CombStruct4Lean: A Formal Combinatorial Benchmark Emphasizing Structures for Automated Theorem Proving

Long Doan
Department of Computer Science
George Mason University
ldoan5@gmu.edu

ThanhVu Nguyen
Department of Computer Science
George Mason University
tvn@gmu.edu

Abstract

Formal theorem proving with large language models (LLMs) has demonstrated promising results, yet combinatorial problems remain a notable challenge due to their reliance on intricate, problem-specific structures and definitions. AlphaProof, a notable LLM-based system for automated theorem proving, has shown strong performance in the International Mathematical Olympiad (IMO), obtaining a silver-medalist performance by solving all questions but two combinatorics problems. Existing formal benchmarks have limited combinatorial coverage and often overlook the importance of combinatorial constructions. To address these gaps, we introduce CombStruct4Lean, a novel benchmark composed of 383 combinatorial math-word problems formalized in the Lean4 proof assistant. CombStruct4Lean emphasizes the creation and usage of combinatorial structures, presenting significantly greater complexity and diversity than existing datasets. We evaluate state-of-the-art autoformalization and neural theorem proving methods on our benchmark, revealing substantial room for improvement. Our findings highlight both the difficulties inherent in formalizing combinatorial problems and the need for further research in structured combinatorial reasoning within automated theorem proving.

1 Introduction

Large language models (LLMs) have recently shown remarkable progress in formal theorem proving, achieving strong results on challenging mathematical tasks. Notably, AlphaProof [1] and AlphaGeometry2 [2] obtained a silver-medalist performance at the International Mathematical Olympiad (IMO) 2024 competition. However, both systems failed on two combinatorics problems, which highlight challenges and limitations of LLMs on this domain.

Combinatorics is a branch of mathematics that focuses on reasoning over discrete structures such as graphs, partitions, and permutations with specific constraints, which often require problem-specific definitions and constructions that are difficult to formalize [3]. More broadly, formal theorem proving involves two core tasks: autoformalization—translating a natural language problem into a formal statement—and automated theorem proving—finding a formal proof from that statement. In both cases, the output must be verified by proof assistants like Coq [4], Isabelle [5], or Lean4 [6].

While there have been multiple works tackled on both tasks in both general mathematical domain [7–10] and specific branches [11–13, 2], only a few focused on combinatorics [14, 15]. A significant factor contributing to this limitation is the current state of formal benchmarks, which offer limited coverage of combinatorics. For instances, miniF2F [3], ProofNet [16], and FIMO [17] contain

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Table 1: Comparison of combinatorics problems across different benchmarks.

Benchmark	#. combinatorics	Problem Type	Custom Definitions
miniF2F-test [3]	0 (0.0%)	-	-
ProofNet [16]	0 (0.0%)	-	-
PutnamBench [18]	29 (4.4%)	Math-word problem	-
LeanComb-test [15]	100 (100%)	Combinatorial identities	✓(shared)
CombStruct4Lean (ours)	383 (100%)	Math-word problem	✓

no combinatorial problems, and PutnamBench [18] includes only a small fraction (29 out of 657) dedicated to this area.

Furthermore, these benchmarks often overlook the aspect of combinatorial constructions. In the formalization of IMO 2024 Problem 5 [19], one of two problems that AlphaProof failed, over 20% of the formalization was focused on defining specific combinatorial objects and structures, with a substantial portion of the remaining code consisting of lemmas directly related to these new constructs. Despite the importance, none of the mentioned benchmarks included any problem-specific definitions dedicated to combinatorics. LeanComb [15], a recent formal benchmark on combinatorics, only focused on combinatorial identities with pre-defined constructions, which limits its ability to evaluate a model's capacity to invent or define new combinatorial structures.

