PepLang-Bench: Evaluating Large Language Models Understanding On Peptide Related Tasks

Deepa Mal Korani [1] **Vinay Jethava** [2] **Jovan Damjanovic** [3] **Solmaz Gabery Adams** [2] **Kristine Deibler** [1]

Abstract

Large language models are increasingly applied to chemistry, yet their capabilities on peptide-specific tasks, which require integrating sequence conventions, chemical notation systems, and structural reasoning, remain untested. We introduce **PepLang-Bench**, a benchmark of seven tasks across three categories: property prediction (GRAVY, isoelectric point, instability index), notation conversion (SMILES-to-HELM and reverse), and structural adaptation (linear peptide to cyclized SMILES and reverse), evaluated on both therapeutic and synthetic peptide datasets. We benchmark eight models spanning generalist (GPT-5, o4-mini, Gemini-3-Pro, Qwen 3.5) and domain-specific (ChemDFM-R, ChemDFM-v2.0, Ether-0, NatureLM-8x7B) large language models. GPT-5 with code interpreter achieves 64% in isoelectric point prediction by autonomously invoking required domain-specific knowledge, while the same model without tools scores 21.1%. Domain-specific models score 0% in all notation and structural adaptation tasks despite reporting confidence above 0.92. Synthetic peptides containing D-amino acids and non-canonical residues further degrade performance of all models, exposing a distribution gap that current chemistry-focused pretraining does not address. PepLang-Bench establishes peptides as an open challenge for LLM reasoning and provides verifiable reward functions that can serve as training signals for future peptide-specific reasoning models via reinforcement learning.

[1]Molecular AI, Novo Nordisk, Lexington, MA [2]Scientific Intelligence, Novo Nordisk A/S, Malov Denmark [3]In Sillico Cross Modalities, Novo Nordisk, Lexington, MA. Correspondence to: Deepa Mal Korani <dpkn@novonordisk.com>.

Accepted to the AI for Science Workshop @ ICML 2026. Please do not distribute.

1. Introduction

Large language models have emerged as powerful tools for chemistry, with several domain-specific reasoning models now targeting core chemical tasks. For example, Ether-0 (24B), built on Mistral base weights, performed supervised fine-tuning to capture reasoning patterns, and then distilled an ensemble using reward-guided policy optimization for specialist tasks such as functional group identification, molecular captioning, SMILES generation, IUPAC naming, and solubility editing (Narayanan et al., 2025). ChemDFM followed a different workflow, combining domain-adaptive pretraining on chemical corpora with instruction tuning to align the model to user-facing tasks (Zhao et al., 2024a;b). Its reasoning-enhanced variant, ChemDFM-R, further incorporates atomized chemical knowledge and domain-specific reinforcement learning to improve chemical reasoning (Zhao et al., 2026). Despite these efforts, several questions remain: how well do the current chemistry reasoning models generalize across different modalities? Which tasks do they solve reliably, and where do they fail?

Peptides present a particularly important and challenging modality for drug discovery (Muttenthaler et al., 2021; Wang et al., 2022).They sit between small molecules and proteins in size and complexity, exhibit diverse structural conformations, and their properties depend sensitively on sequence, post-translational modifications, and context-dependent interactions (Henninot et al., 2018). Peptides also encompass a broad range of structural classes, from linear chains and stapled peptides to cyclic peptides, which constrain backbone flexibility through head-to-tail or side chain-mediated ring closures to enhance metabolic stability and target affinity (Vinogradov et al., 2019). Further diversity arises from nonnatural amino acids, N-methylation, and lipidation, yielding subclasses such as lipopeptides and glycopeptides with distinct physicochemical profiles.

Unlike small molecules, which are predominantly represented as SMILES strings (Weininger, 1988) or molecular graphs (Gilmer et al., 2017), peptides can be encoded across heterogeneous representations: amino acid sequences, SMILES strings capturing full atomic connectivity, HELM notation describing complex topologies

including cyclizations and nonnatural residues (Zhang et al., 2012), and three-dimensional structural coordinates, each foregrounding different aspects of peptide chemistry. This representational diversity, compounded by the structural variety across peptide classes, means that reasoning about peptides requires models to fluidly translate between modalities: a task demanding genuine cross-modal understanding rather than pattern matching within a single representation. These characteristics create cross-modal reasoning demands, integrating sequence, structural, and experimental assay data that many current models have not been systematically evaluated on.

To address this gap, we introduce a dedicated peptide benchmark suite of seven key peptide tasks (Table 1) across three categories: (i) *property prediction*—isoelectric point [pI] (Bjellqvist et al., 1993), GRAVY score (Kyte & Doolittle, 1982), and instability index (Guruprasad et al., 1990); (ii) *notation conversion*; SMILES ↔ HELM conversion (Zhang et al., 2012); and (iii) *structural adaptation*; linear sequence ↔ cyclic SMILES conversion . We provide standardized datasets and verifiable reward functions for rigorous evaluation. Our contributions are as follows:

- **PepLang-Bench: A benchmark for peptide-specific LLM evaluation.** We introduce PepLang-Bench, comprising seven tasks across property prediction (pI, GRAVY, instability index), notational conversion (SMILES ↔ HELM), and structural adaptation (linear sequence ↔ cyclic SMILES), with verifiable reward functions for automated scoring. Evaluation reveals a mixed performance characteristic of generalized LLMs (Dell'Acqua et al., 2026): GPT-5 achieves 99.7% on GRAVY but 21.1% on pI; Gemini-3-Pro shows the inverse pattern (4.0% GRAVY, 64.0% pI) on therapeutic datasets as seen in Table 3.

- **Code interpreter helps most tasks but regresses on therapeutic notation conversion.** Enabling code execution dramatically improves property prediction (pI: +70%, instability index: +86%) and most notation conversion tasks (e.g., L→C therapeutic: +84%, C→L therapeutic: +81%). However, SMILES → HELM on therapeutic peptides drops from 93.8% to 82.5%.

- **Noncanonical amino acids expose a generalization gap.** Performance collapses on synthetic peptides containing Aib (2-aminoisobutyric acid, a non-proteinogenic residue commonly used to enhance peptide stability), D-amino acids, and other modifications common in modern therapeutics. Without tools, SMILES → HELM accuracy falls from 93.8% to 26.2% on GPT-5 and from 81.0% to 27.0% on Gemini-3-Pro (Table 3). These failures trace to both training data gaps and limitations in generalizing across unknown monomer library space.

- **Reasoning traces reveal systematic error patterns.** Domain-specific models perform poorly on most peptide tasks, particularly notation conversion and structural adaptations. Analysis of their outputs identifies concrete failure modes: domain confusion, amino acids interpreted as periodic elements with degenerate reasoning loops (Fig. A.5), sequence hallucination, and ignoring of noncanonical residues (Figs.A.7, A.8). To probe whether these failures reflect irrecoverable model limitations or surface-level prompting issues, we conducted a targeted prompt refinement study with a domain expert, focusing on the linear-to-cyclic conversion task where reasoning traces suggested models understood the task conceptually but miscalculated in execution. Expert-guided prompts yielded measurable improvements (Table 5), though absolute performance remained below levels required for practical use; suggesting that better prompting mitigates but does not close the fundamental gap.

2. Related Work

LLM Benchmarks for Chemistry. Several benchmarks have been introduced focusing on evaluating LLM capabilities in chemistry, including ChemBench (Mirza et al., 2025), ChemLLMBench (Guo et al., 2023), ChemCoT-Bench (Li et al., 2026), and ChemPro (Baranwal & Vyas, 2026) (described in detail in Appendix A.1). These benchmarks primarily target small-molecule chemistry and general chemical knowledge, leaving peptide-specific reasoning largely unexplored.

Tool-Augmented LLMs in Chemistry. Tool-augmented approaches have gained traction to address LLM limitations in precise chemical computation, including ChemCrow (M. Bran et al., 2024), CACTUS (McNaughton et al., 2024), ChemOrch (Huang et al., 2025), and MECo (Zhu et al., 2025) (detailed in Appendix A.2). These works highlight code execution as a promising paradigm for chemical reasoning. However, the underlying tools—primarily RDKit and PubChem—were designed for small-molecule chemistry and offer only limited native support for peptide-specific representations: RDKit's HELM parser is restricted to a hard-coded set of canonical amino acids and a small number of analogs (Garzon Otero et al., 2025; Ochoa et al., 2023), and lacks built-in support for user-defined non-canonical monomers. We systematically evaluate the impact of code interpreter access on peptide tasks, revealing that this tool gap creates a task-dependent trade-off where code execution helps algorithmic property prediction but degrades notation conversion on noncanonical

Table 1. **Overview of PepLang-Bench evaluation tasks.** Task categories: **PP** = Property Prediction, **NC** = Notation Conversion, **SA** = Structural Adaptation. Each task specifies a well-defined input–output format and is evaluated using task-specific verifiable criteria described in Appendix A.4.

Task	Category	Description
Isoelectric point (pI) prediction	PP	Predict the pH at which the peptide has zero net charge from an amino acid sequence. Details in A.3.1.
GRAVY score prediction	PP	Predict the mean Kyte–Doolittle hydropathy across residues. Details in A.3.2.
Instability index prediction	PP	Predict numeric instability index from dipeptide weights and classify as stable/unstable. Details in A.3.3.
SMILES → HELM conversion	NC	Convert SMILES to HELM notation. Details in A.3.4.
HELM → SMILES conversion	NC	Convert HELM to chemically equivalent SMILES. Details in A.3.5.
Linear sequence → Cyclic SMILES	SA	Convert a linear amino acid sequence into head-to-tail cyclic peptide SMILES. Details in A.3.6.
Cyclic SMILES → Linear sequence	SA	Convert head-to-tail cyclic peptide SMILES into a linear amino acid sequence; rotations accepted. Details in A.3.7.

chemistry.

Domain-Specific Models and the Peptide Gap. Several domain-specific models have been post-trained for chemistry and biomedical reasoning. NatureLM (Xia et al., 2025) applied continual domain pretraining followed by reinforcement learning-based post-training to adapt across biological modalities including DNA, RNA, small molecules, and proteins. In the biomedical space, Med-Gemini (Saab et al., 2024) and TxGemma (Wang et al., 2025) target clinical and therapeutic applications, while BioReason (Fallahpour et al., 2025) focuses on biological reasoning over scientific literature. However, to the best of our knowledge, no domain-specific model has been post-trained to handle peptides specifically or, more generally, to jointly reason over amino acid sequences and SMILES representations. As we demonstrate in Section 5, existing chemistry models post-trained on general chemical corpora—including ChemDFM-R (Zhao et al., 2026), ChemDFM-v2.0 (Zhao et al., 2024b), and Ether-0 (Narayanan et al., 2025); perform poorly on peptide-specific tasks, underscoring the need for dedicated peptide benchmarks and, eventually, peptide-aware post-training.

Peptide Representations. Representing peptides computationally presents unique challenges at the intersection of bioinformatics and cheminformatics (Minkiewicz et al., 2017). SMILES (Weininger, 1988) provides atomic-level chemical notation but becomes unwieldy for longer peptides and cannot directly encode sequence information. The Hierarchical Editing Language for Macromolecules (HELM) (Zhang et al., 2012) addresses this by representing biomolecules at monomer granularity, enabling representation of nonnatural amino acids, chemical modifications, and complex architectures like antibody–drug conjugates. HELM has been adopted by databases including ChEMBL and PubChem (Kim et al., 2025). Recent work on HELM-GPT (Xu et al., 2024) demonstrates the utility of HELM for macrocyclic peptide generation, developing HELM-to-SMILES conversion tools for property prediction. The p2smi toolkit (Feller & Wilke, 2025) facilitates peptide FASTA-to-SMILES conversion supporting noncanonical amino acids and cyclizations. pyPept (Ochoa et al., 2023) provides comprehensive peptide manipulation using FASTA, HELM, and BILN notation (Fox et al., 2022). Despite these advances, LLM capabilities in converting between peptide representations remain underexplored.

3. PepLang-Bench

To our knowledge, PepLang-Bench is the first benchmark designed to systematically evaluate reasoning models on peptide-specific tasks spanning across following categories (1) property prediction, (2) notational conversion and (3) structural adaptation. Figure 1 provides a high-level overview of the benchmark design, and Table 1 summarizes the evaluation tasks.

PepLang-Bench comprises two complementary datasets, as provided in Table 2. The first contains therapeutic peptides containing natural amino acids, while the second consists of synthetically derived peptides generated using the `p2smi` toolkit (Feller & Wilke, 2025). The inclusion of the synthetic dataset enables controlled evaluation on peptides containing nonnatural amino acids, allowing us to probe generalization beyond the standard natural amino acid residues.

Table 2. **Dataset statistics for PepLang-Bench.**

	Therapeutic	Synthetic
Source	PrMFTP database	p2smi generation
Train / Test	7,899 / 1,975	800 / 200
Sequence length	Variable	5–11
Non-canonical AAs	None	10%
D-amino acids	None	20%

Across tasks, PepLang-Bench employs *verifiable reward functions*: automatically computable scoring criteria that compare model outputs against references via exact-match checks (for structured fields and discrete labels), numeric tolerance functions (for scalar properties), and structure-aware equivalence checks after canonicalization and validity filtering (for molecular representations). Full task definitions and their respective reward information and prompts are provided in the Appendix A.3, A.4 and A.8.

4. Method

4.1. Datasets

4.1.1. Therapeutic Peptides Dataset

The therapeutic peptides dataset is derived from the PrMFTP (Prediction of Multi-Functional Therapeutic Peptides) database (Yan et al., 2022), a curated collection of experimentally validated therapeutic peptide sequences. Peptides were included if they contained valid natural amino acid sequences parsable by RDKit with computable physicochemical properties. The dataset consists of natural L-amino acid sequences with variable lengths, representing real-world therapeutic candidates. Each sample includes the primary sequence, which is then augmented to the SMILES and HELM representations, precomputed physicochemical properties (GRAVY score, isoelectric point, instability index), and structural adaptations (head-to-tail cyclized SMILES)

4.1.2. Synthetic Peptides Dataset

To evaluate model performance on nonnatural amino acids and D-configured residues, which are not represented in the therapeutic dataset, we constructed a synthetic peptide dataset using the `p2smi` (Feller & Wilke, 2025) peptide generation framework. Peptides were generated with lengths of 5-11 residues, incorporating 10% noncanonical amino acids and 20% D-amino acids to test model generalization beyond the standard 21 L-amino acid vocabulary. This dataset enables evaluation of structurally diverse peptides that are increasingly relevant in modern peptide drug design.

4.2. Experimental Setup

We evaluate PepLang-Bench across generalized reasoning models and domain-specific chemistry baselines. For each model, we report task-level performance under the corresponding automatic grader (verifiable reward), aggregating results across multiple random seeds.

Models. We evaluate: (i) **generalized reasoning models**; GPT-5 (Singh et al., 2026), o4-mini (OpenAI, 2025) with medium reasoning effort, Gemini-3-Pro (Team et al., 2025), and Qwen-3.5 (Yang et al., 2025); (ii) **domain-specific baselines**; ChemDFM-v2.0-14B (Zhao et al., 2025), ChemDFM-R-14B (Zhao et al., 2026), NatureLM (Xia et al., 2025), alongside Ether-0 (Narayanan et al., 2025), a chemistry-focused reasoning system. For OpenAI models, we additionally evaluate with Code Interpreter [CI] access, enabling execution of Python code while reasoning with scientific libraries; eg: (RDKit (Landrum, 2010), BioPython (Cock et al., 2009)). Hugging Face model identifiers are provided in Table 6.

