LEARNING THE GRAPH LANGUAGE OF MOLECULES: GRAMMAR TOKENIZATION FOR CHEMICAL LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have been explored for integrating molecular graphs, but joint autoregressive modeling of text and molecules remains challenging. A key barrier is the lack of principled molecular graph tokenizers (MGTs): string representations like SMILES rely on brittle syntax and canonicalization, limiting robustness, flexibility, and interpretability. We introduce ALPHAGRAMMAR, a neural-symbolic MGT that rewrites molecular graphs as linear derivations over a formal graph grammar with theoretical guarantees. Training optimizes the parameters of a potential function for grammar construction, and inference amortizes the cost of parsing during tokenization. We then introduce GRAMOLE, a multimodal LLM built on ALPHAGRAMMAR, which interleaves text and molecules as rule sequences without the need for modality-specific adapters. GRAMOLE offers 100% validity via type-constrained decoding, expressivity by sampling high-diversity molecular sets, transferability by improving FDA drugs' predicted properties on all six endpoints under an independent property predictor (vs. four for the strongest baseline), with edits that stay $1.9\times$ closer to the parent drug, controllability using natural language interfaces, and synthesizability by achieving $1.3\times$ drug and $3.1\times$ material retrosynthesis success rate of the best multimodal LLM baseline at matched Qwen3-8B backbone. ALPHAGRAMMAR autonomously acquires rules that recapitulate known functional groups, offering a layer of interpretability absent in prior MGTs. We benchmark GRAMOLE across 8 diverse tasks in the design-make-test-analysis (DMTA) cycle and probe ALPHAGRAMMAR's mechanism in 4 quantitative and qualitative case studies.

1 INTRODUCTION

LLMs trained on chemical texts capture chemical knowledge and support molecular comprehension, but they do not explicitly model graph topology and may underperform on tasks where chemical properties depend on graph structure. Chemical Language Models (CLMs) rely on MGTs to convert molecules into sequential tokens, enabling joint reasoning over text and molecular structure. In cheminformatics, string representations like SMILES or SELFIES linearize a molecule via depth-first search (DFS) traversal of the molecular graph, with ad-hoc measures to mark cycles. Despite their popularity, studies show strings are brittle for representing molecules, as they are (i) syntactically brittle, (ii) chemically unintelligible, and (iii) disrespectful of permutation invariance because there can be up to exponential DFS traversals on the same graph topology (Fig. 1). The central difficulty is tokenizing graphs losslessly, while mitigating the combinatorial explosion in equivalent representations. An ideal MGT should (i) compose a modular vocabulary that an LLM can easily enforce chemical validity, (ii) build an interpretable vocabulary of chemically meaningful rules and (iii) encode and decode arbitrary molecules losslessly into a canonical sequence of tokens, without heuristics that violate graph invariances (e.g. SMILES canonicalization).

Recently, fragment-based linear representations have been introduced to capture more domain knowledge than string representations Geng et al. (2023); Wu et al. (2024); Noutahi et al. (2024); Sun et al. (2024a); Edwards et al. (2025). For instance, RandomWalk Sun et al. (2024a) fragments a molecule into interpretable building blocks, and generates molecules by taking random walks on the hierarchical space of building blocks. While intuitive, these methods share a fundamental problem common to all linear representations built from tree traversal: alternate traversals multiply the representation, making autoregressive modeling less principled and more data-intensive – enumerating

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

alternate strings for the same molecule is a well-known means of data augmentation Bjerrum (2017); Priyadarsini et al. (2025).

We develop ALPHAGRAMMAR, a neural-symbolic tokenizer that learns to losslessly scaffold molecules into "sentences" of rules by learning a reusable and composable vocabulary. The vocabulary subsumes a finite set of atomic rules, which covers all chemically reasonable molecules. We couple the learned symbolic grammar with a parameterized potential function, which meets three desiderata of an MGT: (i) learn desirable tokens from feedback, (ii) optimize for the most stable tokenization when alternate tokenizations theoretically exist and (iii) generalize to tokenizing new molecules at inference time. To meet these requirements, ALPHAGRAMMAR trains the potential function to sample desirable grammar rules and parses using group relative policy optimization (GRPO) Shao et al. (2024) with per-step attributions.

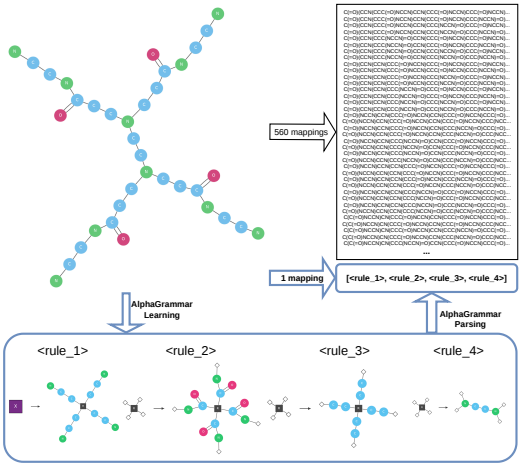


Figure 1: Illustrative: the following parsimonious grammar admits a single parse of this dendrimer, whereas SMILES admits 560 strings.

We develop GRAMOLE, the first multimodal LLM over molecules and text that does autoregressive modeling over a unified vocabulary of text and grammar rule tokens. Our contributions are (1) a novel neural-symbolic tokenizer of molecules, (2) a tokenizer learning paradigm with theoretical guarantees, (3) a new multimodal LLM framework that is free of modality-specific alignment models or adapters.

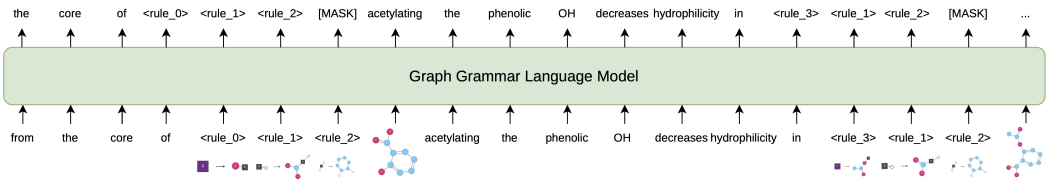


Figure 2: GRAMOLE integrates grammar rule tokens into the vocabulary of an LLM for interleaved generation of text and molecules.

2 RELATED WORKS

Chemical Language Models. CLMs typically represent molecules as sequential tokens and train them autoregressively. Early approaches therefore adopt string encodings such as SMILES or SELFIES, which make molecular generation compatible with standard language modeling: MolGPT trains a transformer decoder directly on SMILES Bagal et al. (2021). SmileyLlama extends this paradigm to general LLMs through prompting, supervised fine-tuning, and DPO Cavanagh et al. (2024). KnowMol generates SELFIES from chemical annotation Yang et al. (2025). ether0 pushes the same interface toward explicit natural-language reasoning before emitting molecular strings Narayanan et al. (2025). Later works seek to bridge graph encoders to LLMs, motivated by the view that graph representations can model structure-property relationships more faithfully than linearized strings. These methods align graph embeddings to text embeddings through GNNs, adapters, or projector layers Kim et al. (2025); Yan et al. (2025); Jing et al. (2026); Ha et al. (2025), but they generally target molecular understanding rather than unified generation. Llamole combines graph-based encoding with a dedicated graph generative decoder, but relies on a relatively complex orchestration of multiple graph modules Liu et al. (2024). Toward unified autoregressive modeling, UniMoT expands the LLM vocabulary with discrete molecular latent tokens, enabling interleaved molecule-text generation in a single token stream Guo et al. (2024); however, these tokens are quantized latent codes rather than explicit functional units, and the VQ pipeline adds training and inference complexity Guo et al. (2024). mCLM addresses this partly by replacing latent

codes with a modular vocabulary of functionally interpretable, synthesis-oriented building blocks Edwards et al. (2025), but its language remains only partially complete in practice, with limited parse coverage, a long-tailed block distribution, and heuristic fallback procedures for reconstruction Edwards et al. (2025). In contrast, our approach treats grammar rules as first-class tokens alongside text, yielding a unified autoregressive language whose tokens are losslessly reconstructive, complete over reasonable molecules, deterministically parseable, and functionally meaningful, while also supporting optimization at both the token and vocabulary levels for interpretability, parsimony, and usability.

Learning Molecular Grammars. Molecular grammars are appealing because they codify chemical patterns and make it easy to enforce constraints on the outputs to ensure generation validity. Early methods such as GrammarVAE Kusner et al. (2017) and SD-VAE Dai et al. (2018) use predefined string grammars to improve syntactic validity. SELFIES Krenn et al. (2020) and STONED Nigam et al. (2021) replace SMILES with a robust hand-designed molecular string language. Grammar learning methods begin with MHG Kajino (2019), which induces rules automatically from data and ensures generation is always valid. DEG and FMG use the same formalism but introduce reinforcement learning and LLMs to optimize grammars towards desirable outputs Guo et al. (2022); Sun et al. (2025b). Sun et al. (2024a) bring experts in-the-loop to induce interpretable grammars. Overall, the field has progressed from enforcing chemical validity with fixed formalisms Kusner et al. (2017); Krenn et al. (2020); Nigam et al. (2021) to flexible learning and exploration of semantic objectives Sun et al. (2024b).

3 AlphaGrammar: Neural Grammar Parsing Tokenizer

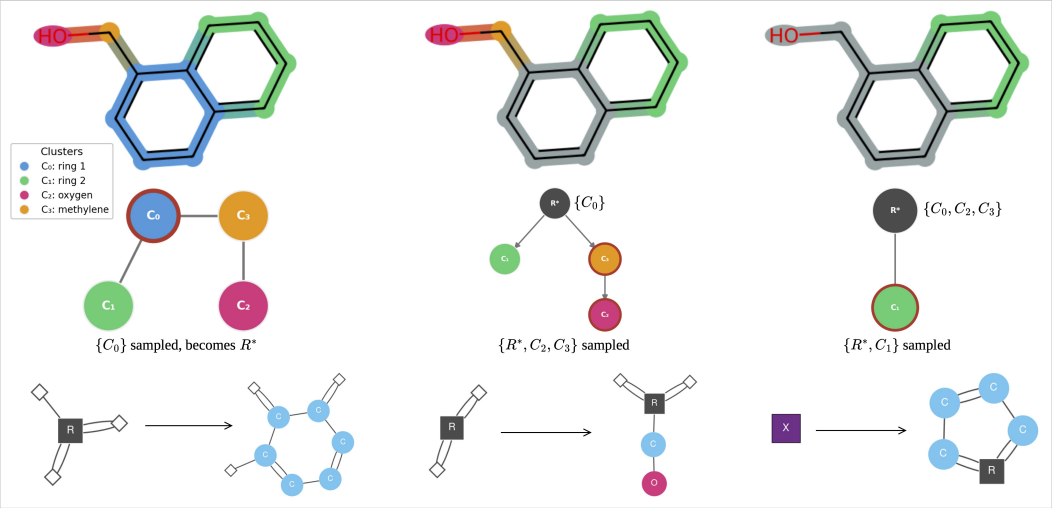


Figure 3: AlphaGrammar decomposes a molecule into base clusters (atoms and rings) which are sampled without replacement per step to form grammar rules. Each cluster node C_i is a Bernoulli RV ($C_i = 1$ means selected for contraction) of the junction tree graphical model. The joint factorizes over the tree (e.g., step 2: $P(C_0)\,P(C_1|C_0)\,P(C_3|C_0) \times P(C_2|C_0, C_3)$) which enables linear-time sampling and inference. The first step roots the tree by categorical selection. Each step thereafter contracts a subtree rooted at R^*.

Grammar Preliminaries. A hyperedge replacement grammar (HRG) constructs a molecular graph by repeatedly replacing a typed nonterminal with a molecular fragment. Each rule specifies the fragment to add, how it attaches to the existing graph, and optionally contains one nonterminal from which generation continues. We restrict every rule to at most one such nonterminal. Consequently, a molecular derivation is a sequence rather than a branching parse tree, allowing each rule to function as an ordinary autoregressive token. App. A gives the formal definitions.

Linear HRGs Sequentially Tokenize Molecules. For a principled MGT that supports autoregressive modeling, we are interested in linear HRGs. Since there is at most one non-terminal label (R^* in Fig. 3), the parse tree is a path, or a sequence of rules. This idea takes inspiration from Sun et al. (2025a),

but our approach works for the general class of undirected, attributed graphs. ALPHAGRAMMAR addresses the two main requirements: (1) given a dataset of molecules $\mathcal{D}$, learn $\mathcal{G}$ such that $\mathcal{D}$ is part of the language $L(\mathcal{G})$, (2) given a grammar $\mathcal{G}$, parse *any* molecule with $\mathcal{G}$.

Grammar Construction as Junction Tree Contraction. We obtain rules for our HRG by collapsing subtrees of a junction tree T_M for each molecule M. In the first step, we choose one node (e.g. C_0 in Fig. 3 (c)) as the root. Every step after, we may collapse any subtree rooted at R^* (e.g. Fig. 3 (e), the nodes C_3 and C_2 are contracted 3). The sampled clusters are absorbed into the root cluster (grey highlighted region). The induced subgraph from the contracted region is the hypergraph fragment used to assemble a rule, and the rule's non-terminal typed by the list of external bond types.

Junction Tree Determines Contraction Dependencies. We formulate a graphical model, where each η has a Bernoulli random variable X_η, which $X_\eta = 1$ if it is sampled, and 0 if not. For the HRG to be linear, every sampled region must include the non-terminal R^*, to avoid a second non-terminal R^* being introduced. This constraint can be expressed in our graphical model as parent-child dependencies, rooted at the only cluster containing R^*: $X_\eta = 1 \to X_{\text{parent}(\eta)} = 1$. Most importantly, we can perform exact sampling and inference over T_M in linear time Koller & Friedman (2009).