To address this gap, we introduce CombStruct4Lean, a benchmark of formal combinatorial problems with an emphasis on combinatorial structures. CombStruct4Lean consists of 383 combinatorial math-word problems, sourced from high-school olympiad-level competitions and formalized in the Lean4 proof assistant. Our benchmark creation process incorporates a LLM-based feedback-driven formalization pipeline that iteratively refines the formal statement by analyzing compilation failures and retrieving relevant premises. Unlike prior methods that rely on a single-pass generation [7, 10, 16], this process enables the model to define and adapt problem-specific structures, which are essential for combinatorial problems. To ensure quality, we incorporate a two-stage semantic checking strategy that checks the consistency between the informal problem and its formal counterpart, followed by manual review by human experts. We illustrate the differences between our CombStruct4Lean and existing benchmarks in Tab. 1. Our analysis shows CombStruct4Lean posess a significantly higher diversity in terms of formalization length and the number of custom definitions required than other widely used formal benchmarks. Through experiments on both autoformalization and automated theorem proving tasks, we demonstrate limitations of current models on our benchmark.

Contributions This paper makes the following contributions: (i) We introduce CombStruct4Lean, a benchmark consisting of 383 formalized combinatorial problems sourced from high-school Olympiad-level competitions. A key feature of our benchmark is its strong emphasis on the creation and utilization of combinatorial structures. (ii) We describe our benchmark construction process, including the iterative formalization pipeline and the semantic checking strategy, along with an analysis highlighting how CombStruct4Lean differs from existing benchmarks. (iii) We evaluate state-of-the-art autoformalization and automated theorem proving methods to demonstrate the complexity and difficulty inherent in CombStruct4Lean, which make our benchmark a suitable testbed for future research on formal combinatorics.

2 Related Work

2.1 Autoformalization

Early LLM-based explorations in autoformalization task adopted in-context learning methods [7] and later incorporated techniques such as back-translation to enrich training sets [16, 20]. More recent work began tackling other aspects of autoformalization, such as fidelity and correctness. RAutoformalizer [21] introduced premise retrieval to ground generated formalization with premises information. Process-Driven Autoformalization [20] included Lean4 compiler's traceback information to verify the quality of a formalization. While both methods focused on checking the correctness of a formal statement, AutoForm4Lean [14] leveraged LLMs to evaluate the formal code based on multiple criteria, whereas Li et al. [22] proposed two self-consistency approaches: symbolic equivalence and semantic equivalence. However, their symbolic approach is primarily designed

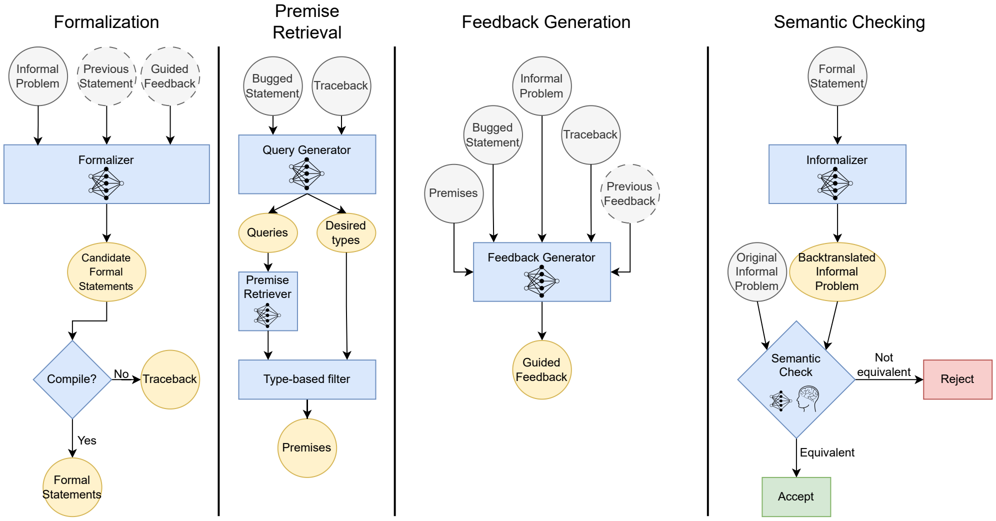


Figure 1: Illustrations of each step in benchmark creation.

for problems involving numerical expressions, making its extension to the combinatorics domain non-trivial.

2.2 Combinatorics in Formal Benchmarks

Current formalization benchmarks, including MiniF2F [3], ProofNet [16], and FIMO [17], largely focus on foundational areas such as algebra, number theory, and analysis, with minimal coverage of combinatorics. For example, MiniF2F, ProofNet and FIMO have no combinatorial problems, while only 29 out of 657 instances in PutnamBench [18] are in combinatorics domain. This underrepresentation occurs because combinatorial problems often require intricate, problem-specific definitions and constructions, making them particularly challenging to formalize [3].