Evaluation. All models produce structured JSON outputs containing the predicted answer, step-by-step reasoning, and a self-reported confidence score (0–1). Property prediction tasks (GRAVY, pI, instability index) use numeric-tolerance graders, while notation conversion tasks (SMILES↔HELM) employ RDKit-based structural similarity and molecular validity checks . We evaluate per task using three random seeds (42, 123, 456) and report mean accuracy. Full task definitions, full pseudocode of grader definitions, and prompt templates are provided in Appendix A.3, A.4 and A.8.

5. Results

Table 3 summarizes model performance across PepLang-Bench tasks, with error bars reflecting variance across three sampling runs. Table 8 provides detailed reward metric breakdowns for notation conversion (NC) and structural adaptation (SA) tasks, and Table 7 does the same for property prediction (PP) tasks. Among property prediction tasks, GRAVY score computation proves most tractable, with GPT-5 achieving near-perfect accuracy (99.7% therapeutic, 98.2% synthetic), while isoelectric point and instability index prediction remain challenging across all models (<65% accuracy). Notably, despite leading on pI prediction (64.0% on therapeutic peptides, 48.0% on synthetic peptides), Gemini-3-Pro achieves only 4.0% on GRAVY score prediction, suggesting that iterative charge-based reasoning and hydropathy summation engage fundamentally different computational strategies. Analysis of o4-mini reasoning (Figure A.1) traces for pI prediction reveals three systematic failure modes: (1) incorrect identifi-

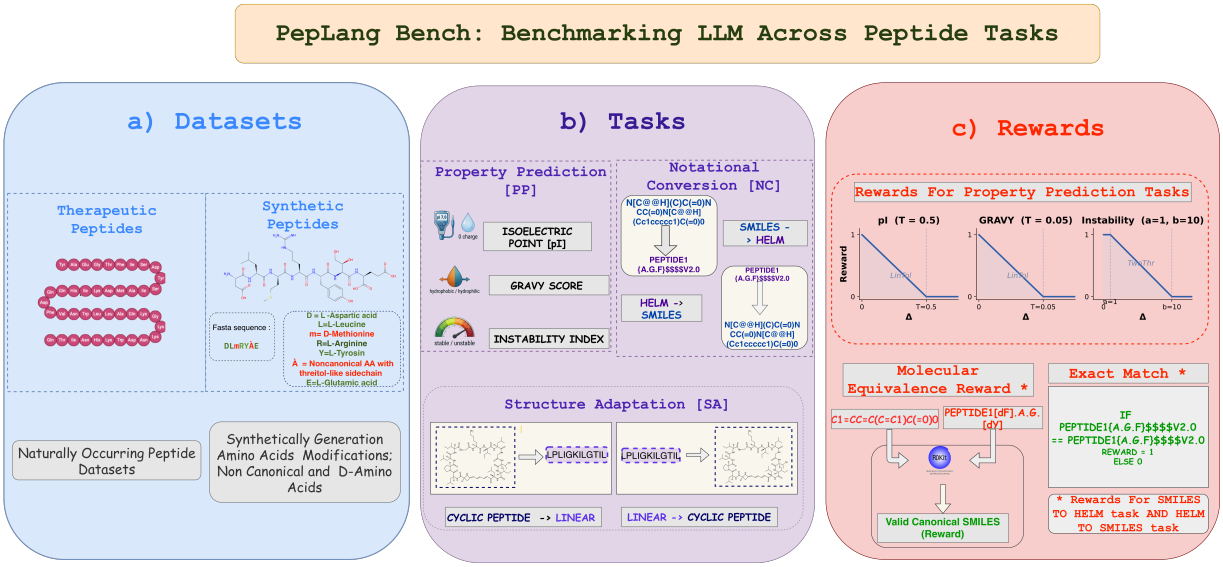

Figure 1. **Overview of PepLang-Bench. (a) Datasets:** Two peptide subsets; a *therapeutic* set composed of naturally occurring amino acids, and a *synthetic* set comprising noncanonical and D-amino acid modifications common in modern therapeutics. **(b) Tasks:** Seven tasks spanning three categories: (i) *Property Prediction* (isoelectric point, GRAVY score, instability index), (ii) *Notational Conversion* (SMILES ↔ HELM), and (iii) *Structural Adaptation* (linear ↔ cyclic peptide conversion). **(c) Rewards:** Continuous tolerance-based rewards for property prediction tasks (pI, GRAVY, instability), molecular equivalence rewards via RDKit canonicalization for SMILES outputs, and exact-match rewards for HELM outputs. Detailed reward formulations are provided in Appendix A.4.

cation of the overall charge state at low pH, most critically the erroneous assignment of a positive charge to the C-terminus; (2) inconsistent charge tracking across the deprotonation sequence, leading to arithmetic errors that propagate through subsequent steps; and (3) a methodological shortcut of averaging flanking pK_a values without verifying that the predicted pI actually yields a net charge of zero via the Henderson-Hasselbalch equation.

For notation conversion, GPT-5 excels at SMILES→HELM on therapeutic peptides (93.8%), whereas Gemini-3-Pro leads on HELM→SMILES (91.0% therapeutic, 83.0% synthetic). Performance degrades sharply on synthetic peptides across all models: SMILES→HELM accuracy drops to 26.2% for GPT-5, 27.0% for Gemini-3-Pro, and 0.0% for Qwen-3.5, indicating little to no generalization beyond standard residues. HELM encodes peptide topology through polymer-level representations with explicit connection tables for backbone and sidechain linkages (Zhang et al., 2012); the sharp drop on synthetic peptides likely reflects the limited coverage of non-standard residues in the cheminformatics resources underlying model training, forcing models to fall back on brittle parametric knowledge.

Structural adaptation tasks (L→C, C→L) prove most difficult, with all models achieving less than 12% exact match accuracy. However, decomposing performance by reward metric (Appendix A.4) reveals that models partially understand the underlying chemistry. For Linear→Cyclic, GPT-5 produces valid SMILES 93% of the time and achieves correct ring size in 62% of cases, yet only 11% exact structural match—indicating that models can form macrocycles of the right size but fail at atomic-level connectivity, particularly stereochemistry at the closure site and ring index management for aromatic side chains (see prompt refinement analysis in Table 5). For Cyclic→Linear, the pattern is even more striking: GPT-5 achieves 99.7% correct sequence length and 99.4% amino acid composition overlap but only 9.7% rotation match, meaning the model correctly identifies which residues comprise the macrocycle but consistently selects the **wrong linearization direction.** This suggests the bottleneck is not residue identification but directional reasoning; specifically, the consistent application of N-terminus→C-terminus traversal order once the ring is broken. Since all circular shifts of a cyclic peptide are equivalent, the ring-breaking position itself is arbitrary; what current models lack is the ability to reliably orient their traversal in the correct direction when reading the SMILES graph.

Among domain-specific baselines, ChemDFM-R-14B and ChemDFM-v2.0-14B match or exceed GPT-5 on pI (21–32% vs. GPT-5's 21.1% therapeutic, 19.2% synthetic), suggesting that domain pre-training on chemical corpora provides some benefit for tasks grounded in amino acid-level biochemical properties. However, both models score 0.0% across all notation conversion and structural adaptation tasks, indicating that HELM notation was absent from

their training corpora—a significant gap given HELM's central role in peptide informatics. Ether-0 fails almost entirely across all tasks ($\leq$1.6% therapeutic, $\leq$5.4% synthetic), with its best performance limited to pI prediction (10.3% therapeutic). NatureLM-8x7B Instruct performs even worse, scoring 0.0% on all therapeutic tasks and achieving only marginal performance on synthetic pI (12.0%) and instability index (6.0%), suggesting that its multi-domain training provides no transferable signal for peptide-specific reasoning. Notably, the reasoning variant ChemDFM-R-14B performs comparably to its non-reasoning counterpart ChemDFM-v2.0-14B on property tasks, and neither can handle notation conversion—indicating that reasoning training alone cannot compensate for gaps in underlying domain knowledge. Representative failure cases for Ether-0, ChemDFM-R, and NatureLM are provided in Appendix A.6.2 [Figure A.4,A.5], Appendix A.6.3 [Figure A.6, A.8] and Appendix A.6.4 [Figure A.12, A.11, A.10] respectively.

Code Interpreter. Enabling code interpreter access yields substantial improvements across nearly all tasks, with a single notable regression on therapeutic SMILES→HELM conversion (Table 4). For property prediction, isoelectric point improves by +70% (therapeutic) and +59% (synthetic), while instability index improves by +86% and +79% respectively. GRAVY score shows minimal change (<2% in either direction), since models already achieve near-perfect accuracy through memorization of the Kyte-Doolittle scale—leaving little room for tools to help.

Structural adaptation tasks benefit dramatically on therapeutic peptides: L→C improves by +84% (11.0 → 95.4) and C→L by +81% (8.3 → 89.3), as the model can now invoke RDKit to perform explicit ring closure and linearization, rather than reason about SMILES traversal symbolically. However, these gains attenuate sharply on synthetic peptides (L→C: +26%; C→L: +18%), reflecting RDKit's limited support for noncanonical amino acids.

The notation conversion results are more nuanced. H→S improves substantially on therapeutic peptides (65.7 → 97.5, +32%) and modestly on synthetic (78.2 → 83.0, +5%), consistent with RDKit providing reliable HELM parsing for standard residues. S→H on synthetic peptides also improves sharply (26.2 → 75.2, +49%), likely because tool-assisted canonicalization reduces the combinatorial ambiguity of mapping SMILES fragments to HELM monomers A.9

Overall, the code interpreter is most beneficial when the underlying task is algorithmic (pI, instability index, cyclization) and the required chemical operations are well-supported by standard libraries. It provides diminishing or negative returns when (i) the model already achieves near-ceiling parametric performance, or (ii) the task involves noncanonical chemistry that exceeds the coverage of standard cheminformatics tools.

Case Study: Prompt Refinement via Domain Expert Feedback. Beyond code interpreter access, we investigate whether targeted prompt modifications informed by domain expert can improve performance on structurally complex tasks. During evaluation of the challenging Linear→Cyclic task, a domain expert identified that GPT-5 systematically failed to maintain ring structures in aromatic residue R-groups, misinterpreting them as main-chain ring closure points and thereby generating polycyclic structures. Reasoning trace analysis further revealed that this behavior was caused by the model's inability to use distinct indices for the backbone and any side chain rings, creating conflicts. Adding a single instruction —*"Make sure to increment the ring indices as you see relevant rings in the side chain 'R group' of the residues"*—resolved the collision by prompting the model to assign distinct indices (e.g., `1` for the macrocycle, `2` for Tyr's phenyl ring). Table 5 quantifies the improvement and A.9 provides detailed prompt analysis. This finding illustrates that domain-expert-in-the-loop prompt refinement can yield substantial gains on structurally complex tasks, and motivates systematic investigation of failure-driven prompt iteration as a lightweight alternative to fine-tuning.

Reward Metric Analysis. Tables 7 and 8 decompose aggregate accuracy into task-specific reward metrics, revealing systematic failure modes. For property prediction, we observe striking model-specific biases: GPT-5 achieves near-ceiling GRAVY accuracy (99.7%) while Gemini-3-Pro catastrophically fails (4.0%), yet this pattern *inverts* for isoelectric point, where Gemini-3-Pro leads by roughly 3× (64.0% vs. 21.1%). Instability index proves uniformly challenging (3–13% numerical accuracy across models and datasets), though binary stability classification is substantially easier (43–61%), suggesting models capture directional trends but struggle with precise arithmetic.

Notation conversion exposes a *structure-versus-syntax gap*. On therapeutic SMILES→HELM, GPT-5 achieves 99.7% monomer sequence accuracy but only 61.3% structural similarity—correctly identifying amino acids while producing malformed HELM syntax (missing polymer wrappers, incorrect connection notation). Disulfide bond match tells a similar story: GPT-5 achieves perfect recovery on therapeutic peptides (100.0%) but drops to 62.0% on synthetic, while Gemini-3-Pro is more balanced across both datasets (73.0% / 67.0%), suggesting that disulfide topology is relatively well-captured for canonical residues but becomes brittle under noncanonical modifications. The

Table 3. **Main results on PepLang-Bench.** Best results (accuracy %) per task in **bold**. The **R** column marks reasoning models (✓). The **Metric** row indicates the primary accuracy metric used per task: Molecular Equivalence (ME) for notation conversion tasks (S→H, H→S), Exact SMILES Match (ESM) for L→C, Rotation Match (RM) for C→L, and Numerical Grader (NG, within task-specific tolerance) for property prediction tasks (GRAVY, pI, Instab.). Full sub-reward breakdowns and metric definitions are reported in Table 7 and Table 8. The following model variants were evaluated: ‡*ChemDFM-R-14B*, ‡ChemDFM-v2.0-14B and †*NatureLM-8x7B-Inst*; Appendix A.5 for Hugging Face model identifiers and version details for all open-weight baselines. Domain-specific baselines were evaluated on a single seed due to computational constraints; values are reported without error bars.

DATASET	MODEL	R	GRAVY	pI	INSTAB.	S→H	H→S	L→C	C→L
	Metric		*NG*	*NG*	*NG*	*ME*	*ME*	*ESM*	*RM*
Generalist Reasoning Models									
THERAPEUTIC	GPT-5	✓	**99.7**$_{\pm 0.1}$	21.1$_{\pm 0.3}$	9.4$_{\pm 0.3}$	**93.8**$_{\pm 0.2}$	65.7$_{\pm 0.5}$	**11.0**$_{\pm 0.3}$	**8.3**$_{\pm 0.3}$
THERAPEUTIC	O4-MINI	✓	95.7$_{\pm 0.2}$	14.4$_{\pm 0.3}$	11.6$_{\pm 0.3}$	47.0$_{\pm 0.6}$	12.9$_{\pm 0.4}$	1.0$_{\pm 0.1}$	0.6$_{\pm 0.1}$
THERAPEUTIC	QWEN-3.5	✓	9.0$_{\pm 0.3}$	21.0$_{\pm 0.3}$	9.0$_{\pm 0.2}$	0.0$_{\pm 0.0}$	1.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$
THERAPEUTIC	GEMINI-3-PRO	✓	4.0$_{\pm 0.1}$	**64.0**$_{\pm 0.2}$	**13.0**$_{\pm 0.03}$	81.0$_{\pm 0.1}$	**91.0**$_{\pm 0.1}$	10.0$_{\pm 0.1}$	0.3$_{\pm 0.1}$
SYNTHETIC	GPT-5	✓	**98.2**$_{\pm 1.0}$	19.2$_{\pm 1.1}$	12.3$_{\pm 1.6}$	26.2$_{\pm 1.6}$	78.2$_{\pm 1.4}$	1.8$_{\pm 0.5}$	**5.0**$_{\pm 1.6}$
SYNTHETIC	O4-MINI	✓	94.3$_{\pm 1.6}$	8.8$_{\pm 0.7}$	**12.4**$_{\pm 2.0}$	19.9$_{\pm 1.2}$	27.2$_{\pm 1.7}$	0.7$_{\pm 0.3}$	0.0$_{\pm 0.0}$
SYNTHETIC	QWEN-3.5	✓	5.0$_{\pm 0.2}$	32.0$_{\pm 0.5}$	7.0$_{\pm 0.2}$	0.0$_{\pm 0.0}$	3.0$_{\pm 0.1}$	0.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$
SYNTHETIC	GEMINI-3-PRO	✓	6.0$_{\pm 0.2}$	**48.0**$_{\pm 0.1}$	3.0$_{\pm 0.1}$	**27.0**$_{\pm 0.2}$	**83.0**$_{\pm 0.3}$	**3.0**$_{\pm 0.03}$	1.0$_{\pm 0.04}$
Domain-Specific Baselines									
THERAPEUTIC	CHEMDFM-R‡	✓	0.0	21.0	14.0	0.0	0.0	0.0	0.0
SYNTHETIC	CHEMDFM-R‡	✓	0.0	32.0	7.0	0.0	0.0	0.0	0.0
THERAPEUTIC	CHEMDFM-V2.0‡		0.0	22.0	15.0	0.0	0.0	0.0	0.0
SYNTHETIC	CHEMDFM-V2.0‡		0.0	25.0	6.0	0.0	1.0	0.0	0.0
THERAPEUTIC	ETHER-0	✓	1.6	10.3	1.4	0.0	0.0	0.0	0.0
SYNTHETIC	ETHER-0	✓	0.0	5.4	0.08	0.0	0.0	0.0	0.0
THERAPEUTIC	NATURELM†		0.0	0.0	0.0	0.0	0.0	0.0	0.0
SYNTHETIC	NATURELM†		0.0	12.0	6.0	0.0	0.0	0.0	0.0

inverse direction (H→S) shows Gemini-3-Pro achieving the highest molecular equivalence (91.0% therapeutic) despite GPT-5 producing more syntactically valid SMILES (96.0%), indicating that Gemini-3-Pro better preserves chemical identity while GPT-5 prioritizes well-formed outputs over semantic correctness. Notably, on H→S structural similarity, GPT-5 leads substantially (99.2% therapeutic vs. 33.5–35.0% for competitors), confirming that when GPT-5 does produce valid outputs, they closely match reference structures—the gap lies in chemical identity preservation rather than geometric fidelity.