Parameterized Potential Function. We define a cluster-weight function, $\mathcal{F}_\theta : T_M \to \mathbb{R}$, mapping each node of T_M to a weight (Fig. 4 Left). This is similar to the potential function in graphical models Barber (2012). For each cluster η, the potential defines a node weight $\phi(\eta;\theta) = \sigma(-\mathcal{F}_\theta(f(\eta))) \in (0,1)$, where $f(\eta)$ aggregates a pretrained GNN's per-atom features into T_M's clusters. The GNN is called *once* per molecule and its atom features are cached. $\mathcal{F}_\theta$ is a single attention pass masked by the current junction-tree ancestry (Fig. 4 Left). ALPHAGRAMMAR can also run parameter-free via "atomic" contractions (Theorem 1) which ablates the GNN and θ altogether. We then sample from the factorized distribution (Fig. 3 (c)), and compute log-probabilities for reinforcement learning. We detail the implementation and complexity of $\mathcal{F}_\theta$ in App. H.

Parameterized Grammar Construction. The contraction process completely determines the grammar rules (and the grammar). Thus, θ parameterizes grammar construction while satisfying requirement (1) by construction. Training θ optimizes the grammar $\mathcal{G}(\mathcal{D};\theta)$ according to metrics.

Exact Parse Enumeration. Exact parsing for general HRGs is intractable, but Lautemann (1990) and Chiang et al. (2013) show for bounded degree hypergraphs, parsing is polynomial-time. We extend an implementation of the algorithm, to molecular hypergraphs. In App. C, we state and prove the complexity of our parser. Similar to the CYK parser Sakai (1961), we can use it as an oracle to enumerate all derivations (or prove none exist) to train $\mathcal{F}_\theta$ to construct desirable grammars.

Potential Function as a Neural Parser. Once we have trained θ, we can use the parameterized contraction process to sample a valid parse, hence amortizing the cost of parsing. However, a sound tokenizer *must guarantee each sampled rule is in* $\mathcal{G}$; sampling a contraction that emits a rule not in $\mathcal{G}$ leads to the Out of Vocabulary (OOV) scenario. We prove in Theorem 1 that for *reasonable* molecules, there is a finite, enumerable set of rules, which any smallest possible contraction (sampling a single cluster) would yield. Because this atomic set is finite and enumerable, it defines a closed rule vocabulary; any reasonable molecule can be tokenized down to atomic contractions, so we never encounter OOV scenarios.

Optimization Objectives. We optimize θ along five desired property axes of grammars, including two rule-level metrics (b, c), one grammar-level metric (a), and two as periodic bonuses (d, e): (a) *parsimony* – $\mathcal{G}$ parses efficiently; (b) *compositionality* – rules should be reused often within the batch; (optional c) *quality* – rules should align with known functional groups and reaction patterns; (d) *coverage* – rules from parsing independent samples of data are present in the vocabulary; (e) *stability* – parses across rollouts should agree. These metrics are scored by active oracles $\mathcal{M}_1, \ldots, \mathcal{M}_L$ with weights $\lambda_1, \ldots, \lambda_L$. We use (a, b, d, e) as the default ALPHAGRAMMAR training setting for every non-MolQA benchmark, and (a, b, c, d, e) as the default for MolQA (RQ6, Table 19, Fig. 7). For MolQA, we instantiate (c) using the CRM quality oracle. Exact details, definitions, and hyperparameter settings are in App. D.

Reinforcement Learning. At contraction step t, each currently available cluster $\eta_j^{(t)}$ is assigned a Bernoulli random variable $X_j^{(t)} \in \{0,1\}$, where $X_j^{(t)} = 1$ means $\eta_j^{(t)}$ is selected into the rooted subtree contracted at step t. The selected rooted subtree is contracted into R^*, producing one HRG rule; repeating this yields a trajectory $\mathbf{X} = (X^{(1)}, \ldots, X^{(T)})$, with $X^{(t)} := (X_1^{(t)}, X_2^{(t)}, \ldots)$. Each cluster carries a potential $\phi_j^{(t)}(\theta) := \sigma(-\mathcal{F}_\theta(f(\eta_j^{(t)}))) \in (0,1)$ which induces a constrained policy $p_\theta(X^{(t)} \mid X^{(<t)}, M)$ over valid rooted subtrees at step t; its exact normalized form, linear-time

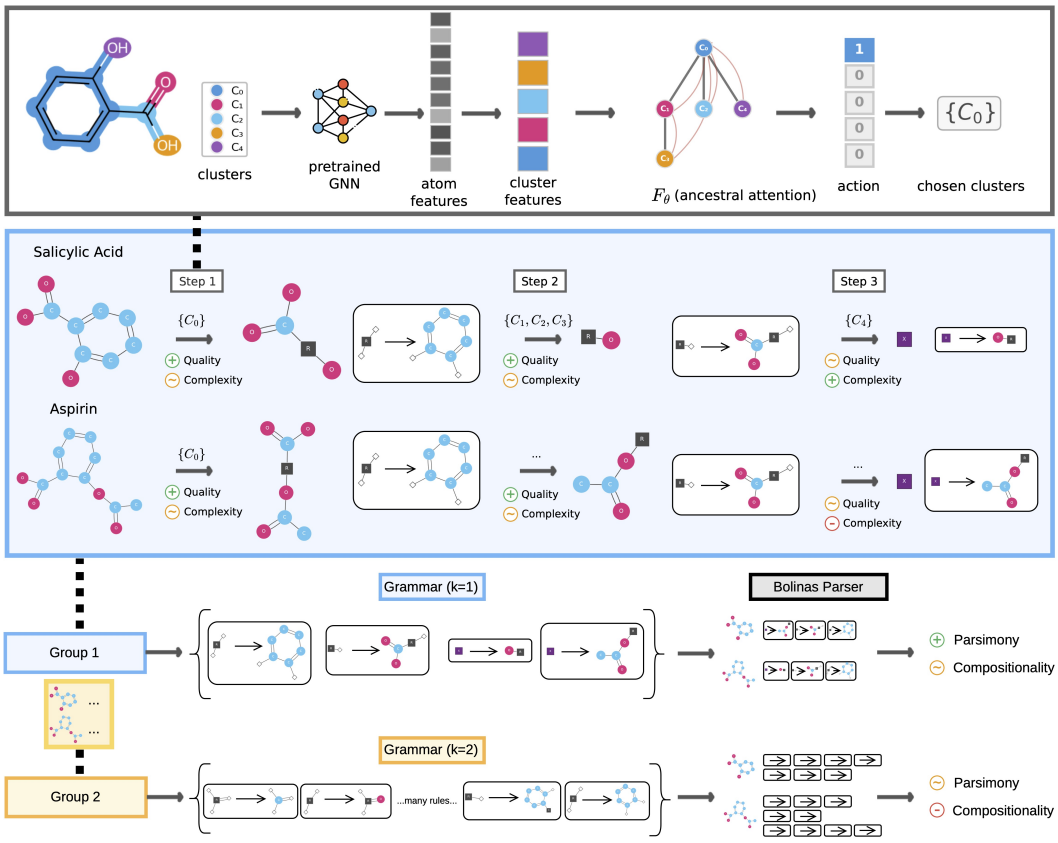


Figure 4: (Top) We parameterize $\mathcal{F}_\theta$ as an attention layer masked by graphical model dependencies. (Middle) Each rollout is a sequence of contraction decisions, or equivalently, a sequence of rules, that emits per-step rewards. (Bottom) Each rollout forms a "batch" grammar, which is evaluated using our oracle. Group normalized advantages are computed across rollout groups for GRPO.

normalizer, and differentiable log-probability are derived in App. D. The trajectory log-probability is therefore $\log p_\theta(\mathbf{X} \mid M) = \sum_{t=1}^{T} \log p_\theta(X^{(t)} \mid X^{(<t)}, M)$, where the support of p_θ is restricted to decisions satisfying the contraction dependencies. For each molecule, we sample K contraction trajectories, each inducing a grammar. We score the resulting grammars for parsimony, rule reuse, coverage, stability, and optionally chemical quality. GRPO normalizes these scores within each rollout group (Fig. 4 Right), while step-level attribution assigns credit to the contractions responsible for individual rules (Fig. 4 Middle). We optimize the log-probability of rewarded contraction decisions; Appendix C gives the complete objective and reward definitions.

4 GRAMOLE: LANGUAGE MODELING WITH GRAMMAR VOCABULARY

Theorem 1 (Universal Coverage of Reasonable Molecules using Atomic Rules)**.** There exists a finite set of rules, whose language includes all reasonable molecules, of cardinality no greater than

$$576 \cdot \binom{V(C+1)+3}{3} + \sum_{r=3}^{R} \frac{10^r + (2r-1) \cdot 10^{\lceil r/2 \rceil + 1}}{2r} \cdot \binom{V(C+1)+2}{2} \cdot [G(r, V(C+1)) + 20 \cdot r \cdot G(r-1, V(C+1))]$$

where G is the per-atom valence generating function. We call this enumerated set of rules *atomic rules* for reasonable constants R, V, C (Def. A.7). Full proof and enumeration procedure in App. N.

Corollary 1 (Vocabulary Closure on Reasonable Molecules)**.** If a grammar $\mathcal{G}$ includes the atomic rules, then every reasonable molecule admits a parse in $\mathcal{G}$. A model whose vocabulary tokens are rules of such a $\mathcal{G}$ never encounters out-of-vocabulary tokens on reasonable molecules.

Autoregressive Modeling Over Unified Vocabulary. A closed and complete vocabulary enables a new paradigm for graph language modeling free of graph adapters. Graph adapters may see

out-of-distribution graph tokens unseen during training. GRAMOLE removes this by construction. Its rule vocabulary $\mathcal{V}_r$ is the enumerated set of atomic rules (Theorem 1), comprising of $50{,}082{,}289$ rules for our reasonableness constants (App. N), together with the composed rules the grammar learns. Every rule is a token whose embedding is learned as the rule is used during training, so the materialized table grows with the rules encountered rather than the full dictionary. Because the atomic set is complete over reasonable molecules, by Corollary 1 no molecule is ever out-of-vocabulary. Augmenting the LLM's word vocabulary $\mathcal{W}$ with $\mathcal{V}_r$ and a small set of special tokens $\mathcal{V}_s$ yields a single unified vocabulary $\mathcal{V} = \mathcal{W} \cup \mathcal{V}_r \cup \mathcal{V}_s$. Under this vocabulary, parsing molecule M with $\mathcal{F}_\theta$ produces a rule sequence $H_M \mapsto [\rho_1, \rho_2, \ldots, \rho_K]$, and each rule is already a token in $\mathcal{V}$. Generating M reduces to generating a sequence of rule tokens; a single next-token distribution $p_\theta(y_i \mid Y_{<i})$ over $\mathcal{V}$ suffices.

GRAMOLE Supports Heterogeneous Tasks Across the DMTA Cycle. GRAMOLE makes interleaved text-and-molecule processing and generation a first-class capability rather than a coordination problem across specialized encoders and decoders. This enables a single base model to interleave generation of text and molecules in one flat sequence $Y = (y_1, \ldots, y_N)$ over $\mathcal{V}$, with special tokens demarcating different phases. Outside of rule regions, the model generates ordinary text, supporting planning and analysis. We train a base LLM via supervised finetuning (SFT) across different benchmarking setups, to demonstrate the holistic strengths of the GRAMOLE framework.

GRAMOLE Guarantees Molecule Validity. Each rule ρ_k carries type signature $\mathrm{intype}(\rho_k)$ and $\mathrm{outtype}(\rho_k)$. A derivation begins with an *initial* rule, $\mathrm{intype} = []$; every subsequent rule must *type-chain*, $\mathrm{intype}(\rho_{k+1}) = \mathrm{outtype}(\rho_k)$, and the derivation closes the first time a *terminal* rule ($\mathrm{outtype} = \perp$) is emitted. Inside a rule region, we mask every rule token whose in-type disagrees with the most recent out-type. Outside a rule region, rule tokens are masked, and the model generates natural language normally. No search, backtracking, or architectural change is needed: each decoding step remains $\mathcal{O}(1)$, and the LLM produces grammatically-valid molecules by construction. We enforce decoding and length constraints only at inference time; for more details, see App. F.

5 EVALUATION

Research Questions. Our experiments answer the following questions:

- **Tokenizer Properties:** (RQ1) Does our grammar tokenizer meet the requirements and desiderata of an ideal MGT better than string and fragment-based linear representations?
- **Unconditional Generation:** (RQ2) Can our learned grammar generate desirable molecules from random priors? (RQ3) Does blank prompting after SFT generate coherent molecules?
- **Molecular Optimization:** (RQ4) Can GRAMOLE optimize existing drug candidates for specific properties by critical reasoning of molecular structure in both the input and output?
- **Prompt Following and Inverse Design:** (RQ5) Can GRAMOLE follow property specifications via prompting after instruction tuning? Can GRAMOLE design molecules to satisfy natural language specifications with chain-of-thought traces?
- **Synthesis Planning:** (RQ6) Can GRAMOLE enable synthesis planning in the design loop? Are generated molecules synthesis-friendly?
- **Interpretability:** (RQ7) Do our grammar rules capture interpretable fragments? Do our learned parses align with expert intuition?