Recent research, such as AutoForm4Lean [14] and LeanComb [15], aim to address this by introducing methods that can synthesize new combinatorial benchmarks. AutoForm4Lean proposed a dataset construction pipeline focused on both syntactically and semantically correctness of the formalization. LeanComb developed a data augmentation approach that can automatically generate new theorems from a complete formal proof and introduced a benchmark dedicated to combinatorial identities. However, these combinatorial identities can be solved by applying algebraic techniques without consideration of combinatorial reasoning or combinatorial structures.

3 Benchmark Creation

In this section, we detail the construction process of CombStruct4Lean, which consists of 383 competition-level combinatorial problems formalized in Lean4 proof assistant [6]. To our knowledge, CombStruct4Lean is the first benchmark dedicated on formalizing math-word problems in combinatorics domain with a focus on problem-specific combinatorial structures.

3.1 Informal Problem Sources

To ensure the quality of CombStruct4Lean, we select informal combinatorial problems from high-school olympiad-level competitions [23]. We avoid problems that require computing a solution (e.g., how many arrangements satisfy a certain constraints) and choose only problems that requires to prove a statement. At the end of this process, we obtained 8608 combinatorial problems. Among those, we randomly sample 1000 problems to create the benchmark.

Table 2: Notations.

Symb	Meaning
p	Informal problem
s	Formal statement candidate
f	Guided feedback
t	Traceback from Lean
P	Retrieved premises

Algorithm 1: Formalization Process

```
Input: Informal combinatorics problem $p$
Output: A statement $s$ if formalization succeeds
$f \leftarrow \emptyset$, $s \leftarrow \emptyset$;
while not terminated do
    $s \leftarrow$ FORMALIZATION$(p, s, f)$;
    $t \leftarrow$ COMPILE$(s)$;
    if $t = \emptyset$ then return $s$;
    $P \leftarrow$ RETRIEVEPREMISE$(s, t)$;
    $f \leftarrow$ GENERATEFEEDBACK$(p, s, t, f, P)$
return $\emptyset$;
```

3.2 Formalization Process

To improve clarity, we introduce the following notations that will be used consistently throughout the benchmark creation pipeline in Tab. 2. The formalization process for each informal combinatorial problem is described in Algorithm 1. Here, we discuss each of the step included in this process, namely *formalization*, *premise retrieval*, *feedback generation*. Fig. 1 also provides an overview on how each step work.

Formalization. Given an informal combinatorics problem p, the formalization step uses the previous formal attempt s and guided feedback f to produce a new candidate statement. This process is handled by the Formalizer module. In the first iteration, both s and f is empty. The Formalizer first determines whether additional definitions or structures are needed and generates them accordingly, then constructs a formal statement based on these elements. In our implementation, the formalization step is performed using `Claude-3.5-Sonnet` API as the LLM. We generate a single candidate formal statement in each iteration with a temperature of 0.3.

If the generated statement s compiles successfully, it proceeds to semantic checking (Sec. 3.3). Otherwise, the compiler returns a traceback t, which is used in the next phase, *premise retrieval*. Fig. 2 provides an example input to the Formalizer, including the problem p, the previous s, and guided feedback f.

Premise Retrieval. A common errors we found during the formalization is the incorrect usages of existing premises, which can be occurred because of the lack of grounding between the LLM and the Mathlib library. To address this, we use a retrieval-augmented generation approach via the premise retriever module, which provide the LLM's knowledge with relevant premises documentation. Although previous work has applied RAG to the autoformalization task [21], none has used it explicitly to correct buggy formal statements.