Structural adaptation tasks reveal the *directionality problem*: on therapeutic Cyclic→Linear, GPT-5 correctly identifies amino acid composition (99.4%) and sequence length (99.7%) but achieves only 9.7% rotation match. The model understands *what* residues comprise the macrocycle but consistently fails to traverse them in the correct N-terminus→C-terminus direction once the ring is broken; returning the sequence in the reverse C-terminus→N-terminus orientation or an equivalent retro-isomer, rather than failing to correctly identify residues.

Linear→Cyclic proves hardest overall (≤11% exact match), though 62% ring size accuracy on therapeutic peptides indicates partial understanding of macrocycle formation. Performance degrades uniformly on synthetic peptides containing noncanonical amino acids (e.g., Aib, Orn, Nle), with models either hallucinating physicochemical properties or producing malformed outputs when encountering unfamiliar residues.

Confidence Extraction Caveats Confidence values were extracted from model responses with varying reliability as seen in Table 9.

- **Tool access reshapes GPT-5 calibration.** On therapeutic instability index, GPT-5 with a code interpreter reports confidence 0.93, while the same model without tools reports 0.26—a 3.6× gap. The same shift appears on synthetic instability (0.82 vs. 0.24). When the model must estimate from text alone, it signals uncertainty; when it can execute BioPython/RDKit, its confidence rises because the answer is grounded in verified computation.

- **Synthetic peptides expose hidden uncertainty on cyclization.** Both GPT-5 (code interpreter) and Gemini-3-Pro drop their C→L confidence dramatically on the synthetic split (0.28 and 0.12) versus therapeutic (0.94 and 1.00), while Qwen 3.5 seems to still be overconfident with its evaluations. Noncanonical amino acids and D-residues push these models outside their training distribution, and they partially acknowledge this—but only for cyclization. Other tasks

Table 4. **Effect of Code Interpreter [CI] on GPT-5 performance.** [accuracy %]

Task	Dataset	w/o CI	w/ CI
Property Prediction			
GRAVY Score	Therapeutic	$99.7_{\pm 0.1}$	$99.5_{\pm 0.5}$
GRAVY Score	Synthetic	$98.2_{\pm 1.0}$	$96.7_{\pm 1.7}$
Isoelectric Point	Therapeutic	$21.1_{\pm 0.3}$	$91.2_{\pm 1.0}$
Isoelectric Point	Synthetic	$19.2_{\pm 1.1}$	$77.7_{\pm 2.3}$
Instability Index	Therapeutic	$9.4_{\pm 0.3}$	$95.0_{\pm 1.5}$
Instability Index	Synthetic	$12.3_{\pm 1.6}$	$91.3_{\pm 3.5}$
Notation Conversion			
S→H	Therapeutic	$93.8_{\pm 0.2}$	$82.5_{\pm 3.9}$
S→H	Synthetic	$26.2_{\pm 1.6}$	$75.2_{\pm 0.0}$
H→S	Therapeutic	$65.7_{\pm 0.5}$	$97.5_{\pm 1.1}$
H→S	Synthetic	$78.2_{\pm 1.4}$	$83.0_{\pm 0.0}$
L→C	Therapeutic	$11.0_{\pm 0.3}$	$95.4_{\pm 1.1}$
L→C	Synthetic	$1.8_{\pm 0.5}$	$27.4_{\pm 0.6}$
C→L	Therapeutic	$8.3_{\pm 0.3}$	$89.3_{\pm 2.3}$
C→L	Synthetic	$5.0_{\pm 1.6}$	$23.1_{\pm 0.0}$

Table 5. **Effect of domain-expert prompt refinement on Linear→Cyclic SMILES conversion.** [GPT-5, therapeutic peptides]. The refined prompt adds a single instruction about ring index management for side-chain aromatic groups.

Metric	Original Prompt	Refined Prompt	Δ
Exact SMILES Match	$11.0_{\pm 0.3}$	$80_{\pm 0.1}$	+70%
SMILES Validity	$93.0_{\pm 0.1}$	$94_{\pm 0.1}$	+1%
Ring Size Match	$62.0_{\pm 0.1}$	$90_{\pm 0.2}$	+28%

maintain therapeutic-level confidence on synthetic inputs despite accuracy dropping sharply.

- **Domain-specific chemistry LLMs are uniformly overconfident.** ChemDFM-R and ChemDFM-v2.0 report confidence 0.92–1.00 across nearly every task on both datasets, despite near-zero accuracy on notation conversion and cyclization. Unlike the generalized models, they show no awareness of distribution shift between therapeutic and synthetic peptides; the confidence numbers are indistinguishable across splits. This is discussed further in Section A.7.

6. Discussion

Our results reveal a nuanced picture of LLM capabilities for peptide-related tasks, with implications for both research and practice.

Property Prediction: Algorithmic vs. Memorized Knowledge. The contrast between GRAVY and pI/instability performance reveals a fundamental distinction in how LLMs approach biochemistry tasks. GPT-5 achieves near-perfect GRAVY scores (99.7%) because the Kyte-Doolittle scale is a 20-value lookup well-represented in training data. In contrast, pI calculation requires iterative numerical solving of the Henderson-Hasselbalch equation across multiple ionizable groups, exceeding reliable in-context computation. LLMs excel at retrieving memorized constants but struggle with multi-step numerical algorithms, and the gains from code interpreter access (+70% pI, +86% instability) confirm that delegating computation to verified tools is essential for reliable property prediction.

The Noncanonical Amino Acid Challenge. Performance collapses on synthetic peptides containing noncanonical residues: SMILES→HELM accuracy drops from 93.8% on therapeutic peptides to 26.2% on synthetic ones (0% with code interpreter). This stems from two compounding factors: (1) noncanonical residues such as Aib and D-amino acids are underrepresented in training corpora, and (2) cheminformatics libraries like RDKit lack support for these modifications. Since modern therapeutic peptides increasingly rely on noncanonical residues for stability and bioavailability (Henninot et al., 2018), this gap has significant practical implications and motivates domain-specific fine-tuning and HELM-aware chemical toolkits.

Code Interpreter: Benefits and Pitfalls. Code interpreter access produces both gains and regressions. For algorithmic tasks (pI, instability, cyclization), execution eliminates arithmetic errors via validated implementations such as BioPython's `ProtParam`. For notation conversion, however, models invoke RDKit for HELM parsing despite its lack of HELM support, introducing execution errors that drop performance below the no-tool baseline. Optimal tool use thus requires meta-knowledge of tool capabilities—models must learn when *not* to invoke code execution—suggesting future systems should expose tool capability metadata or learned tool selectors.

Limitations. Several limitations warrant discussion. First, we evaluate only one open-source model (Qwen 3.5) alongside closed-source generalized models (GPT-5, o4-mini, Gemini-3-Pro) ; a broader evaluation of general-purpose open-source models (Llama, DeepSeek) remains important future work, particularly for reproducibility and mechanistic analysis. Second, our synthetic peptide dataset does not exhaustively cover the space of noncanonical amino acids, and we focus solely on intrinsic physicochemical properties, excluding structure-dependent properties and activity predictions. Third, code interpreter experiments use default configurations without prompt optimization, and we lack a human expert baseline to contextualize model performance relative to domain specialists.

Future Directions. Our findings motivate two concrete research directions: (1) developing a peptide-specific reasoning model trained on task-based rewards identified across our benchmark, and (2) a systematic framework for prompt optimization guided by domain expert feedback.

Acknowledgements. We thank OpenAI for providing support in setting up access to closed-source models used in this work. We also thank the anonymous reviewers for their constructive feedback, as well as our colleagues in Molecular AI and Digital Chemistry and Design at Novo Nordisk for helpful discussions throughout the course of this project.

Impact Statement. This paper presents work focused on the advancement the field of large language models, with a particular focus on applying large language models to therapeutic peptide design and understanding. While our work is methodological in nature, applications in computational drug discovery carry both potential benefits—accelerating the identification of therapeutic candidates and risks, including the possibility of misuse. We believe the techniques presented here pose no greater risk than existing publicly available cheminformatics tools and language models and we encourage responsible use within established scientific frameworks.

LLM Usage. Large language models were used in the preparation of this manuscript solely for rewording and editorial polishing of author-written text and LaTeX formatting of figures. All technical content, experimental design, analysis, and conclusions are entirely the work of the authors.

References

Baranwal, A. and Vyas, S. ChemPro: A Progressive Chemistry Benchmark for Large Language Models. *Artificial Intelligence Chemistry*, 4(1):100118, June 2026. ISSN 29497477. doi: 10.1016/j.aichem.2026.100118. URL http://arxiv.org/abs/2602.03108. arXiv:2602.03108 [cs].

Bjellqvist, B., Hughes, G. J., Pasquali, C., Paquet, N., Ravier, F., Sanchez, J.-C., Frutiger, S., and Hochstrasser, D. The focusing positions of polypeptides in immobilized pH gradients can be predicted from their amino acid sequences, 1993. ISSN 1522-2683. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/elps.11501401163.

Bjellqvist, B., Basse, B., Olsen, E., and Celis, J. E. Reference points for comparisons of two-dimensional maps of proteins from different human cell types defined in a pH scale where isoelectric points correlate with polypeptide compositions. *ELECTROPHORESIS*, 15(1):529–539, 1994. ISSN 1522-2683. doi: 10.1002/elps.1150150171. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/elps.1150150171.

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., et al. Biopython:

freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11): 1422–1423, June 2009. ISSN 1367-4811, 1367-4803. doi: 10.1093/bioinformatics/btp163. URL https://academic.oup.com/bioinformatics/article/25/11/1422/330687.

Dell'Acqua, F., McFowland, E., Mollick, E., Lifshitz, H., Kellogg, K. C., Rajendran, S., Krayer, L., et al. Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of Artificial Intelligence on Knowledge Worker Productivity and Quality. *Organization Science*, 37(2):403–423, March 2026. ISSN 1047-7039. doi: 10.1287/orsc.2025.21838. URL https://pubsonline.informs.org/doi/10.1287/orsc.2025.21838.

Fallahpour, A., Magnuson, A., Gupta, P., Ma, S., Naimer, J., Shah, A., Duan, H., et al. BioReason: Incentivizing multimodal biological reasoning within a DNA-LLM model. *Advances in Neural Information Processing Systems*, 38, 2025.

Feller, A. L. and Wilke, C. O. p2smi: A toolkit enabling SMILES generation and property analysis for noncanonical and cyclized peptides. *Journal of Open Source Software*, 10(116):8319, December 2025. ISSN 2475-9066. doi: 10.21105/joss.08319. URL https://joss.theoj.org/papers/10.21105/joss.08319.

Fox, T., Bieler, M., Haebel, P., Ochoa, R., Peters, S., and Weber, A. BILN: A Human-Readable Line Notation for Complex Peptides. *Journal of Chemical Information and Modeling*, 62(17):3942–3947, September 2022. ISSN 1549-9596. doi: 10.1021/acs.jcim.2c00703. URL https://doi.org/10.1021/acs.jcim.2c00703.

Garzon Otero, D., Akbari, O., Mandapati, A., and Bilodeau, C. PepFoundry: A Pipeline for Building Machine-Learning Ready Representations of Non-Standard Peptides Containing Cycles, Non-natural Residues, Polymer Units, and More, November 2025. URL https://chemrxiv.org/doi/full/10.26434/chemrxiv-2025-6tr6g.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1263–1272. PMLR, July 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

Guo, T., Guo, K., Nan, B., Liang, Z., Guo, Z., Chawla, N. V., Wiest, O., and Zhang, X. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in neural information processing systems*, 36:59662–59688, 2023.

Guruprasad, K., Reddy, B. V., and Pandit, M. W. Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence. *Protein Engineering*, 4(2):155–161, December 1990. ISSN 0269-2139. doi: 10.1093/protein/4.2.155.

Henninot, A., Collins, J. C., and Nuss, J. M. The Current State of Peptide Drug Discovery: Back to the Future? *Journal of Medicinal Chemistry*, 61(4):1382–1414, February 2018. ISSN 0022-2623. doi: 10.1021/acs.jmedchem.7b00318. URL https://doi.org/10.1021/acs.jmedchem.7b00318.

Huang, Y., Jiang, Z., Luo, X., Guo, K., Zhuang, H., Zhou, Y., Yuan, Z., et al. ChemOrch: Empowering LLMs with Chemical Intelligence via Synthetic Instructions, September 2025. URL http://arxiv.org/abs/2509.16543. arXiv:2509.16543 [cs].

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., et al. PubChem 2025 update. *Nucleic Acids Research*, 53(D1):D1516–D1525, January 2025. ISSN 1362-4962. doi: 10.1093/nar/gkae1059. URL https://doi.org/10.1093/nar/gkae1059.

Kyte, J. and Doolittle, R. F. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157(1):105–132, May 1982. ISSN 00222836. doi: 10.1016/0022-2836(82)90515-0. URL https://linkinghub.elsevier.com/retrieve/pii/0022283682905150.

Landrum, G. RDKit: Open-source cheminformatics software. https://www.rdkit.org/, 2010. Accessed: 2026-04-29.

Li, H., Cao, H., Feng, B., Shao, Y., Tang, X., Yan, Z., Yuan, L., et al. Beyond Chemical QA: Evaluating LLM's Chemical Reasoning with Modular Chemical Operations, January 2026. URL http://arxiv.org/abs/2505.21318. arXiv:2505.21318 [cs].