Benchmarks. We use MOSES Polykovskiy et al. (2020) to benchmark random generation from our grammar vocabulary, GuacaMol Brown et al. (2019) to benchmark GRAMOLE's unconditional and in-context generation after SFT on an instruction-following dataset curated from ChEMBL by Cavanagh et al. (2024) (we call it SmileyChEMBL), MolQA to benchmark GRAMOLE's inverse design with retrosynthesis planning abilities after SFT on an instruction-tuning dataset curated from USPTO by Liu et al. (2024), and mCLM-Pretrain-10K to benchmark GRAMOLE on molecular optimization tasks where molecules appear in both the question and answers after SFT on the dataset curated by Edwards et al. (2025). We list all dataset sources in App. L. We train three separate GRAMOLE models, one per benchmark (SmileyChEMBL, MolQA, and mCLM-Pretrain-10K), each matching the respective data splits, model architecture, and evaluation protocols.

Baselines. We selected the following papers for our primary comparisons, on the basis they feature distinct MGT solutions: SmileyLlama Cavanagh et al. (2024) for SMILES tokens, mCLM Edwards et al. (2025) for fragment-based block tokens with graph adapters, and Llamole Liu et al. (2024) for heterogeneous graph models. For training GRAMOLE, we match the respective authors' choice of base

LLM (Qwen3 Llamole, Qwen2.5 mCLM, Llama-3.1 SmileyLlama) to obtain separate checkpoints per benchmark for a fair comparison. In addition, we compare with domain-specific models, CLMs, and ICL/SFT on other leading base LLMs.

Metrics. (RQ1) We adopt or implement parse enumerators for each tokenizer (needed to compute Uniqueness and Usage): Bolinas for ALPHAGRAMMAR (polynomial-time, large constant), and adapted worst-case-exponential algorithms for non-ALPHAGRAMMAR tokenizers (Krenn et al. (2020); Weininger (1988)), for which parse counts are estimated by sampling (App. J.2). Parsimony is $1/$(# of theoretically distinct parses), compositionality the avg. rule usage rate, rule quality the avg. similarity of a rule to a set of compiled functional groups. Each is evaluated $@|V| = 10, 100, 1000$. Details are in App. J. (RQ2-6) We respectively adopt metrics from MOSES, GuacaMol, Edwards et al. (2025)'s FDA drug, and MolQA benchmarks. (RQ7) We use Monte Carlo p-value to test for enrichment of established functional groups. Details are in App. K.

Compute Complexity and Costs. We analyze the theoretical complexity of ALPHAGRAMMAR and GRAMOLE training and inference in App. H. We report GRAMOLE training costs in App. G.

6 RESULTS

(RQ1) ALPHAGRAMMAR exhibits favorable tokenizer properties. We see in Table 14 string MGTs, especially char-level, show the best usage rate, due to their small and fixed alphabet. Fragment MGTs, meanwhile, show high parsimony. Using larger fragments (e.g. mCLM) can boost parsimony by trading off usage rate and rule quality. ALPHAGRAMMAR exhibits comparably high parsimony as fragment MGTs (at $|V|$=100, midpoint parsimony is 252% higher than BRICS and 29% lower than mCLM), *and* comparable to orders of magnitude higher usage rate (e.g. $\sim$3.2–8.2$\times$ the midpoint Use@100 of BRICS and mCLM). mCLM attains higher Uniq than ALPHAGRAMMAR (Table 14), with two caveats. First, exact parse enumeration is intractable for string and fragment tokenizers and can time out even for ALPHAGRAMMAR, so Uniq (and the mean parse count $\mathbb{E}|\mathcal{P}(m)|$) are reported as intervals for all methods, not point comparisons (App. J.2). Second, higher Uniq trades against *universal coverage*: ALPHAGRAMMAR provably tokenizes every reasonable molecule (Thm 1), a guarantee mCLM's fixed block vocabulary cannot offer. Either way, parse multiplicity is a theoretical property; during deployment, ALPHAGRAMMAR caches one canonical parse per molecule, so its token stream is deterministic. Rule quality applies only to MGTs whose tokens are standalone molecular structures, and ALPHAGRAMMAR excels by attaining up to 0.53 Qual@100 to known functional groups, which is $60 - 229\%$ higher than the other structure-bearing MGTs. For autoregressive modeling, parsimony coupled with rule quality is very desirable for quality generations, while high usage ensures efficient learning, making ALPHAGRAMMAR a strong candidate MGT.

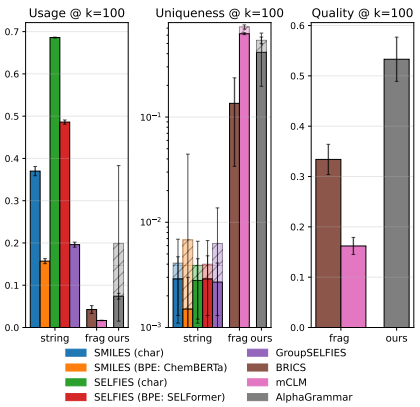


Figure 5: Tokenizer properties (usage, uniqueness, rule quality) at $|V|$=100, each $[0, 1]$ (higher $\uparrow$ better); bars show worst–best ranges where exact enumeration is intractable. Full results in App. J.

(RQ2) ALPHAGRAMMAR generates diverse and novel molecules. As a representation-level sanity check, we find sampling random type-chaining rule sequences from ALPHAGRAMMAR achieve 100% validity and the highest diversity and novelty on MOSES; full results are in App. M.2.

(RQ3, RQ5) GRAMOLE is distribution-aligned and meets prompt requirements. Gramole achieves 100% validity and favorable novelty–realism trade-off on unconditional generation, without benefits of SMILES pretraining. It also follows prompts specifying molecular structures, physicochemical properties, and required or excluded substructures. Full results are in App. M.3.

(RQ4) GRAMOLE optimizes FDA approved drugs by modifying the grammar rule composition. We compare GRAMOLE against mCLM along two independent FDA drug sets (Table 1; details in App. L.2). First, for a faithful comparison to Edwards et al. (2025), we obtained their unreleased 122-drug subset and in-house oracle through correspondence and evaluate GRAMOLE under their exact protocol: GRAMOLE improves predicted ADMET properties by $+43.32\%$ vs. mCLM's $+14.57\%$. Second, to remove any dependence on private artifacts, we reproduce the case study with public substitutes, resulting in 84 DrugBank drugs scored by ADMET-AI (Swanson et al., 2024): GRAMOLE

Table 1: We follow Edwards et al. (2025), curate FDA-approved drugs composed of synthesis-guaranteed blocks, and report avg. ADMET predicted properties before/after model-proposed modifications. (Top): the nine rows of Edwards et al. (2025), plus a GRAMOLE row evaluated on their 122-drug set and in-house oracle (obtained via correspondence). (Middle, Bottom): an independently reproducible setup on 84 public DrugBank drugs, scored by two independent public predictors. *Tanim.* is Tanimoto-to-parent (higher = more faithful edit). Best per column **bolded**.

Model	ADMET Predictions						Predicted Property Improv.			Structure		
	AMES (↓)	BBBP (↑)	CYP3A4 (↓)	DILI (↓)	HIA (↑)	PGP (↓)	Mean %	Median %	Win Rate (/6)	Valid. (↑)	SA (↓)	Tanim. (↑)
Ensemble predictor, N=122 (mCLM's released oracle) — faithful reproduction												
FDA Drug (n=122)	47.81	61.40	2.05	60.08	98.96	64.58	+0.00%	+0.00%	0	**100%**	2.70	1.00
MoleculeSTM	47.1	63.4	2.2	59.3	98.76	64.1	+0.31%	—	—	93.80%	2.64	—
FineMolTex	47.5	66.0	2.4	59.5	98.84	64.0	-0.73%	—	—	94.20%	2.58	—
LDMol	49.0	63.5	2.5	57.5	99.00	67.6	-3.07%	—	—	—	—	—
GPT-4o	46.0	72.1	2.2	60.8	99.30	65.2	+2.45%	—	—	—	—	—
GPT-5	49.1	70.2	2.4	61.2	99.20	65.5	-0.82%	—	—	—	—	—
Claude-3.5-Haiku	49.3	72.2	2.2	58.6	98.30	65.7	+1.64%	—	—	—	—	—
Gemini-2.5-Flash	45.0	72.5	1.8	58.1	99.10	64.5	+6.98%	—	—	—	—	—
mCLM	44.48	85.24	1.42	53.66	98.99	64.42	+14.57%	+8.83%	**6**	**100%**	2.44	**0.210**
GRAMOLE (Ours)	**34.98**	**99.98**	**0.16**	**30.17**	**100.00**	**47.00**	**+43.32%**	**+38.50%**	**6**	**100%**	2.13	0.118
ADMET-AI, N=84 — in-family predictor (saturation-prone on CYP3A4/PGP; see App.∼L.2)												
FDA Drug (n=84)	31.11	82.39	30.02	48.61	97.07	37.77	+0.00%	+0.00%	0	**100%**	2.69	1.00
mCLM	22.35	96.36	**1.06**	**15.69**	99.71	**0.60**	+51.73%	+47.94%	**6**	**100%**	2.38	0.083
GRAMOLE (Ours)	**4.74**	**98.56**	1.78	23.07	**100.00**	2.66	**+57.83%**	**+68.65%**	**6**	**100%**	**2.21**	**0.154**
ADMETlab 3.0 Fu et al. (2024), N=84 — independent predictor (robustness screen)												
FDA Drug (n=84)	39.03	51.64	40.56	52.70	93.69	20.27	+0.00%	+0.00%	0	**100%**	2.69	1.00
mCLM	25.58	96.05	1.98	24.65	91.98	25.02	+40.59%	+43.84%	4	**100%**	2.38	0.083
GRAMOLE (Ours)	38.07	**93.89**	6.48	32.34	**97.38**	8.65	**+44.70%**	**+47.98%**	**6**	**100%**	**2.21**	**0.154**

improves $+57.83\%$ vs. mCLM's $+51.73\%$. Because ADMET-AI empirically saturates on the smallest-baseline endpoints CYP3A4 & PGP (saturation analysis in App. M.4), we add a third, fully independent predictor ADMETlab 3.0 (Fu et al., 2024); different corpus, architecture, and institution) and report denominator-insensitive metrics: GRAMOLE improves all six endpoints over the FDA baseline vs. four for mCLM, with median relative improvement $+47.98\%$ vs. $+43.85\%$ and edits that stay $1.9\times$ closer to the parent drug (Tanimoto 0.154 vs. 0.083). Both mCLM and GRAMOLE *constrain* for validity, but mCLM also constrains for synthesizable-guaranteed blocks at inference-time, yet GRAMOLE optimized molecules have 0.17 lower SA score, a heuristic for synthetic accessibility Ertl & Schuffenhauer (2009). Since the data and model architecture are the same, we can attribute our performance to two advantages of ALPHAGRAMMAR tokenizer. First, mCLM's training uses only the top 1k most frequent block fragments, and the adapter has to generalize to synthesis-guaranteed blocks, which may be out-of-distribution. In contrast, GRAMOLE's vocabulary is consistent across training and inference, which eliminates any generalization gaps due to modality adapters. Second, rule tokens are more reusable and modular, offering more flexibility than BRICS-style fragments. For instance, in Fig. 6, we see that GRAMOLE suggests a modification to Emedastine, an H1 antihistamine, to lower its mutagenicity risk. In particular, GRAMOLE removes a chain-extending rule and modifies the 7-membered amine ring, reducing the likelihood of forming reactive intermediates. The nonlinear construction order of the ring backbone, made possible by grammar, allows the flexibility to selectively modify outer moieties sequentially before inner rules that organize their relationship.

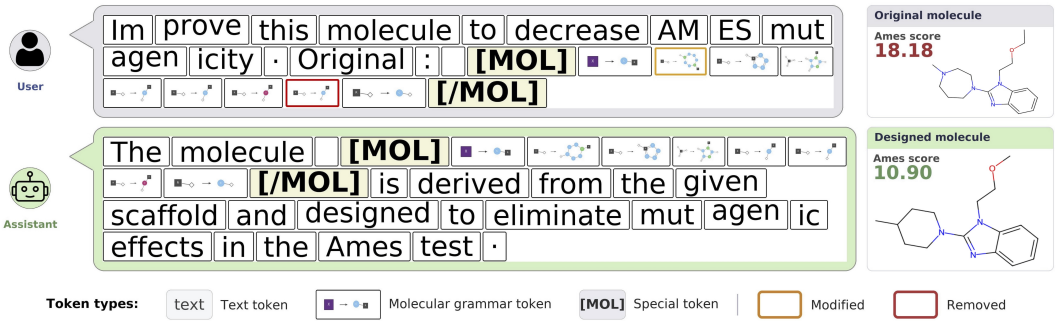


Figure 6: Case study example from our Table 1 benchmarking; full analysis in App. O.

(RQ5, RQ6) GRAMOLE is an end-to-end inverse design with retrosynthesis solution. In Table. 19, we see GRAMOLE nearly guarantee validity (95% to 100%), *substantially* outperform all baselines in synthesis planning of its designs. At the default setting with matched Qwen3-8B baselines,

GRAMOLE leads on both drug RS (48.1 ± 11.8 vs. 36.7, 3 seeds) and material RS (78.7 ± 4.8 vs. 25.3, $3.1\times$) and matches or exceeds all non-Llamole baselines on controllability (best GRAMOLE result achieve 1.3% and 9.3% better property adherence metric than best non-Llamole result). Since molecular design is multi-objective in nature, we plot the Pareto-efficient methods over these two dimensions in Fig. 7, and we find GRAMOLE and Llamole form the Pareto frontier. In particular, Llamole *pretrains* three specialized graph models separately, requiring additional training stages that we do not require. We attribute GRAMOLE's strengths in synthesis planning to the ALPHAGRAMMAR tokenizer, as rules are symbolic patterns that can capture hard constraints in real synthesis, reducing chance of hallucinating unfeasible structures. Overall, GRAMOLE offers a streamlined solution that analyzes property requirements, designs molecular structures, outputs textual analysis, and plans under real-world synthesis constraints, all in one unified autoregressive model.