Given the bugged statement s and traceback t, the query generator produces queries q and associated desired types $T(q)$. Each query q and Mathlib premise p_i are encoded using Dense Passage Retrieval [24], and their cosine similarity is computed as:

$$\text{sim}(q, p_i) = \frac{f(q) \cdot f(p_i)}{\|f(q)\| \cdot \|f(p_i)\|}$$

We select the top-k entries from the corpus of Mathlib premises with the highest similarity and match with the desired types:

$$P = \{p_i \mid \text{sim}(q, p_i) \text{ among top-}k \ \wedge \ T(q) = T(p_i)\}$$

We forward these retrieved premises P to the next phase for generating feedback. See Fig. 3 for a detailed example. To generate the queries and desired types for each query, we use `Claude-3.5-Haiku` as the LLM with a temperature of 0.3. We use `CodeRankEmbed` [25] to embed the query and signatures of each premise in the Mathlib library. For each pair of query and expected type, the retriever returns at most $k = 5$ relevant premises, though there is no restriction on the number of queries or types that can be generated.

Translate the following problem into a formal theorem in Lean4:

```
/--
Consider a graph where each vertex is connected to exactly three other
    vertices (a 3-regular graph). Prove that it is possible to color the
    edges of such a graph using only three colors in such a way that no two
    adjacent edges share the same color if the graph is bipartite.
-/
```

Previous formalization attempt:

```
structure ThreeRegularGraph {V : Type} where
  ...
def EdgeColoring {V : Type} (G : ThreeRegularGraph V)
  ...
def IsValidColoring {V : Type} (G : ThreeRegularGraph V) (c : EdgeColoring V
    G)
  ...
def IsBipartite {V : Type} (G : ThreeRegularGraph V)
  ...
theorem three_regular_edge_coloring {V : Type} (G : ThreeRegularGraph V) :
  (∃ c : EdgeColoring V G, IsValidColoring G c) → IsBipartite G :=
  sorry
```

Guided feedback from expert:

- The `three_regular` field in ‘ThreeRegularGraph‘ has multiple problems ...
- The `EdgeColoring` definition has type mismatches ...
- `IsBipartite` definition could be improved ...

Figure 2: Example of an input prompt to the Formalizer LLM. Detailed feedback and implementations are abbreviated for brevity.

Feedback Generation. The feedback generation module produces guided feedback f using the original problem p, the current formalization s, the traceback t, retrieved premises P, and prior feedback. This guided feedback helps refine s in the next iteration by (i) diagnosing root causes of compilation failure using t; (ii) analyzing whether custom definitions align with p; (iii) demonstrating correct use of retrieved premises P with a code snippet.

We provide an example of a generated feedback in Fig. 2. The feedback f is reused in the next call to *formalization* step, continuing the iterative refinement loop. We use `Claude-3.5-Sonnet` to generate the feedback, producing one feedback candidate per iteration with a temperature of 0.7.

For each informal problem, we perform the formalization process for a maximum of 5 iterations. At the end, we obtained 634 examples that compile successfully. Among those, we removed 127 examples that contain placeholder `sorry` in their definitions, resulted in 507 examples to perform semantic checking.

3.3 Semantic Checking

Successfully compiling a formal statement does not guarantee its semantic correctness. We provide an example of this issue in Fig. 4. In the first formalization, the code includes an implementation for valid subsets `valid_subsets` and compares the output of the function f with the number of valid subsets returned by `valid_subsets`. This formalization faithfully captures all mathematical objects described in the informal problem, making it semantically correct. In contrast, the second formalization lacks an implementation for valid subsets and does not reference any set S in the theorem's statement. Although it compiles, it fails to reflect the structure of the original problem and is thus semantically incorrect.

To verify the correctness of a formal statement, we adopt a two-stage semantic checking strategy. Similar to semantic equivalence [22], we first informalize the formal statement, then compare the back-

Bugged Statement:

```
structure ThreeRegularGraph (V : Type) where
  three_regular : ∀ v : V, (({w | (v, w) ∈ edges}).card = 3)
  ...
```

Traceback:

```
Text: (({w | (v, w) ∈ edges}).card
Error: invalid field ‘card’, the environment does not contain ’Set.card’
  {w | (v, w) ∈ edges} has type Set V
```

Step 1: Query Generation
Queries: [“Set.card”, “Set to finset”]
Desired types: [“Set”, “null”]
Step 2: Premise Retrieval
Premises related to “Set.card”:

- `def card : Cardinal`
- `def card (α) [Fintype α] : Nat`

Premises related to “Set to finset”:

- `def toFinset (s : Set α) [Fintype s] : Finset α`
- `def toFinset (s : Multiset α) : Finset α`

