected excerpts from Claude's reasoning. Technical report, 2026d. URL https://www-cdn.anthropic.com/9e431dff043da6538d99d6c2d231b670aa3da263.pdf.

Shuze Chen, Kunal Marwaha, Xiaoyang Lu, Henry Yuen, and Tianyi Peng. Prove2Me: An open collaborative platform for scaling math formalization. arXiv:2608.28433v2, 2026. URL https://arxiv.org/abs/2608.28433v2.

Frenzymath. Archon: AI-assisted Lean project automation. Public software repository, 2026. URL https://github.com/frenzymath/Archon.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012. URL https://www.cvlibs.net/publications/Geiger2012CVPR.pdf.

Google Research. Ramsey number bounds: Improved bounds. Public certificate archive, accessed September 25, 2026, 2026. URL https://github.com/google-research/google-research/tree/master/ramsey_number_bounds/improved_bounds.

Jiewen Hu, Thomas Zhu, and Sean Welleck. miniCTX: Neural theorem proving with (long-)contexts. arXiv:2408.03350, 2024. URL https://arxiv.org/abs/2408.03350.

ICPC Foundation. Problem set for the 2025 ICPC world finals baku, 2025. URL https://worldfinals.icpc.global/problems/2025/finals/index.html.

Austin Letson, Leopoldo Sarra, Auguste Poiroux, Oliver Dressler, Paul Lezeau, Dhyan Aranha, Frederick Pu, Aaron Hill, Miguel Corredera Hidalgo, Julian Berman, George Tsoukalas, and Lenny Taelman. SorryDB: Can AI provers complete real-world Lean theorems? arXiv:2603.02668v2, 2026. URL https://arxiv.org/abs/2603.02668v2.

Prove2Me. Prove2Me agent workspace and API documentation. Public agent workspace and documentation, 2026. URL https://github.com/prove2me/prove2me_workspace.

Amitayush Thakur, Jasper Lee, George Tsoukalas, Meghana Sistla, Matthew Zhao, Stefan Zetzsche, Greg Durrett, Yisong Yue, and Swarat Chaudhuri. CLEVER: A curated benchmark for formally verified code generation. arXiv:2505.13938, 2025. URL https://arxiv.org/abs/2505.13938.

Yuandong Tian, Jerry Ma, Qucheng Gong, Shubho Sengupta, Zhuoyuan Chen, James Pinkerton, and C. Lawrence Zitnick. ELF OpenGo: An analysis and open reimplementation of AlphaZero. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 2019. URL https://arxiv.org/abs/1902.04522.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2407.11214.

USA Computing Olympiad. 2025 US Open contest: Problems, solutions, and test data, 2025. URL https://usaco.org/index.php?page=open25results.

Huajian Xin, Luming Li, Xiaoran Jin, Jacques Fleuriot, and Wenda Li. APE-Bench: Evaluating automated proof engineering for formal math libraries. arXiv:2504.19110v3, 2026. URL https://arxiv.org/abs/2504.19110v3.

Zhe Ye, Zhengxu Yan, Jingxuan He, Timothe Kasriel, Kaiyu Yang, and Dawn Song. VERINA: Benchmarking verifiable code generation. arXiv:2505.23135, 2025. URL https://arxiv.org/abs/2505.23135.

Charlie Yu. Ramsey number $R(4, 20) \geq 252$. arXiv:2608.18169v2, 2026. URL https://arxiv.org/abs/2608.18169v2.

Yuanhe Zhang, Yuekai Sun, Taiji Suzuki, Jason D. Lee, and Fanghui Liu. LeanMarathon: Toward reliable AI co-mathematicians through long-horizon Lean autoformalization. arXiv:2606.05400, 2026. URL https://arxiv.org/abs/2606.05400.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: A cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

A PRELIMINARY AGENTS AND DATASETS

The observations in Section 2 come from historical complete-answer experiments and a later FLT exploration. We distinguish their agents and task collections here so that each observation can be read in its own experimental setting and relative to the appropriate collection of tasks.

Agents. The preliminary PSSWARM contest, code-verification, Putnam, and Ramsey experiments used `gpt-5.5`. The FLT trajectory in Figure 1 comes from a later exploratory run using `gpt-6-astra` with `max` reasoning for the shared FLT construction.