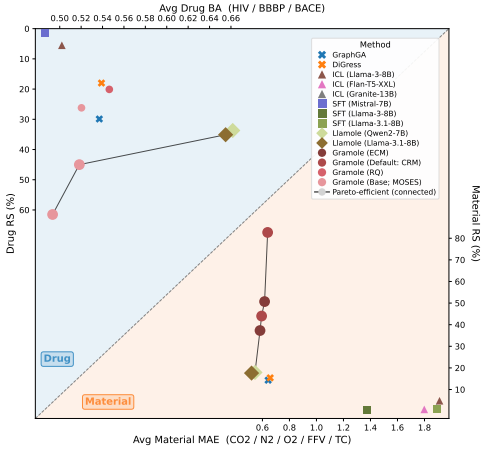


Figure 7: RS vs Balanced Acc. / MAE (Drug/Material). Full table in App. M.

(RQ7) ALPHAGRAMMAR autonomously discovers functional groups. We probe whether θ parameterized contractions are more likely to recapitulate known functional groups (the positives satisfy exact match criterion in App. D), by computing standard ranking metrics $@k$ under the Null hypothesis that probabilities are random ($\phi_j^{(t)}(\theta)$ replaced by $\mathrm{Uniform}[0,1]$). In Table 24, we see that ALPHAGRAMMAR with the default rule usage and success rewards increases the ranks of rules whose RHS match exactly with known functional groups compared to the null, increasing the average percentile rank by 0.244. This suggests that ALPHAGRAMMAR is learning interpretable rules autonomously, even without explicitly providing functional-group-based rewards. Adding per-step rewards for functional group (ECM) and reaction matching (CRM) further increases the ranks of selection probability for rules matching functional groups. We examine a few cases in App. P where the trained potential function upweights functional group associated contractions.

7 DISCUSSION

Ablation Studies. We ablate key ALPHAGRAMMAR modules, to explore the tokenizer's relation to GRAMOLE in Table 19. First, we swap out $\mathcal{D}_{\text{MolQA}}$ for $\mathcal{D}_{\text{MOSES}}$ and observe a significant improvement in Avg. RS (51.2%), more cohesive text ($+49\%$ ROUGE-L), and only a minor drop in structure similarity (-0.007). This suggests ALPHAGRAMMAR can be pretrained without overfitting; a well-trained tokenizer positively transfers. Next, we ablate ALPHAGRAMMAR altogether, replacing θ with the greedy heuristic of contracting minimal clusters, i.e. yielding atomic rules only (Theorem 1). An atomic vocabulary coupled with a greedy contraction procedure produces a closed and complete tokenizer, but GRAMOLE (Default) improves average property adherence by 7.0% and 8.6%, while avg. RS increases by 146%, connecting ALPHAGRAMMAR training to downstream performance gains. Finally, we add rule quality oracles and ablate its variants (RQ/ECM/CRM in App. D), since we observe interpretability gains in Table 24. We see noticeable improvements across *all dimensions* (Material RS as high as 83%), suggesting strong correlations between token quality and performance across inverse design and synthesis planning, where expert knowledge is indispensable. In App. Q, we show ALPHAGRAMMAR can adjust contraction decisions for the same local topology depending on surrounding context, offering flexibility which heuristic fragmentation algorithms don't have.

Empirical Analysis of Theoretical Assumptions. We test whether our assumptions hold using the MOLQA corpus (128,140 molecules) in App. I. These assumptions hold broadly in MolQA: all molecules satisfy the measured pathwidth bound, 98.6% admit our default junction-tree initialization, and the observed atomic support contains just under 10k rules (App. I).

Limitations. (i) GRAMOLE was trained using LoRA-style SFT only, so it may not reach its performance ceiling compared to LLMs that pretrained on SMILES. (ii) We do not post-train beyond SFT: first, for fair comparison against cited papers and second, because we aim to prove the concept is compute-accessible. We expect performance to improve from data/compute scaling and post-training.

Ethics Statement

All experiments use public benchmark datasets and *predicted* (not experimentally measured) properties, involve no human subjects or personally identifiable information, and release no hazardous artifacts. Our multimodal language model can model text and molecular structures, which can lead to conception of exciting new compounds. However, eventual outcomes depend on real-world institutions. We do not foresee any direct negative impact of our work, but understand our work can have real-world impacts.

Reproducibility Statement

We detail the benchmarks, datasets, training parameters, evaluation proto-1684 cols, and optimization setups needed to reproduce our results. We cite previous works whose benchmarking setups we extend, and we detail the changes we made to the standard setups, if any. We attach as supplementary material a self-contained package for training AlphaGrammar and using it as a tokenizer for Gramole, with detailed usage instructions. We use data sources from established benchmarks, and describe our benchmarking setup in detail. We provide architecture, training and compute details in sections like App. H and App. G.

AI Use Statement

In this work, we used generative AI tools to assist in editing code, assisting in writing proofs, cleaning up figures, and revising the manuscript. We have not used generative AI tools to propose novel contributions, make hypotheses, develop theoretical models or conceptual frameworks, design research methodology or experiments, or analyze qualitative or quantitative results. Synthetic data sets, translation, clean and reformat dataset are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. Molgpt: molecular generation using a transformer-decoder model. *Journal of Chemical Information and Modeling*, 62(9): 2064–2076, 2021.

David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

Esben Jannik Bjerrum. Smiles enumeration as data augmentation for neural network modeling of molecules. *arXiv preprint arXiv:1703.07076*, 2017.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3): 1096–1108, 2019.

Joseph M Cavanagh, Kunyang Sun, Andrew Gritsevskiy, Dorian Bagni, Yingze Wang, Thomas D Bannister, and Teresa Head-Gordon. Smileyllama: Modifying large language models for directed chemical space exploration. *arXiv preprint arXiv:2409.02231*, 2024.

Austin H Cheng, Andy Cai, Santiago Miret, Gustavo Malkomes, Mariano Phielipp, and Alán Aspuru-Guzik. Group selfies: a robust fragment-based molecular string representation. *Digital Discovery*, 2(3):748–758, 2023.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 924–932, 2013.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, 2018.

Jorg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem*, 3(10):1503, 2008.

Dariusz Dereniowski. From pathwidth to connected pathwidth. *SIAM Journal on Discrete Mathematics*, 26(4):1709–1732, 2012.

Carl Edwards, Chi Han, Gawon Lee, Thao Nguyen, Sara Szymkuć, Chetan Kumar Prasad, Bowen Jin, Jiawei Han, Ying Diao, Ge Liu, et al. mclm: A modular chemical language model that generates functional and makeable molecules. *arXiv preprint arXiv:2505.12565*, 2025.

Peter Ertl. An algorithm to identify functional groups in organic molecules. *Journal of cheminformatics*, 9(1):36, 2017.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):8, 2009.

Li Fu, Shaohua Shi, Jiacai Yi, Ningning Wang, Yuanhang He, Zhenxing Wu, Jinfu Peng, Youchao Deng, Wenxuan Wang, Chengkun Wu, et al. ADMETlab 3.0: an updated comprehensive online ADMET prediction platform enhanced with broader coverage, improved performance, API functionality and decision support. *Nucleic Acids Research*, 52(W1):W422–W431, 2024.

Zijie Geng, Shufang Xie, Yingce Xia, Lijun Wu, Tao Qin, Jie Wang, Yongdong Zhang, Feng Wu, and Tie-Yan Liu. De novo molecular generation via connection-aware motif mining. *arXiv preprint arXiv:2302.01129*, 2023.

Minghao Guo, Veronika Thost, Beichen Li, Payel Das, Jie Chen, and Wojciech Matusik. Data-efficient graph grammar learning for molecular generation. *arXiv preprint arXiv:2203.08031*, 2022.

Shuhan Guo, Yatao Bian, Ruibing Wang, Nan Yin, Zhen Wang, and Quanming Yao. Unimot: Unified molecule-text language model with discrete token representation. *arXiv preprint arXiv:2408.00863*, 2024.

Sumin Ha, Jun Hyeong Kim, Yinhua Piao, and Sun Kim. Mv-clam: Multi-view molecular interpretation with cross-modal projection via language model. *arXiv preprint arXiv:2503.04780*, 2025.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*, 2021.

Zihao Jing, Qiuhao Zeng, Ruiyi Fang, Yan Sun, Boyu Wang, and Pingzhao Hu. Entropy-guided dynamic tokens for graph-llm alignment in molecular understanding. *arXiv preprint arXiv:2602.02742*, 2026.

Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization. In *International Conference on Machine Learning*, pp. 3183–3191. PMLR, 2019.

Dongki Kim, Wonbin Lee, and Sung Ju Hwang. Mol-llama: Towards general understanding of molecules in large molecular language model. *arXiv preprint arXiv:2502.13449*, 2025.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pp. 1945–1954. PMLR, 2017.

Clemens Lautemann. The complexity of graph languages generated by hyperedge replacement. *Acta Informatica*, 27(5):399–421, 1990.

Gang Liu, Michael Sun, Wojciech Matusik, Meng Jiang, and Jie Chen. Multimodal large language models for inverse molecular design with retrosynthetic planning. *arXiv preprint arXiv:2410.04223*, 2024.

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

AkshatKumar Nigam, Robert Pollice, Mario Krenn, Gabriel dos Passos Gomes, and Alan Aspuru-Guzik. Beyond generative models: superfast traversal, optimization, novelty, exploration and discovery (stoned) algorithm for molecules using selfies. *Chemical science*, 12(20):7079–7090, 2021.

AkshatKumar Nigam, Serhii Tretiakov, and Robert Pollice. Exhaustive molecular string enumeration for data augmentation and structure exploration. 2026.

Emmanuel Noutahi, Cristian Gabellini, Michael Craig, Jonathan SC Lim, and Prudencio Tossou. Gotta be safe: a new framework for molecular design. *Digital Discovery*, 3(4):796–804, 2024.

Rıza Özçelik, Sarah de Ruiter, Emanuele Criscuolo, and Francesca Grisoni. Chemical language modeling with structured state space sequence models. *Nature Communications*, 15(1):6176, 2024.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:565644, 2020.

Indra Priyadarsini, Seiji Takeda, Lisa Hamada, Emilio Vital Brazil, Eduardo Soares, and Hajime Shinohara. Selfies-ted: a robust transformer model for molecular representation using selfies. 2025.

Itiroo Sakai. Syntax in universal translation. In *Proceedings of the International Conference on Machine Translation and Applied Language Analysis*, 1961.

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

Michael Sun, Minghao Guo, Weize Yuan, Veronika Thost, Crystal Elaine Owens, Aristotle Franklin Grosz, Sharvaa Selvan, Katelyn Zhou, Hassan Mohiuddin, Benjamin J Pedretti, et al. Representing molecules as random walks over interpretable grammars. In *International Conference on Machine Learning*, pp. 46988–47016. PMLR, 2024a.

Michael Sun, Alston Lo, Minghao Guo, Jie Chen, Connor Coley, and Wojciech Matusik. Procedural synthesis of synthesizable molecules. *arXiv preprint arXiv:2409.05873*, 2024b.

Michael Sun, Orion Foo, Gang Liu, Wojciech Matusik, and Jie Chen. Directed graph grammars for sequence-based learning. In *International Conference on Machine Learning*, pp. 57391–57418. PMLR, 2025a.

Michael Sun, Weize Yuan, Gang Liu, Wojciech Matusik, and Jie Chen. Foundation molecular grammar: Multi-modal foundation models induce interpretable molecular graph languages. In *International Conference on Machine Learning*, pp. 57886–57937. PMLR, 2025b.

Kyle Swanson, Parker Walther, Jeremy Leitz, Souhrid Mukherjee, Joseph C Wu, Rabindra V Shivnaraine, and James Zou. Admet-ai: a machine learning admet platform for evaluation of large-scale chemical libraries. *Bioinformatics*, 40(7):btae416, 2024.

David Weininger. 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, 1988.

Juan-Ni Wu, Tong Wang, Yue Chen, Li-Juan Tang, Hai-Long Wu, and Ru-Qin Yu. t-smiles: a fragment-based molecular representation framework for de novo ligand design. *Nature Communications*, 15(1):4993, 2024.

Mingliang Yan, Yanhua Yu, Ruochi Zhang, Zhiyuan Liu, Ruicheng Zhang, Yimeng Ren, Kangkang Lu, Zhiyong Huang, Feng Luo, and Zhen Cai. Deepmoltex: Deep alignment of molecular graphs with large language models via mixture of modality experts. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 2323–2332, 2025.

Zaifei Yang, Hong Chang, Ruibing Hou, Shiguang Shan, and Xilin Chen. Knowmol: Advancing molecular large language models with multi-level chemical knowledge. *arXiv preprint arXiv:2510.19484*, 2025.

Atakan Yüksel, Erva Ulusoy, Atabey Ünlü, and Tunca Doğan. Selformer: molecular representation learning via selfies language models. *Machine Learning: Science and Technology*, 4(2):025035, 2023.