Step 3: Type-Based Filtering

- “Set.card”: Matching premises: N/A
- “Set to finset”: Matching premises:
  `def toFinset (s : Set α) [Fintype s] : Finset α`
  `def toFinset (s : Multiset α) : Finset α`

Figure 3: Example of Premise Retrieval step

translated version with original informal problem. However, instead of computing cosine similarity between their sentence embeddings, we leverage LLMs to assess their semantic alignment based on multiple criteria: combinatorial objects and structures, constraints, goals, scope, and equivalence (i.e., can we restate one version by using the other?). This approach also shares similarities with AutoForm4Lean [14], which uses LLMs to compare formal and informal representations. However, our method avoids the challenge of cross-modality comparison by evaluating two informal statements, thereby reducing the complexity introduced by differences in syntax and representation between code and natural language. The entire semantic checking process is supervised by a human experts. We use `Claude-3.5-Haiku` with a temperature of 1.0 for both informalization and semantic checking. At the end of this stage, we obtained 383 examples for CombStruct4Lean.

3.4 Benchmark Analysis

As mentioned in Sec. 1, a key challenge in formalizing combinatorial problems is the need to define new concepts to support the main theorem. We analyze this issue in our benchmark using two aspects: formalization length in Fig. 5a and number of definitions created in Fig. 5b. For formalization length, we remove all comments and the header block (e.g, `import`, `open`) and count only the code related to the theorem statement. For number of definitions, we count code blocks beginning with one of the following keywords `def`, `structure`, `class`, `inductive`, `coinductive`, `abbrev`, `instance`, `mutual`, `constant`, `axiom`.

From the figures, we observe that CombStruct4Lean is much more diverse than existing benchmarks in both of formalization length and number of custom definitions support each theorem. Over 85% of problems in miniF2F and PutnamBench require fewer than 10 lines of code, whereas CombStruct4Lean shows a broader distribution, with many examples exceeding this range. Notably, only 14% of CombStruct4Lean problems fall within the 0–9 LoC range. A similar pattern

```
/--
From a set S of n elements, prove that there are f(n,m,k) ways to select a
    subset s of k elements such that m < k elements cannot be together in s.
-/
def f (n m k : Nat) : Nat :=
  ...
-- First approach, semantically correct
def containsForbidden (m : Nat) (s : Finset a) : Prop :=
  ...
def valid_subsets (S : Finset a) (k m : Nat) : Finset (Finset a) :=
  ...
theorem count_valid_subsets_general
  (S : Finset a) (n m k : Nat) (hS : S.card = n) :
  (valid_subsets S k m).card = f n m k := by sorry
-- Second approach, semantically incorrect
def subsetsWithConflicts (n m k : Nat) : Nat :=
  ...
theorem subsetsWithConflicts_eq (n k m : Nat) :
  subsetsWithConflicts n k m = f n m k := by sorry
```

Figure 4: Example of formal statements that are semantically correct and incorrect. Implementations of definitions are abbreviated for brevity.

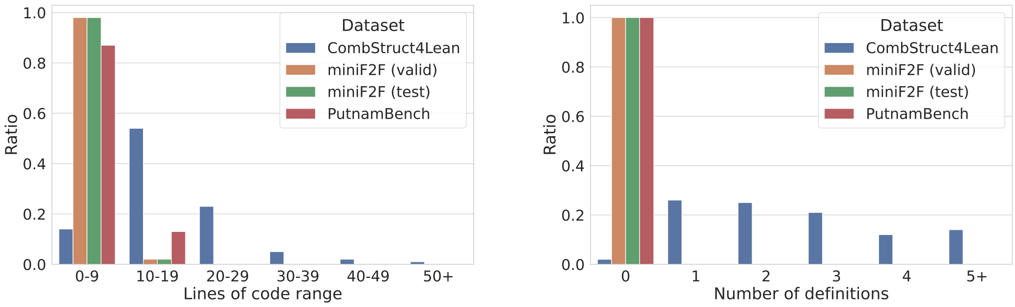
appears in the number of custom definitions: all problems in miniF2F and PutnamBench define no new concepts, while only 2% of examples in CombStruct4Lean exhibit similar behaviors. The rest of CombStruct4Lean require varying numbers of new definitions, with the highest count of definitions reaching 12 in one example. These findings highlight the diversity and complexity of CombStruct4Lean, presenting a realistic and challenging setting for autoformalization and formal reasoning in combinatorics.