Datasets. The programming collections contain the twelve problems from the ICPC World Finals 2025 (ICPC Foundation, 2025) and the twelve USACO 2025 US Open problems across Bronze, Silver, Gold, and Platinum (USA Computing Olympiad, 2025). The code-verification collections select twelve hard tasks each from Verina (Ye et al., 2025) and CLEVER (Thakur et al., 2025). The formal-mathematics collection contains the twelve problems of PutnamBench 2025 (Tsoukalas et al., 2024). The Ramsey exploration uses $R(4, 20)$ and $R(4, 21)$ construction tasks, with starting bounds from Google Research (2026) as the initial reference for the reported bounds.

B RAMSEY CONSTRUCTIONS

Section 2.1 and Table 2 report two Ramsey lower bounds. We describe the underlying colorings and explain why adding five vertices to the first construction yields the second.

A 251-vertex construction for $R(4, 20) \geq 252$. Let G be the red graph on $\mathbb{Z}/251\mathbb{Z}$, with an edge between u and v exactly when $v - u \in \pm S$, where

$$S = \{1, 2, 4, 9, 10, 13, 18, 25, 26, 33, 36, 37, 43, 45, 50, 52,\\ 65, 66, 71, 72, 74, 79, 86, 90, 93, 100, 103, 104, 107, 109, 119, 121\}.$$

All other edges are blue. This recovered construction is attributed to Yu (2026). The red graph is 64-regular, with 8,032 red edges. Translation invariance maps any monochromatic clique to one containing zero. The red neighborhood of zero contains no triangle, so G contains no red K_4. Exact maximum-clique search on the blue neighborhood of zero gives 18; therefore the full blue graph has clique number 19. This proves that the coloring contains neither a red K_4 nor a blue K_{20}.

A five-vertex extension for $R(4, 21) \geq 257$. Add vertices a, b, c_0, c_1, c_2 to G. Preserve every base edge color and define the additional red edges by

$$\{a, b\}, \qquad \{a, c_i\},\ \{b, c_i\} \quad (i = 0, 1, 2), \qquad \{c_i, v\} \quad \text{for } v \in N_G(i).$$

All other new edges are blue. Thus a and b have no red neighbors in the base graph, and the three c_i are pairwise blue; c_i copies the red base neighborhood of vertex i. The resulting 256-vertex coloring has 8,231 red edges and 24,409 blue edges, including the base and added vertices.

No new red K_4 is introduced. A red clique containing a or b cannot contain a base vertex, and among the five new vertices there is no red K_4. A clique using two c_i is not red. A red K_4 with just one c_i would require a red triangle in $N_G(i)$, contradicting the base construction.

For the blue condition, a clique containing a or b contains none of the other new vertices and has size at most $1 + 19 = 20$. A clique containing at most one c_i also has size at most 20. It remains to consider two or three c_i. For $I \subseteq \{0, 1, 2\}$, define their common blue base neighborhood as

$$B_I = \{v \in \mathbb{Z}/251\mathbb{Z} :\ v \notin N_G(i) \text{ for every } i \in I\}.$$

Exact maximum-clique search in the blue graph induced by B_I gives the values in Table 6. These cases exhaust the possible new-vertex subsets of a blue clique, so no blue K_{21} exists. A blue 19-clique of the base together with a gives a 20-clique, establishing that the extended blue graph has clique number exactly 20. The coloring therefore witnesses $R(4, 21) \geq 257$.

Table 6: **The remaining blue-clique cases stay below the forbidden size of 21.** The 256-vertex Ramsey coloring adds a, b, c_0, c_1, c_2 to a 251-vertex base whose largest blue clique has size 19. For $I \subseteq \{0, 1, 2\}$, B_I is the set of base vertices joined in blue to every c_i with $i \in I$. Rows cover cliques using two or three such new vertices. Exact clique search gives the largest blue clique in B_I; adding $|I|$ gives the last column, always at most 20. A blue clique containing a or b contains no other new vertex; all cases with at most one new vertex are also bounded by $19 + 1 = 20$. Together these bounds exclude every possible blue K_{21} in the extended coloring.