CONTENTS

1 Introduction **1**

2 Related Works **2**

Chemical Language Models 2

Learning Molecular Grammars 3

3 ALPHAGRAMMAR: Neural Grammar Parsing Tokenizer **3**

4 GRAMOLE: Language Modeling with Grammar Vocabulary **5**

5 Evaluation **6**

Research Questions 6

Benchmarks 6

Baselines 6

Metrics 7

Compute Complexity and Costs 7

6 Results **7**

(RQ1) ALPHAGRAMMAR exhibits favorable tokenizer properties 7

(RQ2) ALPHAGRAMMAR generates diverse and novel molecules 7

(RQ3, RQ5) GRAMOLE is distribution-aligned and meets prompt requirements 7

(RQ4) GRAMOLE optimizes FDA approved drugs by modifying the grammar rule composition 7

(RQ5, RQ6) GRAMOLE is an end-to-end inverse design with retrosynthesis solution . . . 8

(RQ7) ALPHAGRAMMAR autonomously discovers functional groups 9

7 Discussion **9**

Ablation Studies 9

Empirical Analysis of Theoretical Assumptions 9

Limitations 9

A Formal Grammar Preliminaries **17**

B Details of ALPHAGRAMMAR Formulation **17**

B.1 Junction Tree Initialization 17

B.2 Atomic Rule Bound 17

B.3 Implementation Details 18

B.4 Handling Non-Reasonable Molecules 19

B.5 Sampling and Inference 19

C Details on Adapting Bolinas Parser **20**

C.1 Bolinas parser 20
C.2 Implementation choices in Bolinas 20
C.3 Our changes 20
C.4 Complexity 21

D Details of ALPHAGRAMMAR Training **21**
D.1 GRPO Derivation 21
D.2 Training Metrics 22
D.3 Periodic Bonuses 23
D.4 Hyperparameter Settings 24

E Ablations **24**
E.1 Training Trajectory 24
E.2 Per-Reward Ablation 24
E.3 Advantage-Mode Ablation 25
E.4 Parsing Budget and the Parsimony Reward 26
E.5 Rule-Quality Oracle Transfer 27
E.6 Canonical-Parse Determinism 27

F Details of GRAMOLE Inference **27**
F.1 Generating Valid Molecules via Type Chaining 27
F.2 Generating Valid Polymers via Star Counting 28
F.3 Inference Complexity 28
F.4 Rule Vocabulary 29

G Compute Cost **29**

H Computational Complexity **29**

I Empirical Analysis of Theoretical Assumptions **31**
I.1 A1: Reasonableness is satisfied for >98% of the dataset 31
I.2 A2: The default junction-tree initialization is almost always acyclic 31
I.3 A3: The learned grammar is a *tiny* fraction of the theoretical atomic set. 31
I.4 A4: Rule usage is heavy-tailed, motivating compositionality 32
I.5 Takeaways 32

J Tokenizer Properties **33**
J.1 Definitions 33
J.2 Parse estimation for intractable tokenizers 33
J.3 Parse enumeration per baseline 34
J.4 Rule quality per fragment-based baseline 34

J.5 Worst/best bounds for sampled parses 34

K Interpretability Evaluation **35**

K.1 Metric Definitions 35

K.2 Statistical Test 35

L Benchmarking Details **36**

L.1 SmileyLlama Cavanagh et al. (2024) 36

Background 36

GRAMOLE Extension 36

Evaluation Protocol 36

L.2 mCLM Edwards et al. (2025) 37

Background 37

GRAMOLE Extension 37

Evaluation Protocol 37

L.3 Llamole Liu et al. (2024) 37

Background 37

GRAMOLE Extension 38

Evaluation Protocol 38

M Additional Results **38**

M.1 (RQ1) Tokenizer Evaluation 38

M.2 (RQ2) Grammar Sampling for Molecular Generation 39

M.3 (RQ3, RQ5) Unconditional Sampling and Prompt Following 39

M.4 (RQ4) FDA-Drug Optimization 40

M.5 (RQ6) Inverse Design with Retrosynthesis 41

M.6 (RQ7) Functional Group Enrichment 42

N Atomic Rule Enumeration **42**

N.1 Category A 44

N.2 Category B 45

N.3 Category C 47

N.4 Combined bound 47

N.5 Numerical evaluation 47

O Case Study: AMES Improvement **48**

P Case Study: Rule Interpretability **48**

Q Case Study: Context-Aware Contraction **50**

A Formal Grammar Preliminaries

A hyperedge replacement grammar (HRG) for molecular hypergraphs consists of nonterminal and terminal atom labels, a start symbol, and a finite set of production rules. Each production rule replaces a nonterminal atom with a molecular hypergraph fragment whose external bonds specify how the fragment attaches to the rest of the molecule. The process terminates when no nonterminal atoms remain, yielding a fully terminal molecular hypergraph.

Definition A.1. Given a molecule M, its hypergraph $H_M = (A_M, B_M)$ consists of:

(1) Finite set of bonds B_M
(2) Finite set of atoms A_M together with attachment map $\mathrm{att}\colon A_M \to B_M^*$
(3) Atomic label assignment $l_A\colon A_M \to L_A$ (set of atomic types)
(4) Bond label assignment $l_B\colon B_M \to L_B$ (set of bond types).

When (5) L_A is expanded to include non-terminal labels, and (6) *external bonds* may attach one end to atoms in A_M, we promote A.1 to a *hypergraph fragment*, typed by its list of external bond types.

Definition A.2. A HRG $\mathcal{G}$ for molecular hypergraphs consists of:

(1) Terminal labels L_A and non-terminal labels N
(2) Type function $\mathrm{type}\colon N \to L_B^*$, the list of external bond types required for attachment
(3) Production rules ρ, each of the form $A_\rho \to R_\rho$, where $A_\rho \in N$, R_ρ is a hypergraph fragment, and $\mathrm{type}(A_\rho) = \mathrm{type}(R_\rho)$
(4) Start symbol $S \in N$, with $\mathrm{type}(S) = []$.

Definition A.3. A HRG is *linear* if the hypergraph fragment of every production rule contains at most 1 atom with non-terminal label, and further consists of two rule type functions:

(5) $\mathrm{intype}(\rho) = \mathrm{type}(R_\rho)$,
(6) $\mathrm{outtype}(\rho) \in L_B^*$, the external bond types of the sole non-terminal atom if present, $\perp$ otherwise.

Definition A.4. A junction tree of a molecule is a tree T, each of whose nodes η is labeled with $A_\eta \subseteq A$ and $B_\eta \subseteq B$, such that the following properties hold:

(1) Atom cover: $\forall a \in A$, there is a node $\eta \in T$ s.t. $a \in A_\eta$.
(2) Bond cover: $\forall b \in B$, there is exactly one node $\eta \in T$ s.t. $b \in B_\eta$, and $\{a \in A | b \text{ bonds } a\} \subseteq A_\eta$.
(3) Running intersection: $\forall a \in A$, the set $\{\eta \in T | a \in A_\eta\}$ is connected.

Remark 1. We refer to the induced substructure of η (bonds B_η and atoms A_η) as a *cluster*.

Definition A.5. The *width* of a junction tree is $\max_{\eta \in T}(|A_\eta| - 1)$. The *treewidth* of a molecular hypergraph is the minimal width of any junction tree.

Definition A.6. A *connected path decomposition* of a molecular hypergraph is a junction tree T whose underlying tree is a path $\eta_1, \eta_2, \ldots, \eta_m$, and for every prefix $1 \le i \le m$, the sub-hypergraph induced by $A^{(i)} := \bigcup_{j=1}^{i} A_{\eta_j}$ and $B^{(i)} := \bigcup_{j=1}^{i} B_{\eta_j}$ is connected. The *connected pathwidth* of a molecular hypergraph is the minimum width, $\max_{\eta \in T}(|A_\eta| - 1)$, of any connected path decomposition.

Definition A.7 (Reasonable Molecule)**.** A molecule M is *reasonable* if it is chemically valid, every atom has valence $\le V$, every minimal ring has size $\le R$, and H_M has connected pathwidth $\le C$.

B Details of AlphaGrammar Formulation

B.1 Junction Tree Initialization

If H_M has treewidth k, we can convert H_M into a junction tree T_M with every node containing no more than $k + 1$ atoms. In practice, we initialize B_η as the bonds of a minimal ring (from `rdkit.Chem.GetSymmSSSR`) or a non-ring bond. This is a heuristic way to construct T_M, but one that is justified by real-world dataset statistics. After rings are collapsed, the resulting connectivity graph (Fig. 3 (b)) is a valid junction tree for $\sim 99\%$ of molecules (see App. I). This ensures coverage by our explicitly enumerated atomic rules in App. N.

B.2 Atomic Rule Bound

Reasonable molecules have bounded connected pathwidth, which implies bounded treewidth. Thus, a junction tree exists with bounded width. Contracting a minimal cluster results in an atomic

rule. However, our explicit atomic rule set in App. N only accounts for cases where the heuristic initialization results in a valid junction tree. If the heuristic is insufficient to produce a junction tree, then the atomic rule set needs to be revised to account for more complex RHS topologies, up to (bounded) size and reasonableness constants. The essential observation from App. N is that under bounded connected pathwidth and fixed reasonableness constants, the total number of rules is bounded and can be enumerated.

B.3 Implementation Details

Our hypergraph encodes stereochemistry: R/S at stereocenters and E/Z at double bonds are carried as node/hyperedge attributes (e.g. Thalidomide's chiral carbon carries an explicit chirality tag), so reconstruction preserves stereochemical information. We materialize each junction-tree node $\eta \in T_M$ as a data record that carries two complementary views of its atom cover A_η:

- an *additive* view, A_η^{add}, storing the original atom indices that belong to η from the outset. When two clusters are merged by a contraction, the merged additive cover is the union of the originals; atoms that become internal to the contracted region R^* are kept, never stripped. This preserves chemical context such as complete aromatic rings, which is required whenever the fragment is re-exported to an atom-typed form for rule emission.
- a *terminal* view, $A_\eta^{\mathrm{term}} \subseteq A_\eta^{\mathrm{add}}$, consisting of the atoms of η that are still unvisited at the current contraction step, i.e. $A_\eta^{\mathrm{term}} = A_\eta^{\mathrm{add}} \setminus A_{R^*}$. It shrinks monotonically as contractions proceed and drives both (i) the pooling of per-atom GNN features into the cluster feature $f(\eta)$ that the potential $\mathcal{F}_\theta$ consumes and (ii) any visualization of the partial hypergraph.

Separating the two views is important: the additive view is what makes pairwise cluster overlap and boundary-bond computations cheap and exact throughout the contraction process, while the terminal view is what isolates the still-active substructure whose features should inform the selection potential $\phi_\eta(\theta)$.

A boolean flag $r_\eta \in \{0, 1\}$ and the set $\mathrm{NT}(H_M) \subseteq A_M$ of currently non-terminal atoms together encode R^*: at any contraction step exactly one node has $r_\eta = 1$ and $A_\eta^{\mathrm{term}} = \varnothing$, and for that node $A_\eta^{\mathrm{add}} = \bigcup_{\eta' \in R^*} A_{\eta'}$. We call this the *standalone* R^* node; it is the sole root of the rooted tree of Fig. 3 (c) and the only node for which the potential reads the learned embedding $e_{R^*} \in \mathbb{R}^d$ in place of pooled atom features.

The contraction step of Sec.3 is executed in two passes. Given the Bernoulli selection vector $[X_\eta^{(t)}]$ drawn by the policy, a union-find over pairwise bond-overlap groups selected clusters into connected components, and each component is merged into a new node whose additive cover is the union of the members, whose terminal cover is the union of the members' terminal covers, and whose R^* flag is the OR of the members' flags. A second pass then (a) marks the contracted atoms and bonds as *visited* on H_M, (b) updates $\mathrm{NT}(H_M)$ to the boundary of the contracted region, (c) advances a *watershed* bookkeeping, tagging each newly contracted bond node with the integer index t of the step at which it was absorbed and memoising the boundary bond nodes per watershed level, and (d) transforms the merged node into the standalone R^* by emptying its terminal cover and setting $r_\eta = 1$, while stripping the same indices from the terminal covers of all overlapping neighbors and leaving *their* additive covers intact.

Rule emission reads off one production ρ per step by walking the hyperedges inside the contracted cluster. Unvisited atom edges become terminal edges of the RHS; bond nodes that either (i) sit on an external attachment to the rest of the molecule or (ii) touch a visited atom inside the cluster contribute external nodes. The type-(ii) boundary bonds are collected into a single unified R^* nonterminal edge on the RHS whose bond-symbol list records $\mathrm{intype}(\rho) \in L_B^*$, while the type-(i) external bonds become the single LHS R^* edge (or an empty LHS for an initial rule). Rules are deduplicated into the grammar corpus by a Morgan-refinement hash on their bipartite representation that coincides with graph isomorphism modulo external-node labeling, so repeated rule instances across a dataset share a single entry and its usage counts contribute to the vocabulary statistics of Appendix F.

On the RL side the connectivity graph G of Fig. 3 (b) is built by the rule "two clusters share an edge whenever their bond sets intersect at more than one bond, or at exactly one non-ring bond". The R^* standalone is attached to every cluster that shares a boundary bond with the contracted region, which

matches the parent–child dependency $X_{R^*} = 1 \Rightarrow X_\eta = 1$ for each child η of R^*. The potential function computes $\phi_\eta(\theta)$ with a small MLP regularized by a self-attention layer whose keys and values at the R^* node are driven by e_{R^*}, and whose attention mask is the ancestor mask of the rooted tree so that each node's logit depends only on its ancestors. Sampling and log-probability replay share a single upward dynamic program $A_v = \exp(u_v) \prod_{c \in \mathrm{ch}(v)} (1 + A_c)$ with $u_v = \log \phi_v(\theta)$; after this, the "at least one child of R^* is selected" constraint reduces to a sequential conditional Bernoulli pass over $\mathrm{ch}(R^*)$. The per-node conditional is $q_v(\theta) = A_v/(1 + A_v)$, and the factors $1 + A_v$ are the local partition functions whose product yields the policy normalizer Z_θ; the replayed log-probability is therefore exactly normalized, with $-\log Z_\theta$ differentiated in full (App. D). All probabilities are clamped to $[\epsilon,\ 1 - \epsilon]$ before taking $\log(\cdot)$ or $\log(1 - \cdot)$ so that log-probabilities, and therefore policy gradients, remain finite even when the policy becomes highly peaked.

