Protein Design with Agent Rosetta: A Case Study for Specialized Scientific Agents

Jacopo Teneggi[1] **Tanya Marwah**[1,2] **Alberto Bietti**[1,3] **P. Douglas Renfrew**[4]
Vikram Khipple Mulligan[4] **Siavash Golkar**[1,5]
[1]Polymathic AI
[2]Simons Foundation
[3]Center for Computational Mathematics, Flatiron Institute
[4]Center for Computational Biology, Flatiron Institute
[5]New York University

Abstract

Large language models (LLMs) are increasingly capable of emulating reasoning and using tools, creating opportunities for autonomous agents that execute complex scientific tasks. Protein design provides a natural case study: existing deep learning models achieve strong results, but they are typically restricted to canonical amino acids and narrow objectives, leaving space for a generalist tool for broad design pipelines. We introduce *Agent Rosetta*, an LLM agent built on top of the Rosetta suite—the leading physics-based software for heteropolymer design, capable of modeling non-canonical building blocks and geometries. Agent Rosetta is a single-agent, multi-turn framework that iteratively refines heteropolymers to achieve the goals of a user-defined task brief, combining the biophysical knowledge of modern LLMs with the accuracy of Rosetta's physics-based methods. In evaluations, Agent Rosetta achieves performance comparable to specialized deep learning models, especially when combined with inference-time techniques such as *best-of-n* sampling. Interestingly, we find that prompt engineering alone is insufficient for reliably producing RosettaScripts actions. This underscores the need for building a comprehensive environment that, for example, simplifies the most challenging aspects of RosettaScripts syntax. These results demonstrate that combining frontier LLMs with established domain-specific scientific tools can yield flexible agentic frameworks that not only lower barriers to use but also achieve performance competitive with specialized deep learning models.

1 Introduction

Autonomous agents built on large language models (LLMs) are becoming increasingly capable of executing complex, multi-turn tasks that demand the emulation of reasoning and the capability to use tools [100, 101]. A key strength of these agents lies in their ability to write, debug, and execute code [21, 39, 89], which makes them well-suited for aiding the automation of scientific discovery workflows that rely on code-based interfaces, spanning chemistry [7, 15], mathematics [92], physics [5, 63], and biology [33, 46, 83]. By combining sustained reasoning with iterative feedback, agents offer new ways to address scientific challenges where machine learning methods remain limited.

Protein design—and, more generally, heteropolymer design—is a central challenge in science, with broad implications for developing nanomaterials [35, 45, 52, 53], medically- or industrially-relevant enzymes [36, 49, 82, 95], and drugs [43, 74]. While machine learning-based (ML) protein modeling methods, such as AlphaFold [51], RFdiffusion [110], and ProteinMPNN [26], have advanced the

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Science: The Reach and Limits of AI for Scientific Discovery.

field, they are often restricted to the 20 canonical amino acids, specialized to particular pipelines, and dependent on large training datasets. Even inversion of a chiral center results in poor ML performance, for want of training data for non-canonical D-amino acid peptides [22]. In contrast, the Rosetta Macromolecular Modeling Suite [58], built on a largely physics-based energy function [6], requires minimal training data and can accurately model proteins containing non-canonical building blocks that have not been observed in experimentally-determined structures [10, 25, 29, 67, 73, 85, 86, 109]. This dramatically expands the accessible design space and has enabled advances in, for example, drug development [11, 42, 43, 69, 74]. Yet effective use of Rosetta demands not only deep biophysical expertise and coding proficiency, but also familiarity with its idiosyncratic input formats, such as the RosettaScripts scripting language [32]. Because these barriers limit accessibility, hybrid approaches that combine ML models trained on prior knowledge with general physics-based methods could be transformative for heteropolymer design [70]. Here, we identify an opportunity for LLM-based agents: by combining multi-turn reasoning with code generation and tool use, such agents could accelerate Rosetta protocol development even for advanced users, while making Rosetta's powerful capabilities available to non-expert users in the broader scientific community.

In this work, we develop *Agent Rosetta*, a single-agent, multi-turn agentic framework that follows a user-defined brief and progressively refines protein designs through structured dialogue and feedback with a RosettaScripts [32] environment. To enable this, we built an OpenAI Gym-like environment [16] in which the agent can perform multiple types of actions and receive feedback from Rosetta, allowing for iterative design refinement. Our approach leverages the biophysical knowledge of the base LLM through artificial reasoning, combines it with the constraints of XML scripting in Rosetta, and uses the code-writing capabilities of LLMs. We find this framework an interesting test case for building autonomous agents aimed at real-world scientific workloads. For example, we find that even though many examples of RosettaScripts are certainly included in modern LLM training corpora, its rich and complex syntax—unlike that of mainstream software packages—creates many challenges. In specific cases, we found that these difficulties proved significant enough that prompt engineering alone was insufficient to extract scientifically sound outputs from frontier language models.

We briefly summarize the main contributions of this work:

- We introduce Agent Rosetta, an LLM agent capable of executing general user-defined task briefs via multi-turn interactions with a RosettaScripts environment. In building our agent, we found that prompting alone may be insufficient to bridge general-purpose LLMs with specialized scientific software. Instead, tool and environment design play a crucial role.
- We evaluate Agent Rosetta on designing amino acid sequences to stabilize input polypeptide backbone conformations, and we study how scaling inference-time compute affects the quality of the designs. In particular, we investigate longer multi-turn interactions and more parallel designs in *best-of-n* sampling. Our results show that increasing compute overall improves design quality.
- We compare Agent Rosetta with ProteinMPNN—an existing deep learning model trained specifically for fixed-backbone design of canonical protein sequences. The best designs of our agent are comparable with ProteinMPNN's, highlighting that general-purpose LLM agents using scientific software are comparable with specialized models trained to solve particular tasks.

This work represents an important step towards understanding the merits and limitations of LLM-based agents for scientific tasks. Our results demonstrate that coupling frontier LLMs with domain-specific scientific tools can yield flexible frameworks that achieve performance competitive with deep learning models trained for narrowly defined design tasks. This suggests that generalist agentic approaches can combine accuracy with versatility, opening new opportunities for scientific discovery.

We refer readers to Appendix A for a detailed discussion of prior works in designing scientific agents, deep learning approaches to protein design, and heteropolymer design with Rosetta.

2 Agent Rosetta

The Rosetta modeling suite [58] is a general collection of software supporting a wide range of applications such as heteropolymer structure prediction, docking, and design. A central component of Rosetta's methodology is the use of Monte Carlo algorithms guided by an *energy function* [6] to explore either conformation space (*i.e.*, to sample favorable structural conformations given a fixed sequence of chemical building blocks) or sequence space (*i.e.*, to sample favorable sequences given a

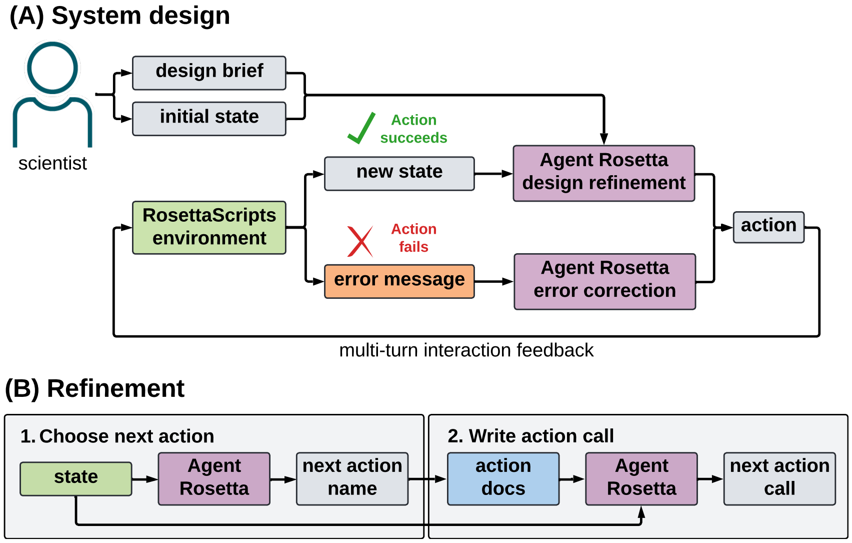


Figure 1: Illustration of our multi-turn agentic system. (A) Overall schematics of an Agent Rosetta trajectory. (B) Design refinement protocol: first, the agent chooses which action to take next and the environment returns the appropriate documentation, then the full action call is generated.

fixed heteropolymer backbone conformation) [14, 56]. To carry out these simulations, Rosetta defines *Movers* (*i.e.*, actions that alter a molecular structure) to guide the sampling process and navigate the energy landscape, as well as *Filters* (*i.e.*, modules that measure some property of a structure and decide whether to continue or to start over).

Scientists typically construct protocols composed of multiple Movers and Filters that may require hours to days to execute. Because individual Movers can take tens of minutes—depending on polymer length and structural complexity—intermediate results are seldom examined. Consequently, researchers must commit to executing entire protocols before assessing their success, often foregoing potentially valuable information about intermediate states.

In contrast, LLM agents can interact with Rosetta in a sequential manner, selecting each subsequent Mover immediately after the preceding one completes. This enables adaptive guidance of the design process informed by the properties of intermediate structures. When parallelized across multiple design trajectories, such an approach also facilitates a broader and more diverse exploration of protocols than is typically feasible for human researchers. To realize these advantages, we introduce Agent Rosetta, a multi-turn framework that analyzes the current structure at each step and determines the next action (see Fig. 1 and Appendix B).

The design brief. Rosetta offers a versatile toolkit that can be used for a variety of tasks. For example, it can modify natural proteins to alter their function [68, 103, 104], construct antibodies that can bind to particular antigens [4, 91], or design synthetic exotic peptides that can bind to target proteins of therapeutic interest [43, 74]. Agent Rosetta is instructed to follow a design brief that states these goals in terms of both quantitative and qualitative objectives. For example, an entry-level design brief could be to *"Redesign ubiquitin so that its core is well-packed and its stability is maximized"*. Together with the design brief, the user can define an initial structure the agent should start from: in our example, the Protein Data Bank (PDB) file `1UBQ`, describing the 3D conformation of ubiquitin. In other tasks, this could be left to the agent to generate with internal Rosetta modeling tools.

The environment state. At any step of the design process, the structure of a candidate polymer in Rosetta is stored as a data structure called a `Pose` [58], which can be written to a PDB file. `Pose` objects are highly complex data structures, and PDB files can easily contain hundreds of lines—making either impractical for direct use as context for the agent. Instead, we use summary statistics such as the amino acid sequence, the individual scores in the Rosetta energy function, and key structural information. In particular, at each step, we compute the radius of gyration and the cavity volume [93] of the current design.

Stability is one of the most important criteria for assessing the quality of protein designs, yet it remains difficult to quantify without experimental validation. To approximate it, we rely on surrogate metrics. For canonical protein design, at each step, we use ESMFold [64] to predict the native fold of the amino acid sequence, and we compare this predicted fold with the current Rosetta-generated structure in terms of Root Mean Squared Deviation (RMSD). A stable design should yield similar folds, and thus a small RMSD. We complement the RMSD with the average pLDDT of the C_α atoms: values close to 1 indicate high confidence in ESMFold's prediction, providing further evidence for stability. We include the RMSD and pLDDT in the definition of our state as easy-to-compute but imperfect proxies for design stability. Note, however, that since ESMFold and related models are trained on natural proteins, they may not reliably capture the stability of Rosetta designs that deviate from naturalistic sequences. Similarly, Rosetta's `ref2015` energy function [6] also serves as a general but imperfect proxy for design stability.

Finally, we complement the state with task-specific evaluation metrics. For example, when the task is to stabilize a given polymer backbone, we evaluate the RMSD between the current structure and this target conformation to inform the agent on how well the current design adheres to the target conformation.

In Appendix C, we illustrate how the agent uses this information in practice.

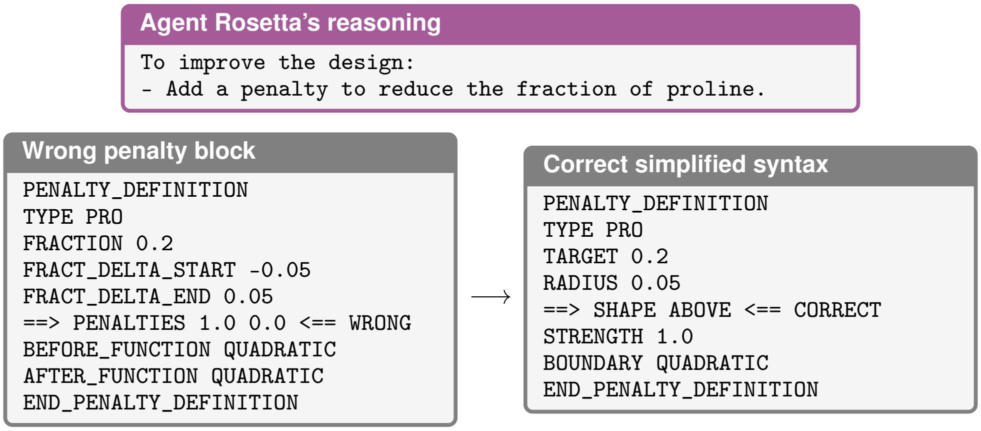
Available actions. Since RosettaScripts accepts XML scripts as actions, a reasonable choice would be to let Agent Rosetta generate them from scratch. This approach, however, leads to unnecessarily long generations that are prone to errors, harder for the agent to correct, and computationally more expensive. In this work, instead, we leverage the well-structured RosettaScripts action schema [32], which allows us to define *types* of actions with their respective XML templates that are populated at runtime with the action parameters generated by the agent. This removes a significant overhead from the agent, focusing on the scientifically-relevant parameters of each action only, without having to devote considerable computation to the overall syntax for the action type. We define three general types of actions that fit several design pipelines and allow Agent Rosetta to explore a vast sequence space (see Appendix D for example action calls):

- `rotamer_change` uses Rosetta's `FastDesign` Mover [10] to change the side chains of the molecule, relaxing the structure in the process. It also allows the agent to specify amino acid composition constraints that penalize deviations from a target composition and guide sequence sampling [71]. Beyond compositional constraints, the agent can use *ResidueSelectors* to select which regions of the structure to penalize (*e.g.*, the core, the surface), and *TaskOperations* to restrict the search space (*e.g.*, by limiting the residues allowed at certain positions).
- `backbone_change` includes three RosettaScripts Movers that randomly perturb the torsion angles of the backbone to modify its conformation: `Small`, `Shear`, and `Backrub` [96]. Backbone changes can be used to escape sub-optimal conformations difficult to stabilize with any choice of sequence, permitting discovery of nearby conformations compatible with much more stabilizing sequences. Each Mover accepts specific parameters and residue selectors, which the agent is free to specify.
- `go_back_to` allows the agent to revert to a previous node in the trajectory in case the design has diverged too far from the goal.

The multi-turn agentic workflow. Agent Rosetta iteratively refines the design by interacting with the environment. Each turn of our framework follows these general steps:

1. **Structured artificial reasoning and action selection:** First, the agent receives the current state of the environment as described above, and a summary of the previous actions. History summarization is important to prevent the context of the underlying LLM from saturating over long trajectories with multiple interactions [76, 94, 99, 120]. Here, we use a tabular summary of the previous actions and their effect on key design metrics (see Fig. C.1 for an example).

   Then, we prompt Agent Rosetta to reflect on the progress towards the design goals and choose the next action. We provide structured reasoning instructions, explicitly mentioning to take into consideration the individual Rosetta energy terms, the structural information, and the trajectory history. This is important to elicit the biophysical knowledge in the LLM and to verify that the action chosen by the agent is coherent with the objectives of the design brief.
2. **Structured reasoning and parameter generation:** After the agent selects the next action, the environment responds with the necessary RosettaScripts documentation. This ensures access to



```
To improve the design:
- Add a penalty to reduce the fraction of proline.
```

```
PENALTY_DEFINITION
TYPE PRO
FRACTION 0.2
FRACT_DELTA_START -0.05
FRACT_DELTA_END 0.05
==> PENALTIES 1.0 0.0 <== WRONG
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION
```

```
PENALTY_DEFINITION
TYPE PRO
TARGET 0.2
RADIUS 0.05
==> SHAPE ABOVE <== CORRECT
STRENGTH 1.0
BOUNDARY QUADRATIC
END_PENALTY_DEFINITION
```

Figure 2: A representative example of a failure mode of prompting for generation of the compositional penalty blocks for the `rotamer_change` action. Even though Agent Rosetta wants to reduce the number of prolines in the sequence, the penalty block achieves the opposite effect due to the wrong `PENALTIES 1.0 0.0` line, which encourages more than 15% of proline content. We also include the correct penalty block using our simplified syntax, which is easier for the agent to follow.

the appropriate syntax and parameter information, which is different from the majority of code LLMs are trained on, and it can vary significantly across different types of actions. Instead of the online RosettaScript's documentation, we opted to write *ad-hoc* summaries tailored to clarify and emphasize the aspects the agent struggles the most with.

We include the action's documentation in the context of the LLM, and we prompt the agent to reason again and generate the full action call with its parameters. We use the same structured reasoning steps as for action selection.

3. **Action parsing, execution, and feedback:** We parse the action parameters, format them in the corresponding XML template, and execute the script in the environment. If the action was successful, the state is updated, and the agent proceeds to the next design refinement turn. If not, we pass the error message to the agent to fix its mistakes (see Appendix E for examples).