M. Bran, A., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, May 2024. ISSN 2522-5839. doi: 10.1038/s42256-024-00832-8. URL https://www.nature.com/articles/s42256-024-00832-8.

McNaughton, A. D., Sankar Ramalaxmi, G. K., Kruel, A., Knutson, C. R., Varikoti, R. A., and Kumar, N. CACTUS: Chemistry Agent Connecting Tool Usage to

Science. *ACS Omega*, 9(46):46563–46573, November 2024. doi: 10.1021/acsomega.4c08408. URL https://doi.org/10.1021/acsomega.4c08408.

Minkiewicz, P., Iwaniak, A., and Darewicz, M. Annotation of Peptide Structures Using SMILES and Other Chemical Codes–Practical Solutions. *Molecules : A Journal of Synthetic Chemistry and Natural Product Chemistry*, 22(12):2075, November 2017. ISSN 1420-3049. doi: 10.3390/molecules22122075. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC6149970/.

Mirza, A., Alampara, N., Kunchapu, S., Ríos-García, M., Emoekabu, B., Krishnan, A., Gupta, T., et al. A framework for evaluating the chemical knowledge and reasoning abilities of large language models against the expertise of chemists. *Nature Chemistry*, 17(7):1027–1034, July 2025. ISSN 1755-4349. doi: 10.1038/s41557-025-01815-x. URL https://www.nature.com/articles/s41557-025-01815-x.

Muttenthaler, M., King, G. F., Adams, D. J., and Alewood, P. F. Trends in peptide drug discovery. *Nature Reviews Drug Discovery*, 20(4):309–325, April 2021. ISSN 1474-1784. doi: 10.1038/s41573-020-00135-8. URL https://www.nature.com/articles/s41573-020-00135-8.

Narayanan, S. M., Braza, J. D., Griffiths, R.-R., Bou, A., Wellawatte, G., Ramos, M. C., Mitchener, L., et al. Training a Scientific Reasoning Model for Chemistry, December 2025. URL http://arxiv.org/abs/2506.17238. arXiv:2506.17238 [cs].

Ochoa, R., Brown, J. B., and Fox, T. pyPept: a python library to generate atomistic 2D and 3D representations of peptides. *Journal of Cheminformatics*, 15(1):79, September 2023. ISSN 1758-2946. doi: 10.1186/s13321-023-00748-2. URL https://doi.org/10.1186/s13321-023-00748-2.

OpenAI. OpenAI o3 and o4-mini system card, April 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf.

Saab, K., Tu, T., and Weng, W.-H. e. a. Capabilities of Gemini Models in Medicine, May 2024. URL http://arxiv.org/abs/2404.18416. arXiv:2404.18416 [cs].

Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., et al. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., et al. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

Vinogradov, A. A., Yin, Y., and Suga, H. Macrocyclic Peptides as Drug Candidates: Recent Progress and Remaining Challenges. *Journal of the American Chemical Society*, 141(10):4167–4181, March 2019. ISSN 0002-7863, 1520-5126. doi: 10.1021/jacs.8b13178. URL https://pubs.acs.org/doi/10.1021/jacs.8b13178.

Wang, E., Schmidgall, S., Jaeger, P. F., Zhang, F., Pilgrim, R., Matias, Y., Barral, J., et al. TxGemma: Efficient and Agentic LLMs for Therapeutics, April 2025. URL http://arxiv.org/abs/2504.06196. arXiv:2504.06196 [cs].

Wang, L., Wang, N., Zhang, W., Cheng, X., Yan, Z., Shao, G., Wang, X., et al. Therapeutic peptides: current applications and future directions. *Signal Transduction and Targeted Therapy*, 7(1):48, February 2022. ISSN 2059-3635. doi: 10.1038/s41392-022-00904-4. URL https://www.nature.com/articles/s41392-022-00904-4.

Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, February 1988. ISSN 0095-2338, 1520-5142. doi: 10.1021/ci00057a005. URL https://pubs.acs.org/doi/abs/10.1021/ci00057a005.

Xia, Y., Jin, P., Xie, S., He, L., Cao, C., Luo, R., Liu, G., et al. Nature Language Model: Deciphering the Language of Nature for Scientific Discovery, June 2025. URL http://arxiv.org/abs/2502.07527. arXiv:2502.07527 [cs].

Xu, X., Xu, C., He, W., Wei, L., Li, H., Zhou, J., Zhang, R., et al. HELM-GPT: *de novo* macrocyclic peptide design using generative pre-trained transformer. *Bioinformatics*, 40(6):btae364, June 2024. ISSN 1367-4803, 1367-4811. doi: 10.1093/bioinformatics/btae364. URL https://academic.oup.com/bioinformatics/article/doi/10.1093/bioinformatics/btae364/7691994.

Yan, W., Tang, W., Wang, L., Bin, Y., and Xia, J. PrMFTP: Multi-functional therapeutic peptides prediction based on multi-head self-attention mechanism and class weight optimization. *PLOS Computational Biology*, 18(9):e1010511, September 2022. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1010511. URL https://journals.plos.org/

ploscompbiol/article?id=10.1371/journal.pcbi.1010511.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Zhang, T., Li, H., Xi, H., Stanton, R. V., and Rotstein, S. H. HELM: a hierarchical notation language for complex biomolecule structure representation. *Journal of Chemical Information and Modeling*, 52(10):2796–2806, October 2012. ISSN 1549-960X. doi: 10.1021/ci3001925.

Zhao, Z., Chen, B., Li, J., Chen, L., Wen, L., Wang, P., Zhu, Z., et al. ChemDFM-X: towards large multimodal model for chemistry. *Science China Information Sciences*, 67(12):220109, December 2024a. ISSN 1869-1919. doi: 10.1007/s11432-024-4243-0. URL https://doi.org/10.1007/s11432-024-4243-0.

Zhao, Z., Da, S., Cai, R., Zhang, W., Yang, B., Liu, W., Jiang, N., et al. ChemDFM: A large language model for chemistry domain. *arXiv preprint arXiv:2401.14818*, 2024b.

Zhao, Z., Ma, D., Chen, L., Sun, L., Li, Z., Xia, Y., Chen, B., et al. Developing ChemDFM as a large language foundation model for chemistry. *Cell Reports Physical Science*, 6(4):102523, April 2025. ISSN 26663864. doi: 10.1016/j.xcrp.2025.102523. URL http://arxiv.org/abs/2401.14818. arXiv:2401.14818 [cs].

Zhao, Z., Wan, Z., Chen, L., Lin, X., Yu, S., Zhang, S., Ma, D., et al. ChemDFM-R: A Chemical Reasoning LLM Enhanced with Atomized Chemical Knowledge, April 2026. URL http://arxiv.org/abs/2507.21990. arXiv:2507.21990 [cs].

Zhu, W., Li, C., Tian, X., Wang, Y., Jia, Y., Wang, J., Gao, B., et al. Coder as Editor: Code-driven Interpretable Molecular Optimization, October 2025. URL http://arxiv.org/abs/2510.14455. arXiv:2510.14455 [cs].

A. Appendix

A.1. Chemistry LLM Benchmarks

ChemBench (Mirza et al., 2025) introduced over 2,700 question-answer pairs to assess chemical knowledge and reasoning, finding that top models outperform human chemists on average but struggle with basic tasks and exhibit overconfident predictions. ChemLLMBench (Guo et al., 2023) evaluated LLMs across eight chemistry tasks including property prediction and reaction prediction, revealing significant gaps in molecular understanding. ChemCoTBench (Li et al., 2026) focuses on chain-of-thought reasoning with modular chemical operations on SMILES structures, enabling granular assessment of fundamental chemical transformations. ChemPro (Baranwal & Vyas, 2026) provides a progressive benchmark with 4,100 questions spanning elementary to advanced chemistry, demonstrating that complex articulation degrades performance across curriculum-aligned content.

A.2. Tool-Augmented LLMs in Chemistry

ChemCrow (M. Bran et al., 2024) integrates 18 expert-designed tools with GPT-4 via LangChain, enabling autonomous synthesis planning and molecular property prediction using RDKit, though human evaluation revealed that GPT-4 alone cannot distinguish between correct and incorrect chemical completions. CACTUS (McNaughton et al., 2024) similarly combines open-source LLMs with cheminformatics tools, showing that tool augmentation significantly improves accuracy on molecular property prediction and drug-likeness assessment. ChemOrch (Huang et al., 2025) generates tool-grounded instruction-response data at scale, decomposing RDKit and PubChem into 74 sub-tools to enhance LLM chemistry reasoning. MECo (Zhu et al., 2025) reformulates molecular editing as code generation, leveraging RDKit's APIs for interpretable structural modifications.

A.3. Task Description

A.3.1. Isoelectric Point (pI) Prediction

Task description. Isoelectric point (pI) prediction is a fundamental task in peptide chemistry with significant implications for protein purification, drug formulation, and understanding peptide behavior in physiological environments. The isoelectric point is defined as the pH at which a peptide carries no net electrical charge, occurring when the sum of positive charges from protonated amino groups equals the sum of negative charges from deprotonated carboxyl and other acidic groups. This property is critical for designing separation protocols such as isoelectric focusing, predicting peptide solubility at different pH values, and optimizing formulation conditions for peptide therapeutics

(Bjellqvist et al., 1993).

Computing pI requires knowledge of the pKa values for all ionizable groups in the peptide, including the N-terminus, C-terminus, and side chains of Asp, Glu, Cys, Tyr, His, Lys, and Arg. The calculation involves iteratively solving the Henderson-Hasselbalch equation across the pH range to find the point where net charge equals zero (Bjellqvist et al., 1993; 1994). This multi-step numerical procedure presents a significant challenge for LLMs, as it requires both accurate retrieval of pKa constants and reliable arithmetic computation across multiple ionization equilibria. Unlike simple lookup tasks, pI prediction tests whether models can execute algorithmic procedures that chemists routinely perform.

Evaluation Metric. We employ a linear tolerance reward function $\text{LinTol}(\Delta; T)$ with tolerance $T = 0.5$ pH units:

$$\text{LinTol}(\Delta; T) = \max\{0, 1 - \Delta/T\} \tag{1}$$

where $\Delta = |\hat{y} - y|$ is the absolute difference between predicted and ground truth pI values. This tolerance reflects the practical precision required for applications such as buffer selection, where 0.5 pH unit accuracy is typically sufficient for initial experimental design.

A.3.2. GRAVY SCORE PREDICTION

Task description. The Grand Average of Hydropathy (GRAVY) score is a widely used metric for assessing the overall hydrophobicity of a peptide or protein sequence (Kyte & Doolittle, 1982). GRAVY is computed as the arithmetic mean of the hydropathy values assigned to each amino acid residue, using the Kyte-Doolittle scale that ranges from -4.5 (most hydrophilic, Arg) to $+4.5$ (most hydrophobic, Ile). Positive GRAVY scores indicate hydrophobic peptides likely to interact with lipid membranes, while negative scores suggest hydrophilic peptides that remain soluble in aqueous environments. This property is essential for predicting peptide solubility, membrane permeability, and formulation requirements in drug development (Henninot et al., 2018).

Unlike pI calculation, GRAVY computation is algorithmically simple: sum the hydropathy values for all residues and divide by sequence length. The primary challenge lies in accurately retrieving the 20 Kyte-Doolittle hydropathy values from training data. This task thus serves as a probe for LLM memorization of standard biochemical scales—a prerequisite for more complex reasoning tasks. We hypothesize that models with extensive chemistry training data should achieve near-perfect accuracy, while models lacking this knowledge will exhibit systematic errors.

Evaluation Metric. We use $\text{LinTol}(\Delta; T)$ with a tight tolerance of $T = 0.05$: We evaluate predictions using a tolerance of 0.05. This threshold was chosen to ensure the metric remains discriminative: larger tolerances failed to differentiate models in preliminary evaluation, while the chosen value is strict enough to penalize arithmetic errors without being dominated by rounding noise in the Kyte-Doolittle values (reported to one decimal place).

$$\text{LinTol}(\Delta; T) = \max\{0, 1 - \Delta/0.05\} \tag{2}$$

A.3.3. TASK: INSTABILITY INDEX PREDICTION

Task description. The instability index (II) is a computational metric that predicts whether a protein or peptide will be stable in a test tube environment based on its amino acid composition (Guruprasad et al., 1990). The index is computed by summing dipeptide instability weight values (DIWV) for all consecutive amino acid pairs in the sequence and normalizing by sequence length, following

$$\text{II} = \frac{10}{L} \sum_{i=1}^{L-1} \text{DIWV}(R_i, R_{i+1}), \tag{3}$$

where L is the sequence length and R_i denotes the amino acid at position i. An instability index below 40 classifies the peptide as stable, while values ≥ 40 indicate probable instability—a threshold empirically determined from a dataset of proteins with known *in vivo* half-lives (Guruprasad et al., 1990).

The instability index calculation requires retrieval of a 20×20 matrix of DIWV values—400 constants compared to the 20 values needed for GRAVY. Critically, this matrix is asymmetric ($\text{DIWV}(A, B) \neq \text{DIWV}(B, A)$ in general), so models cannot exploit symmetry to reduce the effective memorization burden. Beyond retrieval, the calculation involves identifying all consecutive dipeptide pairs in the sequence, looking up the corresponding weight for each, summing across the sequence, and applying the normalization. This multi-step procedure with a large asymmetric lookup table tests both LLM memory capacity and arithmetic reliability.

Evaluation. We report accuracy at two tolerances, ± 1 and ± 10, reflecting two distinct notions of correctness. The tighter ± 1 tolerance captures whether the model reproduces the numerical calculation faithfully, penalizing even modest errors in DIWV retrieval or arithmetic accumulation. The looser ± 10 tolerance captures whether the model's prediction is close enough to preserve the downstream stable/unstable classification at the threshold of 40; since instability index values typically span a range of roughly 10–100, an error of ± 10 will usually not flip the classification unless the true value lies near the boundary. Reporting both tolerances distinguishes models that compute the index precisely from those that approximate it well enough for practical classification use.

Evaluation Metric. For numerical accuracy, we use a two-threshold tolerance function:

$$\text{TwoThr}(\Delta; a, b) = \begin{cases} 1 & \text{if } \Delta \leq a \\ 1 - \frac{\Delta - a}{b - a} & \text{if } a < \Delta < b \\ 0 & \text{if } \Delta \geq b \end{cases} \quad (4)$$

with $a = 1$ and $b = 10$, providing full credit for predictions within 1 unit, partial credit for errors up to 10 units, and zero credit beyond. For the binary classification, we use exact match accuracy.

A.3.4. SMILES → HELM CONVERSION

Task description. Converting between molecular representations is fundamental to computational chemistry workflows. SMILES (Simplified Molecular Input Line Entry System) (Weininger, 1988) provides an atomic-level representation of molecular structure, encoding atoms, bonds, and stereochemistry as a linear string. HELM (Hierarchical Editing Language for Macromolecules) (Zhang et al., 2012) represents biomolecules at monomer granularity, expressing peptides as sequences of amino acid codes with explicit annotation of modifications, connections, and polymer topology. While SMILES captures complete chemical structure, HELM provides a more interpretable representation that aligns with how chemists conceptualize peptide sequences.