B.4 Handling Non-Reasonable Molecules

For a reasonable molecule (Def. A.7) the connectivity graph G produced by our junction-tree initialization is a forest: cycles in G are iteratively fused by union–find on their atom clusters until $\mathrm{cycle_basis}(G) = \varnothing$, and the resulting tree has $|A_\eta| \le k + 1$ on every bag for a connected path decomposition of width $k \le C$. Exact sampling and inference over the rooted tree of Fig. 3(c) then follow in linear time via the upward DP introduced above.

Non-reasonable molecules violate Def. A.7 typically via a connected pathwidth that exceeds our enumeration budget C, so the cluster fusion either does not terminate at an acyclic G within the allotted iterations, or produces a G that becomes cyclic again after a later contraction because R^* reintroduces a back edge to an already adjacent cluster. We preserve the underlying generative algorithm and fall back to a *spanning-tree approximation* at the graphical modeling layer, while leaving the grammar, the contraction, and the rule-emission steps untouched. Concretely, whenever the current G is non-tree, we orient a spanning tree $T \subseteq G$ rooted at R^* by $T = \text{BFS-tree}(G, R^*)$; in the initial step where R^* does not yet exist we attach a virtual root to every cluster and first sample the seed cluster categorically, $r^* \sim \mathrm{Cat}\Big(\phi_\eta(\theta) / \sum_{\eta'} \phi_{\eta'}(\theta)\Big)$, before performing BFS from r^*. The edges of $G \setminus T$ are dropped from the factor graph.

This replaces the exact rooted-tree factorization $p_\theta(\mathbf{X}) = \prod_\eta p_\theta(X_\eta \mid X_{\mathrm{parent}(\eta)})$ over the true junction tree T_M by the analogous factorization over the spanning tree T, multiplied by the categorical factor for r^* in the virtual-root branch:

$$\tilde{p}_\theta(\mathbf{X}) \;=\; \frac{\phi_{r^*}(\theta)}{\sum_{\eta'} \phi_{\eta'}(\theta)} \prod_{\eta \in T} p_\theta\big(X_\eta \mid X_{\mathrm{parent}_T(\eta)}\big)\,.$$

Three properties survive this compromise. *(i) Sampling and inference remain exact for $\tilde{p}_\theta$*: the same upward DP and "at least one child selected" top-down pass on T draw unbiased samples from $\tilde{p}_\theta$ in $\mathcal{O}(N)$ time, and the log-probability replay inverts the same traversal to return a differentiable $\log \tilde{p}_\theta(\mathbf{X})$ suitable for REINFORCE/GRPO. *(ii) The rooted dependency $X_\eta = 1 \Rightarrow X_{\mathrm{parent}_T(\eta)} = 1$ is still enforced on T*, so every sampled contraction yields a subtree of T rooted at R^* and therefore a connected subgraph of G: the contraction introduces exactly one new R^*, so the linear-HRG constraint of Def. A.3 is preserved. *(iii) Emitted rules remain valid HRG productions*: the rule-emission procedure operates on (A_M, B_M) and the visited markers, not on the connectivity-graph topology, so any dropped back edge in $G \setminus T$ that reconnects two clusters simply contributes additional external bond nodes on the RHS of a future rule, possibly enlarging $\mathrm{outtype}(\rho)$. The price paid is a bias: the RL policy evaluates advantages under $\tilde{p}_\theta$ rather than under the clique-tree factorization that would be available for a reasonable molecule. The KL gap to the exact posterior is bounded by $|E(G) \setminus E(T)|$ and is empirically small on our datasets, because non-reasonable molecules violate Def. A.7 by at most a few extra fused rings and the BFS orientation tends to place the missing edges among siblings in T rather than across deep ancestor–descendant paths.

B.5 Sampling and Inference

Sampling and inference over the junction tree is exact for reasonable molecules and approximate for non-reasonable molecules. In both cases, the algorithmic complexity is $O(N)$ (tree traversal in linear time). The more reasonable the molecules, the more atomic the worst-case rules become while guaranteeing exact sampling and inference.

C Details on Adapting Bolinas Parser

C.1 Bolinas parser

The Bolinas parser of Chiang et al. (2013) is a CKY-style chart parser for hyperedge replacement grammars. Given an HRG $\mathcal{G}$ and an input hypergraph $H_M = (A_M, B_M)$, it decides whether $H_M \in L(\mathcal{G})$ and produces a parse forest of derivations of H_M. Its central idea is that, rather than searching directly over H_M, the parser precomputes for every production $\rho\colon A_\rho \to R_\rho$ a junction tree T_ρ of R_ρ, and uses T_ρ to dictate the order in which the atoms of R_ρ are matched against atoms of H_M. Each chart item is a partial derivation keyed by $(\rho, \eta, \sigma, \phi)$ where η is a node of T_ρ, σ is the already-recognized sub-hypergraph of H_M summarised by its boundary bonds, and $\phi\colon B_\eta \hookrightarrow B_M$ is an injective mapping from the bonds in the bag at η to bonds of H_M. Items are combined by four inference rules corresponding to walking the tree decomposition: Terminal consumes a terminal atom of R_ρ by binding it to an atom of A_M, Nonterminal composes a passive sub-derivation at a nonterminal atom of R_ρ, Binary merges two sibling sub-trees of T_ρ, and Root closes a completed rule application, producing a passive item for further composition. Using the boundary-bond representation (Sec. 3.1 of Chiang et al. (2013)), each bag carries at most $k+1$ boundary bonds, where k is the maximum width of any T_ρ, and each boundary bond carries a constant-size state summarising which of its incident atoms have already been consumed. The resulting complexity claim is $O\big(|\mathcal{G}| \cdot (3^d n)^{k+1}\big)$, where $n = |B_M|$ and d is a bound on bond degree.

C.2 Implementation choices in Bolinas

Bolinas's implementation hard-codes a *directed, rooted* variant of HRG. Atoms are stored as triples $(a_{\text{head}},\ l_A(a),\ (a_{\text{tail}1}, \ldots, a_{\text{tail}a-1}))$ in which the first bond is singled out as the "head" and the remainder as an ordered tail tuple, and every hypergraph fragment R_ρ is required to carry at least one distinguished root bond. These conventions are exploited in three places. First, the junction tree T_ρ is built by a DFS starting from R_ρ's root and following `out_edges`, so the traversal direction is well-defined and the running-intersection property is enforced by a single top-down pass. Second, the inference rule Terminal matches a rule atom $e' = (a'_h, l, (a'_{t_1}, \ldots))$ to an input atom $e = (a_h, l, (a_{t_1}, \ldots))$ by the deterministic positional bijection $a'_h \mapsto a_h,\ a'_{t_i} \mapsto a_{t_i}$, with no enumeration over reorderings. Third, the list external nodes $[b_0, \ldots, b_{k-1}]$ is interpreted as an ordered type vector, and Nonterminal binds it positionally to the tail of the nonterminal atom being expanded. At the input side the graph H_M is itself required to be rooted; the parser's goal condition is that a Root item exists whose subgraph covers all of A_M and whose rule symbol is S. Matching is otherwise oblivious to bond labels l_B: the Bolinas data structure carries *atom* labels but treats bonds as opaque identifiers.

C.3 Our changes

Molecular hypergraphs (Def. A.1) are undirected and unrooted, and they carry labels on *both* A_M and B_M. To obtain a parser consistent with this definition, we relax the Bolinas formulation in three ways. First, we store each atom as a pair $(l_A(a),\ (b_{i_1}, \ldots, b_{i_a}))$ where the tuple is the ordered list of adjacent bonds that is part of the definition of $a \in A_M \subseteq B_M$; there is no privileged head. This preserves the textbook HRG semantics that each atom has an ordered endpoint list Lautemann (1990) while dropping the artificial head/tail split. Second, because no bond of H_M or R_ρ is distinguished, the junction tree is seeded at a pseudo-root b chosen by maximum incident-atom degree, and its DFS follows all atoms incident to a bond rather than only out-edges; the three junction tree properties in Def. A.4 still hold, since they make no reference to direction.

Third, the Terminal and Nonterminal inference rules no longer admit a deterministic positional bijection: a rule atom e' of arity a may legitimately bind to an input atom e under any order-preserving injection of e''s adjacent-bond sequence into a reordering of e's, so we enumerate all $a!$ reorderings of each input atom's endpoints and keep those consistent with the accumulated ϕ and with the bond-label constraint $l_B(\phi(b')) = l_B(b')$ (and analogously l_A for atoms). This is the one place where the textbook HRG definition forces a real cost over Bolinas's head-rooted variant. Our live API additionally threads l_B through matching via the `nodelabels` flag and a `node_to_concepts` map, so that bond-type equality is checked at every positional binding rather than only at the atom-label level as in Bolinas.

C.4 COMPLEXITY

Theorem 2. Let H_M be a reasonable molecule, so $|A_M| = n$, every atom has degree at most $a \le V$, and H_M admits a connected path decomposition of width at most $k \le C$ (we use a and k to match Bolinas notation, for easier cross-reference). Let $\mathcal{G}$ be a linear HRG whose production RHSs admit junction trees of width at most k. Our parser decides $H_M \in L(\mathcal{G})$ in time

$$O\Big(|\mathcal{G}| \cdot \big(3 \cdot a! \cdot n\big)^{k+1}\Big),$$

which is polynomial in n whenever a, k are bounded by constants (in particular, for reasonable molecules, with $a \le V$ and $k \le C$).

Proof. We bound the chart size and the cost per item. A chart item is a tuple $(\rho, \eta, \phi, \mathbf{s})$: the rule ρ and bag η contribute a factor of $|\mathcal{G}|$ (summed over bags of all T_ρ). The remaining state is associated with the atoms in the bag, which by Def. A.6 number at most $|A_\eta| \le k+1$. Atoms are the entities that satisfy the running-intersection property (Def. A.4, Property 3) and therefore must be tracked across bags, whereas bonds are localized by the bond-cover property (Property 2) and incur no separate factor; this is the dual of the standard Chiang et al. (2013) accounting, in which nodes satisfy running intersection and hyperedges are localized. *Atom injection:* the injection $\phi\colon A_\eta \hookrightarrow A_M$ has at most n^{k+1} values, since $|A_\eta| \le k+1$ and $|A_M| = n$. *Endpoint permutations:* for each atom $a' \in A_\eta$ paired by ϕ with an input atom $\phi(a')$, the parser enumerates orderings of $\mathrm{att}(a')$ that align with $\mathrm{att}(\phi(a'))$; since $|\,\mathrm{att}(a')| \le a$, this contributes a factor of at most $(a!)^{|A_\eta|} \le (a!)^{k+1}$. This is the cost of replacing Bolinas's directed head/tail matching with the order-preserving bijection required by Def. A.1. *Boundary state:* for each $a' \in A_\eta$, a flag in $\mathbf{s}$ records whether a' is unrecognized, on the boundary (still to be re-encountered along its connected occurrence subtree), or fully recognized, contributing at most $3^{|A_\eta|} \le 3^{k+1}$ states, as in Chiang et al. (2013, Section 3.1). *Bonds:* once ϕ and the endpoint permutations are fixed at the unique bag η containing a bond b, the bond mapping is determined and never reconsidered, by bond cover (hence no further factor). Multiplying, the per-item state has at most $n^{k+1} \cdot (a!)^{k+1} \cdot 3^{k+1} = (3 \cdot a! \cdot n)^{k+1}$ values, and the chart-population work per item is constant in n. The total runtime is $O\big(|\mathcal{G}| \cdot (3 \cdot a! \cdot n)^{k+1}\big)$, as claimed. □

Remark 2. For reasonable molecules, $a \le V$ and $k \le C$ are absolute constants, so $(3 \cdot a!)^{k+1} \le (3 \cdot V!)^{C+1}$ is a constant and the bound collapses to $O(|\mathcal{G}| \cdot n^{C+1})$, polynomial in n.

D DETAILS OF ALPHAGRAMMAR TRAINING

This section fills in details of Sec. 3, which introduced the GRPO formulation and the five-axis reward. First, we continue the GRPO loss derivation. Then, we turn our attention to how the metrics are computed. App. C equips each rollout with a parser that enumerates $\mathcal{D}_{k,m}$ and exposes the sampled derivation $d^\star_{k,m}$ (the reverse of the rollout's contraction sequence). The remainder of the section specifies how the three in-batch metrics are computed from $\mathcal{D}_{k,m}$, how the two periodic bonuses (d, e) are maintained between batches, and how we track the data budgets consumed by training and evaluation.