I	$\lvert B_I\rvert$	Blue clique number on B_I	Total with new vertices
$\{0, 1\}$	137	18	20
$\{0, 2\}$	136	17	19
$\{1, 2\}$	137	18	20
$\{0, 1, 2\}$	100	16	19

C Reductions and reuse in the FLT exploration

The FLT exploration in Section 2.2 illustrates why reusing a reduction can leave the main mathematical difficulty unchanged. Let F_p denote $\forall a, b, c \in \mathbb{N}_{>0},\ a^p + b^p \neq c^p$. Define

$$\begin{aligned} D:&\quad \forall n \geq 3,\ \exists d \mid n, \quad d = 4 \ \vee\ (\mathrm{Prime}(d) \wedge d \geq 3),\\ O:&\quad \forall p \geq 3, \quad \mathrm{Prime}(p) \Rightarrow F_p,\\ C:&\quad F_4 \wedge O. \end{aligned}$$

Statement D says that every exponent $n \geq 3$ has a divisor that is either four or an odd prime. If $d \mid n$, a solution at exponent n gives one at exponent d by replacing a, b, c with $a^{n/d}, b^{n/d}, c^{n/d}$. Consequently, $D \wedge C$ implies FLT for every exponent $n \geq 3$. Since Mathlib already provides the exponent-four theorem, `fermatLastTheoremFour`, proving O also suffices to establish C. These reductions concentrate the unresolved mathematics in the odd-prime statement O.

Of the 20 conditional proposals shown in Figure 1, 17 relied on this same prime-or-four core. They used two equivalent forms of the exponent-divisor lemma, so their different formulations did not remove the shared obligation. The other three proposals used either a restatement of FLT or a primitive-case reduction that still required a proof of its remaining premises.

The run also included reuse across workers. One worker retrieved another's shared candidate; a proof of D was subsequently used in the reduction $D \wedge C \Rightarrow \forall n \geq 3, F_n$, and the existing exponent-four theorem supplied the route $O \Rightarrow C$. The resulting state had proved D but left O, C, and full FLT open. Thus the workers reused mathematical work and established valid implications, while the premise carrying the remaining difficulty was still unresolved.

D Mathematical background for FLTBench

D.1 What the checkpoints ask for

The selection follows the Frey–Serre–Ribet–Wiles route documented with the public formalization (Anthropic, 2026b). Assuming an FLT counterexample, one constructs a Frey elliptic curve, proves the required representation irreducible and the curve modular, and lowers the modular level to two. This would produce a nonzero weight-two cusp form at level two, where every such form is zero. The tasks expose both intermediate interfaces and larger combinations within this argument; they do not assert that this is the only possible proof of any target.

Level I: focused constructions and interfaces. Q01 takes nonzero integers a, b, c, a prime $p \geq 5$, and $a^p + b^p = c^p$, and constructs a nonempty Frey package. This package collects the normalized arithmetic data from which the curve is defined. Its reference uses Mathlib and the supplied Frey definitions and lemmas. This makes it a useful anchor for distinguishing local proof search from larger reconstruction. Q02 turns reducibility of the Frey curve's mod-p representation into a Galois-stable line on whose quotient the action is trivial (a cofixed line). Q03 constructs a local filtration of prime-power torsion under multiplicative reduction, together with its cardinality, Galois stability,

and inertia properties. Q04 constructs a quotient by a Galois-stable subgroup of prime order, including the quotient map, its kernel, equivariance, and compatibility with multiplicative reduction away from that prime. These are useful interfaces because subsequent arguments need the properties of the constructed objects to agree, not merely the objects to exist.

Level II: assembling the irreducibility argument. Q05 is the particular Mazur Step 3 used by this formalization: under its stated torsion and reduction hypotheses, a rational prime-order point is excluded from the identity component at a further prime of multiplicative reduction. Its hypotheses include the corresponding exclusions at two and three; it is not a task to prove all of Mazur's torsion theorem. Q06 uses the large-prime argument to rule out a cofixed line for a Frey package with $p \geq 17$. Q07 establishes the required Frey representation's irreducibility for all prime exponents represented by the package, combining this argument with the remaining cases. Q03, Q04, and Q05 occur inside the reference cone for Q06; Q02 and Q06 both occur inside that for Q07. Thus this level samples both substantial internal constructions and the point at which their interfaces must fit together to establish the representation's irreducibility.