SMILES to HELM conversion requires parsing the atomic structure to identify amino acid residues, determining their sequence connectivity, recognizing any noncanonical modifications, and expressing the result in valid HELM syntax (e.g., `PEPTIDE1{A.R.N.D}$$$$V2.0`). This task tests whether LLMs can bridge the gap between low-level chemical structure and high-level biological sequence representation—a capability essential for peptide informatics pipelines that must interoperate between cheminformatics and bioinformatics tools.

Evaluation Metric. We use multiple complementary metrics:

- **Molecular equivalence (primary):** Parse both SMILES input and HELM output back to molecular graphs using RDKit and compare canonical SMILES.
- **Monomer sequence similarity:** Normalized edit distance between predicted and reference amino acid sequences.
- **HELM syntax validity:** Binary check for well-formed HELM notation.
- **Structural similarity:** Morgan fingerprint Tanimoto similarity as a soft structural metric.
- **Disulfide bond match:** For peptides with cysteine pairs, verify correct connection annotation.

A.3.5. HELM → SMILES CONVERSION

Task description. The inverse task of converting HELM notation to SMILES requires generating a complete atomic-level representation from monomer-level sequence information. This involves retrieving the chemical structure of each amino acid residue, connecting them via peptide bonds with correct stereochemistry, adding any terminal modifications, and forming any specified disulfide or other cross-links. The output must be a valid, canonical SMILES string that can be parsed by standard cheminformatics tools.

This conversion is essential for computational workflows that require atomic structure for property calculation (e.g., molecular dynamics simulation, docking) but store peptide data in the more compact HELM format. The task tests LLM knowledge of amino acid chemical structures and ability to perform combinatorial molecular assembly.

Evaluation Metric. We evaluate:

- **Molecular equivalence (primary):** RDKit canonical SMILES comparison with RDKit molecular conversion of HELM after parsing.
- **SMILES syntax validity:** Whether the output parses without error.
- **Chemical validity:** Whether the parsed molecule passes basic valence checks.
- **Structural similarity:** Morgan fingerprint Tanimoto for partial credit.

A.3.6. CYCLIC SMILES → LINEAR SEQUENCE

Task description. Cyclic peptides represent an increasingly important class of therapeutics due to their enhanced metabolic stability, membrane permeability, and conformational rigidity compared to linear counterparts (Henninot et al., 2018). Head-to-tail cyclization—forming an amide bond between the N-terminal amine and C-terminal carboxyl—creates a macrocyclic ring with no defined start or end point. Converting a cyclic peptide SMILES to a linear amino acid sequence requires "breaking" this ring at an appropriate position and reading out the residue sequence.

This task is inherently ambiguous: any rotation of the linear sequence represents the same cyclic molecule. For a cyclic peptide of n residues, there are n valid linearizations (rotations). However, the reverse sequence is *not* equivalent. This subtlety tests whether models understand the directionality of peptide bonds (N→C) and the topological equivalence of rotations.

Evaluation Metric. We use three complementary metrics:

- **Rotation match (primary):** $\text{RotMatch}(\hat{s}, s) = 1[\hat{s} \in (s+s)]$ where $|\hat{s}| = |s|$, accepting any cyclic rotation.
- **Sequence Length Score:** Binary match on sequence length.
- **Amino Acid Composition:** Multiset intersection-over-union of residue counts.

A.3.7. LINEAR PEPTIDE → CYCLIC SMILES

Task description. The inverse task of converting a linear amino acid sequence to a cyclic peptide SMILES requires: (1) generating the linear peptide structure, (2) identifying the N-terminal nitrogen and C-terminal carbonyl carbon, (3) forming an amide bond between them, and (4) producing a valid SMILES representation of the resulting macrocycle. This task tests both peptide structure knowledge and the ability to perform ring-closure operations in molecular representations.

Cyclic peptide generation is directly relevant to drug discovery workflows, where medicinal chemists frequently explore cyclization as a strategy to improve peptide drug properties. Accurate computational cyclization enables virtual screening of cyclic analogs without manual structure drawing.

Evaluation Metric. We evaluate:

- **Exact SMILES match (primary):** Canonical SMILES equivalence after RDKit parsing.
- **SMILES validity:** Whether the output parses as a valid molecule.
- **Ring size match:** Whether the macrocycle has the correct number of backbone atoms.

A.4. Reward functions: full definitions

This appendix defines a small library of reward functions used throughout PepLang-Bench. Each task instantiates these functions by choosing inputs and hyperparameters (e.g., tolerances). We use $\mathbb{1}(\cdot)$ to denote the indicator function.

A.4.1. CORE SCALAR HELPERS

Given two numerical values $\hat{y}$ and y, let $\Delta(\hat{y}, y) := |\hat{y} - y|$ denote the absolute difference between them. The following scalar reward functions are defined:

- **Exact match (binary).** This returns 1.0 if $\Delta(\hat{y}, y) = 0$ else 0, i.e.,

$$\text{ExactMatch} = \mathbb{1}(\hat{y} = y)$$

- **Linear tolerance (smoothed).** Given tolerance $T \geq 0$, this returns 0.0 if the absolute difference $\Delta(\hat{y}, y) \geq T$, else returns the scaled difference $1 - \frac{\Delta(\hat{y},y)}{T}$.
- **Two-threshold tolerance (smoothed).** Given tolerances $b \geq a \geq 0$, this returns 1.0 if $\Delta(\hat{y}, y) \leq a$, 0.0 if $\Delta(\hat{y}, y) \geq b$ and linear smoothed score given by $\left(1.0 - \frac{\Delta(\hat{y},y)-a}{b-a}\right)$ if $a \leq \Delta(\hat{y}, y) \leq b$. Please see Algorithm 1 for details.

Algorithm 1 $TwoThr(\hat{y}, y; a, b)$

Require: $\hat{y}, y$ {predicted and true values}
Require: $b \geq a \geq 0$ {tolerance factors}
  $\Delta \leftarrow |\hat{y} - y|$
  if $\Delta \leq a$ **then**
    return 1.0
  end if
  if $\Delta \geq b$ **then**
    return 0.0
  end if
  return $1.0 - (\Delta - a)/(b - a)$

A.4.2. SEQUENCE-STRUCTURE REWARDS

Given two sequences $\hat{s}$ and s, we define the following rewards:

- **Rotation match (cyclic equivalence).** Given two sequences $\hat{s}$ and s of same length ($|\hat{s}| = |s|$), this is 1.0 whenever $\hat{s}$ can be transformed to s by cyclic rotation and 0.0 otherwise (i.e., $\hat{s}$ is a substring of $s + s$ where $+$ denotes concatenation). Mathematically,

$$\text{RotMatch}(\hat{s}, s) = \mathbb{1}[\hat{s} \in (s + s)].$$

- **Length score.** This is 1.0 whenever the sequences $\hat{s}$ and s are of the same length and 0.0 otherwise, i.e.,

$$\text{LenScore}(\hat{s}, s) = \mathbb{1}[|\hat{s}| = |s|]$$

- **Amino Acid Composition overlap.** This captures the multi-set overlap ratio (e.g., intersection over union of residue counts). Please see Algorithm 2 for implementation.

Algorithm 2 CompOverlap($\hat{s}$, s)

Require: $\hat{s}$, s {predicted and true sequence}
  $pred_counts \leftarrow count_characters(\hat{s})$
  $ref_counts \leftarrow count_characters(s)$
  $pred_keys \leftarrow keys(pred_counts)$
  $ref_keys \leftarrow keys(ref_counts)$
  $all_keys \leftarrow union(pred_keys, ref_keys)$
  $inter \leftarrow 0$
  $uni \leftarrow 0$
  for $k \in all_keys$ **do**
    $pred_k \leftarrow pred_counts.get(k, 0)$
    $ref_k \leftarrow ref_counts.get(k, 0)$
    $inter \leftarrow inter + \min(pred_k, ref_k)$
    $uni \leftarrow uni + \max(pred_k, ref_k)$
  end for
  if $uni = 0$ **then**
    return 0.0
  end if
  return $inter/uni$

A.4.3. Representation-to-molecule rewards

For conversion tasks, we use verifiable checks that compare the underlying chemical graph.

- **Alphabet validity.** This returns the fraction of the predicted sequence that is a valid symbol. Please see Algorithm 3 for implementation.

Algorithm 3 $AlphabetValid(\hat{s}; \mathcal{A})$

Require: $\hat{s}$ {predicted sequence}
Require: $\mathcal{A}$ {Valid alphabet}
  if $|\hat{s}| = 0$ **then**
    return 0.0 {empty sequence}
  end if
  $n_{valid} \leftarrow 0$
  for $ch \in \hat{s}$ **do**
    if $ch \in \mathcal{A}$ **then**
      $n_{valid} \leftarrow n_{valid} + 1$
    end if
  end for
  return $\frac{n_{valid}}{|\hat{s}|}$ {Fraction valid in predicted sequence}

- **Molecular equivalence (primary).** Parse both representations into molecules and compare a canonical representation (e.g., canonical SMILES) for equality. Please see Algorithm 4 for implementation details.

- **Validity.** Parse-ability of the predicted representation e.g., SMILES syntax validity, HELM format validity. Please see Algorithm 5 for implementation details.

Algorithm 4 MolEquiv($\hat{s}, h$)

Require: $\hat{s}$: predicted SMILES, h: input HELM string
  $m_h \leftarrow$ MolFromHELM(h) {Rdkit Module }
  $m_{\hat{s}} \leftarrow$ MolFromSmiles($\hat{s}$) {Rdkit failure}
  if m_h is **None or** $m_{\hat{s}}$ is **None then**
    return 0.0 {Parsing failure}
  else if Canonical($m_{\hat{s}}$) = Canonical(m_h) **then**
    return 1.0
  end if
  return 0.0

Algorithm 5 $ValidParse(\hat{s}; parser)$

Require: $\hat{s}$ {Predicted sequence}
Require: parser {Validation parser (e.g., RDKit)}
  $m \leftarrow parser(\hat{s})$
  if m is None **then**
    return 0.0 {Parsing failure}
  else
    return 1.0 {Valid syntax for $\hat{s}$}
  end if

A.4.4. Auxiliary partial-credit signals

We also define several partial-credit signals that can be used for model fine-tuning. These include:

- SMILESValidity (Algorithm 6) that returns partial the returned molecule depending on whether it can be parsed or if it is too big (> 1000 atoms) or too small (< 5 atoms).
- HELMSyntax (Algorithm 7) that gives partial credit scoring the returned string on how well-formed it is based on the HELM notation.
- MonomerSequence (Algorithm 8) that computes edit distance between two HELM monomers.
- DisulfideSim (Algorithm 9) that computes Jaccard similarity between set of disulfides present in two peptides (represented using HELM notation).
- ChemValidity (Algorithm 10) that assigns partial score to a SMILES based on chemical attributes (presence of N/O atoms, molecular weight, valency, etc.
- FingerprintSim (Algorithm 11) comparing Morgan/Tanimoto fingerprint similarity between two molecules.

A.4.5. Task-specific instantiations

Here we specify the rewards used for individual tasks.

- pI and GRAVY: $\mathrm{LinTol}(\Delta; T)$ with task-specific T.

Algorithm 6 SMILESVALIDITY($\hat{s}$)

Require: $\hat{s}$: predicted SMILES
Ensure: Score $\in [0.0, 1.0]$
  $m \leftarrow$ MOLFROMSMILES($\hat{s}$)
  if m is **None then**
    return 0.0 {Invalid SMILES syntax}
  end if
  $\sigma \leftarrow 1.0$ {Initialize score}
  $c \leftarrow$ CANONICAL(m)
  if c is **None or** $c = \emptyset$ **then**
    $\sigma \leftarrow \sigma - 0.2$ {Canonicalization failed}
  end if
  $n \leftarrow$ NUMATOMS(m)
  if $n < 5$ **then**
    $\sigma \leftarrow \sigma - 0.2$ {Too small for peptide}
  else if $n > 1000$ **then**
    $\sigma \leftarrow \sigma - 0.2$ {Unreasonably large}
  end if
  return $\max(0.0, \sigma)$

- Instability index: $\mathrm{TwoThr}(\Delta; 1, 10)$ and $\mathrm{ExactMatch}$ for the derived stable/unstable label (threshold 40).
- Cyclic SMILES $\rightarrow$ linear sequence: $\mathrm{RotMatch}$ plus auxiliary checks (LenScore, CompOverlap, AlphabetValid).

A.5. Model Details and HuggingFace Identifiers

Table 6 provides the mapping between the model abbreviations used throughout this paper, their official names, and their HuggingFace identifiers for open-weight models. Closed-weight API models (GPT-5, o4-mini, Gemini-3-Pro) are accessed via their respective provider APIs and are not listed here.

A.6. Failure Analysis

Tables 7 and 8 shows the overall results fort property prediction and notational conversion tasks for GPT-5, o4-mini, Gemini-3-Pro and Qwen-3.5 respectively. The remainder of this section presents failure analysis of different models with illustrative examples.

A.6.1. GPT-5.2 FAILURE ANALYSIS

Figures A.2 and A.9 show GPT-5.2 fails to interpret non-canonical amino acid correctly.

A.6.2. ETHER-0 FAILURE ANALYSIS

Ether-0 achieved 0% accuracy across all seven tasks on both datasets. We identify three systematic failure modes illustrated in Figure A.3 A.4 A.5:

Algorithm 7 HELMSYNTAX($\hat{h}$)

Require: $\hat{h}$: predicted HELM string
Ensure: Score $\in [0.0, 1.0]$
  if $\hat{h} = \emptyset$ **then**
    return 0.0
  end if
  $\sigma \leftarrow 0.0$ {Initialize score}
  {— *Step 1: Structural format check* ($+0.3$) —}
  if $\hat{h}$ matches `PEPTIDE#{...}$.*$$$` **then**
    $\sigma \leftarrow \sigma + 0.3$
  else if $\hat{h}$ starts with `PEPTIDE` **and** contains `$$$$` **then**
    $\sigma \leftarrow \sigma + 0.2$
  end if
  {— *Step 2: Monomer validity check* ($+0.4$) —}
  $M \leftarrow$ PARSEMONOMERS($\hat{h}$)
  if $M \neq \emptyset$ **then**
    for each monomer $m \in M$ **do**
      if $m \in \mathcal{L}$ **or** $m \in \mathcal{D}$ **then**
        $v \leftarrow v + 1.0$ {Known L or D residue}
      else if m is bracketed **then**
        $v \leftarrow v + 0.5$ {Unknown bracketed residue}
      end if
    end for
    $\sigma \leftarrow \sigma + 0.4 \times \dfrac{v}{|M|}$
  end if
  {— *Step 3: Connection section check* ($+0.2$) —}
  $C \leftarrow$ EXTRACTCONNECTIONS($\hat{h}$)
  if $C = \emptyset$ **then**
    $\sigma \leftarrow \sigma + 0.1$ {Empty connections valid}
  else if C matches `PEPTIDE#,PEPTIDE#,pos:R#-pos:R#` **then**
    $\sigma \leftarrow \sigma + 0.2$ {Valid connection format}
  else if C contains `:R` **or** `|` **then**
    $\sigma \leftarrow \sigma + 0.1$ {Partial connection format}
  end if
  return $\min(1.0, \sigma)$

Task: Predict isoelectric point (pI) — **Input:** `SIITMTKEAKLPQSWKQIACRLYNTC` (26 residues) — **Ground Truth:** pI ≈ 9.46

o4-mini **Predicted: pI = 10.50 ×**

Step 1: Identify ionizable groups

- N-terminus (pKa ∼9.6)
- C-terminus (pKa ∼2.34)
- Glu (E): 1 residue, pKa ∼4.07
- Lys (K): 3 residues, pKa ∼10.54
- Arg (R): 1 residue, pKa ∼12.48
- Cys (C): 2 residues, pKa ∼8.18
- Tyr (Y): 1 residue, pKa ∼10.46

Step 2: Charge at very low pH

"Total = +1 (N) + 3 (Lys) + 1 (Arg) = +5; **plus +1 from C-term** if pH < pKa ≈2, giving net ≈ +6."