3 Results

Agent Rosetta can be used to generate diverse structures—including cyclic peptides and those with non-canonical residues. However, in order to allow rigorous comparison with existing baselines, we evaluate its performance on the simpler task of stabilizing fixed backbone conformations using canonical amino acids only.

Since we are interested in focusing on Rosetta-based exploration of sequence space for this task, we use RFdiffusion [110] to sample backbone conformations and to generate input structures for Rosetta. RFdiffusion generates backbone-only PDB files, where each residue is, by default, glycine. The initial state of the trajectory is the target poly-glycine conformation, and the task brief instructs Agent Rosetta to change its side chains so that it is stable, it folds in the same shape as the target, and it has low Rosetta energy (see Appendix B for the design brief). We compare the quality of the designs generated with our agent with those generated by ProteinMPNN [26], an ML encoder-decoder model trained specifically for fixed-backbone design. Unlike Rosetta's physics-based sampling, ProteinMPNN was trained on known protein sequences and structures to approximates the conditional distribution of amino acid sequences given an input conformation (in our case, the one generated with RFdiffusion).

We consider 4 sequence length ranges: from 40 to 60 residues, from 60 to 80, 80 to 100, and 100 to 120. For each range, we sample 5 independent backbones. For each backbone, we run Agent Rosetta 16 times for at most 40 iterations—including successful and failed turns—and sample 128 sequences with ProteinMPNN with temperature parameter set to 0.3 to increase sequence diversity. Herein, we use GPT-4.1 [3] as the underlying LLM.

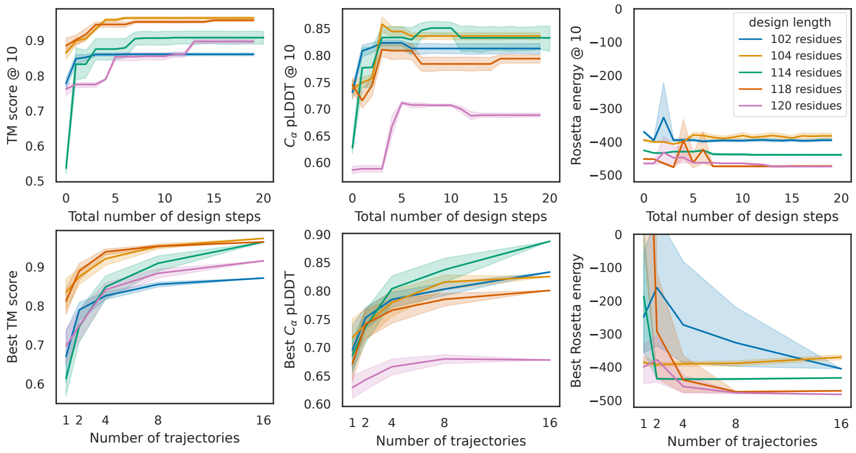


Figure 3: Quality of designs generated by Agent Rosetta as a function of number of successful design steps (top row) and number of parallel trajectories in best-of-n sampling (bottom row). We show results for 5 backbone conformations with 100 to 120 residues, and defer remaining backbones to the appendix. We plot quality metrics for the design with highest TM score, averaging results over 50 independent replicates. Error bars represent 95% confidence intervals.

3.1 Prompting can fail to generate specialized code

Rosetta and RosettaScripts have extensive online presence, including their source code, documentation, and discussion forums. It is reasonable to assume, then, that they are in the training set of frontier LLMs, thus prompting should suffice to bring forth the necessary knowledge from the training data.

We found that LLMs know general facts about Rosetta and RosettaScripts syntax, but they struggle to generate semantically correct action calls with prompting alone. Consider the composition penalty blocks for the `rotamer_change` action: these follow a particular syntax that defines which residues to select, a target fractional or absolute range, and a sequence of magnitudes that specify penalties for deviations from the desired range. Furthermore, the number of magnitudes changes according to the width of the penalty range, and whether the target composition is fractional or absolute. All these factors are both somewhat counter-intuitive to first-time RosettaScripts users, and also sources of potential errors for the agent, even if its reasoning is plausible. We include a representative example from a trajectory in our experiments in Fig. 2. The agent correctly recognizes that excessive proline content may be causing steric clashes, and wants to correct it. However, the line `PENALTIES 1.0 0.0`, even if syntactically correct, *increases* proline content. In our experiments, extending prompting instructions and clarifications only yielded minor improvements, without removing semantic errors that are computationally wasteful: the LLM generates an action call, RosettaScripts runs successfully, but the outcome is counterproductive.

To solve these issues, we simplified the syntax of composition penalty blocks to align with the high-level intentions of the agent. We include the corrected equivalent of the penalty block with our simplified syntax in Fig. 2. Not only is our syntax simpler to generate for an agent, but also more intuitive to a user. Agent Rosetta generates a `TARGET` and `RADIUS` parameters that define the target range, chooses the shape of the penalty between `BELOW`, `ABOVE`, or `OUTSIDE`, and specifies one `STRENGTH` value. Then, we automatically translate this syntax into its RosettaScripts equivalent. We expand on this with examples in Appendix F. These findings highlight that prompting and specialized documentation can fail to bridge general-purpose LLMs with specialized scientific software.

3.2 Scaling inference-time compute improves design quality

Inference-time techniques such as step-by-step reasoning [111], best-of-n sampling [23, 97], Monte Carlo tree search [116], or evolutionary algorithms [79, 88] have emerged as successful approaches to improve the performance of LLMs for solving challenging tasks. Given the iterative nature of protein design, we investigate how longer multi-turn interactions and parallelizing more trajectories with best-of-n sampling affect the quality of designs generated with Agent Rosetta.

Recall the objective of the agent is to stabilize a target backbone conformation. Hence, at each step, we use three metrics to evaluate quality:

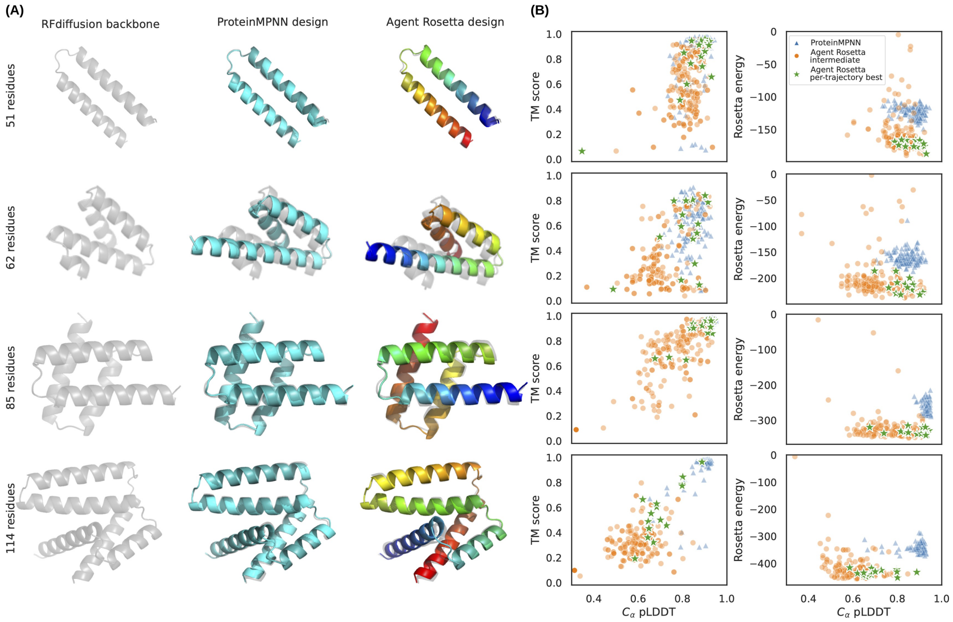


Figure 4: Comparison of designs generated with ProteinMPNN and Agent Rosetta for 5 backbone conformations of increasing length. (A) Visual inspection of designs. The leftmost column includes the initial backbone, the center column the ESMFold predicted structure of the best sequence sampled with ProteinMPNN, and, similarly, the rightmost column shows the ESMFold predicted structure of the best sequence designed with Agent Rosetta. Designs are successful if the align well with the initial conformation. (B) Quantitative comparison of the distribution of designs, we include both the best design for each Agent Rosetta trajectory and intermediate steps.

- **Structural similarity with the target:** We predict the native fold of the current design with ESMFold [64] and compute its template modeling (TM) score [118] with the target RFdiffusion backbone. The TM score is a normalized notion of similarity (1 being a perfect match) that, compared to the RMSD, is less sensitive to local differences. A high TM score means that the predicted fold aligns with the target conformation.
- **Confidence of ESMFold predicted structure:** We complement the TM score with the average pLDDT of the C_α atoms. This is a heuristic quantity between 0 and 1 that conveys the confidence of the model in predicting the 3D coordinates of each residue. A low pLDDT may indicate both an intrinsically flexible region but also an uncertain prediction.
- **Total Rosetta energy:** We report the total energy of the current Rosetta design. This evaluates whether the design is favorable, such as having a compact hydrophobic core, no clashes between side chains, and hydrogen bonds. If the Rosetta Monte Carlo algorithm is proceeding towards a minimum of the energy landscape, then energy should decrease.

Fig. 3 includes results for the 5 longest backbone conformations, and we include results for the remaining ones in Fig. H.1. For performance as a function of total successful design steps, we randomly sample 10 trajectories, remove failed action calls, and truncate them to the desired length. We show quality metrics for the design with highest structural similarity with the target conformation. In particular, we report average and 95% confidence intervals over 50 independent replicates. Overall, we found that longer trajectories generally lead to better designs in terms of TM score and pLDDT. During the first interactions, both rapidly increase, demonstrating that the agent is able to efficiently steer the trajectory towards plausible sequences. For later interactions, results vary across designs, with potential saturation. This may be due to the fact that not all RFdiffusion backbones are designable. Overall, designs converge, providing evidence that the agent is able to sustain multiple interactions. We note that the total Rosetta energy shows the least amount of improvement as a function of number of steps. This is not surprising, as the energy landscape is very rugged, with several local minimizers that have similar energies but different physical properties.

For results as a function of number of parallel trajectories, we randomly subsample the pool of 16 trajectories per backbone. As above, we report average quality metrics and confidence intervals of the design with highest TM score over 50 independent replicates. We found that increasing the number of trajectories is also beneficial. This is important for two reasons. First, one of the main advantages of LLM agents compared to human scientists is scalability. Second, reinforcement learning methodologies for reasoning tasks (*e.g.*, GRPO [92]) rely on multiple parallel trajectories to explore the upper tail of the reward for a given task. We consider studying the extent to which this is true for smaller, open source models as next steps. This would enable training smaller, specialized agents capable of aiding automation of general scientific discovery pipelines.

3.3 Comparing Agent Rosetta designs with those produced by ProteinMPNN

We study the tradeoffs between scientific agents using general-purpose tools and deep learning models trained for specific design pipelines. In particular, we compare the quality of designs generated with Agent Rosetta and ProteinMPNN [26]—a model trained specifically for fixed-backbone design.

We visually inspect the ESMFold predicted structure of a few designs for each method in Fig. 4 (A). For each target backbone, we show the best design in terms of all three quality metrics, confirming they align well with the target. Both methods miss a turn for the backbone with 62 residues, which suggests that it may not be possible to stabilize all conformations generated by RFdiffusion.

Quantitatively, we compare the distribution of designs with Agent Rosetta and ProteinMPNN in Fig. 4 (B). For Agent Rosetta, we show both the best designs for each trajectory and intermediate structures. For ProteinMPNN, we show all 128 sequences. We found that, in terms of TM score and pLDDT, the best designs of both methods are comparable. Furthermore, the distributions of designs mostly overlap for shorter sequences but diverge for longer (results for all backbones are in Fig. H.2).

The designs of our agents generally have lower Rosetta energy, even those with worse pLDDT. Although Rosetta's energy function is an imperfect proxy to stability, this highlights the capacity to discover potentially favorable sequences that are very different from the natural sequences on which ESMFold was trained. There is reason to believe that, even in the realm of canonical protein design, approaches like Agent Rosetta that produce diverse designs—some of which do not resemble known sequences—could have advantages over ML approaches that focus exclusively on the realm of sequences resembling known ones. While natural proteins are evolutionarily optimized for marginal stability, with melting temperatures in the 40-60 °C [48] to permit degradation, Rosetta-designed proteins routinely show extreme stability (often a desirable property in engineered proteins), with melting temperatures of 90 °C or higher [54, 87]. This suggests that a great deal of diversity (with favorable properties) exists beyond the natural proteins on which ML models are trained.

Finally, both energy functions and sequence optimization algorithms are ongoing fields of research. Rosetta features extensible architecture (particularly when linked against the newly-introduced Masala software suite [117], which features enhanced optimizers), which means that it can readily make state-of-the-art algorithms available to the agent without need for retraining.

4 Conclusions

In this work, we introduced Agent Rosetta, an LLM agent capable of using the RosettaScripts language for protein design. Through multi-turn feedback with the environment, Agent Rosetta follows a user-defined brief to refine protein designs. Our framework is general, and it encompasses design pipelines with non-canonical building blocks and geometries.

In building our framework, we found that prompt engineering alone can fail to bridge general-purpose LLMs with specialized scientific software that have rich and complex syntax different from mainstream code. Instead, careful tool and environment design are necessary to bring forth the desired capabilities. We compared the quality of the designs generated with Agent Rosetta and ProteinMPNN on stabilizing fixed backbone conformations with canonical amino acids. Our results showed that general-purpose LLM agents using specialized software are comparable to deep learning models trained specifically to solve the same task.

As important next steps, we consider studying design tasks with non-canonical structures. Since these follow the same underlying principles as canonical proteins, we expect Agent Rosetta's skills to generalize well. Furthermore, our work builds the necessary infrastructure to fine-tune smaller, open-source models, and subsequently train them with reinforcement learning techniques. Training specialized models opens up several exciting directions. For example, multi-modal agents could combine information from several sources, going beyond a text-based context with summary statistics.

Overall, our work contributes to the broader scientific community by studying what it takes to develop specialized scientific agent with the capability to aid scientists in real-world tasks, saving experts person-hours during methods development, while providing entry for biochemists and biologists with little Rosetta experience into the world or rational heteropolymer design.

Acknowledgments

Polymathic AI was supported by the Simons Foundation and Schmidt Sciences. PDR and VKM are wholly funded by the Simons Foundation. We thank the Flatiron Institute's Scientific Computing Core for ongoing support. The computations reported in this paper were performed in-part using resources made available by the Flatiron Institute. The Flatiron Institute is a division of the Simons Foundation.

References

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[2] Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. Agent-e: From autonomous web navigation to foundational design principles in agentic systems. *arXiv preprint arXiv:2407.13032*, 2024.

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[4] Jared Adolf-Bryfogle, Oleks Kalyuzhniy, Michael Kubitz, Brian D. Weitzner, Xiaozhen Hu, Yumiko Adachi, William R. Schief, and Roland L. Dunbrack Jr. RosettaAntibodyDesign (RAbD): A general framework for computational antibody design. *PLOS Computational Biology*, 14(4):e1006112, April 2018. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006112. URL `https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006112`. Publisher: Public Library of Science.

[5] Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: a preliminary study using gpt-4, 2023. URL `https://arxiv.org/abs/2311.07361`.

[6] Rebecca F Alford, Andrew Leaver-Fay, Jeliazko R Jeliazkov, Matthew J O'Meara, Frank P DiMaio, Hahnbeom Park, Maxim V Shapovalov, P Douglas Renfrew, Vikram K Mulligan, Kalli Kappel, et al. The rosetta all-atom energy function for macromolecular modeling and design. *Journal of chemical theory and computation*, 13(6):3031–3048, 2017.

[7] Sören Arlt, Haonan Duan, Felix Li, Sang Michael Xie, Yuhuai Wu, and Mario Krenn. Meta-designing quantum experiments with language models, 2024. URL `https://arxiv.org/abs/2406.02470`.

[8] Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. Researchagent: Iterative research idea generation over scientific literature with large language models. *arXiv preprint arXiv:2404.07738*, 2024.

[9] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[10] Gaurav Bhardwaj, Vikram Khipple Mulligan, Christopher D. Bahl, Jason M. Gilmore, Peta J. Harvey, Olivier Cheneval, Garry W. Buchko, Surya V. S. R. K. Pulavarti, Quentin Kaas, Alexander Eletsky, Po-Ssu Huang, William A. Johnsen, Per Jr Greisen, Gabriel J. Rocklin, Yifan Song,

Thomas W. Linsky, Andrew Watkins, Stephen A. Rettie, Xianzhong Xu, Lauren P. Carter, Richard Bonneau, James M. Olson, Evangelos Coutsias, Colin E. Correnti, Thomas Szyperski, David J. Craik, and David Baker. Accurate de novo design of hyperstable constrained peptides. *Nature*, 538(7625):329–335, October 2016. ISSN 1476-4687. doi: 10.1038/nature19791. URL https://www.nature.com/articles/nature19791. Publisher: Nature Publishing Group.

[11] Gaurav Bhardwaj, Jacob O'Connor, Stephen Rettie, Yen-Hua Huang, Theresa A. Ramelot, Vikram Khipple Mulligan, Gizem Gokce Alpkilic, Jonathan Palmer, Asim K. Bera, Matthew J. Bick, Maddalena Di Piazza, Xinting Li, Parisa Hosseinzadeh, Timothy W. Craven, Roberto Tejero, Anna Lauko, Ryan Choi, Calina Glynn, Linlin Dong, Robert Griffin, Wesley C. van Voorhis, Jose Rodriguez, Lance Stewart, Gaetano T. Montelione, David Craik, and David Baker. Accurate de novo design of membrane-traversing macrocycles. *Cell*, 185(19):3520–3532.e26, September 2022. ISSN 1097-4172. doi: 10.1016/j.cell.2022.07.019.