4 Evaluation

We evaluate CombStruct4Lean using two tasks: autoformalization and automated theorem proving. Given the inherent difficulty of combinatorics, even in formalizing statements, we assess how well current models perform the autoformalizing task by using our benchmark as a reference. To demonstrate the significant challenge of the benchmark, we also perform evaluation on the automated theorem proving task with different theorem provers. We conduct all experiments on a machine with 2 A100 80GB GPUs, 32-cores AMD CPU and 100GB of RAM. We public our benchmark and the code for experiments at `https://github.com/dynaroars/CombStruct4Lean`.

4.1 Autoformalization

Experiment Setting. Due to the lack of existing models for autoformalization, we use Goedel-Formalizer [10] as the main model for this experiment. We consider two versions of the model, one of them is trained on the open Numina dataset [23] annotated with the CLAUDE-3.5-SONNET API, denoted as SONNETANNOTATED, and the other is finetuned on LeanWorkbook [26] dataset, denoted as LEANWORKBOOKANNOTATED. For both models, we sample $n \in \{1, 16\}$ formal statement candidates with a temperature of 1.0. We use two metrics for evaluation, one is `#.Compiled` indicating the number of examples with at least one candidate successfully compile, and the other is `Ground-truth Alignment` where we use another LLM to check whether each candidate align with the ground truth formalization and the informal problem, and count the number of examples that have an aligned candidate. The alignment is judged based on two criteria: (i) Mathematical equivalence–Does the candidate aim to formalize the same mathematical problem as the ground truth and the informal problem? (ii) Fidelity–Does the candidate accurately represents the mathematical entities, scope, assumptions, and definitions of the informal problem, using the ground truth as a reference? We use `Claude-3.5-Haiku` with a temperature of 1.0 as the evaluator LLM here. Due to



(a) Distribution of formalization lengths.

(b) Distribution of number of custom definitions.

Figure 5: Benchmark Analysis.

the differences in Lean4 environments, we do not include RAutoformalizer model and BEq metric [21] despite being a closely related work.

We also conduct a small ablation study on our formalization process (Sec. 3.2) by removing either the *premise retrieval* step (denoted as No Premise) or the *feedback generation* step (denoted as No Guided Feedback) to understand their impact on performance. In particular, in the No Premise setting, we exclude retrieved premises from the inputs to *guided feedback* step. In the No Guided Feedback setting, we directly use the traceback as feedback for the Formalizer LLM. In both settings, we sample a single candidate with a temperature of 0.3.

Results. Tab. 3 presents our experiment results. We observe that the SonnetAnnotated model obtained a substantial higher compiled rate than the LeanWorkbookAnnotated model: 293 vs. 11 when sampling 1 candidate, and 383 vs. 112 when sampling 16 candidates. However, note that the SonnetAnnotated model share the same data source and the same foundation LLM as our benchmark, which may introduce some potential data leakage. Nonetheless, despite a high compiled rate, the alignment with the ground-truth formal statement remains relatively low for SonnetAnnotated, with only 106/293 and 311/383 candidates aligned for $n = 1$ and $n = 16$ settings, respectively. This further reinforce our argument in Sec 3.3 that successful compilation does not necessarily indicate semantic correctness.

Interestingly, the No Premise model obtained a higher compiled rate than the No Guided Feedback model (347 vs. 285). Upon closer inspection, we found that when given with only the compiler traceback, the Formalizer LLM tends to iteratively refine its original formalization to make the code compile. In contrast, the guided feedback in No Premise setting is more likely to suggest radical changes to the formalization, many of which fail to compile due to the lack of grounding with premise documentation from the Mathlib library. However, when considering the percentage of compiled candidates that are semantically aligned with the ground truth formalization, the No Premise model outperforms No Guided Feedback, with 76.8% vs. 71.1% alignment, highlighting the importance of guided feedback in improving semantic correctness. A similar trend is observed when comparing No Premise with SonnetAnnotated ($n = 1$): although their compiled rates are similar, No Premise yields more than twice the number of aligned candidates, further demonstrating the impact of guided feedback on formalization quality. These findings show the importance of guided feedback and premise retrieval in producing both syntactically and semantically correct formal statements.