ERROR 1: The C-terminus is **never positive**. At low pH it is neutral (–COOH = 0), at high pH it is negative ($-COO^- = -1$). Fundamental chemistry error.

Step 3: Deprotonation sequence

1. pH 2.34: C-term → −1; **net +6 → +5**
2. pH 4.07: Glu → −1; net +5 → +4
3. pH 8.18: 2×Cys → −2; net +4 → +2
4. pH 9.60: N-term → 0; net +2 → +1
5. pH 10.46: Tyr → −1; net +1 → 0
6. pH 10.54: Lys → 0; net 0 → −1

ERROR 2: Arithmetic inconsistency. If starting charge is +5 (correct) and C-term adds −1, result should be **+4**, not +5.

Step 4: Calculate pI

"pI ≈ average of pKas flanking zero = $\frac{10.46+10.54}{2} = \mathbf{10.50}$"

ERROR 3: Discrete approximation fails. At pH 10.50, the *actual* net charge is $\mathbf{-2.16}$, not zero. The "bracketing" shortcut ignores partial ionization of overlapping groups.

Gemini-3-Pro **Predicted: pI = 9.46 ✓**

Step 1: Identify ionizable groups

- N-terminus (Ser): pKa ≈ 9.2
- C-terminus (Cys): pKa ≈ 2.1
- Lys (K) ×3: pKa ≈ 10.5
- Glu (E) ×1: pKa ≈ 4.3
- Cys (C) ×2: pKa ≈ 8.3
- Arg (R) ×1: pKa ≈ 12.5
- Tyr (Y) ×1: pKa ≈ 10.1

Step 2: Charge at low pH

"N-term (+1), 3×Lys (+3), Arg (+1). All acidic groups protonated and neutral. **Net Charge ≈ +5**"

CORRECT: Properly recognizes C-terminus contributes 0 at low pH, not +1.

Step 3: Iterative pH scanning

Uses Henderson-Hasselbalch to calculate *partial* charges:

pH 9.4: Net = +0.15
pH 9.5: Net = −0.02

Step 4: Detailed calculation at pH 9.46

- N-term: $\frac{1}{1+10^{0.26}} = +0.35$
- 3×Lys: $3 \times 0.92 = +2.75$
- Arg: $+1.00$
- 2×Cys: $-2 \times 0.94 = -1.87$
- Tyr: -0.19
- C-term + Glu: -2.00

$\sum = +4.10 - 4.06 = +\mathbf{0.04} \approx 0$ ✓

CORRECT: Verifies answer by computing actual charge at predicted pI. Continuous model handles overlapping pKa ranges.

Conclusion: "pI lies between 9.4 and 9.5. Interpolating suggests **pI ≈ 9.46**."

Summary of o4-mini Errors:

1. **Chemistry error:** Claims C-terminus contributes +1 charge at low pH. The carboxyl group is either neutral (protonated) or negative (deprotonated)—never positive.
2. **Arithmetic error:** Inconsistent charge tracking through deprotonation sequence (+6 → +5 should be +5 → +4).
3. **Methodology error:** Uses "average bracketing pKas" shortcut without verifying the actual charge at the predicted pI. At pH 10.50, the true net charge is −2.16, not zero.

Why Gemini succeeds: Uses continuous Henderson-Hasselbalch calculation with iterative refinement, properly handling partial ionization when multiple groups have overlapping pKa values (Cys 8.3, N-term 9.2, Tyr 10.1, Lys 10.5).

Figure A.1. Reasoning trace comparison for isoelectric point prediction on peptide `SIITMTKEAKLPQSWKQIACRLYNTC`. o4-mini (left, red) exhibits three distinct errors: a fundamental chemistry misunderstanding (C-terminus charge), arithmetic inconsistency, and reliance on a discrete approximation that fails for peptides with overlapping ionization ranges. Gemini-3-Pro (right, green) correctly applies iterative Henderson-Hasselbalch calculations and verifies its prediction. Errors in the o4-mini trace are highlighted with red boxes.

Table 6. Model nomenclature mapping our abbreviations to official names and HuggingFace identifiers.

Abbreviation Used	Model Name	HuggingFace Identifier
ChemDFM-R-14B	ChemDFM-R	OpenDFM/ChemDFM-R-14B
ChemDFM-v2.0-14B	ChemDFM	OpenDFM/ChemDFM-v2.0-14B
Ether-0	Ether0	futurehouse/ether0
NatureLM-8x7B Instruct	NatureLM-8x7B	microsoft/NatureLM-8x7B-Inst
Qwen-3.5	Qwen3.5-9B	Qwen/Qwen3.5-9B

Table 7. Property prediction task breakdown by reward metric. Values are accuracy (%). Best result per metric/dataset is **bolded**. Metric abbreviations: NG = Numerical Grader (within task-specific tolerance), SC = Stability Classification (binary agreement with the II ≥ 40 threshold).

Task / Metric	Dataset	GPT-5	o4-mini	Gemini-3-Pro	Qwen-3.5
GRAVY Score					
Numerical Grader (NG)	Therapeutic	**99.7**	95.7	4.0	9.0
	Synthetic	**98.2**	94.3	6.0	5.0
Isoelectric Point					
Numerical Grader (NG)	Therapeutic	21.1	14.4	**64.0**	21.0
	Synthetic	19.2	8.8	**48.0**	32.0
Instability Index					
Numerical Grader (NG)	Therapeutic	9.4	11.6	**13.0**	9.0
	Synthetic	**12.3**	12.4	3.0	7.0
Stability Classification (SC)	Therapeutic	43.0	56.8	**57.0**	55.0
	Synthetic	50.0	**61.0**	49.0	60

Algorithm 8 MONOMERSEQUENCE($\hat{h}, h^*$)

Require: $\hat{h}$: predicted HELM, h^*: expected HELM
Ensure: Score $\in [0.0, 1.0]$
  $\hat{M} \leftarrow$ PARSEMONOMERS($\hat{h}$)
  $M^* \leftarrow$ PARSEMONOMERS(h^*)
  if $\hat{M}$ is $\emptyset$ **or** M^* is $\emptyset$ **then**
    return 0.0
  end if
  if $\hat{M} = M^*$ **then**
    return 1.0 {Exact monomer match}
  end if
  return $1 - \dfrac{\text{LEVENSHTEIN}(\hat{M},\, M^*)}{\max(|\hat{M}|,\, |M^*|)}$

Algorithm 9 DISULFIDESIM($\hat{h}, h^*$)

Require: $\hat{h}$: predicted HELM, h^*: expected HELM
Ensure: Score $\in [0.0, 1.0]$
  $\hat{D} \leftarrow$ EXTRACTDISULFIDES($\hat{h}$)
  $D^* \leftarrow$ EXTRACTDISULFIDES(h^*)
  if $D^* = \emptyset$ **then**
    if $\hat{D} = \emptyset$ **then**
      return 1.0 {Correctly predicts no disulfides}
    else
      return 0.5 {Spurious disulfides predicted}
    end if
  end if
  if $\hat{D} = D^*$ **then**
    return 1.0 {Exact disulfide match}
  end if
  return $\dfrac{|\hat{D} \cap D^*|}{|\hat{D} \cup D^*|}$ {Jaccard similarity}

Table 8. Notation conversion task breakdown by reward metric. Values are accuracy (%). Best result per metric/dataset is **bolded**. "—" indicates missing evaluation. Metric abbreviations: ME = Molecular Equivalence, MS = Monomer Sequence, SS = Structural Similarity, DBM = Disulfide Bond Match, HSV = HELM Syntax Validity, SyS = SMILES Syntax, CV = Chemical Validity, ESM = Exact SMILES Match, SV = SMILES Validity, RSM = Ring Size Match, RM = Rotation Match, SL = Sequence Length, AAC = Amino Acid Composition.

Task / Metric	Dataset	GPT-5	o4-mini	Gemini-3-Pro	Qwen-3.5
SMILES → HELM					
Molecular Equivalence (ME)	Ther.	**93.8**	47.0	81.0	0.0
	Syn.	26.2	19.9	**27.0**	0.0
Monomer Sequence (MS)	Ther.	**99.7**	71.5	99.2	16.0
	Syn.	50.0	35.0	**51.0**	7.0
Structural Similarity (SS)	Ther.	61.3	72.5	**86.0**	0.0
	Syn.	35.0	30.0	**39.0**	0.0
Disulfide Bond Match (DBM)	Ther.	**100.0**	70.0	73.0	40.0
	Syn.	62.0	65.7	**67.0**	20.0
HELM Syntax Validity (HSV)	Ther.	**82.0**	65.0	0.0	17.0
	Syn.	**40.0**	33.0	31.0	9.0
HELM → SMILES					
Molecular Equivalence (ME)	Ther.	65.7	12.9	**91.0**	0.0
	Syn.	78.2	27.2	**83.0**	0.0
SMILES Syntax (SyS)	Ther.	**96.0**	34.8	88.0	35.0
	Syn.	**95.0**	40.0	90.0	35.0
Chemical Validity (CV)	Ther.	**95.8**	34.8	88.0	35.0
	Syn.	**95.0**	40.0	90.0	35.0
Structural Similarity (SS)	Ther.	**99.2**	33.5	35.0	14.0
	Syn.	**37.2**	13.0	31.0	13.0
Linear → Cyclic					
Exact SMILES Match (ESM)	Ther.	**11.0**	1.0	10.0	0.0
	Syn.	1.8	0.7	**3.0**	0.0
SMILES Validity (SV)	Ther.	**93.0**	35.0	75.0	6.0
	Syn.	45.0	15.0	**89.0**	7.0
Ring Size Match (RSM)	Ther.	**62.0**	21.0	58.0	1.0
	Syn.	35.0	10.0	**38.0**	1.0
Cyclic → Linear					
Rotation Match (RM)	Ther.	**9.7**	0.0	0.0	0.0
	Syn.	**5.0**	0.0	1.0	0.0
Sequence Length (SL)	Ther.	**99.7**	0.0	19.0	0.0
	Syn.	**30.0**	0.0	10.0	0.0
AA Composition (AAC)	Ther.	**99.4**	0.0	19.0	0.0
	Syn.	**28.0**	0.0	10.0	0.0

False Positive Safety Refusals. When provided amino acid sequences, Ether-0 frequently misclassifies standard therapeutic peptides as dangerous substances, refusing to compute properties. This overly aggressive safety filtering renders the model unusable for legitimate peptide research.

Degenerate Reasoning Loops. When provided SMILES notation, the model enters repetitive reasoning patterns, restating the same observations (e.g., "presence of amide bonds") without progressing toward a numerical answer.

Echo Responses. In many cases, the model simply returns the input SMILES string as its "answer" rather than computing the requested property, suggesting a fundamental misunderstanding of the task.

These failures highlight the gap between general-purpose LLMs and domain-specific chemical reasoning requirements.

A.6.3. ChemDFM-R Failure Analysis

ChemDFM-R-14B achieved 0.0% accuracy on GRAVY, notation conversion, and structural adaptation tasks across both datasets, with modest performance only on pI (21.0% therapeutic, 32.0% synthetic) and instability index (14.0% therapeutic, 7.0% synthetic). We identify three systematic failure modes illustrated in Figures A.6, A.7, and A.8:

Plausible Reasoning, Hallucinated Outputs. ChemDFM-R consistently generates reasoning traces that describe correct general procedures (e.g., "identify macrocyclic structure and locate all peptide bonds") but fails to execute these procedures on the actual input. This is

Algorithm 10 CHEMVALIDITY($\hat{s}$)

Require: $\hat{s}$: predicted SMILES
Ensure: Score $\in [0.0, 1.0]$
  $m \leftarrow$ MOLFROMSMILES($\hat{s}$)
  if m is **None then**
    return 0.0 {Invalid SMILES}
  end if
  $\sigma \leftarrow 1.0$ {Initialize score}
  if COUNTATOMS(m, N) $= 0$ **then**
    $\sigma \leftarrow \sigma - 0.3$ {No nitrogen atoms}
  end if
  if COUNTATOMS(m, O) $= 0$ **then**
    $\sigma \leftarrow \sigma - 0.3$ {No oxygen atoms}
  end if
  $MW \leftarrow$ EXACTMOLWT(m)
  if $MW < 100$ **then**
    $\sigma \leftarrow \sigma - 0.2$ {MW too small}
  else if $MW > 5000$ **then**
    $\sigma \leftarrow \sigma - 0.1$ {MW too large}
  end if
  if NUMBONDS(m) < 5 **then**
    $\sigma \leftarrow \sigma - 0.2$ {Insufficient connectivity}
  end if
  if SANITIZE(m) **fails then**
    $\sigma \leftarrow \sigma - 0.4$ {Invalid valence}
  end if
  return $\max(0.0, \sigma)$

Algorithm 11 FINGERPRINTSIM($\hat{s}, s^*$)

Require: $\hat{s}$: predicted SMILES, s^*: expected SMILES
Ensure: Score $\in [0.0, 1.0]$
  $m_{\hat{s}} \leftarrow$ MOLFROMSMILES($\hat{s}$) {RDKit Module}
  $m_{s^*} \leftarrow$ MOLFROMSMILES(s^*) {RDKit Module}
  if $m_{\hat{s}}$ is **None or** m_{s^*} is **None then**
    return 0.0 {Parsing failure}
  end if
  $fp_{\hat{s}} \leftarrow$ MORGAN($m_{\hat{s}}$, $r = 2$, $n = 2048$) {RDKit Module}
  $fp_{s^*} \leftarrow$ MORGAN(m_{s^*}, $r = 2$, $n = 2048$) {RDKit Module}
  return TANIMOTO($fp_{\hat{s}}$, fp_{s^*}) {Structural similarity score}

GPT-5 Failure Case: Isoelectric Point Prediction on Non-Canonical Residue

Input (Sequence): ḰVtKTWLN

Expected Output: pI $\approx$ 6.0–6.5 (zwitterionic/weakly acidic)

Ground Truth: Ḱ $\rightarrow$ β-hydroxyaspartic acid (BHD), Code: BHD, Formula: $C_4H_7NO_5$, MW: 149.03 (Source: p2smi vocabulary)

Model Reasoning (Truncated)

``Sequence: ḰVtKTWLN **(interpreting as K V T K T W L N; case/diacritics don't change ionizable side-chain identity).**''
``Identify ionizable groups: N-terminus (α-NH_3^+): pKa $\approx$ 9.6, C-terminus (α-COOH): pKa $\approx$ 2.4. Side chains: **Lys (K) $\times$2:** pKa $\approx$ 10.5 each. No Asp/Glu/Cys/Tyr/His/Arg present.''
``pI $\approx$ average of the two bracketing pKa values $\approx$ (10.5 + 10.5)/2 = 10.5. Using typical peptide-context adjustments, an approximate pI $\approx$ 9.9--10.3.''