[12] Aadyot Bhatnagar, Sarthak Jain, Joel Beazer, Samuel C Curran, Alexander M Hoffnagle, Kyle Ching, Michael Martyn, Stephen Nayfach, Jeffrey A Ruffolo, and Ali Madani. Scaling unlocks broader generation and deeper functional understanding of proteins. *bioRxiv*, pages 2025–04, 2025.

[13] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[14] R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. E. Strauss, and D. Baker. Rosetta in CASP4: progress in ab initio protein structure prediction. *Proteins*, Suppl 5:119–126, 2001. ISSN 0887-3585. doi: 10.1002/prot.1170.

[15] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

[16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. 2016. eprint. *arXiv preprint arXiv:1606.01540*, 50, 2016.

[17] Chai Discovery. Chai-1: Decoding the molecular interactions of life. *bioRxiv*, 2024. doi: 10.1101/2024.10.10.615955. URL https://www.biorxiv.org/content/early/2024/10/11/2024.10.10.615955.

[18] Chai Discovery. Zero-shot antibody design in a 24-well plate. *bioRxiv*, 2025. doi: 10.1101/2025.07.05.663018. URL https://www.biorxiv.org/content/early/2025/07/06/2025.07.05.663018.

[19] Sidhartha Chaudhury, Sergey Lyskov, and Jeffrey J Gray. Pyrosetta: a script-based interface for implementing molecular modeling algorithms using rosetta. *Bioinformatics*, 26(5):689–691, 2010.

[20] Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. Mlr-bench: Evaluating ai agents on open-ended machine learning research. *arXiv preprint arXiv:2505.19955*, 2025.

[21] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[22] Henry Childs, Pei Zhou, and Bruce R. Donald. Has AlphaFold 3 Solved the Protein Folding Problem for D-Peptides? *bioRxiv: The Preprint Server for Biology*, page 2025.03.14.643307, March 2025. ISSN 2692-8205. doi: 10.1101/2025.03.14.643307.

[23] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[24] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. In *International Conference on Learning Representations (ICLR)*, 2023.

[25] Bobo Dang, Haifan Wu, Vikram Khipple Mulligan, Marco Mravic, Yibing Wu, Thomas Lemmin, Alexander Ford, Daniel-Adriano Silva, David Baker, and William F. DeGrado. De novo design of covalently constrained mesosize protein scaffolds with unique tertiary structures. *Proceedings of the National Academy of Sciences*, 114(41):10852–10857, September 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1710695114. URL `http://www.pnas.org/lookup/doi/10.1073/pnas.1710695114`.

[26] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[27] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

[28] Joseph Dodd-O, Amanda M. Acevedo-Jake, Abdul-Rahman Azizogli, Vikram Khipple Mulligan, and Vivek A. Kumar. How to Design Peptides. *Methods in Molecular Biology (Clifton, N.J.)*, 2597:187–216, 2023. ISSN 1940-6029. doi: 10.1007/978-1-0716-2835-5_15.

[29] Kevin Drew, P. Douglas Renfrew, Timothy W. Craven, Glenn L. Butterfoss, Fang-Chieh Chou, Sergey Lyskov, Brooke N. Bullock, Andrew Watkins, Jason W. Labonte, Michael Pacella, Krishna Praneeth Kilambi, Andrew Leaver-Fay, Brian Kuhlman, Jeffrey J. Gray, Philip Bradley, Kent Kirshenbaum, Paramjit S. Arora, Rhiju Das, and Richard Bonneau. Adding Diverse Noncanonical Backbones to Rosetta: Enabling Peptidomimetic Design. *PLoS ONE*, 8(7):e67051, July 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0067051. URL `http://dx.plos.org/10.1371/journal.pone.0067051`.

[30] Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, et al. Protein complex prediction with alphafold-multimer. *biorxiv*, pages 2021–10, 2021.

[31] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.

[32] Sarel J Fleishman, Andrew Leaver-Fay, Jacob E Corn, Eva-Maria Strauch, Sagar D Khare, Nobuyasu Koga, Justin Ashworth, Paul Murphy, Florian Richter, Gordon Lemmon, et al. Rosettascripts: a scripting language interface to the rosetta macromolecular modeling suite. *PloS one*, 6(6):e20161, 2011.

[33] Alireza Ghafarollahi and Markus J Buehler. Protagents: protein discovery via large language model multi-agent collaborations combining physics and machine learning. *Digital Discovery*, 3(7):1389–1409, 2024.

[34] Alireza Ghafarollahi and Markus J Buehler. Sciagents: automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. *Advanced Materials*, 37(22):2413523, 2025.

[35] Shane Gonen, Frank DiMaio, Tamir Gonen, and David Baker. Design of ordered two-dimensional arrays mediated by noncovalent protein-protein interfaces. *Science (New York, N.Y.)*, 348(6241):1365–1368, June 2015. ISSN 1095-9203. doi: 10.1126/science.aaa9897.

[36] Sydney R. Gordon, Elizabeth J. Stanley, Sarah Wolf, Angus Toland, Sean J. Wu, Daniel Hadidi, Jeremy H. Mills, David Baker, Ingrid Swanson Pultz, and Justin B. Siegel. Computational design of an α-gliadin peptidase. *Journal of the American Chemical Society*, 134(50):20513–20520, December 2012. ISSN 1520-5126. doi: 10.1021/ja3094795.

[37] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

[38] Mourad Gridach, Jay Nanavati, Khaldoun Zine El Abidine, Lenon Mendes, and Christina Mack. Agentic ai for scientific discovery: A survey of progress, challenges, and future directions. *arXiv preprint arXiv:2503.08979*, 2025.

[39] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

[40] Sikun Guo, Amir Hassan Shariatmadari, Guangzhi Xiong, Albert Huang, Myles Kim, Corey M Williams, Stefan Bekiranov, and Aidong Zhang. Ideabench: Benchmarking large language models for research idea generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5888–5899, 2025.

[41] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

[42] Parisa Hosseinzadeh, Gaurav Bhardwaj, Vikram Khipple Mulligan, Matthew D. Shortridge, Timothy W. Craven, Fátima Pardo-Avila, Stephen A. Rettie, David E. Kim, Daniel-Adriano Silva, Yehia M. Ibrahim, Ian K. Webb, John R. Cort, Joshua N. Adkins, Gabriele Varani, and David Baker. Comprehensive computational design of ordered peptide macrocycles. *Science*, 358(6369):1461–1466, December 2017. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aap7577. URL http://www.sciencemag.org/lookup/doi/10.1126/science.aap7577.

[43] Parisa Hosseinzadeh, Paris R. Watson, Timothy W. Craven, Xinting Li, Stephen Rettie, Fátima Pardo-Avila, Asim K. Bera, Vikram Khipple Mulligan, Peilong Lu, Alexander S. Ford, Brian D. Weitzner, Lance J. Stewart, Adam P. Moyer, Maddalena Di Piazza, Joshua G. Whalen, Per Jr Greisen, David W. Christianson, and David Baker. Anchor extension: a structure-guided approach to design cyclic peptides targeting enzyme active sites. *Nature Communications*, 12 (1):3384, June 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-23609-8.

[44] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8):1–79, 2024.

[45] Yang Hsia, Jacob B. Bale, Shane Gonen, Dan Shi, William Sheffler, Kimberly K. Fong, Una Nattermann, Chunfu Xu, Po-Ssu Huang, Rashmi Ravichandran, Sue Yi, Trisha N. Davis, Tamir Gonen, Neil P. King, and David Baker. Design of a hyperstable 60-subunit protein icosahedron. *Nature*, 535(7610):136–139, June 2016. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature18010. URL http://www.nature.com/doifinder/10.1038/nature18010.

[46] Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, et al. Biomni: A general-purpose biomedical ai agent. *biorxiv*, pages 2025–05, 2025.

[47] Ian R Humphreys, Jimin Pei, Minkyung Baek, Aditya Krishnakumar, Ivan Anishchenko, Sergey Ovchinnikov, Jing Zhang, Travis J Ness, Sudeep Banjade, Saket R Bagde, et al. Computed structures of core eukaryotic protein complexes. *Science*, 374(6573):eabm4805, 2021.

[48] Anna Jarzab, Nils Kurzawa, Thomas Hopf, Matthias Moerch, Jana Zecha, Niels Leijten, Yangyang Bian, Eva Musiol, Melanie Maschberger, Gabriele Stoehr, Isabelle Becher, Charlotte Daly, Patroklos Samaras, Julia Mergner, Britta Spanier, Angel Angelov, Thilo Werner, Marcus Bantscheff, Mathias Wilhelm, Martin Klingenspor, Simone Lemeer, Wolfgang Liebl, Hannes Hahne, Mikhail M. Savitski, and Bernhard Kuster. Meltome atlas-thermal proteome stability across the tree of life. *Nature Methods*, 17(5):495–503, May 2020. ISSN 1548-7105. doi: 10.1038/s41592-020-0801-4.

[49] Lin Jiang, Eric A. Althoff, Fernando R. Clemente, Lindsey Doyle, Daniela Röthlisberger, Alexandre Zanghellini, Jasmine L. Gallaher, Jamie L. Betker, Fujie Tanaka, Carlos F.

Barbas, Donald Hilvert, Kendall N. Houk, Barry L. Stoddard, and David Baker. De Novo Computational Design of Retro-Aldol Enzymes. *Science (New York, N.Y.)*, 319 (5868):1387–1391, March 2008. ISSN 0036-8075. doi: 10.1126/science.1152692. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3431203/.

[50] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[51] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[52] Neil P. King, William Sheffler, Michael R. Sawaya, Breanna S. Vollmar, John P. Sumida, Ingemar André, Tamir Gonen, Todd O. Yeates, and David Baker. Computational design of self-assembling protein nanomaterials with atomic level accuracy. *Science (New York, N.Y.)*, 336(6085):1171–1174, June 2012. ISSN 1095-9203. doi: 10.1126/science.1219364.

[53] Neil P. King, Jacob B. Bale, William Sheffler, Dan E. McNamara, Shane Gonen, Tamir Gonen, Todd O. Yeates, and David Baker. Accurate design of co-assembling multi-component protein nanomaterials. *Nature*, 510(7503):103–108, June 2014. ISSN 1476-4687. doi: 10.1038/nature13404.

[54] Nobuyasu Koga, Rie Tatsumi-Koga, Gaohua Liu, Rong Xiao, Thomas B. Acton, Gaetano T. Montelione, and David Baker. Principles for designing ideal protein structures. *Nature*, 491 (7423):222–227, November 2012. ISSN 1476-4687. doi: 10.1038/nature11600.

[55] Rohith Krishna, Jue Wang, Woody Ahern, Pascal Sturmfels, Preetham Venkatesh, Indrek Kalvet, Gyu Rie Lee, Felix S Morey-Burrows, Ivan Anishchenko, Ian R Humphreys, et al. Generalized biomolecular modeling and design with rosettafold all-atom. *Science*, 384(6693): eadl2528, 2024.

[56] Brian Kuhlman and David Baker. Native protein sequences are close to optimal for their structures. *Proceedings of the National Academy of Sciences of the United States of America*, 97(19):10383–10388, September 2000. ISSN 0027-8424. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC27033/.

[57] Brian Kuhlman, Gautam Dantas, Gregory C. Ireton, Gabriele Varani, Barry L. Stoddard, and David Baker. Design of a novel globular protein fold with atomic-level accuracy. *Science (New York, N.Y.)*, 302(5649):1364–1368, November 2003. ISSN 1095-9203. doi: 10.1126/science.1089427.

[58] Andrew Leaver-Fay, Michael Tyka, Steven M. Lewis, Oliver F. Lange, James Thompson, Ron Jacak, Kristian W. Kaufman, P. Douglas Renfrew, Colin A. Smith, Will Sheffler, Ian W. Davis, Seth Cooper, Adrien Treuille, Daniel J. Mandell, Florian Richter, Yih-En Andrew Ban, Sarel J. Fleishman, Jacob E. Corn, David E. Kim, Sergey Lyskov, Monica Berrondo, Stuart Mentzer, Zoran Popović, James J. Havranek, John Karanicolas, Rhiju Das, Jens Meiler, Tanja Kortemme, Jeffrey J. Gray, Brian Kuhlman, David Baker, and Philip Bradley. Rosetta3. In *Methods in Enzymology*, volume 487, pages 545–574. Elsevier, 2011. ISBN 978-0-12-381270-4. doi: 10.1016/B978-0-12-381270-4.00019-6.

[59] Jin Sub Lee, Osama Abdin, and Philip M Kim. Language models for protein design. *Current Opinion in Structural Biology*, 92:103027, 2025.

[60] Julia Koehler Leman, Brian D. Weitzner, Steven M. Lewis, Jared Adolf-Bryfogle, Nawsad Alam, Rebecca F. Alford, Melanie Aprahamian, David Baker, Kyle A. Barlow, Patrick Barth, Benjamin Basanta, Brian J. Bender, Kristin Blacklock, Jaume Bonet, Scott E. Boyken, Phil Bradley, Chris Bystroff, Patrick Conway, Seth Cooper, Bruno E. Correia, Brian Coventry, Rhiju Das, René M. De Jong, Frank DiMaio, Lorna Dsilva, Roland Dunbrack, Alexander S. Ford, Brandon Frenz, Darwin Y. Fu, Caleb Geniesse, Lukasz Goldschmidt, Ragul Gowthaman, Jeffrey J. Gray, Dominik Gront, Sharon Guffy, Scott Horowitz, Po-Ssu Huang, Thomas Huber, Tim M. Jacobs, Jeliazko R. Jeliazkov, David K. Johnson, Kalli Kappel, John

Karanicolas, Hamed Khakzad, Karen R. Khar, Sagar D. Khare, Firas Khatib, Alisa Khramushin, Indigo C. King, Robert Kleffner, Brian Koepnick, Tanja Kortemme, Georg Kuenze, Brian Kuhlman, Daisuke Kuroda, Jason W. Labonte, Jason K. Lai, Gideon Lapidoth, Andrew Leaver-Fay, Steffen Lindert, Thomas Linsky, Nir London, Joseph H. Lubin, Sergey Lyskov, Jack Maguire, Lars Malmström, Enrique Marcos, Orly Marcu, Nicholas A. Marze, Jens Meiler, Rocco Moretti, Vikram Khipple Mulligan, Santrupti Nerli, Christoffer Norn, Shane Ó'Conchúir, Noah Ollikainen, Sergey Ovchinnikov, Michael S. Pacella, Xingjie Pan, Hahnbeom Park, Ryan E. Pavlovicz, Manasi Pethe, Brian G. Pierce, Kala Bharath Pilla, Barak Raveh, P. Douglas Renfrew, Shourya S. Roy Burman, Aliza Rubenstein, Marion F. Sauer, Andreas Scheck, William Schief, Ora Schueler-Furman, Yuval Sedan, Alexander M. Sevy, Nikolaos G. Sgourakis, Lei Shi, Justin B. Siegel, Daniel-Adriano Silva, Shannon Smith, Yifan Song, Amelie Stein, Maria Szegedy, Frank D. Teets, Summer B. Thyme, Ray Yu-Ruei Wang, Andrew Watkins, Lior Zimmerman, and Richard Bonneau. Macromolecular modeling and design in Rosetta: recent methods and frameworks. *Nature Methods*, 17(7):665–680, July 2020. ISSN 1548-7105. doi: 10.1038/s41592-020-0848-2. URL https://www.nature.com/articles/s41592-020-0848-2. Publisher: Nature Publishing Group.

[61] Julia Koehler Leman, Brian D. Weitzner, P. Douglas Renfrew, Steven M. Lewis, Rocco Moretti, Andrew M. Watkins, Vikram Khipple Mulligan, Sergey Lyskov, Jared Adolf-Bryfogle, Jason W. Labonte, Justyna Krys, RosettaCommons Consortium, Christopher Bystroff, William Schief, Dominik Gront, Ora Schueler-Furman, David Baker, Philip Bradley, Roland Dunbrack, Tanja Kortemme, Andrew Leaver-Fay, Charlie E. M. Strauss, Jens Meiler, Brian Kuhlman, Jeffrey J. Gray, and Richard Bonneau. Better together: Elements of successful scientific software development in a distributed collaborative community. *PLOS Computational Biology*, 16(5):e1007507, May 2020. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1007507. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1007507. Publisher: Public Library of Science.

[62] Long Li, Weiwen Xu, Jiayan Guo, Ruochen Zhao, Xingxuan Li, Yuqian Yuan, Boqiang Zhang, Yuming Jiang, Yifei Xin, Ronghao Dang, et al. Chain of ideas: Revolutionizing research via novel idea development with llm agents. *arXiv preprint arXiv:2410.13185*, 2024.

[63] Shanda Li, Tanya Marwah, Junhong Shen, Weiwei Sun, Andrej Risteski, Yiming Yang, and Ameet Talwalkar. Codepde: An inference framework for llm-driven pde solver generation. *arXiv preprint arXiv:2505.08783*, 2025.

[64] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *BioRxiv*, 2022: 500902, 2022.

[65] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[66] Thang Luong and Edward Lockhart. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad. https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the- July 2025. Google DeepMind Blog.