4.2 Automated Theorem Proving

Experiment Setting. We evaluate different theorem provers on our CombStruct4Lean with two types of models, specialized LLMs finetuned on the automated theorem proving task and general-purpose LLMs. For specialized LLMs, we choose DeepSeek-Prover-V1.5 [9] and Goedel-Prover [10]. For general-purpose LLMs, we perform evaluation on standard model Claude-3.5-Sonnet and reasoning model o4-mini. We follow evaluation in ProofNet [16] and use `Pass@`K as the evaluation metric, with $K \in \{1, 5, 10\}$. Considering the computational cost, we adopt the whole-proof generation approach for all theorem provers. Specifically, we sample K candidate proofs,

Table 3: Evaluation on Autoformalizing task.

Models	#. Compiled	Ground-truth Alignment
Goedel-Formalizer		
- SonnetAnnotated ($n = 1$)	293	106
- SonnetAnnotated ($n = 16$)	383	311
- LeanWorkbookAnnotated ($n = 1$)	11	1
- LeanWorkbookAnnotated ($n = 16$)	112	8
Claude-3.5-Sonnet		
- No Premise	285	219
- No Guided Feedback	347	247

Table 4: Evaluation on Automated Theorem Proving task.

Models	Pass@1	Pass@5	Pass@10
Deepseek-Prover-V1.5	0	0	0
Goedel-Prover	0	0	0
Claude-3.5-Sonnet	3	7	10
o4-mini	0	0	0

remove all candidates that violates the integrity of the original formal statement and candidates with placeholder proof (i.e., `sorry`), then check whether each proof compile or not.

Results. Tab. 4 presents the results. We observe that except for CLAUDE-3.5-SONNET, all theorem provers failed to solve any problems in CombStruct4Lean. For two specialized LLMs, this weak performance may be due to the limited exposure to combinatorics during their finetuning. However, in the case of the reasoning model O4-MINI, we found that a majority of responses are either stopped due to excessively long reasoning chains or the model decide that the problem is not solvable. Interestingly, in multiple examples, O4-MINI just flat out decline to generate an answer. In contrast, CLAUDE-3.5-SONNET was able to solve 10 problems in our benchmark. Upon closer inspection, we found that CLAUDE-3.5-SONNET has a tendency of placing a placeholder `sorry` for tactics that creating new subgoals, such as `have` or `by_cases`. If we include such proofs for evaluation, the number of problems solved by Claude increased to 75, or 19.6% of the benchmark, suggesting potential research directions involving subgoal based approaches.

5 Conclusion

We introduced CombStruct4Lean, a benchmark of combinatorial problems formalized in Lean4 proof assistant with a focus on combinatorial structures. We detailed the benchmark construction process, including an iterative formalization pipeline and semantic checking strategy, and showed how CombStruct4Lean differs from existing benchmarks. Our experiments reveal that current autoformalization and theorem proving methods struggle significantly on CombStruct4Lean, especially on the theorem proving task. These results demonstrate the complexity and difficulty inherent in CombStruct4Lean, which make our benchmark a suitable testbed for future research on formal combinatorics.

Broader Impacts CombStruct4Lean improves research in formal reasoning by providing a structured, challenging suite of formalized combinatorial problems. As combinatorial reasoning plays a vital role in areas such as cryptography, algorithm design, and network theory, our work may also support the broader adoption of formal methods in safety-critical and high-assurance domains.

Limitations CombStruct4Lean is built specifically for Lean4, which restricts the applicability to other proof assistants like Coq or Isabelle. Our benchmark is also developed on specific Lean4 and Mathlib versions, giving concerns in incompatibilities unless actively maintained. Finally, our work focuses on formalizing problem statements only and omit informal proofs, which allow us to isolate and evaluate the challenges of translating informal problem into formal statements.

References

[1] AlphaProof and AlphaGeometry teams. Ai achieves silver-medal standard solving international mathematical olympiad problems. `https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/`, 2024.

[2] Yuri Chervonyi, Trieu H Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Vikas Verma, Quoc V Le, and Thang Luong. Gold-medalist performance in solving olympiad geometry with alphageometry2. *arXiv preprint arXiv:2502.03544*, 2025.