Level III: combining large theories and closing FLT. Q08 proves the Frey curve modular, through the semistable modularity argument. Q09 takes a Frey package together with modularity and irreducibility hypotheses and produces a nonzero weight-two cusp form on $\Gamma_0(2)$. The reference proof establishes modularity again while constructing its conductor-level representation; consequently its explicit modularity hypothesis is unused (Anthropic, 2026b). Q10 states FLT for positive natural numbers and every exponent $n \geq 3$. Its reference also includes the reduction to prime exponents and the final contradiction. Q10 is therefore more than a concatenation of the other nine task submissions, and no task imports results from another evaluated run.

D.2 Reading the reference profiles

Table 3 summarizes the part of the public reference associated with each target (Anthropic, 2026c). N_Q counts named theorems, including the target, with each named statement counted once. D_Q counts definition modules. H_Q is the number of theorem nodes along the longest chain, and S_Q counts theorem nodes used by at least two immediate theorem dependents in the source-import graph. Mathlib and the Lean library form the external foundation and are excluded from these counts, which describe the reference construction above that shared foundation.

The four quantities describe different aspects of a construction. For example, Q03 contains fewer theorem nodes than Q02 (66 versus 83), but its longest chain is deeper (11 versus 9). The three levels group tasks by their mathematical scope: a focused interface, assembly within irreducibility, and connections between large theories. The profiles describe one reference construction; another proof may use fewer results or organize them differently.

D.3 The supplied mathematical foundation

The solver receives Mathlib and the definitions needed to state its target, together with their permitted supporting lemmas. These supplied modules form only part of the reference vocabulary counted by D_Q. For example, Q01 receives the Frey-package definitions and eight supporting lemmas, whereas Q10 starts from Mathlib and the statement of FLT. Following the reference route for Q10 therefore requires developing the mathematical vocabulary as well as proving the intermediate results needed to connect that vocabulary to the final theorem.

The theorem statements and import relations provide a framework for choosing intermediate goals. The system reconstructs the supporting proofs needed by its chosen route and may introduce new definitions and lemmas. All ten tasks use the same principle: complete the given target from its permitted foundation. Q01–Q09 each ask for a particular theorem along the development; Q10 asks for FLT itself, with no prescribed route from its permitted mathematical foundation.

D.4 Anonymous supplementary package

The anonymous archive `FLT-Bench.zip` provides the ten task statements and Lean starter files, the supplied definition and frontier modules, and the metadata identifying the removed proof depen-

dencies. It also includes fixed dependency revisions, SHA-256 integrity manifests, and third-party attribution. These materials specify the benchmark tasks and their public mathematical context. System implementations and complete experimental trajectories are outside the scope of this archive.

The package fixes Lean 4.33.1 and Mathlib revision `db584cd6`; the full revision is recorded in `source/lake-manifest.json`. As in the experiments, Mathlib is built from source with Lean 4.33.1. Although this Mathlib revision declares Lean 4.33.0 upstream, the experiment environment uses the stated 4.33.1 toolchain. The command `lake exe cache get` must therefore be omitted: Mathlib's cache tool requires matching toolchain declarations and rejects this combination. With `elan` and Git installed, extract the archive and run:

```
cd FLT-Bench/source
lake build
lake env lean ../Q01/baseline/Q01.lean
```

The first build can take hours and require approximately 15 GB of local disk space, depending on the machine; the compressed archive is approximately 5.6 MB. This environment setup precedes the four-hour solving budget in Appendix E.2 and is not charged to the reconstruction run.

The last command loads the Q01 starter file; the same check applies to Q02–Q10. Each starter file intentionally contains one `sorry` placeholder. Loading it checks that the statement elaborates in its supplied context. A completed submission must preserve the target and its imports and pass kernel checking without `sorry`, `admit`, unsafe code, or other proof bypasses. The archive's README gives the complete build and checking instructions.

E Additional experimental details

E.1 Compared systems

The four systems organize mathematical work in different ways. PSSwarm uses decentralized workers with persistent task contexts and shared experience. Archon organizes Lean projects through planning, proving, and review. LeanMarathon maintains a shared proof blueprint and coordinates proof construction and repair. Prove2Me-Doc implements the public Prove2Me workflow of sharing statements, decomposing goals, and reusing results across workers pursuing parts of a common proof (Frenzymath, 2026; Zhang et al., 2026; Prove2Me, 2026).