[67] Jeremy H. Mills, Sagar D. Khare, Jill M. Bolduc, Farhad Forouhar, Vikram Khipple Mulligan, Scott Lew, Jayaraman Seetharaman, Liang Tong, Barry L. Stoddard, and David Baker. Computational design of an unnatural amino acid dependent metalloprotein with atomic level accuracy. *Journal of the American Chemical Society*, 135(36):13393–13399, September 2013. ISSN 1520-5126. doi: 10.1021/ja403503m.

[68] Rocco Moretti, Brian J. Bender, Brittany Allison, and Jens Meiler. Rosetta and the Design of Ligand Binding Sites. In Barry L. Stoddard, editor, *Computational Design of Ligand Binding Proteins*, pages 47–62. Springer, New York, NY, 2016. ISBN 978-1-4939-3569-7. doi: 10.1007/978-1-4939-3569-7_4. URL https://doi.org/10.1007/978-1-4939-3569-7_4.

[69] Vikram K. Mulligan. The emerging role of computational design in peptide macrocycle drug discovery. *Expert Opinion on Drug Discovery*, 15(7):833–852, July 2020. ISSN 1746-045X. doi: 10.1080/17460441.2020.1751117.

[70] Vikram Khipple Mulligan. Current directions in combining simulation-based macromolecular modeling approaches with deep learning. *Expert Opinion on Drug Discovery*, 16(9):1025–1044, September 2021. ISSN 1746-045X. doi: 10.1080/17460441.2021.1918097.

[71] Vikram Khipple Mulligan. Computational Methods for Peptide Macrocycle Drug Design. In Seetharama D. Jois, editor, *Peptide Therapeutics: Fundamentals of Design, Development, and Delivery*, volume 47, pages 79–161. Springer International Publishing, New York, 2022. ISBN 978-3-031-04543-1 978-3-031-04544-8. doi: 10.1007/978-3-031-04544-8_3. URL https://link.springer.com/10.1007/978-3-031-04544-8_3. Series Title: AAPS Advances in the Pharmaceutical Sciences Series.

[72] Vikram Khipple Mulligan and Parisa Hosseinzadeh. Computational Design of Peptide-Based Binders to Therapeutic Targets. In Swapnil V. Ghodge, Kaustav Biswas, and Andrei A. Golosov, editors, *Approaching the Next Inflection in Peptide Therapeutics: Attaining Cell Permeability and Oral Bioavailability*, volume 1417 of *ACS Symposium Series*, pages 55–102. American Chemical Society, Washington, DC, August 2022. ISBN 978-0-8412-9761-6 978-0-8412-9760-9. doi: 10.1021/bk-2022-1417.ch003. URL https://pubs.acs.org/doi/abs/10.1021/bk-2022-1417.ch003.

[73] Vikram Khipple Mulligan, Christine S. Kang, Michael R. Sawaya, Stephen Rettie, Xinting Li, Inna Antselovich, Timothy W. Craven, Andrew M. Watkins, Jason W. Labonte, Frank DiMaio, Todd O. Yeates, and David Baker. Computational design of mixed chirality peptide macrocycles with internal symmetry. *Protein Science*, 29(12):2433–2445, 2020. ISSN 1469-896X. doi: https://doi.org/10.1002/pro.3974. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/pro.3974. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/pro.3974.

[74] Vikram Khipple Mulligan, Sean Workman, Tianjun Sun, Stephen Rettie, Xinting Li, Liam J. Worrall, Timothy W. Craven, Dustin T. King, Parisa Hosseinzadeh, Andrew M. Watkins, P. Douglas Renfrew, Sharon Guffy, Jason W. Labonte, Rocco Moretti, Richard Bonneau, Natalie C. J. Strynadka, and David Baker. Computationally designed peptide macrocycle inhibitors of New Delhi metallo-β-lactamase 1. *Proceedings of the National Academy of Sciences of the United States of America*, 118(12):e2012800118, March 2021. ISSN 1091-6490. doi: 10.1073/pnas.2012800118.

[75] Siddharth M Narayanan, James D Braza, Ryan-Rhys Griffiths, Albert Bou, Geemi Wellawatte, Mayk Caldas Ramos, Ludovico Mitchener, Samuel G Rodriques, and Andrew D White. Training a scientific reasoning model for chemistry. *arXiv preprint arXiv:2506.17238*, 2025.

[76] Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, et al. Mlgym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 2025.

[77] Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978, 2023.

[78] Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao-yong Wei, Shanru Lin, Hui Liu, Philip S Yu, et al. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6140–6150, 2025.

[79] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[80] Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, David Kwabi-Addo, Dominique Beaini, Tommi Jaakkola, and Regina Barzilay. Boltz-2: Towards accurate and efficient binding affinity prediction. *bioRxiv*, 2025. doi: 10.1101/2025.06.14.659707.

[81] Kevin Pu, KJ Kevin Feng, Tovi Grossman, Tom Hope, Bhavana Dalvi Mishra, Matt Latzke, Jonathan Bragg, Joseph Chee Chang, and Pao Siangliulue. Ideasynth: Iterative research idea development through evolving and composing idea facets with literature-grounded feedback. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–31, 2025.

[82] Ingrid Swanson Pultz, Malcolm Hill, Joanne M. Vitanza, Clancey Wolf, Lasse Saaby, Tina Liu, Peter Winkle, and Daniel A. Leffler. Gluten Degradation, Pharmacokinetics, Safety, and Tolerability of TAK-062, an Engineered Enzyme to Treat Celiac Disease. *Gastroenterology*, 161(1):81–93.e3, July 2021. ISSN 00165085. doi: 10.1053/j.gastro.2021.03.019. URL https://linkinghub.elsevier.com/retrieve/pii/S0016508521005278.

[83] Yuanhao Qu, Kaixuan Huang, Ming Yin, Kanghong Zhan, Dyllan Liu, Di Yin, Henry C Cousins, William A Johnson, Xiaotong Wang, Mihir Shah, et al. Crispr-gpt for agentic automation of gene-editing experiments. *Nature Biomedical Engineering*, pages 1–14, 2025.

[84] Marissa Radensky, Simra Shahid, Raymond Fok, Pao Siangliulue, Tom Hope, and Daniel S Weld. Scideator: Human-llm scientific idea generation grounded in research-paper facet recombination. *arXiv preprint arXiv:2409.14634*, 2024.

[85] P. D. Renfrew, T. W. Craven, G. L. Butterfoss, K. Kirshenbaum, and R. Bonneau. A rotamer library to enable modeling and design of peptoid foldamers. *Journal of the American Chemical Society*, 136:8772–8782, 2014. doi: 10.1021/ja503776z.

[86] P. Douglas Renfrew, Eun Jung Choi, Richard Bonneau, and Brian Kuhlman. Incorporation of Noncanonical Amino Acids into Rosetta and Use in Computational Protein-Peptide Interface Design. *PLoS ONE*, 7(3):e32637, March 2012. ISSN 1932-6203. doi: 10.1371/journal.pone.0032637. URL http://dx.plos.org/10.1371/journal.pone.0032637.

[87] Gabriel J. Rocklin, Tamuka M. Chidyausiku, Inna Goreshnik, Alex Ford, Scott Houliston, Alexander Lemak, Lauren Carter, Rashmi Ravichandran, Vikram K. Mulligan, Aaron Chevalier, Cheryl H. Arrowsmith, and David Baker. Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science*, 357(6347):168–175, July 2017. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aan0693. URL http://www.sciencemag.org/lookup/doi/10.1126/science.aan0693.

[88] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[89] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

[90] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. *arXiv preprint arXiv:2309.15817*, 2023.

[91] Clara T. Schoeder, Samuel Schmitz, Jared Adolf-Bryfogle, Alexander M. Sevy, Jessica A. Finn, Marion F. Sauer, Nina G. Bozhanova, Benjamin K. Mueller, Amandeep K. Sangha, Jaume Bonet, Jonathan H. Sheehan, Georg Kuenze, Brennica Marlow, Shannon T. Smith, Hope Woods, Brian J. Bender, Cristina E. Martina, Diego del Alamo, Pranav Kodali, Alican Gulsevin, William R. Schief, Bruno E. Correia, James E. Jr. Crowe, Jens Meiler, and Rocco Moretti. Modeling Immunity with Rosetta: Methods for Antibody and Antigen Design. *Biochemistry*, 60(11):825–846, March 2021. ISSN 0006-2960. doi: 10.1021/acs.biochem.0c00912. URL https://doi.org/10.1021/acs.biochem.0c00912. Publisher: American Chemical Society.

[92] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[93] Will Sheffler and David Baker. Rosettaholes: rapid assessment of protein core packing for structure prediction, refinement, design, and validation. *Protein Science*, 18(1):229–239, 2009.

[94] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[95] Justin B. Siegel, Alexandre Zanghellini, Helena M. Lovick, Gert Kiss, Abigail R. Lambert, Jennifer L. St Clair, Jasmine L. Gallaher, Donald Hilvert, Michael H. Gelb, Barry L. Stoddard, Kendall N. Houk, Forrest E. Michael, and David Baker. Computational design of an enzyme catalyst for a stereoselective bimolecular Diels-Alder reaction. *Science (New York, N.Y.)*, 329 (5989):309–313, July 2010. ISSN 1095-9203. doi: 10.1126/science.1190239.

[96] Colin A. Smith and Tanja Kortemme. Backrub-Like Backbone Simulation Recapitulates Natural Protein Conformational Variability and Improves Mutant Side-Chain Prediction. *Journal of Molecular Biology*, 380(4):742–756, July 2008. ISSN 0022-2836. doi: 10.1016/j.jmb.2008.05.023. URL `https://www.sciencedirect.com/science/article/pii/S0022283608005779`.

[97] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[98] Eva-Maria Strauch, Steffen M Bernard, David La, Alan J Bohn, Peter S Lee, Caitlin E Anderson, Travis Nieusma, Carly A Holstein, Natalie K Garcia, Kathryn A Hooper, Rashmi Ravichandran, Jorgen W Nelson, William Sheffler, Jesse D Bloom, Kelly K Lee, Andrew B Ward, Paul Yager, Deborah H Fuller, Ian A Wilson, and David Baker. Computational design of trimeric influenza-neutralizing proteins targeting the hemagglutinin receptor binding site. *Nature Biotechnology*, 35(7):667–671, July 2017. ISSN 1087-0156, 1546-1696. doi: 10.1038/nbt.3907. URL `http://www.nature.com/articles/nbt.3907`.

[99] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2023.

[100] Gemini Team. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[101] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[102] NovelSeek Team, Bo Zhang, Shiyang Feng, Xiangchao Yan, Jiakang Yuan, Zhiyin Yu, Xiaohan He, Songtao Huang, Shaowei Hou, Zheng Nie, et al. Novelseek: When agent becomes the scientist–building closed-loop system from hypothesis to verification. *arXiv preprint arXiv:2505.16938*, 2025.

[103] David F. Thieker, Jack B. Maguire, Stephan T. Kudlacek, Andrew Leaver-Fay, Sergey Lyskov, and Brian Kuhlman. Stabilizing proteins, simplified: A Rosetta-based webtool for predicting favorable mutations. *Protein Science*, 31(10):e4428, 2022. ISSN 1469-896X. doi: 10.1002/pro.4428. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/pro.4428`. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/pro.4428.

[104] Summer Thyme and David Baker. Redesigning the Specificity of Protein–DNA Interactions with Rosetta. In David R. Edgell, editor, *Homing Endonucleases: Methods and Protocols*, pages 265–282. Humana Press, Totowa, NJ, 2014. ISBN 978-1-62703-968-0. doi: 10.1007/978-1-62703-968-0_17. URL `https://doi.org/10.1007/978-1-62703-968-0_17`.

[105] Christine E. Tinberg, Sagar D. Khare, Jiayi Dou, Lindsey Doyle, Jorgen W. Nelson, Alberto Schena, Wojciech Jankowski, Charalampos G. Kalodimos, Kai Johnsson, Barry L. Stoddard, and David Baker. Computational design of ligand-binding proteins with high affinity and selectivity. *Nature*, 501(7466):212–216, September 2013. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature12443. URL `http://www.nature.com/articles/nature12443`.

[106] 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.

[107] Eric Wang, Samuel Schmidgall, Paul F Jaeger, Fan Zhang, Rory Pilgrim, Yossi Matias, Joelle Barral, David Fleet, and Shekoofeh Azizi. Txgemma: Efficient and agentic llms for therapeutics. *arXiv preprint arXiv:2504.06196*, 2025.

[108] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[109] A. M. Watkins, T. W. Craven, P. D. Renfrew, P. S. Arora, and R. Bonneau. Rotamer libraries for the high-resolution design of β-amino acid foldamers. *Structure*, 25(11):1771–1780.e3, 2017. doi: 10.1016/j.str.2017.09.005.

[110] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[111] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[112] Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*, 2025.

[113] Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, and Regina Barzilay. Boltz-1: Democratizing biomolecular interaction modeling. *bioRxiv*, 2024. doi: 10.1101/2024.11.19.624167.

[114] Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

[115] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

[116] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[117] Tristan Zaborniak, Noora Azadvari, Qiyao Zhu, SM Bargeen A Turzo, Parisa Hosseinzadeh, P Douglas Renfrew, and Vikram Khipple Mulligan. The open-source masala software suite: Facilitating rapid methods development for synthetic heteropolymer design. *bioRxiv*, pages 2025–07, 2025.

[118] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

[119] Yu Zhang, Xiusi Chen, Bowen Jin, Sheng Wang, Shuiwang Ji, Wei Wang, and Jiawei Han. A comprehensive survey of scientific large language models and their applications in scientific discovery. *arXiv preprint arXiv:2406.10833*, 2024.

[120] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731, 2024.

[121] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Related works

In this section, we provide a detailed overview of relevant related works.

LLM agents. Through text generation, modern LLMs are capable of using external tools such as web browsers [2, 41, 78, 112, 121], email [27, 90], and, more broadly, code-based interfaces [44, 50, 115]. Coupled with reasoning and *chain-of-thought* prompting [92, 111], LLMs can act as efficient *agents*, automating repetitive workflows and solving complex tasks that require multi-turn interactions [108].

Scientific agents. Scientific discovery workflows are generally well-structured, which makes them attractive for the development of *AI scientists* [37, 65, 102]. Recent works have explored LLMs agents for idea and hypothesis generation [8, 40, 62, 81, 84], experiment design [13, 33], and paper writing [20, 34, 114]. These approaches have found several applications in math [66, 106], chemistry [13, 15, 75], and therapeutics [107]. Most closely related to our work, ProtAgents [33] is a multi-agent system that orchestrates the use of existing machine learning tools for *de novo* protein discovery; CRISPR-GPT [83] studies LLM agents for gene editing tasks; and Biomni [46] develops an agentic framework for general biomedical tasks. We refer interested readers to [38, 119] for comprehensive reviews.

Manually-crafted Rosetta protocols for heteropolymer design. Rosetta has been extensively used for rational design of new protein folds [54, 57, 87],rigidly-structured peptides [10, 25, 42, 73], and other heteropolymers [29, 85, 86]. Proteins have been rationally designed with Rosetta that self-assemble into sheets [35] or cages [45, 53], that bind metals [67] or small molecules [105], that catalyze new enzymatic reactions [49, 95], and that specifically recognize target proteins [98]. Non-canonical peptides have also been designed which bind target proteins [43, 74] or passively diffuse across cell membranes [11]. Rosetta's infrastructure, energy function, and development are described in [6, 58, 60, 61]. Rosetta's scripting interfaces are described in [19, 32]. Non-canonical Rosetta design methods are reviewed in [28, 69, 71, 72]. Recently, the Masala libraries were introduced to permit easy, plugin-based development of new optimizers and other algorithms to extend Rosetta or other heteropolymer modelling software [117].

ML for protein design. Deep learning methodologies have proved successful at training models for particular design pipelines. For example, models exist for both monomer (AlphaFold 2 [51], RoseTTAFold [9, 47], ESMFold [64]) and multimer structure prediction (AlphaFold 3 [1] and Multimer [30], RosettaFold All-Atom [55], Chai-1/2 [17, 18], Boltz-1/2 [80, 113]); sequence generation (ProteinMPNN [26], ProGen2/3 [12, 77], ProtGPT2 [31]); and structure generation and docking (RFdiffusion [110], DiffDock [24]). We point to [59] for a recent review on language models for protein design, and [70] for a review of earlier ML approaches used for protein modeling.

B Prompts

In this section, we include all prompt templates. In the prompts, fields enclosed within curly brackets (*e.g.*, `{field_name}`) are populated at run time.