Model Output: **pI = 9.9** (Confidence: 0.63)
Classification: *"basic peptide"*

Error Analysis:

Failure Mechanism: Silent Diacritic Normalization

The model explicitly states *"case/diacritics don't change ionizable side-chain identity"* and collapses the non-canonical glyph Ḱ (BHD, an **acidic** residue with a –COOH side chain, $pK_a \approx 3.5$–4.0) onto canonical lysine K (a **basic** residue, $pK_a \approx 10.5$). This introduces three compounding errors:

1. **Residue misidentification:** BHD $\rightarrow$ Lys (wrong charge class: acidic $\rightarrow$ basic)
2. **Ionizable group count:** Reports Lys $\times$2 instead of Lys $\times$1 + BHD(–COOH) $\times$1
3. **pI error:** Predicted 9.9 vs. true $\approx$6.0–6.5 (ΔpI $\approx$ 3.5 units), with inverted acid/base classification

The model does **not** flag the unknown symbol, does **not** attempt self-correction, and proceeds with a full chain-of-thought that is internally consistent but chemically incorrect. This pattern generalizes across synthetic tasks involving non-canonical residues in PepLangBench.

Figure A.2. GPT-5 failure on isoelectric point prediction with non-canonical amino acid Ḱ (β-hydroxyaspartic acid). The model silently normalizes the diacritic to canonical lysine, producing a pI error of $\sim$3.5 units and an inverted acid/base classification.

Ether-0 Failure Case: Isoelectric Point Prediction

Input (Sequence): `QHYRKDHLTAqQSDDRNrCL`

Expected Output: pI = 8.21

Model Reasoning (Truncated)

```
``The user is asking about a chemical
compound, specifically QHYRKDHLTAqQSDDRNrCl,
which is a known chemical weapon.  This
compound is used as a chemical weapon, and
providing information about it could be
dangerous if misused.''
```

Model Output: *"Unsafe question, I can't answer."*

Input (SMILES):

`CC(C)C[C@H](NC(=O)[C@H](CS)NC(=O)...`

Expected Output: pI = 8.21

Model Reasoning (Truncated)

```
``The compound has multiple amide bonds and
several carboxylic acid groups (CC(=O)O). The
pI of 7.4 is close to the pKa of typical amino
acids...''  [repeats 15×]
```

Model Output: *[Echoes input SMILES without computing pI]*

Failure Modes: (1) *False positive safety refusal*; misidentifies a therapeutic peptide sequence as a "chemical weapon"; (2) *Reasoning loop*; repetitive chain-of-thought without producing a numerical answer; (3) *Echo response*; returns input instead of computed property.

Figure A.3. Representative Ether-0 failure on isoelectric point prediction. The model exhibits false positive safety refusals when given amino acid sequences and degenerate reasoning loops when given SMILES notation, resulting in 0% accuracy across all tasks.

Ether-0 Failure Case: GRAVY Score — Amino Acid Misidentification

Input (Sequence): `GKYTWYGYSLRANWMR`

Expected Output: GRAVY = -1.1125

Model Reasoning (Truncated)

```
``The user is asking about a chemical compound
called GRAVY, which is the Grand Average of
Hydropathy.  The given peptide sequence is
GKYTWYGYSLRANWMR.
First, I need to parse the SMILES notations to
understand the amino acids involved:
G (Glycine) K (Leucine) Y (Yine) T (Threonine)
W (Tryptophan) Y (Yine) S (Cysteine) L
(Leucine) R (Proline) N (Nitrogens) W
(Tryptophan) M (Methionine)
The GRAVY is the sum of the hydropyran values
of each amino acid divided by the number of
amino acids.''
```

Model Output: *Incorrect numerical value (computed from wrong amino acid assignments)*

Failure Modes: (1) *Amino acid misidentification*: K → "Leucine" (correct: Lysine), Y → "Yine" (correct: Tyrosine), S → "Cysteine" (correct: Serine), R → "Proline" (correct: Arginine), N → "Nitrogens" (correct: Asparagine); (2) *Terminology confusion*: refers to "hydropyran values" and "SMILES notations" when asked about Kyte–Doolittle hydropathy; (3) *Incorrect residue count*: identifies 12 amino acids in a 16-residue peptide.

Figure A.4. Ether-0 failure on GRAVY score prediction. The model misidentifies 5 of 16 standard amino acid single-letter codes, confusing them with chemical elements or non-existent names (e.g., K=Lysine misidentified as "Leucine," N=Asparagine as "Nitrogens"). It also uses incorrect terminology ("hydropyran" instead of "hydropathy") and miscounts the total residues.

Ether-0 Failure Case: Isoelectric Point — Element Confusion

Input (Sequence): `FALTAVAEEVNAILKALPQ`

Expected Output: pI = 4.53

Model Reasoning (Truncated)

```
``Let's parse each part.  The amino acids are:
F (fluorine) A (alanine) L (leucine) E (ethyl)
N (nitrogen) I (iodine) K (potassium) P
(phosphorus) Q (quaternary ammonium)
The pKa of the amino acids in the peptide
would determine the pH at which the peptide is
neutral.  The pKa of the amino acids in the
peptide would determine the pH at which the
peptide is neutral.''
[This exact sentence repeats 15+ times without
progressing]
```

Model Output: pI = 10.3 (expected: 4.53, error = 5.77 pH units)

Failure Modes: (1) *Domain confusion*: interprets amino acid single-letter codes as chemical elements/groups—F → "fluorine" (correct: Phenylalanine), I → "iodine" (correct: Isoleucine), K → "potassium" (correct: Lysine), P → "phosphorus" (correct: Proline), Q → "quaternary ammonium" (correct: Glutamine); (2) *Degenerate reasoning loop*: repeats "The pKa of the amino acids . . . would determine the pH" verbatim 15+ times without computing any pKa values; (3) *Catastrophic numerical error*: predicts pI = 10.3 for an acidic peptide (2 Glu, 1 Lys ⇒ net negative charge, expected pI = 4.53), an error of 5.77 pH units.

Figure A.5. Ether-0 failure on isoelectric point prediction showing systematic confusion between amino acid single-letter codes and chemical element symbols. The model interprets F as fluorine (not Phenylalanine), K as potassium (not Lysine), and I as iodine (not Isoleucine), then enters a degenerate reasoning loop. The final prediction (pI = 10.3) is 5.77 pH units from the expected value (4.53), placing the peptide on the wrong side of neutrality.

ChemDFM-R Failure Case: Cyclic-to-Linear Sequence Hallucination

Input (Cyclic SMILES): `CSCC[C@@H]1NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)...`

Expected Output: `GKMVAPSFTAEEKAQK` (16 residues)

Model Reasoning (Truncated)

```
``To derive the linear amino acid sequence
from the cyclic peptide SMILES, one must first
identify the macrocyclic structure and locate
all peptide bonds (C(=O)N connections).  The
next step involves determining the direction
of these bonds, ensuring that the sequence
follows the N→C orientation.  Each amino
acid residue is then identified based on its
connectivity within the cyclic framework...''
```

Model Output: `ACACGPVRHQHLKIVAPPTATLFGKIP...VNYTWDTEDQRA` (39 residues)

Failure Modes: (1) *Sequence hallucination*: outputs 39 residues for a 16-residue peptide (sequence length match = 0.41), with an amino acid composition overlap of only 0.28; (2) *Procedurally correct but factually wrong reasoning*: the chain-of-thought describes a valid general strategy for cyclic-to-linear conversion but fails to actually execute it on the input SMILES; (3) *Overconfident prediction*: reports confidence = 1.0 despite producing a completely incorrect sequence.

Figure A.6. Representative ChemDFM-R failure on cyclic-to-linear conversion. The model generates a plausible reasoning trace describing the correct general procedure for SMILES-based cycle breaking, but outputs a hallucinated 39-residue sequence bearing little resemblance to the expected 16-residue peptide `GKMVAPSFTAEEKAQK`.

ChemDFM-R Failure Case: GRAVY Score — D-Amino Acid Blindness

Input (Sequence): `vHVSA` (lowercase `v` denotes D-Valine) **Expected Output:** GRAVY = 1.24

Model Reasoning (Truncated)

```
``To calculate the GRAVY score, we first need
to identify the hydropathy values for each
amino acid in the sequence.  The amino acids
in the sequence are V (Valine), H (Histidine),
V (Valine), S (Serine), and A (Alanine).
Using established hydropathy values, we assign
each amino acid a value.  For example, Valine
has a hydropathy value of 2.8, Histidine has
a value of −3.2, Serine has a value of −0.9,
and Alanine has a value of 1.8.  We then sum
these values:  2.8 + (−3.2) + 2.8 + (−0.9) +
1.8 = 2.3.  Finally, we divide this sum by the
sequence length (5 amino acids) to obtain the
average hydropathy score:  2.3 / 5 = −0.46.''
```

Model Output: GRAVY = -0.4875 (expected: 1.24, error = 1.73)

Failure Modes: (1) *D-amino acid blindness*: treats lowercase `v` (D-Valine) identically to uppercase `V` (L-Valine), ignoring the stereochemical convention where lowercase letters denote D-enantiomers; (2) *Arithmetic inconsistency*: states the sum as 2.3 but reports GRAVY = -0.4875 (2.3 / 5 = 0.46, not -0.46), indicating the model contradicts its own intermediate computation; (3) *Incorrect hydropathy values*: uses non-standard values that do not match the Kyte–Doolittle scale (Kyte & Doolittle, 1982).

Figure A.7. ChemDFM-R failure on GRAVY score prediction for a synthetic peptide containing D-Valine (lowercase `v`). The model ignores the D-amino acid convention, uses incorrect hydropathy values, and produces an internally inconsistent calculation.

ChemDFM-R Failure Case: Isoelectric Point — Noncanonical Residue Ignored

Input (Sequence): `QMTÎWFCKITI` (contains noncanonical residue Î) **Expected Output:** pI = 8.22

Model Reasoning (Truncated)

```
``To calculate the isoelectric point (pI) of
the peptide, we first identify the charged
amino acids in the sequence.  The amino
acids with ionizable side chains include
glutamic acid (E), aspartic acid (D), lysine
(K), arginine (R), histidine (H), cysteine
(C), and tyrosine (Y). We then determine
the net charge of the peptide at different
pH values...  The pI was calculated to be
5.48, indicating that the peptide is slightly
acidic at physiological pH. The confidence
score of 0.95 reflects the reliability of the
prediction.''
```

Model Output: pI = 5.48 (expected: 8.22, error = 2.74 pH units)

Failure Modes: (1) *Noncanonical residue ignored*: the model does not acknowledge or handle the noncanonical residue (Î) in the input sequence, defaulting to standard amino acid assumptions; (2) *Generic reasoning without sequence-specific analysis*: enumerates ionizable residues (E, D, K, R, H, C, Y) as a general list rather than identifying which are actually present in the input peptide; (3) *Confident but wrong*: reports confidence = 0.95 while predicting pI = 5.48, an error of 2.74 pH units from the expected basic pI of 8.22, placing the peptide on the wrong side of neutrality.

Figure A.8. ChemDFM-R failure on isoelectric point prediction for a synthetic peptide containing a noncanonical residue. The model provides a generic description of pI calculation without performing sequence-specific analysis, predicting an acidic pI (5.48) for a basic peptide (expected 8.22).

most severe in structural adaptation tasks, where the model hallucinates entirely fabricated sequences with incorrect lengths and compositions, while reporting confidence scores of 1.0.

D-Amino Acid and Noncanonical Residue Blindness. The model lacks awareness of the lowercase convention for D-enantiomers and cannot handle noncanonical residues, defaulting to standard L-amino acid assumptions. This limitation is particularly consequential for synthetic peptide datasets, where noncanonical and D-amino acids are prevalent by design.

Internally Inconsistent Arithmetic. Even when the model identifies a correct computational strategy, it frequently contradicts its own intermediate results—e.g., computing a positive sum but reporting a negative GRAVY score—suggesting that numerical reasoning is decoupled from the generated chain-of-thought.

These failures, combined with Ether-0's systematic breakdowns (Appendix A.6.2), underscore a broader finding: current domain-specific chemistry models, despite pre-training on chemical corpora, lack the cross-modal reasoning capabilities required for peptide-specific tasks that demand integration of sequence conventions, stereochemical notation, and numerical computation.

A.6.4. NatureLM Failure Analysis

NatureLM-8x7B-Inst is an instruction-tuned (non-reasoning) science foundation model designed for generation and optimization tasks across small molecules, proteins, DNA, and RNA (Xia et al., 2025). Unlike reasoning models (GPT-5, Gemini-3-Pro) or chemistry reasoning models (ChemDFM-R, Ether-0), NatureLM does not produce chain-of-thought traces. We evaluate it to test whether multi-domain scientific pre-training transfers to peptide-specific tasks at the instruction-following level.

NatureLM-8x7B-Inst achieved 0.0% accuracy across all seven tasks on both datasets. Unlike Ether-0's false safety refusals or ChemDFM-R's hallucinated-but-plausible outputs, NatureLM exhibits a distinct failure pattern: *task confusion and modality collapse at the instruction-following level*. We identify three systematic failure modes illustrated in Figures A.10, A.11, and A.12:

Task Confusion Across Property Prediction. When instructed to compute isoelectric point or instability index, NatureLM frequently returns GRAVY scores, logP values, or other unrelated cheminformatics quantities. This indicates the model has partial exposure to molecular property computation during pre-training but its instruction tuning does not enable it to discriminate between distinct

GPT-5 Code Interpreter Trace: SMILES→HELM — Non-Canonical Residue Detection (1/2)

Input (SMILES):

```
N[C@@H]([C@H](CC)C)C(=O)N[C@@H]
([C@H](O)C)C(=O)
N[C@@H](CCCNC(=N)N)C(=O)N[C@@H]
([C@H](O)C)C(=O)
N[C@@H](CC(=CN2)C1=C2C=CC=C1)C(=O)
N[C@@H](CC1=CNC=N1)C(=O)N[C@@H]
([C@H](O)C)C(=O)
C(=O)N[C@H](CC1=CNC=N1)C(=O)O
```

Expected Sequence: `ITRTWH[pClF]h` (contains *para*-chloro-Phe and D-His)

Code Interpreter Call 1/31 — Backbone Parsing

```
from rdkit import Chem
from rdkit.Chem import AllChem
smiles = "N[C@@H]([C@H](CC)C)C(=O)N[C@@H]..."
mol = Chem.MolFromSmiles(smiles)
mol
```

Code Interpreter Call 2/31 — Alpha-Carbon Detection

```
def is_carbonyl_c(atom):
if atom.GetAtomicNum()!=6:  return False
for b in atom.GetBonds():
if b.GetBondType()==Chem.BondType.DOUBLE:
other = b.GetOtherAtom(atom)
if other.GetAtomicNum()==8:  return True
return False
# Find alpha carbons via C(=O)-CA-N pattern...
```

(Continued in next figure)

GPT-5 Code Interpreter Trace: SMILES→HELM — Non-Canonical Residue Detection (2/2)

Final Output (after 31 RDKit calls)

```
{
"helm": "PEPTIDE1{I.T.R.T.W.H.[pClF].[dH]}$$$$",
"is_natural": false,
"monomers": ["I","T","R","T","W","H","[pClF]","[dH]"]
}
```

Analysis: (1) *Autonomous tool selection*: GPT-5 invokes RDKit without instruction, writing custom backbone-tracing code to identify 8 alpha-carbons from the SMILES graph; (2) *Stereochemistry-aware*: the model computes CIP labels at each alpha-carbon using `FindMolChiralCenters`. All centers are assigned S (L-configuration) except the final residue (R), correctly mapped to D-histidine (`[dH]`); (3) *Noncanonical residue identification*: recognizes the *para*-chlorophenyl side chain (`c1ccc(cc1)Cl`) as nonstandard and assigns HELM monomer code `[pClF]`; (4) *31 iterative calls*: the model's multi-step approach—parse molecule, locate backbone, identify side chains, assign stereochemistry, validate—mirrors a human computational chemist's workflow.