The systems retain their native organization of work with shared agent settings, time budget, concurrency limit, and set of mathematical tasks for every independent reconstruction run.

E.2 Common experimental settings

Table 7 summarizes the settings shared by all four systems. Every run starts afresh from the target statement and its permitted foundation. Results from other evaluated tasks are not transferred between runs. The four-hour budget runs from the first solving attempt to the end of the run, including checking and revision. Workers and verifiers use CPU machines in standardized Docker environments for proof construction as well as the checking of submitted Lean code.

E.3 Local contributions and their use in a common proof

The cases in Table 5 follow proof text, dependencies, checking outcomes, and the reports available to later workers. The Prove2Me-Doc Q03 and Archon Q02 cases come from their third retained runs; the PSSwarm Q04 case comes from the retained candidate history.

Checked reuse on Q03. One agent proves that the zero-component condition is preserved by the decomposition-group action. A second agent calls this lemma to prove stability of a subgroup assumed to have the required torsion characterization and exact cardinality. Both verifier jobs accept with Lean's standard axioms, and the second submission records its dependency on the first. The target also requires constructing the subgroup and proving the inertia condition. The shared continuation report names these outstanding obligations, so a later worker can apply the stability theorem once its premises are supplied by the remaining construction work.

Table 7: **Shared agent settings, tasks, and run budgets support the four-system comparison.** PSSWARM, Archon, LeanMarathon, and Prove2Me-Doc each receive three fresh runs on all ten FLTBench targets, giving 120 runs. Each run starts from the target statement and permitted mathematical foundation; solved checkpoints are not transferred between runs. The four-hour limit and cap of 16 concurrent agents apply per run. Success requires a complete Lean proof from that foundation within the budget, with all supporting obligations discharged.

Setting	Value
Agent configuration	`gpt-6-astra`
Reasoning setting	`max`
Time budget per run	4 hours
Maximum concurrent agents per run	16
Tasks	Q01–Q10
Runs per system–task pair	3
Total runs	$4 \times 10 \times 3 = 120$

A construction to continue on Q04. The candidate writes the desired map as $\psi \circ \pi$, where $\pi : G \to G/L$ is the abstract quotient map. Its proposed ψ must realize the quotient injectively on an integral curve, preserve the Galois action, and satisfy the required local reduction correspondence. The script records this realization as an open claim. A successor explores direct theorem calls and library search, then returns the same candidate and its stated gap.

A completion report corrected on Q02. The prover tries to infer $b = 0$ over the integers from $b \equiv 0 \pmod 2$. Lean rejects the argument, while the prover reports that the target and its check are complete. The reviewer checks the file, identifies the invalid cast, and marks the target as blocked. The next plan rejects the same route and returns to the cofixed-line obligations. The correction concerns the shared completion status; the target still requires a valid proof.

E.4 COMPLETION ASSESSMENT EXPERIMENT

The observer agent uses `gpt-6-astra` with `max` reasoning. It reads each record without tools, judges whether the presented evidence establishes the exact target, and cites the material supporting its answer. It may answer complete, incomplete, or abstain when the evidence is ambiguous. We score one final assessment per record; all 100 return a complete or incomplete judgment.

Records and assigned status. We pair reference-source candidate material with checking reports written for the ten conditions in Figure 3. A record labelled complete retains an identified, nonempty candidate and a matching report of acceptance, including when its proof excerpt is omitted. The conflicting condition supplies an empty candidate with a success report. The other incomplete conditions provide a supporting lemma, a queued check, a mismatched target, or a wrong Lean environment. Labels specify the status represented by each construction. Figure 4 aggregates one assessment for each of five complete and five incomplete conditions per target. The repeated complete condition changes only the record identifiers. The checking reports and labels are constructed for this diagnostic, not obtained by replaying the candidates in Lean.

Target and candidate presentation. Every record states the requested target separately. Q01 and Q02 include the reference target declaration in the candidate excerpt; Q03–Q10 provide surrounding source material. Reference declarations and benchmark targets retain their respective names. The observer is asked to link the candidate reference, its presented material, and the checking report to the requested target. The records retain an earlier foundation context and are not snapshots of the reconstruction runs in Section 5.2; they form a separate diagnostic of completion assessment.