B.1 System prompt

First, we include Agent Rosetta's system prompt. The `{action_docs}` are populated with Python-style docstrings that summarize the available actions and their parameters:

System prompt

```
You are an expert RosettaScripts coding agent that supports scientists in
    biomolecular design tasks:

- Rosetta is a computational toolkit for modeling, predicting, and designing
    biomolecular structures and interactions, using physics-based energy
    functions and stochastic search.
```

```
- RosettaScripts is Rosetta's XML-based interface for assembling custom modeling
    protocols by combining modular actions (i.e., movers), filters, and scoring
    terms.

Your goal is to follow a user-defined brief and to design a valid RosettaScripts
    protocol that captures the brief precisely.

You have interactive access to a RosettaScripts environment with the following
    actions:

{action_docs}

**Instructions:**

1. Reason about the design brief:
  - Carefully read the design brief.
  - Think about its implications for the peptide sequence from a biophysical
    perspective. Consider both composition and the breakdown of the energy
    scores.
2. Choose the next action:
  - Motivate your choice.
  - If the action requires arguments, only write the action name, the
    environment will display the documentation for the action and re-prompt you
    to write the full action call with all its arguments.
  - If the action does not require arguments, it will be executed immediately.
3. Write the full action call with all its arguments:
  - Carefully read the documentation of the action you chose.
  - Write an implementation plan that translates your thought process into the
    action call. Be specific. The expert biomolecular scientists on your team
    should be able to understand and review your plan.
  - Write the action call in the specified format.

**Formatting:**

- FIRST write your thought process inside `<think>...</think>` tags.
- THEN write your action with the following format (do not include any other
    text, comments, or explanations):
```
<action>
<name tag="choose/run">action_name</name>
<arg_name1>arg_value1</arg_name1>
<arg_name2>arg_value2</arg_name2>
...
</action>
```

**Examples:**

- For an action with arguments, first output:
  ```
  <think> ...your thought process goes here... </think>
  <action tag="choose">
  <name>action_name</name>
  </action>
  ```
  then, after the environment displays the documentation, output:
  ```
  <think> ...your thought process goes here... </think>
  <action tag="run">
  <name>action_name</name>
  <arg_name1>arg_value1</arg_name1>
  <arg_name2>arg_value2</arg_name2>
  ...
  </action>
```
```

```
  ‘‘‘

- For an action without arguments, simply output:
  ‘‘‘
  <think> ...your thought process goes here... </think>
  <action tag="choose">
  <name>action_name</name>
  </action>
  ‘‘‘
  and the action will be executed immediately.

Now, read the design brief and start designing your RosettaScripts protocol!
```

The action docstrings of the three types of actions presented in this works are:

Action docstrings

```
- rotamer_change
"""Perform a Rosetta FastRelax action with design enabled. This action uses the
    compositional penalties in ‘penalties‘ to guide the search, the residue
    selectors defined in ‘residue_selectors‘ to specify which residues to apply
    the penalties to, and the task operations defined in ‘task_operations‘ to
    control the design space.

Arguments:
  - penalties (array, required): a list of compositional penalty definitions.
    Each item includes the following parameters:
    - comp (string, required): the penalty definition blocks specifying the
    compositional constraints.
    - comp_selector_name (string, optional): the name of the residue selector to
    apply the compositional constraints to.
  - residue_selectors (string, optional): the XML definitions of the residue
    selectors.
  - task_operations (string, optional): the XML definitions of the task
    operations to control the design space.
"""

- backbone_change
"""Perturb the backbone of the sequence. This action implements three Rosetta
    backbone movers: ‘small‘, ‘shear‘, and ‘backrub‘. Specify the mover to use
    with ‘mover_name‘, the parameters of the mover in XML format with
    ‘mover_params‘, and the residues to perturb with ‘residue_selector‘.

Arguments:
  - mover_name (string, required): the name of the Rosetta backbone mover to use
    (can be ‘small‘, ‘shear‘, or ‘backrub‘).
  - mover_params (string, required): the parameters of the mover in XML format.
  - residue_selectors (string, optional): the XML definitions of the residue
    selectors.
  - mover_selector_name (string, optional): the name of the residue selector to
    apply the mover to.
"""

- go_back_to_step
"""Revert the environment state to a previous step in the trajectory. This
    action resets the environment to the state at the end of the specified
    step, allowing you to retry actions or explore different paths.

Arguments:
  - step (integer, optional): the number of the step to revert to, starting from
    0.
"""
```

B.2 Multi-turn interaction prompts

At each turn, Agent Rosetta is prompted to either refine the current design if the previous action was successful, or to correct its own mistakes. Here, we include the prompts used in our experiments, starting from design refinement.

Recall that refinement comprises two steps: action selection and parameter generation, each with its own prompt:

Design refinement—action selection prompt

```
The RosettaScripts environment successfully executed your action:

**Step Number:** {step}
**Action Name:** {act_name}
**Results:**

{state}

---

**History Summary:**

This is a summary of the previous steps with their actions and results:

{summary}

---

**Revision Instructions**:

1. Review:
  - Remind yourself of the design brief goal.
  - Carefully read the results from the last action.
2. Judge the results of the last action using the following criteria:
  - COMPOSITION.
  - BREAKDOWN OF ENERGY TERMS.
  - STRUCTURE.
3. Summarize:
  - What went well?
  - What went wrong?
  - What are the next steps to achieve the design goal?
4. Choose the next action, motivating your choice.

Follow the format specified in the system prompt.
```

Design refinement—parameter generation prompt

```
You chose to perform action ‘{act_name}‘. This is the full documentation of the
    action:

{docs}

---

**Action Instructions:**

1. Carefully read the documentation of the action you chose.
2. Write a detailed implementation plan that translates your thought process
    into the action call:
  - Motivate your choice of arguments and their values.
  - The expert biomolecular scientists on your team should be able to
    understand, review, and critique your plan in detail.
3. Write the full action call with all its arguments according to the format in
    the system prompt.
```

Error correction, instead, consists of one prompt only. We found that it is important to remind the model not to change the intended meaning of the wrong action for it not to remove problematic lines of code instead of trying to fix them:

Error correction prompt

```
The RosettaScripts environment encountered an error while executing the action:

{error}

**Instructions:**

1. Carefully read the error message.
2. Think about the possible causes of the error and come up with a solution:
  - Was the action call formatted correctly?
  - Did the action use valid Rosetta syntax?
3. Fix the error in the content of the message without changing its meaning.

Follow the format specified in the system prompt.
```

B.3 Design brief

Finally, we include the task prompt used in this work to stabilize backbone conformations. Task prompts should be precise and avoid ambiguities in order for the agent to correctly understand what the desired goals are.

Task prompt

```
**Design Brief:**

You are given a blank poly-glycine backbone structure generated by RFdiffusion.

Your task is to design the side chains of this structure such that:
1. It contains as few glycines as possible.
2. It is stable and has low Rosetta energy.
3. It folds into the same shape as the initial backbone structure.

At each step, you will be given the following information:
- The current sequence.
- The Rosetta energy breakdown of the current sequence.
- Structural information (all terms are in Angstroms, symbol "Å"):
  - RMSD to the initial structure: this measures the distance between the
    current structure and the initial structure.
  - RMSD to the ESMFold structure: this measures the distance between the
    current structure and the predicted ESMFold structure. This term is a proxy
    for the stability of the current structure.
  - Cavity volume: this measures the volume of the cavity in the current
    structure. A value of 20 Å^3 is approximately equivalent to one carbon atom.
  - Radius of gyration: this measures the compactness of the current structure
    around its center of mass.
- CA pLDDT: this measures the average confidence of the ESMFold prediction for
    the CA atoms of the current structure. It is a value between 0 and 1, where
    higher values indicate higher confidence.

**Starting State:**

{initial_state}
```

C Environment state

In this section, we include an example state of the RosettaScripts environment, and we verify the agent is able to extract meaningful information to guide its reasoning process.

The state includes the amino acid sequence, the breakdown of the Rosetta energy function terms, structural information, and a tabular summary of the previous actions in the design trajectory:

Environment state

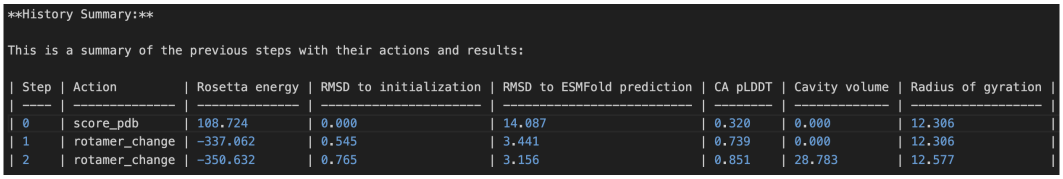
```
- Sequence:

LYS,GLU,GLU,GLU,ILE,LYS,THR,ILE,LEU,TRP,LYS,MET,PHE,LEU,ILE,VAL,MET,LYS,ILE,
    PHE,LEU,ASP,PRO,ARG,LEU,SER,ASP,GLU,GLU,ILE,LEU,TRP,LEU,ILE,ALA,LYS,LEU,
    MET,GLU,GLU,VAL,ALA,LYS,LYS,LEU,ASN,LEU,SER,LEU,ASP,GLN,LEU,ALA,LYS,PHE,
    LEU,ILE,LYS,LEU,ALA,ARG,GLN,TYR,LEU,ASP,GLU,ASP,LEU,ILE,ARG,LYS,LEU,GLU,THR,
    LEU,LEU,LYS,ILE,ILE,LEU,LYS,TRP,LEU,LYS,SER

- Rosetta energy terms:

fa_atr: -559.135 (+9.824)
fa_rep: 61.983 (-4.009)
fa_sol: 319.433 (+14.178)
fa_intra_sol_xover4: 15.113 (+0.029)
lk_ball_wtd: -10.933 (+2.150)
fa_intra_rep: 1.177 (-0.082)
fa_elec: -182.289 (-16.464)
pro_close: 0.134 (+0.134)
hbond_sr_bb: -65.597 (+0.328)
hbond_lr_bb: -1.747 (-0.125)
hbond_bb_sc: -11.760 (-1.938)
hbond_sc: -15.302 (-2.441)
dslf_fa13: 0.000 (+0.000)
rama_prepro: -17.624 (-0.229)
omega: 6.072 (+0.530)
fa_dun: 93.085 (-7.179)
p_aa_pp: -16.813 (+0.026)
yhh_planarity: 0.000 (-0.000)
ref: 33.569 (-8.302)
aspartimide_penalty: 0.000 (+0.000)

total energy: -350.632 (-13.571)

- Composition penalty:

before design: 326.000, after design: 0.000 (-326.000)

- Structure:

RMSD to initial structure: 0.765 (Å)
RMSD to ESMFold predicted structure: 3.156 (Å), CA pLDDT: 0.851 (0-1)
Cavity volume: 28.783 (Å^3)
Radius of gyration: 12.577 (Å)

---

**History Summary:**

see Figure B.1 below
```

for ease of readability, we include a screenshot of the tabular history summary below:

We include the reasoning trace of the agent for the particular example state above:

Agent Rosetta's reasoning

```
Reviewing the design brief and last action:
- The brief is to design side chains on a poly-glycine backbone such that (1)
    glycine is minimized, (2) the structure is stable with low energy, and (3)
    the backbone shape is maintained (low RMSD to initial structure).
```



```
**History Summary:**

This is a summary of the previous steps with their actions and results:

| Step | Action         | Rosetta energy | RMSD to initialization | RMSD to ESMFold prediction | CA pLDDT | Cavity volume | Radius of gyration |
| ---- | -------------- | -------------- | ---------------------- | -------------------------- | -------- | ------------- | ------------------ |
| 0    | score_pdb      | 108.724        | 0.000                  | 14.087                     | 0.320    | 0.000         | 12.306             |
| 1    | rotamer_change | -337.062       | 0.545                  | 3.441                      | 0.739    | 0.000         | 12.306             |
| 2    | rotamer_change | -350.632       | 0.765                  | 3.156                      | 0.851    | 28.783        | 12.577             |
```

Figure C.1: Screenshot of an example tabular history summary appended after the environment state.

```
Judging the results:
- COMPOSITION: The sequence is now almost entirely composed of non-glycine
    residues, with only one glycine remaining in the designed chain. The
    compositional penalty for glycine dropped from 326.0 to 0.0, which is a
    strong positive. There are no excessive Trp/Met residues.
- ENERGY: The total Rosetta energy improved to -350.6, a significant decrease
    from the initial +108.7. Key improvements are seen in fa_atr, fa_elec, and
    the removal of the glycine penalty. Sidechain energies (fa_dun, hbond_sc,
    etc.) are reasonable. There are no major outliers in the energy breakdown.
- STRUCTURE: The RMSD to the initial structure is 0.765 !@\AA{}!@, which is very
    close to the original backbone and within the design goals. RMSD to ESMFold
    prediction is 3.16 !@\AA{}!@ with a high CA pLDDT of 0.85, indicating
    strong predicted stability. Cavity volume is 28.8 !@\AA{}$^3$!@, suggesting
    some small internal cavities, but nothing egregious. Radius of gyration is
    nearly unchanged.

Summary:
- What went well: Glycine content is essentially eliminated, energy is
    favorable, and the structure is highly preserved with high predicted
    confidence.
- What went wrong: There is still a single glycine present in the sequence, and
    a small cavity has appeared (though the volume is not large).
- Next steps: To fully meet the brief, we should eliminate the last glycine
    residue, possibly by explicitly restricting glycine at all positions with a
    task operation. Additionally, a round of relaxation without compositional
    penalties may help optimize packing and reduce cavity volume.

Choice of next action:
- I will perform a rotamer_change with a task operation that explicitly
    prohibits glycine at all positions, ensuring complete glycine elimination.
    I will not apply the compositional penalty this time, as the sequence
    composition is now reasonable; instead, I will focus on the explicit
    exclusion of glycine.
```

We can see that Agent Rosetta is able to reason both about the compositional, energetic, and structural aspects of the current design in the context of the user-defined task brief, and to choose the next action accordingly.

D Actions

Rosetta offers several programming interfaces (*e.g.*, PyRosetta [19]). In this work, we use RosettaScripts [32]—the XML scripting interface—which offers more structured syntax compared to freeform Python code. This is important as it allows us to scaffold a set of flexible *types* of actions (*i.e.*, meta-actions) for which our agent, Agent Rosetta, generates the appropriate parameters. To achieve this, we leverage the ability of instruction-tuned LLMs to generate structured outputs that are easier to parse compared to text.

In this section, we include each action's XML script scaffold and an example action call generated by Agent Rosetta. In the XML scaffolds, '`{field_name}`' indicates a field that is formatted dynamically at runtime depending on the agent's action call. We remark that the `go_back_to` action does not run Rosetta, so it does not need XML scripting.

D.1 Rotamer change

The RosettaScripts XML scaffold for the action is:

`rotamer_change` XML scaffold

```
<ROSETTASCRIPTS>
    <SCOREFXNS>
    </SCOREFXNS>

    <PACKER_PALETTES>
    </PACKER_PALETTES>

    <RESIDUE_SELECTORS>
        {residue_selectors}
    </RESIDUE_SELECTORS>

    <TASKOPERATIONS>
        {task_operations}
    </TASKOPERATIONS>

    <SIMPLE_METRICS>
        <SequenceMetric name="record_sequence" output_mode="basename" />
    </SIMPLE_METRICS>

    <MOVERS>
        {comp_movers}
        <FastDesign name="fdes" repeats="1" {operations_names}>
            <MoveMap name="fdes_mm1">
                <Span begin="1" end="999" bb="true" chi="true" />
            </MoveMap>
        </FastDesign>
    </MOVERS>

    <PROTOCOLS>
        {comp_protocol}
        <Add mover="fdes" />
        <Add metrics="record_sequence" />
    </PROTOCOLS>

    <OUTPUT />
</ROSETTASCRIPTS>
```

The agent can specify the following parameters:

- `comp_movers`: the compositional penalties,
- `residue_selectors`: the regions of the peptide to apply the penalties to,
- `task_operations`: the Rosetta TaskOperations to constraint the design space,

and the remaining parameters are populated automatically by the environment.

An example action call generated by Agent Rosetta that uses all available parameters is:

Example `rotamer_change` action call

```
<action tag="run">
<name>rotamer_change</name>
<penalties>
  <item>
    <comp>
# Strongly penalize presence of glycines anywhere in the sequence
PENALTY_DEFINITION
TYPE GLY
SHAPE ABOVE
TARGET 0
```

```
BOUNDARY QUADRATIC
STRENGTH 100
END_PENALTY_DEFINITION
    </comp>
  </item>
  <item>
    <comp>
# Mildly favor large sidechains in the core to reduce cavity volume
PENALTY_DEFINITION
TYPE TRP TYR PHE MET LEU ILE
SHAPE BELOW
TARGET 0.35
RADIUS 0.1
BOUNDARY QUADRATIC
STRENGTH 5
END_PENALTY_DEFINITION
    </comp>
    <comp_selector_name>core</comp_selector_name>
  </item>
</penalties>
<residue_selectors>
<Layer name="core" select_core="true" select_boundary="false"
    select_surface="false" use_sidechain_neighbors="true"/>
</residue_selectors>
<task_operations>
<RestrictAbsentCanonicalAAS name="no_gly" keep_aas="ACDEFGHIKLMNPQRSTVWY"/>
</task_operations>
</action>
```

D.2 Backbone change

The RosettaScripts XML scaffold for the action is:

`backbone_change` XML scaffold

```
<ROSETTASCRIPTS>
    <SCOREFXNS>
    </SCOREFXNS>

    <PACKER_PALETTES>
    </PACKER_PALETTES>

    <RESIDUE_SELECTORS>
        {residue_selectors}
    </RESIDUE_SELECTORS>

    <MOVE_MAP_FACTORIES>
        {movemap_factory}
    </MOVE_MAP_FACTORIES>

    <TASKOPERATIONS>
    </TASKOPERATIONS>

    <SIMPLE_METRICS>
        <SequenceMetric name="record_sequence" output_mode="basename" />
    </SIMPLE_METRICS>

    <MOVERS>
        <{mover_name} name="backbone_change" {mover_params} />
        <MinMover name="min1" bb="false" chi="true" tolerance="1.0e-07"/>
    </MOVERS>

    <PROTOCOLS>
        <Add mover="backbone_change" />
        <Add mover="min1" />
        <Add metrics="record_sequence" />
    </PROTOCOLS>

    <OUTPUT />
</ROSETTASCRIPTS>
```

The agent can specify the following parameters:

- `mover_name`: the name of the backbone mover to use,
- `mover_params`: the parameters of the backbone mover,
- `residue_selectors`: the regions of the backbone to perturb,

and the remaining parameters are populated automatically by the environment depending on the chosen Mover.