[3] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

[4] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, et al. The coq proof assistant reference manual. *INRIA, version*, 6(11), 1999.

[5] Tobias Nipkow, Markus Wenzel, and Lawrence C Paulson. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer, 2002.

[6] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pages 625–635. Springer, 2021.

[7] Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

[8] Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=4WnqRR915j`.

[9] Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024.

[10] Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, et al. Goedel-prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025.

[11] Jing Xiong, Jianhao Shen, Ye Yuan, Haiming Wang, Yichun Yin, Zhengying Liu, Lin Li, Zhijiang Guo, Qingxing Cao, Yinya Huang, Chuanyang Zheng, Xiaodan Liang, Ming Zhang, and Qun Liu. TRIGO: Benchmarking formal mathematical proof reduction for generative language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL `https://openreview.net/forum?id=ILQnct9H4H`.

[12] Chenrui Wei, Mengzhou Sun, and Wei Wang. Proving olympiad algebraic inequalities without human demonstrations. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL `https://openreview.net/forum?id=8kFctyli9H`.

[13] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[14] Long Doan and ThanhVu Nguyen. AI-Assisted autoformalization of combinatorics problems in proof assistants. *45th International Conference on Software Engineering: New Ideas and Emerging Results*, 2025. URL `https://conf.researchr.org/details/icse-2025/icse-2025-nier/11/AI-Assisted-Autoformalization-of-Combinatorics-Problems-in-Proof-Assistants`.

[15] Beibei Xiong, Hangyu Lv, Haojia Shan, Jianlin Wang, Zhengfeng Yang, and Lihong Zhi. A combinatorial identities benchmark for theorem proving via automated theorem generation. *arXiv preprint arXiv:2502.17840*, 2025.

[16] Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

[17] Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, et al. Fimo: A challenge formal dataset for automated theorem proving. *arXiv preprint arXiv:2309.04295*, 2023.

[18] 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 *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL `https://openreview.net/forum?id=ChKCF75Ocd`.

[19] IMO 2024 P5 formalization. `https://github.com/leanprover-community/mathlib4/blob/master/Archive/Imo/Imo2024Q5.lean`. Accessed: 2025-05-11.

[20] Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, et al. Process-driven autoformalization in lean 4. *arXiv preprint arXiv:2406.01940*, 2024.

[21] Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. Rethinking and improving autoformalization: towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=hUb2At2DsQ`.

[22] Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Xian Zhang, Fan Yang, and Xiaoxing Ma. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=8ihVBYpMV4`.

[23] Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. `[https://huggingface.co/AI-MO/NuminaMath-1.5](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf)`, 2024.

[24] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

[25] Tarun Suresh, Revanth Gangi Reddy, Yifei Xu, Zach Nussbaum, Andriy Mulyar, Brandon Duderstadt, and Heng Ji. Cornstack: High-quality contrastive data for better code ranking. *arXiv preprint arXiv:2412.01007*, 2024.

[26] Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *arXiv preprint arXiv:2406.03847*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our first claim is that CombStruct4Lean is the first formal combinatorial benchmark focused on combinatorial structures, which is demonstrated through the benchmark analysis. Our second claim, state-of-the-art autoformlization and automated theorem proving models perform poorly on CombStruct4Lean, which is shown through the experiment results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations of our work in Sec. 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provided step-by-step how to create our dataset (Sec. 3) and settings used in our experiments (Sec. 4).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provided the the code for creating the dataset (Sec. 3) and experiments (Sec. 4), along with outputs of each experiments.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We detailed our experiment settings in Sec. 4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Given the exploratory nature of our benchmark and the large performance gaps observed, we focus on qualitative evaluation rather than statistical significance.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We detailed resources used in both of our experiments in Sec. 4.1 and Sec. 4.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have acknowledged the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed the impacts of our paper in Sec. 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The topic of our paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Our paper uses Numina dataset [23] that is released under the Apache License, Version 2.0. We have properly cited the original dataset in the manuscript and included the license information as required. The terms of use and redistribution are explicitly respected in accordance with the license.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We publicly release our benchmark at https://github.com/dynaroars/CombStruct4Lean with documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.