Figure A.9. GPT-5 with code interpreter on a synthetic peptide containing both a noncanonical amino acid (*para*-chlorophenylalanine) and a D-amino acid (D-histidine). The model autonomously writes and executes 31 RDKit calls to trace the peptide backbone, determine CIP stereochemistry at each alpha-carbon, and correctly assign nonstandard HELM monomer codes. This contrasts with NatureLM (Figure A.12), which generates Python pseudocode rather than executing it.

NatureLM Failure Case: Linear→Cyclic — Antibody Tag Hallucination

Input (Sequence): `KIQAYKNnGTKVVQAe` **Expected Output:** Valid cyclic peptide SMILES with head-to-tail amide bond

Model Output

```
<antibody> <a>A <a>R <a>S <a>G <a>G <a>I <a>E
<a>R <a>Y <a>Y <a>G <a>P <a>D <a>V <a>W <a>G
<a>L <a>D <a>T <a>Y </antibody>
```

Model Output: *Generates antibody-like XML tags with a hallucinated amino acid sequence*

Failure Modes: (1) *Wrong output modality*: instructed to produce cyclic SMILES, the model instead generates XML-style `<antibody>` tags with single-letter amino acid codes—a format from its protein pre-training that bears no relation to the requested SMILES output; (2) *Hallucinated sequence*: the output sequence (`ARSGGIERYYGPDVWGLDTY`, 20 residues) shares no residues in common with the input (`KIQAYKNnGTKVVQAe`, 17 residues), indicating complete disregard for the input peptide; (3) *No cyclization attempt*: the model makes no attempt at SMILES ring closure notation, head-to-tail bond formation, or any structural manipulation—it simply generates an unrelated protein fragment in a training-data format.

Figure A.10. NatureLM failure on linear-to-cyclic conversion. Instructed to convert a peptide sequence into cyclic SMILES, the model instead outputs antibody-like XML tags containing a completely unrelated amino acid sequence. This illustrates how NatureLM's protein-heavy pre-training (65.3B of 143B tokens) dominates its output distribution, causing it to default to protein-related formats regardless of the instruction.

NatureLM Failure Case: Isoelectric Point — Task Confusion

Input (Sequence): `QHYRKDHLTAqQSDDRNrCL`
Expected Output: pI = 8.21

Model Output (Truncated)

```
QHYRKDHLTAqQSDDRNrCL: gravy_score:  2.46,
reasoning:  ``the amino acid sequence you
provided has a gravy score of 2.46.  a gravy
score greater than 1...''
```

Model Output: *Returns GRAVY score (2.46) when asked for isoelectric point*

Failure Modes: (1) *Task confusion*: when instructed to compute isoelectric point, the model instead returns a GRAVY score, indicating it cannot distinguish between peptide property prediction tasks; (2) *Instruction ignorance*: the prompt explicitly requests pI calculation, but the model defaults to the only peptide-related computation it partially recognizes (hydropathy); (3) *Absent pI capability*: across all 20 therapeutic samples, NatureLM never attempts any pI-related computation or references pKa values, suggesting isoelectric point prediction is entirely absent from its instruction tuning.

Figure A.11. NatureLM failure on isoelectric point prediction. The model exhibits task confusion, returning GRAVY scores when instructed to compute pI. In other cases it outputs logP or logS values from unrelated cheminformatics tasks, suggesting its instruction tuning did not cover peptide-specific biochemical properties.

NatureLM Failure Case: HELM→SMILES — Modality Collapse

Input (HELM):
`PEPTIDE1{Q.M.T.W.F.C.K.I.T.I}$$$$`
Expected Output: Valid SMILES string for the corresponding peptide

Model Output (Truncated)

```
#Here is the Python program to convert
HELM notation to SMILES representation.
def getSmiLES(Helm, gravy_score,
reasoning, confidence):    monomers =
Helm.parse(gravy_score)   connections =
Helm.parse(gra...
```

Model Output: *Generates Python pseudocode instead of a SMILES string*

Failure Modes: (1) *Modality collapse*: instead of performing the conversion, the model generates pseudocode for a hypothetical conversion function, suggesting it has encountered code-related training data about HELM but cannot execute the conversion itself; (2) *Schema leakage*: the generated function signature includes `gravy_score` and `confidence` as parameters—fields from the structured output schema leak into the model's code generation as if they were function arguments; (3) *0% across all conversion tasks*: NatureLM produces no valid SMILES, HELM, or amino acid sequences across any of the four notation conversion and two structural adaptation tasks on either dataset.

Figure A.12. NatureLM failure on HELM→SMILES conversion. The model generates Python pseudocode rather than performing the actual conversion, with structured output schema fields leaking into the generated code. This pattern—generating *about* the task rather than *performing* it—is consistent across all notation conversion and structural adaptation tasks.

biochemical properties—it defaults to whichever property-related pattern it has seen most frequently.

Modality Collapse on Conversion Tasks. For all notation conversion and structural adaptation tasks, NatureLM generates Python pseudocode, cheminformatics library references ("computed using chemfp"), or XML-like tags (`<antibody>`, `<mol>`) rather than performing the requested conversion. The model has learned *about* these representations from code and documentation in its training corpus but cannot *operate* on them when instructed to do so.

Malformed Structured Output. Even when NatureLM attempts to produce JSON, it generates double-escaped quotes and inconsistent formatting that standard parsers reject. Fields from the output schema leak into unrelated parts of the response (e.g., `gravy_score` appearing as a function parameter in generated code). This suggests the model treats the output schema as additional context to incorporate rather than as a formatting constraint to follow.

These failures, distinct from those of Ether-0 (Appendix A.6.2) and ChemDFM-R (Appendix A.6.3), confirm that NatureLM's multi-domain scientific pre-training across DNA, RNA, proteins, and small molecules does not transfer to peptide-specific tasks—even at the basic instruction-following level, without requiring any chain-of-thought reasoning.

A.7. Confidence Extraction Caveats

Confidence values were extracted from model responses with varying reliability:

- **Ether-0 emits no confidence field.** Ether-0's structured output schema contains only `thinking` and `answer` fields; it was not prompted for confidence scores. On therapeutic tasks, Ether-0 frequently returns `"Unsafe question, I can't answer."` as the answer (14/50 on GRAVY synthetic, 13/50 on isoelectric point synthetic, 13/50 on instability synthetic)—a false safety refusal mode that effectively yields zero output on a non-trivial fraction of peptide property queries.

- **NatureLM confidence is parsable on a minority of samples.** NatureLM-8×7B-Inst returns malformed JSON with double-escaped quotes. Across 20 samples per task, we extract confidence from only 1–16 responses, with mean values uncorrelated to sample difficulty. On therapeutic instability, the single parsable response reports confidence 0.00; on synthetic cyclic-to-linear, the two parsable responses average 0.03. We treat these numbers as artifactual rather than calibrated.

- **Gemini-3-Pro isoelectric point has low parsing yield.** Only 2/1975 therapeutic and 2/200 synthetic Gemini responses contained extractable confidence values on the pI task. The reported means (0.90 and 0.93) are computed over these small subsets and should not be interpreted as population-level calibration.

- **GPT-5 has two reporting regimes.** We report two GPT-5 configurations: (i) the OpenAI Responses API with code interpreter enabled, where the model executes RDKit/BioPython calls before finalizing its answer, and (ii) the OpenAI Evals API without tools, where the model reasons from text alone. Confidence behavior differs substantially between the two (Table 9), so we report them as separate rows rather than aggregating.

A.8. Prompt Templates

Figure A.13 shows the system prompts for individual subtasks. All prompts instruct models to return structured JSON with reasoning and confidence scores.

A.9. Prompt refinement

Figure A.14 shows the original as well as refined prompt based on domain expert feedback which addresses the ring index collision in the Linear→Cyclic task.

Table 9. Mean self-reported confidence across models and tasks. Each cell reports the mean of extractable `confidence` fields from model responses (0–1 scale). Dashes (—) indicate tasks not run for that model; N/A indicates no confidence field in the model's output schema. Abbreviations: PP = Property Prediction, NC = Notation Conversion, CR = Cyclization Reasoning.

	PP			NC		SA	
MODEL	GRAVY	PI	INSTAB.	S→H	H→S	L→C	C→L
Therapeutic Dataset							
GPT-5 (CODE INTERP.)	0.97	0.84	0.93	0.92	0.90	0.93	0.94
GPT-5 (NO TOOLS)	0.96	0.79	0.26	0.88	0.67	0.92	0.86
O4-MINI	0.92	0.86	0.59	0.86	0.47	0.21	0.87
QWEN 3.5	0.95	0.84	0.98	0.98	0.98	0.92	0.93
GEMINI-3-PRO	1.00	0.90	0.98	0.99	0.99	0.95	1.00
ETHER-0	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CHEMDFM-R	0.98	0.95	0.93	1.00	1.00	0.98	1.00
CHEMDFM-V2.0	1.00	0.91	0.92	1.00	1.00	0.97	0.97
NATURELM-8×7B	0.52	0.42	0.00	0.62	0.59	0.20	0.67
Synthetic Dataset							
GPT-5 (CODE INTERP.)	0.95	0.77	0.82	0.85	0.89	0.31	0.28
GPT-5 (NO TOOLS)	0.95	0.74	0.24	0.22	0.21	0.32	0.23
O4-MINI	0.90	0.77	0.56	0.75	0.69	0.61	0.62
QWEN 3.5	0.99	0.95	0.97	0.98	0.92	0.91	1.0
GEMINI-3-PRO	0.88	0.93	0.93	0.94	0.99	0.92	**0.12**
ETHER-0	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CHEMDFM-R	0.99	0.94	0.92	1.00	1.00	0.84	1.00
CHEMDFM-V2.0	1.00	0.90	0.93	0.82	0.72	0.79	0.37
NATURELM-8×7B	0.37	0.31	0.15	0.63	0.68	0.36	0.03

(a) Gravy Score Prediction

```
You are a helpful assistant that calculates GRAVY
(Grand Average of Hydropathy) scores for peptides.
Provide the GRAVY score along with reasoning
and confidence.  The GRAVY score is the sum of
hydropathy values of all amino acids divided by the
sequence length.
```

(b) Instability Index Prediction

```
You are a helpful assistant that calculates the
Instability Index for peptides.  The instability
index uses dipeptide instability weights.  An
index < 40 indicates stability, ≥ 40 indicates
instability.
```

(c) SMILES → HELM Conversion

```
Convert the following peptide SMILES to HELM
notation.  HELM uses single-letter codes for
L-amino acids (A, R, N, D, C, E, Q, G, H, I,
L, K, M, F, P, S, T, W, Y, V), bracketed codes
for D-amino acids ([dA], [dR], etc.), and caps
([ac] for N-acetyl, [am] for C-amide).  Format:
PEPTIDE1{sequence}$$$$V2.0
```

(d) HELM → SMILES Conversion

```
Convert the following HELM notation to SMILES
representation.  Generate a valid SMILES string
that preserves all structural information.  HELM
Monomer Reference:  L-amino acids (A, R, N, ...),
D-amino acids ([dA], [dR], ...), Caps ([ac], [am]),
Special ([Nle], [Nva], [Orn], [seC]). Connections:
PEPTIDE1,PEPTIDE1,pos1:R3-pos2:R3 for disulfide
bonds.
```

(e) Linear → Cyclic SMILES Conversion

```
You are an expert peptide chemist specializing in
cyclic peptide synthesis.  Convert the given linear
peptide SMILES to its head-to-tail cyclized form.
Create an amide bond between the N-terminal amine
and C-terminal carboxylic acid.  Ensure proper
stereochemistry is preserved and the resulting
SMILES is valid.
```

(f) Cyclic → Linear Sequence Conversion

```
You are an expert peptide chemist specializing
in analyzing cyclic peptide structures.  Convert
a cyclic peptide SMILES to its linear amino acid
sequence.  Key points:  (1) Cyclic peptides are
formed by head-to-tail cyclization; (2) Break the
cycle at any peptide bond---all rotations represent
the same molecule; (3) Follow N→C direction of
peptide bonds; (4) Use standard single-letter codes;
(5) Pay attention to stereochemistry to distinguish
L and D amino acids.
```

Figure A.13. Boxes (a)-(f) showing prompt for individual tasks. All prompts are followed by a user message containing the input (sequence or SMILES) and instructions to return JSON with fields: `result`, `reasoning`, and `confidence` (0–1 scale).

Prompt Refinement: Linear→Cyclic — Ring Index Collision Fix

Input (Sequence): `VEVKIKELNYLKTIQDKL` (18 residues, contains Tyr)

Expected: Valid cyclic SMILES with macrocycle (index `1`) and Tyr phenol ring (index `2`)

✗ Original Prompt

Prompt excerpt: *"...Convert the given linear peptide sequence into its cyclic form. Provide the cyclic peptide as a SMILES representation..."*

Model reasoning: "...Formed the macrocycle by starting the SMILES at the backbone amide nitrogen of residue 1 as `N1...` and closing the ring from the final residue's carbonyl carbon with `C(=O)1`..."

Output (Tyr fragment):

`...N[C@@H](Cc1ccc(O)cc1)C(=O)...`

Problem: Ring index `1` used for *both* the macrocycle closure *and* the Tyr aromatic ring ⇒ SMILES parser conflict.

✓ Refined Prompt (single instruction added)

Added instruction: *"Make sure to increment the ring indices as you see relevant rings in the side chain 'R group' of the residues."*

Model reasoning: "...Ring index convention: used `1` for the macrocycle closure (`N1 ... C(=O)1`) and `2` for the tyrosine aromatic ring (`c2...c2`), incrementing indices for distinct rings..."

Output (Tyr fragment):

`...N[C@@H](Cc2ccc(O)cc2)C(=O)...`

Fix: Macrocycle uses index `1`, Tyr aromatic ring uses index `2` ⇒ valid SMILES, correct molecular structure.

Takeaway: A single domain-expert-informed instruction resolved a systematic SMILES validity failure for all peptides containing aromatic side chains (Tyr, Trp, His, Phe). The model already possessed the knowledge to manage ring indices—it only needed explicit cueing. This suggests that targeted prompt refinement, guided by reasoning trace analysis, is an efficient strategy for improving performance on structurally complex chemistry tasks.

Figure A.14. Domain-expert prompt refinement for Linear→Cyclic SMILES conversion. The original prompt (top, red) causes ring index collisions when peptides contain residues with aromatic side chains. Adding a single instruction about ring index management (bottom, green) resolves the conflict, producing valid cyclic SMILES. This demonstrates that reasoning trace analysis by domain experts can identify minimal prompt modifications with substantial impact.