We include an example action call generated by Agent Rosetta that uses the Small Mover:

Example `backbone_change` action call—Small mover

```
<action tag="run">
<name>backbone_change</name>
<mover_name>small</mover_name>
<mover_params>temperature="0.8" nmoves="20" angle_max="15"</mover_params>
</action>
```

and an example action call for the Backrub Mover:

Example `backbone_change` action call—Backrub mover

```
<action tag="run">
<name>backbone_change</name>
<mover_name>backrub</mover_name>
<mover_params>max_angle_disp_4="10.0" max_angle_disp_7="6.0"
    max_angle_disp_slope="0.5"</mover_params>
<residue_selectors>
<Layer name="core" select_core="true" select_boundary="true"
    select_surface="false"/>
</residue_selectors>
<mover_selector_name>core</mover_selector_name>
</action>
```

E Error correction

In this section, we include examples of Agent Rosetta reasoning traces when correcting its own mistakes. When an action fails, the agent is prompted with the offending action call and the error message from the RosettaScripts environment. We note that the action documentation is maintained in the LLM context during error correction. If error correction needs multiple interaction turns, we only keep the most recent error in the context.

E.1 Wrong compositional penalty blocks

Recall that the `rotamer_change` action accepts penalty definition blocks that penalize deviations from a target composition. This guides the Monte Carlo algorithm in Rosetta. It can happen, however, that Agent Rosetta tries to penalize structural properties of the chain, which are not allowed and cause the action to fail. We show how the agent reasons in such situations, starting with the wrong action call:

Wrong action call generated by Agent Rosetta

```
<action tag="run">
<name>rotamer_change</name>
<penalties>
```

```
  <item>
    <comp>
# Penalize cavity volume above 20 Å$^3$
PENALTY_DEFINITION
TYPE CAVITY
SHAPE ABOVE
TARGET 20
BOUNDARY QUADRATIC
STRENGTH 25
END_PENALTY_DEFINITION
    </comp>
  </item>
  <item>
    <comp>
# Penalize fa_rep above 60
PENALTY_DEFINITION
TYPE FA_REP
SHAPE ABOVE
TARGET 60
BOUNDARY QUADRATIC
STRENGTH 15
END_PENALTY_DEFINITION
    </comp>
  </item>
</penalties>
<task_operations>
# Restrict design to repacking (no mutations)
RestrictToRepacking name="repack_only"/>
# Add extra rotamers for better sampling
ExtraRotamersGeneric name="ex1_ex2" ex1="1" ex2="1"/>
</task_operations>
</action>
```

The penalty definition blocks include `TYPE CAVITY` and `TYPE FA_REP`, which are not valid, and the environment responds with the following error message:

RosettaScripts error message

```
The RosettaScripts environment encountered an error while executing the action:

<error>
ERROR: Error in core::scoring::aa_composition_energy
    ::AACompositionEnergySetup::parse_a_penalty_definition(): One or more
    three-letter residue type codes must be present after "TYPE" on a "TYPE"
    line.
ERROR:: Exit from:
    src/core/scoring/aa_composition_energy/AACompositionEnergySetup.cc line: 621
[ ERROR ]: Caught exception:
File: src/core/scoring/aa_composition_energy/AACompositionEnergySetup.cc:621
[ ERROR ] UtilityExitException
ERROR: Error in core::scoring::aa_composition_energy
    ::AACompositionEnergySetup::parse_a_penalty_definition(): One or more
    three-letter residue type codes must be present after "TYPE" on a "TYPE"
    line.
</error>
```

The error message specifies that the `TYPE` lines should only contain valid three-letter residue types, and the agent proceeds to correct the action call:

Agent Rosetta's error correction reasoning

```
The error message indicates that the "TYPE" line in the PENALTY_DEFINITION block
    requires one or more three-letter amino acid codes (e.g., GLY, ALA, VAL,
    etc.), but "CAVITY" and "FA_REP" are not valid residue types. These blocks
    are meant for compositional control of amino acid sequence, not for
    structural or energy term constraints like cavity volume or fa_rep.

For this first step, the priority is to reduce the glycine content while
    allowing all other amino acids, except perhaps cysteine (to avoid
    disulfides at this early stage). We'll penalize the presence of glycine and
    encourage the design protocol to replace as many glycines as possible with
    other amino acids, while the FastRelax protocol with side-chain design will
    optimize energy and packing.

I will:
- Apply a strong quadratic penalty on glycine content (TYPE GLY, SHAPE ABOVE,
    TARGET 2, BOUNDARY QUADRATIC, STRENGTH 200), aiming for as few glycines as
    possible.
- Not restrict other amino acids, to maximize design diversity.
- Allow design at all positions.
- Add extra rotamers for sampling.

No residue selectors are needed at this stage, as the penalty is applied
    globally.
```

Corrected action generated by Agent Rosetta

```
<action tag="run">
<name>rotamer_change</name>
<penalties>
  <item>
    <comp>
# Penalize glycine count above 2 to minimize glycine content
PENALTY_DEFINITION
TYPE GLY
SHAPE ABOVE
TARGET 2
BOUNDARY QUADRATIC
STRENGTH 200
END_PENALTY_DEFINITION
    </comp>
  </item>
</penalties>
<task_operations>
<ExtraRotamersGeneric name="ex1_ex2" ex1="1" ex2="1"/>
</task_operations>
</action>
```

E.2 TaskOperations syntax

TaskOperations are useful to restrict the search space for Rosetta's Monte Carlo algorithm. The sequence space grows exponentially with the length of the sequence, but, in many cases, certain positions can be restricted to assume only a subset of the available rotamers. This makes the design step more targeted and efficient.

TaskOperations, however, do not follow a unified nomenclature for parameters. At the same time, it is infeasible to provide prompting instructions about every TaskOperations available. Instead, we only provide general instructions on what TaskOperations can be used for in protein design task, and let the agent converse with the RosettaScripts environment to correct syntactical mistakes.

Here, we provide an example of such a situation. The agent is trying to prohibit the inclusion of proline and cysteine globally in the design. We start by including the offending action call with the error message from the environment:

Wrong action call generated by Agent Rosetta

```
<action tag="run">
<name>rotamer_change</name>
```

```
<penalties>
  <item>
    <comp>
# Penalize glycine content above 4, allow flexibility for backbone constraints
PENALTY_DEFINITION
TYPE GLY
SHAPE ABOVE
TARGET 4
BOUNDARY QUADRATIC
STRENGTH 30
END_PENALTY_DEFINITION
    </comp>
  </item>
  <item>
    <comp>
# Discourage any single non-glycine amino acid from exceeding 8 (softly)
PENALTY_DEFINITION
NOT_TYPE GLY
SHAPE ABOVE
TARGET 8
BOUNDARY QUADRATIC
STRENGTH 6
END_PENALTY_DEFINITION
    </comp>
  </item>
</penalties>
<task_operations>
  <ProhibitSpecifiedBaseResidueTypes name="nopro_cys" aa="PRO CYS"/>
</task_operations>
</action>
```

RosettaScripts error message

```
The RosettaScripts environment encountered an error while executing the action:

<error>
[ ERROR ]: Caught exception:
File: src/protocols/rosetta_scripts/RosettaScriptsParser.cc:1217
Input rosetta scripts XML file "./rotamer_change_11.xml" failed to validate
    against the rosetta scripts schema. Use the option -parser::output_schema
    <output filename> to output the schema to a file to see all valid options.
Your XML has failed validation.  The error message below will tell you where in
    your XML file the error occurred.  Here's how to fix it:
1) If the validation fails on something obvious, like an illegal attribute due
    to a spelling error (perhaps you used scorefnction instead of
    scorefunction), then you need to fix your XML file.
2) If you haven't run the XML rewriter script and this might be pre-2017 Rosetta
    XML, run the rewriter script (tools/xsd_xrw/rewrite_rosetta_script.py) on
    your input XML first.  The attribute values not being in quotes
    (scorefunction=talaris2014 instead of scorefunction="talaris2014") is a
    good indicator that this is your problem.
3) If you are a developer and neither 1 nor 2 worked - email the developer's
    mailing list or try Slack.
4) If you are an academic or commercial user - try the Rosetta Forums
    https://www.rosettacommons.org/forum
Error messages were:
From line 18:
Error: Element 'ProhibitSpecifiedBaseResidueTypes', attribute 'aa': The
    attribute 'aa' is not allowed.
13:     <RESIDUE_SELECTORS>
14:
15:     </RESIDUE_SELECTORS>
16:
17:     <TASKOPERATIONS>
```

```
18:         <ProhibitSpecifiedBaseResidueTypes name="nopro_cys" aa="PRO CYS"/>
19:     </TASKOPERATIONS>
20:
21:     <SIMPLE_METRICS>
22:         <SequenceMetric name="record_sequence" output_mode="basename" />
23:     </SIMPLE_METRICS>
------------------------------------------------------------
Warning messages were:
------------------------------------------------------------
</error>
```

This message shows that the RosettaScripts error indicates the wrong use of the attribute aa in the action call. However, it does not provide any more indications of what should be done to correct it. We stress that this is a nontrivial task for the agent. Now, we include the reasoning trace of Agent Rosetta after being prompted to correct its mistakes in the action call:

Agent Rosetta’s error correction reasoning

```
The error message states: "Element ’ProhibitSpecifiedBaseResidueTypes’,
    attribute ’aa’: The attribute ’aa’ is not allowed." This means that the XML
    syntax for specifying which amino acids to prohibit in the
    ‘ProhibitSpecifiedBaseResidueTypes‘ TaskOperation is incorrect. In modern
    RosettaScripts, the correct attribute is likely ‘amino_acids‘ rather than
    ‘aa‘, or (more commonly) a child element such as ‘<residue_name3>‘ or
    ‘<residue_name>‘ for specifying a list.

Alternatively, in many recent RosettaScripts, the ‘RestrictAbsentCanonicalAAS‘
    operation is used, which takes an attribute ‘keep_aas‘ listing the allowed
    amino acids. This is the most robust way to restrict the design space to
    canonical amino acids, excluding, for example, proline and cysteine.

Plan:
- Replace ‘ProhibitSpecifiedBaseResidueTypes‘ with ‘RestrictAbsentCanonicalAAS‘.
- Use ‘keep_aas‘ to specify all canonical amino acids except PRO and CYS.
- This will prevent the design protocol from introducing proline or cysteine,
    which are special-case residues (proline disrupts backbone flexibility,
    cysteine can form disulfides).
- Keep the compositional penalties unchanged, as they are valid Rosetta
    compositional penalty blocks.
```

The agent reflects on the possible causes of the error and reasons on which options it has to correct its mistake. We stress that, in our prompts, we never include information about the TaskOperations mentioned in the reasoning trace, meaning this was already included in the training data of the underlying LLM. Agent Rosetta correctly recognizes there is not only one way to go about fixing this error, and it reasons on which approach might be the most robust. To conclude, we include the corrected action call, which executed successfully:

Corrected action call generated by Agent Rosetta

```
<action tag="run">
<name>rotamer_change</name>
<penalties>
  <item>
    <comp>
# Penalize glycine content above 4, allow flexibility for backbone constraints
PENALTY_DEFINITION
TYPE GLY
SHAPE ABOVE
TARGET 4
BOUNDARY QUADRATIC
STRENGTH 30
END_PENALTY_DEFINITION
    </comp>
  </item>
  <item>
```

```
    <comp>
# Discourage any single non-glycine amino acid from exceeding 8 (softly)
PENALTY_DEFINITION
NOT_TYPE GLY
SHAPE ABOVE
TARGET 8
BOUNDARY QUADRATIC
STRENGTH 6
END_PENALTY_DEFINITION
    </comp>
  </item>
</penalties>
<task_operations>
  <RestrictAbsentCanonicalAAS name="nopro_cys" keep_aas="ADEFGHIKLMNQRSTVWY"/>
</task_operations>
</action>
```

E.3 Removing offending lines

Agent Rosetta is explicitly prompted to correct mistakes without changing the meaning of the wrong action call. This is important because a trivial way to fix errors is to remove the problematic lines, which may change the intended effect of the action. It can happen, however, that the agent gets stuck in trying to fix syntax errors. For example, consider again TaskOperations syntax. The agent may oscillate between action calls using `aa="PRO CYS"` or `residues="PRO CYS"`, failing to use the correct `keep_aas="..."` syntax. This happens because we only keep the last wrong action in the context of the LLM, and discard previous error correction attempts. This choice is motivated by keeping a context window of fixed size throughout the trajectory, and alternatives exist.

Situations like this cause error correction loops the agent struggles to escape. We noticed that, usually after a few rounds of error correction, the agent removes the wrong lines. We include such an example below:

Wrong action call generated by Agent Rosetta

```
<action tag="run">
<name>rotamer_change</name>
<penalties>
  <item>
    <comp>
# Strong penalty for excess glycine, but tolerate up to 2±2
PENALTY_DEFINITION
TYPE GLY
SHAPE OUTSIDE
TARGET 2
RADIUS 2
BOUNDARY QUADRATIC
STRENGTH 100
END_PENALTY_DEFINITION
    </comp>
  </item>
</penalties>
<task_operations>
  <ProhibitSpecifiedBaseResidueTypes name="no_cys" residues="CYS" />
</task_operations>
</action>
```

RosettaScripts error message

```
The RosettaScripts environment encountered an error while executing the action:

<error>
[ ERROR ]: Caught exception:
File: src/protocols/rosetta_scripts/RosettaScriptsParser.cc:1217
```

```
Input rosetta scripts XML file "./rotamer_change_1.xml" failed to validate
    against the rosetta scripts schema. Use the option -parser::output_schema
    <output filename> to output the schema to a file to see all valid options.
Your XML has failed validation.  The error message below will tell you where in
    your XML file the error occurred.  Here's how to fix it:
1) If the validation fails on something obvious, like an illegal attribute due
    to a spelling error (perhaps you used scorefnction instead of
    scorefunction), then you need to fix your XML file.
2) If you haven't run the XML rewriter script and this might be pre-2017 Rosetta
    XML, run the rewriter script (tools/xsd_xrw/rewrite_rosetta_script.py) on
    your input XML first.  The attribute values not being in quotes
    (scorefunction=talaris2014 instead of scorefunction="talaris2014") is a
    good indicator that this is your problem.
3) If you are a developer and neither 1 nor 2 worked - email the developer's
    mailing list or try Slack.
4) If you are an academic or commercial user - try the Rosetta Forums
    https://www.rosettacommons.org/forum
Error messages were:
From line 18:
Error: Element 'ProhibitSpecifiedBaseResidueTypes', attribute 'residues': The
    attribute 'residues' is not allowed.
13:     <RESIDUE_SELECTORS>
14:
15:     </RESIDUE_SELECTORS>
16:
17:     <TASKOPERATIONS>
18:         <ProhibitSpecifiedBaseResidueTypes name="no_cys" residues="CYS"/>
19:     </TASKOPERATIONS>
20:
21:     <SIMPLE_METRICS>
22:         <SequenceMetric name="record_sequence" output_mode="basename" />
23:     </SIMPLE_METRICS>
------------------------------------------------------------
Warning messages were:
------------------------------------------------------------
</error>
```

Here, the agent reasons that given the early stage of the design, it might be reasonable to drop the wrong line completely:

Agent Rosetta's error correction reasoning

```
The error message indicates that the attribute 'residues' is not allowed for the
    'ProhibitSpecifiedBaseResidueTypes' TaskOperation in the XML. This suggests
    the correct attribute should likely be 'aa' (the usual RosettaScripts
    attribute for specifying amino acids for this TaskOperation), or possibly
    that the syntax for this TaskOperation is slightly different.

Given that the main design goal here is to reduce glycines and maintain
    stability-not specifically to prohibit cysteines-it may be prudent to
    remove the erroneous TaskOperation for now and focus on the compositional
    penalty for glycine, which is the primary requirement of the design brief
    at this stage.

If desired later, a more advanced specification for allowed/disallowed amino
    acids can be reintroduced with valid syntax.

Thus, I will remove the task_operations section for now and focus on a strong
    compositional penalty to minimize glycines.
```

Corrected action call generated by Agent Rosetta

```
<action tag="run">
<name>rotamer_change</name>
<penalties>
```

```
  <item>
    <comp>
# Strong penalty for excess glycine, but tolerate up to 2±2
PENALTY_DEFINITION
TYPE GLY
SHAPE OUTSIDE
TARGET 2
RADIUS 2
BOUNDARY QUADRATIC
STRENGTH 100
END_PENALTY_DEFINITION
    </comp>
  </item>
</penalties>
</action>
```

F Simplified composition penalty blocks syntax

Here, we expand on the simplified syntax for RosettaScripts' `rotamer_change` composition penalty blocks.