D.1 GRPO DERIVATION

For each molecule, we sample a group of K trajectories $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)} \sim p_\theta(\cdot \mid M)$ and let T_k be the number of steps in rollout k. Each trajectory induces a linear HRG $\mathcal{G}^{(k)}$. Let $\widetilde{\mathcal{M}}_{i,k}$ denote the group-normalized score of metric i on rollout k. The rollout-level advantage is $A_k = (\sum_{i=1}^{L} \lambda_i \widetilde{\mathcal{M}}_{i,k})/(\sum_{i=1}^{L} \lambda_i)$. We further use denser step-wise rewards to refine credit assignment within a rollout: for grammar-level metrics, we set $\alpha^{(t)}_{i,k} = 1/T_k$, while for step-wise metrics, $\alpha^{(t)}_{i,k}$ is obtained by normalizing the corresponding per-step reward over t. We then combine them as $A^{(t)}_k = A_k \cdot (\sum_{i=1}^{L} \lambda_i \alpha^{(t)}_{i,k})/(\sum_{i=1}^{L} \lambda_i)$. The loss is

$$\mathcal{L}_{\mathrm{GRPO}} = -\frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T_k} A^{(t)}_k \, \log p_\theta(X^{(t,k)} \mid X^{(<t,k)}, M), \tag{1}$$

where $\log p_\theta(X^{(t)} \mid X^{(<t)}, M)$ is the exact log-probability of the constrained subtree policy, derived next and differentiated in full (normalizer included).

Exact policy and its log-probability. At step t the decision $X^{(t)}$ ranges over $\mathcal{C}^{(t)}$, the set of subtrees of the (spanning) junction tree that are rooted at R^* and select at least one of its children, so that exactly one new R^* is introduced (Def. A.3). The potentials $\phi_j^{(t)}(\theta) = \sigma(-\mathcal{F}_\theta(f(\eta_j^{(t)}))) \in (0,1)$ are *unnormalized node weights*, not marginal selection probabilities. The policy is the distribution they induce over $\mathcal{C}^{(t)}$,

$$p_\theta(X^{(t)} \mid X^{(<t)}, M) = \frac{1}{Z_\theta^{(t)}} \Big(\prod_{j:\, X_j^{(t)}=1} \phi_j^{(t)}(\theta) \Big) \mathbf{1}\big[X^{(t)} \in \mathcal{C}^{(t)}\big], \qquad Z_\theta^{(t)} = \sum_{X \in \mathcal{C}^{(t)}} \prod_{j:\, X_j=1} \phi_j^{(t)}(\theta), \tag{2}$$

hence

$$\log p_\theta(X^{(t)} \mid X^{(<t)}, M) = \sum_{j:\, X_j^{(t)}=1} \log \phi_j^{(t)}(\theta) \; - \; \log Z_\theta^{(t)}. \tag{3}$$

Only selected nodes contribute a potential; unselected frontier nodes contribute weight 1, and the ancestry constraint $X_\eta = 1 \Rightarrow X_{\mathrm{parent}(\eta)} = 1$ is carried by the support indicator $\mathbf{1}[X^{(t)} \in \mathcal{C}^{(t)}]$. A product of independent Bernoulli factors in $\phi_j^{(t)}$ is therefore *not* the policy: the conditioning normalizer $-\log Z_\theta^{(t)}$ is a genuine, θ-dependent term, and we retain it throughout.

Linear-time normalizer and sequential factorization. We never enumerate $\mathcal{C}^{(t)}$. The upward dynamic program of App. B, $A_v = \phi_v^{(t)}(\theta) \prod_{c \in \mathrm{ch}(v)} (1 + A_c)$, computes the local partition function $1 + A_v$ of the subtree rooted at v; these factors telescope, so $\prod_{c \in \mathrm{ch}(R^*)}(1 + A_c)$ is the partition function over *all* subtrees at R^* and $Z_\theta^{(t)} = \prod_{c \in \mathrm{ch}(R^*)}(1 + A_c) - 1$ after excluding the empty selection. The quantity $q_v(\theta) := A_v/(1 + A_v)$ is the exact probability that v is selected given its parent is; sampling and replay both use the resulting sequential (autoregressive, topological-order) factorization

$$\log p_\theta(X^{(t)} \mid X^{(<t)}, M) = \sum_{v \in \mathcal{V}_{\mathrm{off}}(X^{(t)})} \big[X_v^{(t)} \log q_v(\theta) + (1 - X_v^{(t)}) \log\big(1 - q_v(\theta)\big) \big] \\ - \; \log\Big(1 - \prod_{c \in \mathrm{ch}(R^*)} \big(1 + A_c\big)^{-1}\Big), \tag{4}$$

where $\mathcal{V}_{\mathrm{off}}(X^{(t)})$ collects the nodes whose parent is selected (the only nodes at which a Bernoulli decision is made) and the trailing term is the "at least one child of R^*" renormalizer (A_v, A_c here are the upward-DP values, distinct from the advantage A_k). Each $q_v(\theta) \in (0,1)$ is already a normalized conditional, so (4) equals (3) exactly: $-\log Z_\theta^{(t)}$ is not omitted but distributed across the $-\log(1 + A_c)$ terms inside the q_v and the trailing renormalizer. In the virtual-root branch (App. B) an additional categorical factor $\phi_{r^*}(\theta) / \sum_{\eta'} \phi_{\eta'}(\theta)$ for the seed cluster enters (4) identically.

Exactness of the gradient. Sampling and log-probability replay execute the identical dynamic program; the replay recomputes every $\phi_j^{(t)}(\theta)$ with gradients enabled, so automatic differentiation flows through all A_v, through each $q_v(\theta)$, through the "at least one child" denominator, and through the categorical factor, i.e. through $Z_\theta^{(t)}$ in full. Consequently $\nabla_\theta \log p_\theta(X^{(t)} \mid X^{(<t)}, M)$ is the exact score of the constrained policy (2), and the GRPO update (1) is an unbiased policy gradient for that policy, with no dropped conditioning normalizer.

D.2 Training Metrics

Let $\mathcal{B}_k$ be the molecule batch associated with rollout k and $\mathcal{G}^{(k)}$ the grammar it induces. We compute three metrics directly from the batch parses $\{\mathcal{D}_{k,m}\}_{m \in \mathcal{B}_k}$:

- **Parsimony** $\mathcal{M}_a$ (grammar-level, scalar). The fraction of $m \in \mathcal{B}_k$ whose hypergraph H_m is parsed by $\mathcal{G}^{(k)}$ within a per-molecule wall-clock timeout τ_{train}. This definition of parsimony takes into account multiple factors, including (i) the size of the grammar, (ii) the complexity of individual rules, and (iii) the multiplicity of parses. The parser is polynomial-time in theory (see App. C), but in practice timeouts are common for complex grammars. Increasing any of (i)-(iii) while fixing the others on the same batch strictly increases the complexity of the Bolinas parser to return. Because parsimony yields a single number per rollout, its per-step attribution is uniform: $\alpha_{a,k}^{(t)} = 1/T_k$.

- **Compositionality** $\mathcal{M}_b$ (rule-based, per step) We measure how often each rule is reused across the batch's parses. For every $r \in \rho(\mathcal{G}^{(k)})$,

$$\text{usage}_k(r) \;:=\; \sum_{m \in \mathcal{B}_k} \frac{1}{L_{k,m}} \sum_{d \in \mathcal{D}_{k,m}} \#\{\text{occurrences of } r \text{ in } d\},$$

  i.e. the expected number of times r appears in a uniform-random derivation of m, summed over the batch. Step t of rollout k, which applies rule r_t, receives the per-step reward $\mathcal{M}_{b,k,m(t),t} = \text{usage}_k(r_t)$. Because usages are non-negative, the per-step attribution

$$\alpha_{b,k}^{(t)} \;=\; \frac{\text{usage}_k(r_t)}{\sum_{t'=1}^{T_{k,m(t)}} \text{usage}_k(r_{t'})}$$

  (normalized within each molecule's contraction sequence) concentrates gradient on the steps that emit highly-reused rules, pulling the policy toward a small, reusable set of rules that recur across the batch.
- **Quality** $\mathcal{M}_c$ (grammar-level, scalar). We score each $r \in \rho(\mathcal{G}^{(k)})$ by the maximum Tanimoto similarity over a bank of functional-group and average over $\rho(\mathcal{G}^{(k)})$, yielding a scalar per rollout.
  - **RQ** $\mathcal{M}_d$ (grammar-level, scalar). We assign Rule Quality (RQ) scores based on $r \in \rho(\mathcal{G}^{(k)})$. If the RHS matches a functional group in the bank exactly, a score of 1 is assigned, otherwise a score of 0 is assigned. We average over $\rho(\mathcal{G}^{(k)})$ to obtain a scalar per rollout.
  - **ECM** $\mathcal{M}_e$ (per-step). Each step is evaluated to obtain the Exact Contraction Match (ECM) score. If the rule RHS fragment matches exactly with a functional group in the bank Ertl (2017), a score of 1 is applied to the step. A score of 0 is assigned if the rule RHS does not match with any functional group, and some other contraction could result in a functional group match. Steps in which no possible contraction results in a functional group match are assigned a score of None, and they are neither rewarded or penalized. The per-step attribution is normalized within each molecule's contraction sequence, analogously to the Compositionality per-step attribution.
  - **CRM** $\mathcal{M}_f$ (per-step). Each step is evaluated in context for the Coupling Reaction Match (CRM) score. This reward evaluates whether the contraction pattern is similar to known coupling reactions. The first three patterns follow from Edwards et al. (2025):
    - Suzuki-Miyaura Coupling: aromatic C - aromatic C
    - Buchwald-Hartwig Amination: N - aromatic C
    - Amide Coupling: N - carbonyl C
    - Heck Reaction: vinyl C - C
    - Sonogashira Coupling: alkyne C - C
    - Esterification Reaction: O - carbonyl C

    If a contraction introduces a split across a bond that follows any of the six patterns, a score of 1 is applied to the step. A score of 0 is assigned if the contraction does not induce such a split, but where a matching contraction is possible. Steps in which no possible contraction results in a Coupling Reaction Match are assigned a score of None, and they are neither rewarded or penalized. We assign the normalized per-step attributions within each contraction sequence.

Molecules for which the parser returns no derivation within τ_{train} are excluded from the compositionality attribution (they contribute $\alpha_{b,k}^{(t)} = 0$) but still count against parsimony. The three metrics are then weighted by $\lambda_a, \lambda_b, \lambda_c$ and fed into the GRPO loss of Sec. 3 via the combined attribution $\alpha_k^{(t)}$.

D.3 Periodic Bonuses

The three training metrics all score the grammar induced by the *current* batch. They do not test whether the policy generalizes—in particular whether the rules it produces hold up on unseen batches (d) or whether repeated rollouts on the same data agree (e). We add two periodic, non-per-rollout bonuses against a streaming released vocabulary to capture these.

Streaming vocabulary $\mathcal{V}_t$. We maintain a cumulative usage count N_r for every rule hash observed in any training rollout, updated after each training batch from the K rollout grammars. The vocabulary at step t is defined deterministically as the top-L rules by cumulative count, $\mathcal{V}_t \;:=\; \text{top-}L \text{ by } N_r$, with an optional per-LHS top-k variant. Pinning $\mathcal{V}_t$ to cumulative frequency, rather than sampling it each round from a weight estimator, keeps the released vocabulary stable across batches and lets us persist it to disk at the end of every epoch alongside the model checkpoint.

Rollout eval. Every J training batches we snapshot $\mathcal{V}_t$ and draw an auxiliary pass under `torch.no_grad()`: a random training batch $\mathcal{B}'$ of the same size is run through K fresh rollouts, yielding $\{\mathcal{G}^{(k)}, d^{\star}_{k,m}\}_{k=1}^{K}$ without gradient tracking. We compute

$$\mathcal{M}_d \;=\; \text{recall@}\mathcal{V}_t \;=\; \tfrac{1}{K}\sum_{k=1}^{K} \tfrac{|\mathcal{R}_k \cap \mathcal{V}_t|}{|\mathcal{R}_k|},$$

$$\mathcal{M}_e \;=\; \text{stability} \;=\; \tfrac{1}{|\mathcal{B}'|}\sum_{m\in\mathcal{B}'} \Big(1 - \tfrac{U_m - 1}{K-1}\Big),$$

where $\mathcal{R}_k$ is the multiset of rule hashes applied in rollout k and $U_m \in \{1, \ldots, K\}$ is the number of distinct ordered derivation sequences $(d^{\star}_{k,m})_{k=1}^{K}$ observed for molecule m. Recall equals 1 when every rule the policy produces on fresh data is already in the released vocabulary; stability equals 1 when all K rollouts produce identical derivation sequences for every molecule and 0 when they all differ.

EMA-normalised bonus. For each $p \in \{d, e\}$ we maintain running (mean, MAD) estimates (ω_p, ς_p) via an exponential moving average with decay γ, updated after every rollout eval. The resulting bonus

$$\beta_p \;=\; \lambda_p \,\frac{\mathcal{M}_p - \omega_p}{\varsigma_p + \varepsilon}$$

is added to the per-metric normalized advantage for the subsequent J training batches. Using MAD rather than $|\omega_p|$ as the normalization denominator keeps the bonus well-scaled even when ω_p crosses zero during early training.

Validation and checkpointing. When a validation set is available we repeat the rollout eval on a random val batch under the same $\mathcal{V}_t$. The EMA-normalized weighted sum of val metrics is used as our checkpoint criterion, and we train until a patience of 5 epochs is reached without improvement.

D.4 Hyperparameter Settings

Table 2 lists the default hyperparameters used for the AlphaGrammar training run reported in this paper. We randomly sample a batch of $|\mathcal{B}| = 20$ molecules per optimization step and perform $K = 5$ GRPO rollouts per batch, for a maximum of 50 epochs, 50 batches each (maximum 2,500 total batches; 12,500 rollouts).