First, for the sake of completeness, we include the system prompt used to generate the example wrong penalty block included in Fig. 2. We note that this was an earlier iteration compared to the final system prompt included in Appendix B. At this stage in the development of our framework, the objective was to investigate whether general purpose LLMs have been trained on RosettaScripts and can generate syntactically valid penalty definition blocks:

System prompt—wrong penalty block

```
You are a helpful RosettaScripts agent that supports scientists in biomolecular
    design tasks.

Your goal is to translate a high-level design brief (appended after this prompt)
    into a valid ‘*.comp‘ file which contains one or more ‘PENALTY_DEFINITION‘
    blocks.
Each block defines how Rosetta penalizes (or rewards) certain amino acid types
    or residue properties, depending on their relative abundance in the peptide
    sequence.

Your goal is to generate penalty blocks that capture the design goals precisely,
    balancing global and local constraints.
The combined penalties should steer the Monte-Carlo search in Rosetta toward the
    task objective while avoiding mutually impossible requirements.

You have access to a RosettaScripts environment to evaluate your output ‘*.comp‘
    file and then refine it.

---

**Syntax Rules:**

Each penalty definition block must use this valid Rosetta syntax:

PENALTY_DEFINITION
- One or more of:
  TYPE <restype1> <restype2> ...              # a residue should be counted if
    its three-letter code matches ANY of the names provided
  OR_PROPERTIES <property1> <property2> ...   # a residue should be counted if
    it has ANY of the properties listed
  PROPERTIES <property1> <property2> ...      # a residue should be counted if
    it has ALL of the properties listed
  NOT_TYPE <restype1> <restype2> ...          # a residue should NOT be counted
    if its three-letter code matches ANY of the names provided
  NOT_PROPERTIES <property1> <property2> ...  # a residue should NOT be counted
    if it has ANY of the properties listed
- Exactly one of:
  ABSOLUTE <int>                              # target absolute count of
    residues that match the penalty definition
  DELTA_START <int>                           # target range start, relative to
    target count, can be negative
  DELTA_END <int>                             # target range end, relative to
    target count
  -or-
  FRACTION <float>                            # target fraction of the residues
    that match the penalty definition (e.g., 0.05 for 5%)
  FRACT_DELTA_START <float>                   # target range start, relative to
    target fraction, can be negative
  FRACT_DELTA_END <float>                     # target range end, relative to
    target fraction
PENALTIES <float1> <float2> <float3> ...      # one or more values, interpolated
    over the range
BEFORE_FUNCTION <shape>                       # shape before DELTA_START,
    default is QUADRATIC. Can be QUADRATIC, LINEAR, or CONSTANT.
AFTER_FUNCTION <shape>                        # shape after DELTA_END, default
    is QUADRATIC. Can be QUADRATIC, LINEAR, or CONSTANT.
END_PENALTY_DEFINITION

A residue is counted if the following boolean conditions are satisfied:
( any TYPE matches ) OR (( no NOT_TYPE matches ) AND (( no NOT_PROPERTIES
    property is present) AND ((no PROPERTIES or OR_PROPERTIES are defined) OR (
    all PROPERTIES are present ) OR ( any OR_PROPERTIES are present ))))

Each PENALTY_DEFINITION block must:
- Contain one or more of the OR_PROPERTIES, PROPERTIES, TYPE, NOT_TYPE,
    NOT_PROPERTIES definitions.
- Contain exactly one of ABSOLUTE or FRACTION, and never both. The numeric value
    defines the target composition and must be included.
- If the ABSOLUTE option is used, DELTA_START and DELTA_END must be integers.
- If the FRACTION option is used, DELTA_START and DELTA_END must be floats.
- The shape parameter for BEFORE_FUNCTION and AFTER_FUNCTION must be uppercase.

---

**Range Computation Rules:**

This is how RosettaScripts computes the range for each penalty definition:

- Case 1: ABSOLUTE
  - The range is computed as:
    - MIN_RANGE = target count + DELTA_START.
    - MAX_RANGE = target count + DELTA_END.
  - Example: If target count is 10, DELTA_START is -2, and DELTA_END is 3, the
    range is [8, 13].

- Case 2: FRACTION
  Suppose the sequence has N residues. Then,
  - The target count is computed as target count = target fraction * N.
  - The range is computed as:
    - MIN_RANGE = target count + N * floor(FRACT_DELTA_START * N)
    - MAX_RANGE = target count + N * ceil(FRACT_DELTA_END * N)
  - Example: If N is 100, target fraction is 0.1, FRACT_DELTA_START is -0.05,
    and FRACT_DELTA_END is 0.02, the range is [5, 12].

In both cases, MIN_RANGE must be stricly smaller than MAX_RANGE, i.e. MIN_RANGE
    < MAX_RANGE.

---
```

```
**Instructions:**

- Generate penalty blocks based on the given design brief.
- Refine them based on feedback from the RosettaScripts environment.
- Ensure valid Rosetta syntax in every block.

First, read the design brief and think about what penalty definitions would be
    appropriate.
Remember to take into consideration the starting ‘*.pdb‘ backbone strucutre.
Then, output the corresponding ‘*.comp‘ file content.

Your thought process must be enclosed in <think> </think> tags, and the ‘*.comp‘
    file content must be enclosed in <comp> </comp> tags.
```

Next, we compare the original RosettaScripts syntax with our simplified version side-by-side.

Original RosettaScripts syntax

```
PENALTY_DEFINITION
- One or more of:
  TYPE <restype1> <restype2> ...
  OR_PROPERTIES <property1> <property2> ...
  PROPERTIES <property1> <property2> ...
  NOT_TYPE <restype1> <restype2> ...
  NOT_PROPERTIES <property1> <property2> ...
- Exactly one of:
  ABSOLUTE <int>
  DELTA_START <int>
  DELTA_END <int>
  -or-
  FRACTION <float>
  FRACT_DELTA_START <float>
  FRACT_DELTA_END <float>
PENALTIES <float1> <float2> <float3> ...
BEFORE_FUNCTION <shape>
AFTER_FUNCTION <shape>
END_PENALTY_DEFINITION
```

Simplified syntax

```
PENALTY_DEFINITION
- Exactly one of:
  TYPE <restype1> <restype2>
  NOT_TYPE <restype1> <restype2>
  PROPERTIES <property>
  NOT_PROPERTIES <property>
SHAPE <OUTSIDE | ABOVE | BELOW>
TARGET <int or float>
RADIUS <int or float>
BOUNDARY <function>
STRENGTH <int>
END_PENALTY_DEFINITION
```

We can appreciate how the simplified syntax is more concise and abstract in order to better align with the intentions of the agent. The main differences between the two are that:

- The simplified syntax accepts only one of types or properties.
- It unifies fractional and absolute targets with `TARGET` and `RADIUS`.
- It eliminates the `PENALTIES` line, which is the most complicated for LLMs to generate correctly. Instead, it uses a `SHAPE` parameter that directly expresses the shape of the penalty block, which is more intuitive for the agent to correctly generate.

We note that these changes do restrict the expressivity of the syntax. However, they retain the basic block shapes that are most commonly used in design tasks. We include a few example simplified blocks generated by Agent Rosetta and their translation to valid RosettaScripts syntax:

Simplified syntax

```
# Favor hydrophobics in the core
PENALTY_DEFINITION
TYPE ALA VAL LEU ILE MET PHE TRP TYR
SHAPE OUTSIDE
TARGET 0.7
RADIUS 0.15
BOUNDARY QUADRATIC
STRENGTH 30
END_PENALTY_DEFINITION
```

RosettaScripts syntax

```
PENALTY_DEFINITION
TYPE ALA VAL LEU ILE MET PHE TRP TYR
FRACTION 0.7
FRACT_DELTA_START -0.20
FRACT_DELTA_END 0.20
PENALTIES 30 0 30
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION
```

Simplified syntax

```
# Penalize more than 2 Tryptophans
PENALTY_DEFINITION
TYPE TRP
SHAPE ABOVE
TARGET 2
BOUNDARY QUADRATIC
STRENGTH 50
END_PENALTY_DEFINITION
```

RosettaScripts syntax

```
PENALTY_DEFINITION
TYPE TRP
ABSOLUTE 2
DELTA_START -1
DELTA_END 1
PENALTIES 0 0 50
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION
```

Simplified syntax

```
# Favor polar/charged residues at
    the surface (at least 45\%)
PENALTY_DEFINITION
TYPE SER THR ASN GLN ASP GLU LYS ARG
    HIS
SHAPE BELOW
TARGET 0.45
BOUNDARY QUADRATIC
STRENGTH 20
END_PENALTY_DEFINITION
```

RosettaScripts syntax

```
PENALTY_DEFINITION
TYPE SER THR ASN GLN ASP GLU LYS ARG
    HIS
FRACTION 0.45
FRACT_DELTA_START -0.05
FRACT_DELTA_END 0.05
PENALTIES 20 0 0
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION
```

The intentions of the agent in the comment at the top of each block are now in agreement with the shape of the penalty blocks, differently from the example in Fig. 2.

G Example reasoning traces

In this section, we include example reasoning traces generated by the agent while executing our task of stabilizing poly-glycine backbone conformations. We include a representative example for each of the available actions.

G.1 `rotamer_change` action

Here, we include an example interaction step where the agent chose to perform a `rotamer_change` action that overall decreased the energy of the design. We start by including the environment state:

RosettaScripts environment state

```
The RosettaScripts environment successfully executed your action:

**Step Number:** 12
**Action Name:** rotamer_change
**Results:**

- Sequence:

SER,GLU,GLU,GLU,ILE,ARG,ARG,ILE,VAL,PHE,TRP,ALA,ALA,MET,GLU,CYS,ALA,LYS,TYR,ILE,
    PHE,ASP,PRO,ARG,ILE,PRO,ASP,GLU,GLU,ILE,ILE,ARG,ARG,ILE,MET,GLU,ILE,ASN,LYS,
    LYS,THR,LEU,GLU,TRP,ALA,LYS,ILE,SER,PRO,ASP,GLU,LEU,TYR,LYS,TYR,THR,LYS,GLU,
    ILE,ALA,SER,GLN,TYR,TYR,ASP,GLU,GLU,GLN,LEU,LYS,LYS,MET,LEU,LYS,TYR,VAL,LYS,
    ILE,ALA,LEU,ASP,GLN,LEU,LYS,SER

- Rosetta energy terms:

fa_atr: -565.030 (+4.816)
fa_rep: 64.214 (-7.652)
fa_sol: 360.466 (+49.876)
fa_intra_sol_xover4: 19.689 (+2.026)
```

```
lk_ball_wtd: -10.704 (-0.049)
fa_intra_rep: 1.127 (-0.053)
fa_elec: -203.623 (-21.169)
pro_close: 0.229 (+0.102)
hbond_sr_bb: -64.322 (+1.799)
hbond_lr_bb: -1.530 (+0.282)
hbond_bb_sc: -11.683 (-0.748)
hbond_sc: -20.367 (-2.576)
dslf_fa13: 0.000 (+0.000)
rama_prepro: -14.086 (+1.555)
omega: 5.987 (+0.275)
fa_dun: 114.092 (+14.475)
p_aa_pp: -14.607 (+3.305)
yhh_planarity: 0.030 (+0.015)
ref: 5.987 (-33.975)
aspartimide_penalty: 0.000 (+0.000)

total energy: -334.131 (+12.304)

- Composition penalty:

before design: 900.000, after design: 0.000 (-900.000)

- Structure:

RMSD to initial structure: 2.094 (Å)
RMSD to ESMFold predicted structure: 0.422 (Å), CA pLDDT: 0.932 (0-1)
Cavity volume: 23.177 (Å^3)
Radius of gyration: 12.183 (Å)
```

The reasoning of the agent is as follows

Agent Rosetta's reasoning trace to choose the next action

```
Review of the design brief:
- The goal is to design side chains for a poly-glycine backbone so that the
    final sequence has as few glycines as possible, is stable and low in
    energy, and preserves the initial backbone shape.

Results of the last action:
- The current sequence contains zero glycines, fulfilling the "as few glycines
    as possible" criterion.
- Energy: The total energy is -334.131, which is quite favorable, and the
    composition penalty is now zero.
- Structure: The RMSD to the initial backbone is 2.094 Å (slightly increased
    from previous steps but still reasonable for design). The RMSD to the
    ESMFold prediction is extremely low at 0.422 Å, and the CA pLDDT is high
    (0.932), suggesting high predicted stability/confidence. The radius of
    gyration is stable (~12.2 Å), and the cavity volume is small (23 Å^3,
    roughly one residue-sized cavity).

What went well:
- The sequence is now fully designed with no glycines.
- Energetics are favorable and the structure is predicted to be stable by
    ESMFold.
- The protein is compact and lacks large cavities.

What could be improved:
- RMSD to the design backbone has increased a bit (2.094 Å), but this is still
    within a reasonable range.
- The energy is not as low as the best step (-348.882), suggesting room for
    further optimization.
- Cavity volume is low, but could possibly be improved more by repacking.
```

```
Next steps:
- Since the sequence already has no glycines, and the structure is stable with
    high confidence, the main avenue for improvement is to further optimize the
    side-chain packing and overall energy.
- Small, local repacking/design may help lower the energy and perhaps slightly
    reduce the RMSD to the original backbone.
- Therefore, I will run another round of ‘rotamer_change‘ (FastRelax with
    design) without compositional penalties, to allow for local optimization
    and side-chain refinement. This will focus solely on energetic improvement,
    as the composition is already satisfactory.
```

And, after receiving the action's documentation, the agent reasons again before generating the full action call:

Agent Rosetta's reasoning trace to generate the full action call

```
Review and Plan:
- The design brief requires minimal glycine, low energy, and preservation of the
    backbone structure.
- The previous step produced a well-packed, low-energy, diverse sequence with
    only a few proline and no glycine residues. The RMSD to the initial
    backbone is 2.094 Å (excellent), and the RMSD to ESMFold is 0.422 Å with a
    very high CA pLDDT (0.932), indicating a stable, predictable fold.
- The energy is very low (-334), with no composition penalty, and the sequence
    contains very few glycine residues (possibly none) and only a reasonable
    number of prolines (3).
- Cavity volume is small (23 Å$^3$), so packing is good.
- However, some energy terms have room for improvement: fa_sol, fa_rep, and
    fa_dun are somewhat elevated, suggesting that a round of design focusing on
    optimizing packing and rotamers could further lower energy.
- Additional fine-tuning could be achieved by discouraging overuse of
    large/bulky hydrophobics, limiting surface-exposed hydrophobics, and
    ensuring at least a minimal level of glycine (<2) to alleviate possible
    backbone strain.
- No residue selectors or task operations have been used yet. Adding a selector
    for surface residues and penalizing hydrophobics on the surface could help
    further.

Implementation Plan:
- Keep global compositional penalties similar to last time for overall amino
    acid balance.
- Add a penalty for glycine ABOVE 2 (to ensure minimal glycine, in case a few
    are present).
- Use a Layer selector to define "surface" residues and apply a penalty for
    hydrophobic residues (LEU, ILE, VAL, MET, PHE, TRP) on the surface ABOVE a
    small target.
- For task operations, allow all positions to be designable except cysteines and
    prolines (to avoid introducing new CYS/PRO unless already present).
- This should further improve packing, surface polarity, and lower
    solvation/repulsion energies.

Arguments:
- penalties: global balance, glycine penalty, surface hydrophobic penalty.
- residue_selectors: Layer selector for surface.
- task_operations: RestrictToRepacking for CYS/PRO residues.
```

We can see that, initially, the agent is planning on running a round of side chain optimization with no compositional penalty blocks, letting Rosetta optimize for energy with no constraints. Instead, after receiving the action documentation, the agent generates a well-detailed plan, changing its intentions. This stresses the importance of prompting the agent to reason again to generate the full action call. We omit the full action call for the sake of conciseness.

G.2 `backbone_change` action

We include an example interaction step where the agent chose to perform a `backbone_change` action. We start by including the environment state:

RosettaScripts environment state

```
The RosettaScripts environment successfully executed your action:

**Step Number:** 3
**Action Name:** rotamer_change
**Results:**

- Sequence:

GLU,GLU,GLU,GLU,ARG,ARG,ARG,ILE,MET,LEU,TRP,ALA,PHE,GLU,LEU,VAL,MET,LYS,ILE,
    VAL,LEU,ASP,PRO,ARG,LEU,SER,GLU,GLU,GLU,LYS,LEU,ARG,LEU,ILE,ARG,GLU,ILE,
    VAL,ILE,TRP,ALA,ALA,ARG,VAL,LEU,ASN,VAL,PRO,TRP,GLU,TRP,VAL,ASN,LYS,VAL,
    ILE,PHE,TRP,LEU,ALA,THR,LEU,TYR,PHE,ASP,GLU,TRP,TRP,LEU,LYS,ILE,ILE,LEU,
    THR,LEU,LEU,ILE,LYS,THR,LEU,LYS,LEU,MET,LEU,SER

- Rosetta energy terms:

fa_atr: -571.650 (-14.577)
fa_rep: 68.660 (+4.611)
fa_sol: 313.981 (+4.157)
fa_intra_sol_xover4: 16.644 (+0.688)
lk_ball_wtd: -12.839 (-3.026)
fa_intra_rep: 1.235 (+0.008)
fa_elec: -173.336 (+3.040)
pro_close: 0.180 (-0.043)
hbond_sr_bb: -64.824 (-0.651)
hbond_lr_bb: -1.597 (+0.164)
hbond_bb_sc: -10.826 (-0.591)
hbond_sc: -12.111 (+3.466)
dslf_fa13: 0.000 (+0.000)
rama_prepro: -14.994 (+0.677)
omega: 4.814 (-1.668)
fa_dun: 88.577 (-4.761)
p_aa_pp: -18.100 (+0.546)
yhh_planarity: 0.001 (+0.000)
ref: 53.848 (+5.434)
aspartimide_penalty: 0.000 (+0.000)

total energy: -332.336 (-2.525)

- Composition penalty:

before design: 0.000, after design: 0.000 (+0.000)

- Structure:

RMSD to initial structure: 0.784 (Å)
RMSD to ESMFold predicted structure: 10.840 (Å), CA pLDDT: 0.720 (0-1)
Cavity volume: 20.852 (Å$^3$)
Radius of gyration: 12.391 (Å)
```

The reasoning of the agent is as follows:

Agent Rosetta's reasoning trace to choose the next action

```
Reviewing the design brief, our goals are:
1. Minimize glycine content (ideally eliminate all glycines).
2. Achieve a stable, low-energy sequence.
3. Maintain the backbone conformation (low RMSD to the initial structure).
```

From the last action:

- COMPOSITION: All glycines have been eliminated, which is good. The sequence is now diverse, with a mix of hydrophobic and polar residues.
- ENERGY: The total energy is -332.336, which is a favorable score. Most energy terms (e.g., fa_atr, hbond terms) are improved compared to the starting poly-Gly. However, the fa_dun and fa_sol terms are somewhat high, and the cavity volume is >20 $\AA^3$, suggesting the presence of voids inside the structure.
- STRUCTURE: RMSD to the initial structure is 0.784 Å (acceptable, indicating backbone preservation). However, RMSD to the ESMFold prediction has jumped to 10.84 Å, and CA pLDDT dropped to 0.72, indicating a loss of stability and poor confidence in the predicted fold.

Summary:

- What went well: Glycines successfully removed; overall energy is low; backbone shape is still similar to initial structure.
- What went wrong: ESMFold RMSD increased (structure deviates from a stable fold); CA pLDDT confidence dropped; cavity volume increased.
- Next steps: The main priority is to recover structural stability and compactness. The high cavity volume and loss of ESMFold confidence suggest that sidechain packing is suboptimal. We should perform a round of backbone relaxation (with small backbone moves) to optimize both sidechain and backbone conformations, helping close cavities and stabilize the structure.

Action Choice:

- I will use the ‘backbone_change‘ action with the ‘small‘ mover to allow small backbone adjustments throughout the structure to facilitate better sidechain packing while preserving the overall fold.

And, after seeing the action’s documentation, the agent reasons again before generating the full action call:

Agent Rosetta’s reasoning trace to generate the full action call

Review of the design brief and results:

- The goal is to design a sequence with minimal glycine content, low energy, and high structural fidelity to the initial backbone.
- The previous design step replaced all glycines with a diverse set of amino acids, leading to a compositionally rich sequence (no glycines) and a significant drop in total energy (-332.34 vs +108.72).
- However, after the first design step, RMSD to ESMFold prediction increased sharply (0.531 -> 10.84 Å), and CA pLDDT dropped (0.918 -> 0.720), indicating a probable loss of structure or stability.
- This suggests the sidechain introduction perturbed the backbone, and the sequence is currently not optimal for its intended fold. Cavity volume increased (0.0 -> 20.85 $\AA^3$), possibly indicating poor packing or undesired voids.

What went well:

- Glycine minimization was achieved.
- Energy improved dramatically.
- The structure is still fairly close to the starting backbone (RMSD 0.784 $\AA^3$).

What went wrong:

- The structure deviated strongly from the ESMFold-predicted model, and predicted confidence dropped.
- Cavity formation may indicate suboptimal packing.

Next steps:

- The backbone likely needs to adapt to the new side chains to restore stability and fold (as measured by ESMFold RMSD and CA pLDDT).

```
- A gentle backbone relaxation is warranted to allow the structure to
    accommodate new side chains, possibly improving the fold and stability
    without losing the initial global fold (RMSD to initial structure is still
    reasonable).
- I will use the ‘small‘ backbone mover, which makes small torsion angle moves.
    This is less aggressive than backrub and appropriate for global refinement
    after large sequence changes.
- I will apply it globally, as all regions have undergone design.
- I will set a moderate temperature (temperature="1.0"), a reasonable number of
    moves (nmoves="5"), and a small angle (angle_max="8.0"), to allow
    adaptation without radical changes.

Action Plan:
- Use the ‘small‘ mover.
- Set temperature="1.0", nmoves="5", angle_max="8.0".
- Apply to all residues (no selector required).
```

As with the example reasoning trace for a `rotamer_change` action call above, we can see that the plan of the agent after seeing the action's documentation is well-structured and implementable.

G.3 `go_back_to` action

Finally, we include an example scenario where the agent chose to revert to a previous node in the trajectory. This is important, as task objectives such as limiting the deviation of the current design from the initial structure may be difficult to control via compositional constraints. In these scenarios, a reasonable choice is to discard the results of the previous action rather than trying to correct them.

As above, we start by including the environment state:

RosettaScripts environment state

```
The RosettaScripts environment successfully executed your action:

**Step Number:** 3
**Action Name:** rotamer_change
**Results:**

- Sequence:

GLU,GLU,GLU,GLU,ARG,GLN,THR,ILE,TRP,THR,ILE,VAL,PHE,LEU,ILE,THR,LEU,SER,ILE,VAL,
    LEU,ASN,PRO,ASN,LEU,SER,THR,THR,GLN,ILE,ILE,LEU,ILE,LEU,ALA,ALA,LEU,ALA,
    ALA,TRP,THR,MET,LYS,VAL,LEU,GLN,VAL,ASP,LEU,LEU,GLN,ILE,ILE,ILE,TYR,LEU,
    LEU,LEU,LEU,PHE,GLN,GLN,TYR,LEU,ASN,GLU,GLU,GLN,LEU,LYS,TYR,VAL,LEU,TRP,
    VAL,LEU,LEU,VAL,MET,TYR,ALA,TRP,MET,GLN,SER

- Rosetta energy terms:

fa_atr: -559.659 (+7.489)
fa_rep: 67.266 (+0.513)
fa_sol: 292.881 (-22.337)
fa_intra_sol_xover4: 15.324 (-1.029)
lk_ball_wtd: -14.919 (-0.022)
fa_intra_rep: 1.303 (+0.078)
fa_elec: -159.300 (+10.382)
pro_close: 0.130 (-0.099)
hbond_sr_bb: -63.255 (+0.998)
hbond_lr_bb: -1.682 (+0.126)
hbond_bb_sc: -17.451 (-0.883)
hbond_sc: -8.091 (+3.396)
dslf_fa13: 0.000 (+0.000)
rama_prepro: -11.106 (+3.629)
omega: 6.524 (+0.961)
fa_dun: 82.443 (-22.875)
p_aa_pp: -16.877 (-1.241)
```

```
yhh_planarity: 0.002 (-0.014)
ref: 67.262 (+35.126)
aspartimide_penalty: 0.000 (+0.000)

total energy: -319.203 (+14.199)

- Composition penalty:

before design: 150.000, after design: 0.000 (-150.000)

- Structure:

RMSD to initial structure: 0.638 (Å)
RMSD to ESMFold predicted structure: 9.040 (Å), CA pLDDT: 0.661 (0-1)
Cavity volume: 16.841 (Å^3)
Radius of gyration: 12.585 (Å)
```

The reasoning of the agent is as follows:

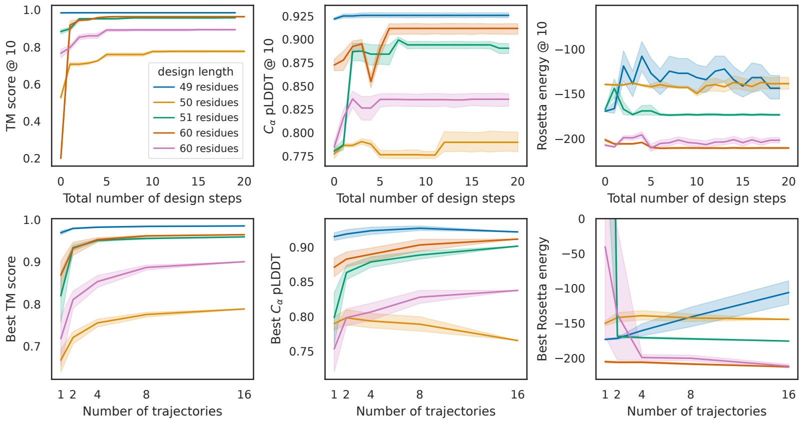
Agent Rosetta's reasoning trace to choose the next action

```
Review of design brief and last action:
- The goal is to design a side chain sequence with minimal glycines, high
    stability (low energy), and structural fidelity to the initial backbone.
- The last rotamer_change (step 3) eliminated almost all glycines and produced a
    diverse, realistic sequence.
- Composition: The glycine content is zero, which is ideal. The penalty for
    composition dropped to zero, confirming the sequence meets the
    compositional constraints.
- Energy: The total energy rose slightly relative to previous designs and is
    less favorable than in earlier steps (-319 vs. -333), with increases in
    repulsion, solvation, and reference energy, but the energy is still quite
    low and reasonable.
- Structure: RMSD to initial backbone remains low (0.638 Å). However, RMSD to
    the ESMFold prediction increased to 9.04 Å (from 2.3 Å in step 2 and ~4 Å
    in step 1), and CA pLDDT dropped slightly. This suggests the structure is
    less ESMFold-predictable/stable than before. Cavity volume is moderate
    (16.8 Å^3), but not excessive.

Summary:
- What went well: Glycine elimination and compositional goal met. Sequence is
    realistic with a good mix of hydrophobics and polar/charged residues.
    Backbone RMSD to initial structure is still low.
- What went wrong: RMSD to ESMFold prediction increased significantly; the
    design is less in agreement with ESMFold, suggesting a loss of
    'designability' or stability. Total energy worsened compared to the best
    previous step.
- Next steps: We need to recover the stability and ESMFold compatibility from
    step 2, while maintaining low glycine content. Step 2 had a good energy,
    low RMSDs, and a cavity volume of 20.5 Å^3, but may have had more glycines
    than desired.

Plan:
- Go back to step 2 (where RMSD to ESMFold and energy were best), and design
    again with a milder glycine penalty (to allow a small number of glycines if
    needed for tight turns), and slightly decrease hydrophobic drive to avoid
    destabilizing the fold. Also, introduce a penalty to avoid excessive cavity
    volume.

Action:
- Use go_back_to_step to revert to step 2.
```

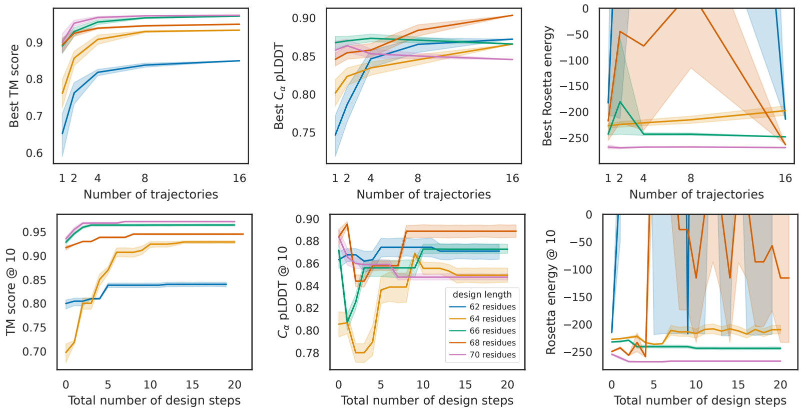
Indeed, the agent reflects on the current design, and recognizes a significant increase in distance to ESMFold predicted structure and drop in pLDDT. Since these changes suggest an overall less stable conformation, the agent chooses to revert to the previous step. Since this action does not require specialized RosettaScripts documentation, we omit the reasoning step after seeing the minimal formatting instructions given to the agent to fit the general multi-turn framework.

H Figures

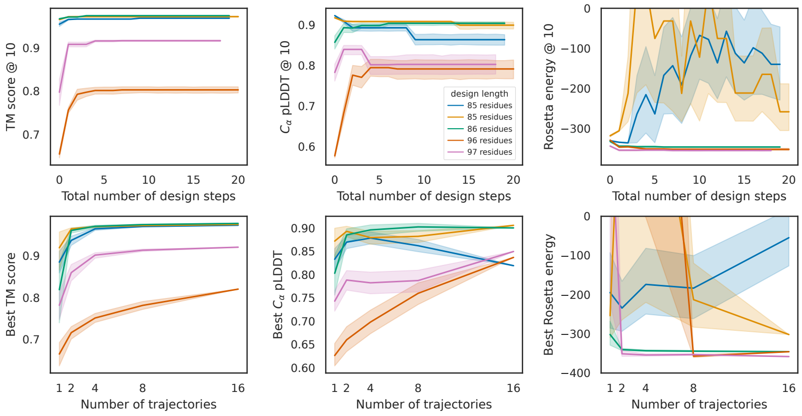
In this section, we include experimental results omitted from the main text for the sake of conciseness.



(a) Backbone conformations with lengths between 40 and 60 residues.

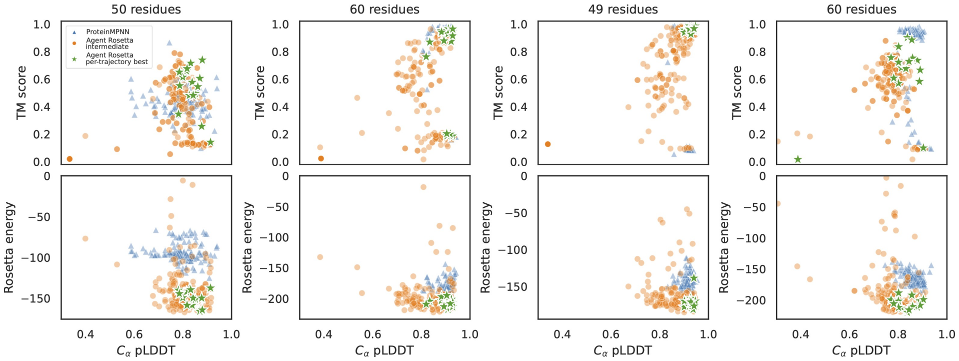


(b) Backbone conformations with lengths between 60 and 80 residues.

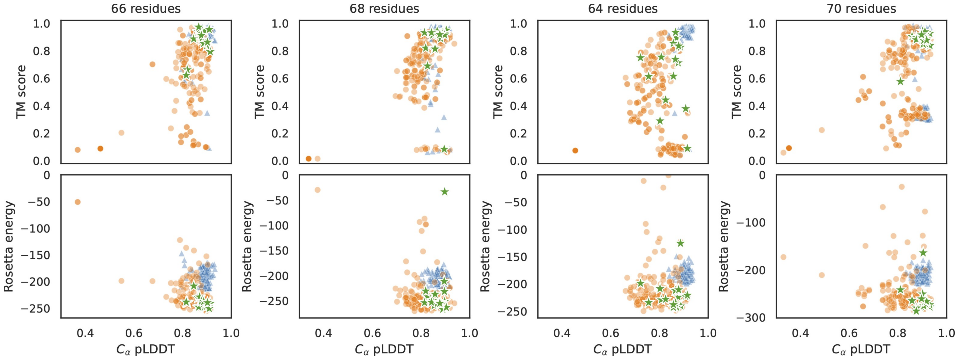


(c) Backbone conformations with lengths between 80 and 100 residues.

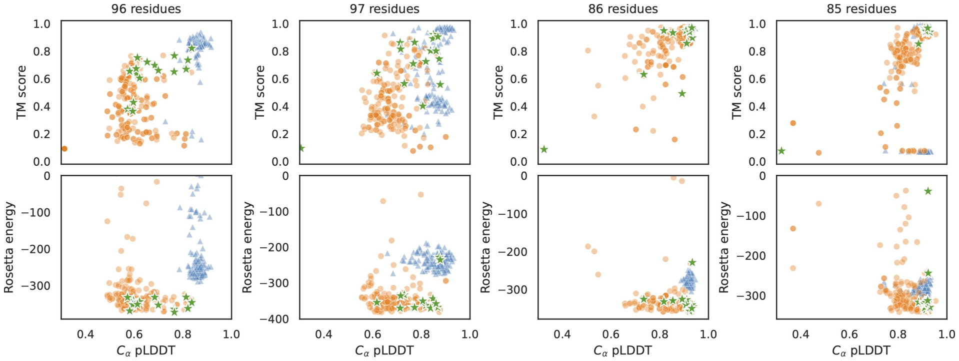
Figure H.1: Quality of designs generated by Agent Rosetta as a function of number of trajectories in best-of-n sampling (top row) and number of steps in each design trajectory (bottom row).



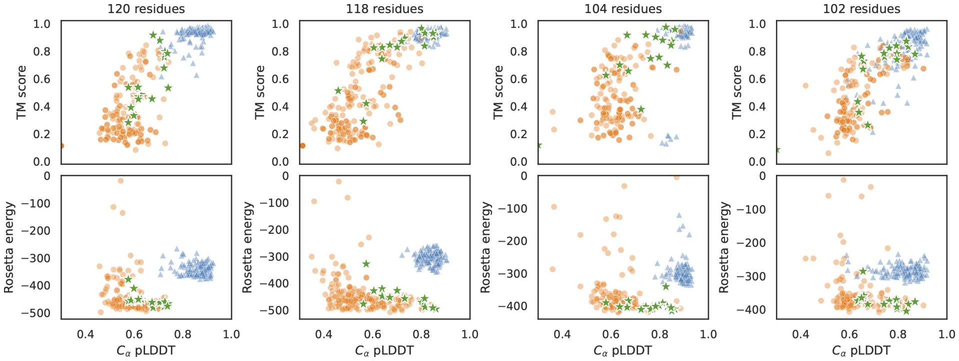
(a) Backbone conformations with lengths between 40 and 60 residues.



(b) Backbone conformations with lengths between 60 and 80 residues.



(c) Backbone conformations with lengths between 80 and 100 residues.



(d) Backbone conformations with lengths between 100 and 120 residues.

Figure H.2: Comparison of the distribution of designs generated with Agent Rosetta and ProteinMPNN for the remaining designs not included in the main text of the manuscript.