The reward is the equal-weight combination of parsimony ($\mathcal{M}_a$, `success`) and compositionality ($\mathcal{M}_b$, `avg_rule_usage`); the other reward axes available in the codebase carry weight zero. Periodic bonuses are computed every $J = 4$ training batches against the streaming top-L vocabulary $\mathcal{V}_t$ with $L = 10{,}000$; for EMA we use decay $\gamma = 0.9$, and we weight rollout-recall (λ_d) and derivation stability (λ_e) at 1.0 and 0.5 respectively. For all non-MolQA benchmarks (MOSES, GuacaMol/SmileyChEMBL, mCLM-Pretrain-10K) we use this (a, b, d, e) default with no rule-quality oracle (c). On MolQA only, whose RQ6 evaluates synthesis planning, we additionally activate the CRM instance of (c) to encourage synthesis-friendly rules. We ablate (c) and swap out CRM for other instances of (c) in our comprehensive ablation and temperature sweep in Table 23. We use a Bolinas timeout of $\tau_{\text{train}} = 10$ seconds per molecule during in-batch metric evaluation. Optimization uses Adam with learning rate 10^{-2} and a numerical $\varepsilon = 10^{-8}$; we early-stop on five epochs without improvement in the checkpoint criterion we defined above.

E Ablations

We ablate AlphaGrammar's reward components, advantage formulation, and training-time parsing budget, and probe the coupling among rule-quality oracles.

E.1 Training Trajectory

E.2 Per-Reward Ablation

Analysis. Removing the parsimony (`success`) reward leaves parsing success essentially unchanged, consistent with the parsimony ablation below. Removing compositionality (`avg_rule_usage`)

Table 2: ALPHAGRAMMAR default training hyperparameters. Reward weights are the (a, b, d, e) default (all quality axes (c) at 0.0); MolQA runs additionally enable the CRM instance of (c). This is the only configuration deviation across benchmarks.

Group	Setting	Value
Data	batch size $\|\mathcal{B}\|$	20
	batches / epoch	50
	max epochs	50
Architecture	GNN backbone	GIN, frozen Hu et al. (2019)
	$\mathcal{F}_\theta$ hidden width	128
RL	method	GRPO
	rollouts per batch K	5
	optimizer	Adam
	learning rate	10^{-2}
	numerical ε	10^{-8}
Reward weights λ_i	parsimony $\mathcal{M}_a$ (`success`)	1.0
	compositionality $\mathcal{M}_b$ (`avg_rule_usage`)	1.0
	all other axes	0.0
	rollout-recall weight λ_d	1.0
	deriv-stability weight λ_e	0.5
Vocabulary & bonuses	estimator interval J	4 batches
	vocab size L	10,000
	EMA decay γ	0.9
Parser timeout	in-batch τ_{train}	10 s / molecule
Early stopping	patience	5 epochs

Table 3: Per-epoch training trajectory of the base ALPHAGRAMMAR checkpoints. Checkpoint criterion is $1.0 \cdot \text{recall}_{\text{val}} + 0.5 \cdot \text{stability}_{\text{val}}$; convergence occurs in 3–6 epochs with early-stopping after 5 epochs without improvement.

dataset	epoch	mean_R	success	usage	recall (tr)	recall (val)	stab. (tr)	stab. (val)
molqa	0	1.55	68.8%	2.41	68.1%	67.0%	11.6%	12.6%
molqa	1	1.80	73.5%	2.86	79.1%	79.6%	14.9%	16.5%
molqa	2	1.73	71.4%	2.74	83.6%	84.7%	17.4%	18.4%
molqa	3	1.78	75.0%	2.80	85.5%	85.3%	18.1%	18.9%
smiley	0	1.57	92.5%	2.21	32.2%	32.5%	5.1%	3.6%
smiley	1	1.59	96.2%	2.22	34.7%	36.3%	5.2%	5.6%
smiley	2	1.60	95.5%	2.25	34.0%	34.7%	5.9%	6.0%
smiley	3	1.51	92.4%	2.09	35.5%	36.0%	6.0%	4.7%
mclm	0	1.47	83.1%	2.11	29.7%	28.9%	5.0%	11.4%
mclm	1	1.48	83.3%	2.12	37.5%	37.8%	6.4%	13.9%
mclm	2	1.61	87.6%	2.34	40.8%	41.3%	8.4%	14.2%

inflates success, but only because rules nearly double in size and attain higher (easier-to-parse) uniqueness, an undesirable regime analogous to the naive reading of mCLM in RQ1. Removing rollout-recall preserves parsimony, compositionality, and quality, yet collapses recall. The full base reward maximizes recall while upholding rule quality, justifying our default.

E.3 ADVANTAGE-MODE ABLATION

Analysis. `global_flat` (basic GRPO) ≈ `global_uniform`: both reach ∼98% success, but the grammar is shaped by large "composed rules" with high diversity and low modularity (rules/parse ∼3.1, rule_size ∼15.6). Adding per-step credit refinement instead encourages smaller, reusable rules (rules/parse ∼7.5, rule_size ∼7.2) and attains higher recall (∼+40 pp). The ∼98% success of the no-per-step modes comes with collapsed recall due to large composed rules that do not generalize.

Table 4: Leave-one-reward-out ablation of the base ALPHAGRAMMAR setting. Removing `avg_rule_usage` (compositionality) inflates success but collapses recall and rule quality — the grammar degenerates into a few large composed rules.

setting	success	rule usage	rules/parse	rule_size	diversity	recall	rule quality
base (full reward)	75.0%	2.80	7.5	7.2	0.84	85.5%	0.53
− success	76.8%	2.98	7.6	7.3	0.84	88.5%	0.52
− avg_rule_usage	98.8%	2.34	3.0	15.7	0.93	40.1%	0.44
− rollout_recall	71.0%	2.67	7.5	7.2	0.83	75.0%	0.53

Table 5: Advantage-mode ablation (per notation in §3). Per-step credit (`global_step`, `step_only`) yields high recall and small reusable rules; without it (`global_flat`=basic GRPO, `global_uniform`) success is ∼98% but recall collapses via a few large composed rules. `global_step (proposed)` is the base full-reward run of Table 4.

advantage_mode	success	rule usage	rules/parse	rule_size	diversity	recall	rule quality
`global_step` (proposed)	75.0%	2.80	7.5	7.2	0.84	85.5%	0.53
`global_flat` (basic GRPO)	98.6%	2.41	3.1	15.7	0.93	42%	0.43
`global_uniform`	98%	2.36	3.1	15.6	0.93	43%	0.44
`step_only`	76%	3.6	7.6	7.1	0.85	82%	0.53

Since our checkpoint criterion prioritizes recall, this ablation justifies our proposed step-level strategy with global modulation (though `step_only` can work comparably).

E.4 PARSING BUDGET AND THE PARSIMONY REWARD

Table 6: Per-molecule Bolinas parse-success rate (τ_{train}=10s) as a function of training batch size, per corpus. At the default batch size 20, ∼13% of molecules time out on MolQA; timeouts steer training toward tractable grammars without affecting global dynamics (App. D), and disappear entirely at inference.

dataset	batch	mean	p10	p25	p50	p75	p90	mean grammar size
mclm	10	92.8%	79.0%	90.0%	100.0%	100.0%	100.0%	35.6
mclm	20	88.5%	70.0%	86.2%	95.0%	95.0%	100.0%	62.8
mclm	30	83.7%	69.7%	74.2%	83.3%	90.0%	100.0%	94.3
mclm	40	79.5%	62.5%	70.6%	80.0%	87.5%	95.0%	123.4
mclm	50	79.2%	62.0%	70.5%	82.0%	90.0%	94.0%	149.8
molqa	10	76.1%	59.0%	70.0%	80.0%	90.0%	91.0%	60.3
molqa	20	74.6%	60.0%	67.5%	75.0%	80.0%	85.2%	100.2
molqa	30	70.5%	50.0%	66.7%	73.3%	78.1%	83.3%	134.3
molqa	40	65.6%	49.9%	57.2%	69.2%	76.8%	79.5%	167.2
molqa	50	60.3%	36.7%	52.6%	62.0%	70.0%	78.2%	195.6
smiley	10	97.0%	90.0%	100.0%	100.0%	100.0%	100.0%	31.4
smiley	20	96.7%	93.3%	95.0%	100.0%	100.0%	100.0%	57.2
smiley	30	95.8%	88.7%	93.2%	96.7%	100.0%	100.0%	79.7
smiley	40	95.8%	91.7%	94.7%	97.3%	99.4%	100.0%	110.5
smiley	50	96.0%	92.0%	94.0%	96.0%	98.0%	100.0%	133.4

We further ablate the parsimony reward entirely (`AG-parsimony`), removing training's dependence on the Bolinas timeout τ. Tokenizer properties:

Downstream GRAMOLE performance:

Usage and coverage are virtually unchanged when parsimony is ablated, and uniqueness and rule quality are, if anything, slightly higher without parsimony (Uniq@100 0.412–0.537 → 0.488–0.866; Qual@100 0.533 → 0.549). Removing the parsimony reward—and with it training's dependence on

Table 7: Tokenizer properties with the parsimony reward ablated.

Method	Use@100	Cov@100	Uniq@100	Qual@100
AG (Table 6)	0.074–0.199	0.080	0.412–0.537	0.533
AG-parsimony	0.072–0.193	0.071	0.488–0.866	0.549

Table 8: Downstream GRAMOLE with the base grammar vs. the base grammar trained without parsimony (`AG-parsimony`).

Gramole Setting	Structure (↑)		Text (↑)		Drug (BA ↑)			Drug (RS ↑)		Material (MAE ↓)						Material (RS ↑)
	Validity	Similarity	BLEU-4	ROUGE-L	HIV	BBBP	BACE	Avg. BA	Success %	COPerm	NPerm	OPerm	FFV	TC	Avg. MAE	Success %
Qwen3-8B (Base AG)	0.999	0.117	0.119	0.258	0.529	0.515	0.513	0.519	26.1	1.164	1.097	0.924	0.028	0.077	0.658	13.3
Qwen3-8B (Base AG-parsimony)	0.993	0.113	0.171	0.327	0.460	0.529	0.492	0.494	33.7	1.107	1.062	0.938	0.027	0.072	0.641	49.3

the Bolinas parser and τ—leaves the learned grammar's tokenizer properties intact: the tractability reward is not load-bearing for quality.

E.5 RULE-QUALITY ORACLE TRANSFER

Table 9: Each rule-quality variant (base/ECM/CRM/RQ) scored against all three reward oracles. Reinforcing functional-group alignment (RQ/ECM) also enriches reaction match (CRM) and vice-versa; all three variants exceed base on vocabulary quality.

method	avg_rule_quality	exact_contraction_match	coupling_rxn_match
base	0.304±0.044	0.650±0.045	0.498±0.070
ecm	0.352±0.036	**0.810±0.058**	0.622±0.077
crm	0.357±0.036	0.703±0.053	**0.665±0.086**
rq	**0.385±0.029**	0.845±0.068	0.670±0.093

Improvements across the RQ, ECM, and CRM oracles couple: reinforcing functional-group alignment also enriches reaction match and vice-versa, and all three exceed the base grammar on vocabulary quality, hence motivating a rule-quality oracle as the MolQA default.

E.6 CANONICAL-PARSE DETERMINISM

Caching a seed-deterministic canonical parse and a greedy (argmax) canonical parse yield negligible downstream differences, confirming that caching *a* canonical parse is what matters, not *which* one (§3).

F DETAILS OF GRAMOLE INFERENCE

F.1 GENERATING VALID MOLECULES VIA TYPE CHAINING

GRAMOLE uses type chaining to ensure rule sequences under a length cap decode to valid molecules. In Def. A.3, each rule ρ has an $(\mathrm{intype}(\rho), \mathrm{outtype}(\rho))$. At inference time, only rule sequences $[\rho_1, \rho_2, \ldots, \rho_K]$ that *type-chain*, i.e. $\mathrm{outtype}(\rho_i) = \mathrm{intype}(\rho_{i+1})$, are permitted. When max rule sequence length, K, is capped (default is 20), we must check for *closure*, i.e. reachability of a terminal rule, satisfying $\mathrm{outtype}(\rho_k) = \perp$. We precompute, for every type, the minimal number of hops to closure via reverse BFS over the type graph. Let $\mu(t)$ denote the smallest k such that some chain $[\rho_1, \ldots, \rho_k]$ with $\mathrm{intype}(\rho_1) = t$ satisfies $\mathrm{outtype}(\rho_k) = \perp$. The BFS proceeds in layers seeded from terminal rules: layer 1 contains every type t admitting at least one terminal rule with $\mathrm{intype} = t$; layer $k+1$ contains every type from which some non-terminal rule transitions into a layer-k type. Each type is finalized at first discovery, so cycles in the type graph are handled implicitly, and types absent from all layers are *dead-ends* from which no closing chain exists. At decoding step i with pending intype $t = \mathrm{outtype}(\rho_{i-1})$ and $K - i + 1$ slots remaining, we admit a rule ρ iff $\mathrm{intype}(\rho) = t$ and either $\mathrm{outtype}(\rho) = \perp$ (closes here) or $\mu(\mathrm{outtype}(\rho)) \leq K - i$ (the successor can still close within the remaining budget). At the final position $i = K$ the second clause

Table 10: Canonical-parse ablation: caching a seed-deterministic parse (CRM) vs. a greedy argmax-contraction parse (CRM-greedy). Differences are negligible — caching *a* canonical parse is what matters, not which one — supporting the deployment-determinism claim (§3).

GRAMOLE Setting	Structure (↑)		Text (↑)		Drug (BA ↑)				Drug RS	Material (MAE ↓)						Mat. RS
	Valid.	Sim.	BLEU-4	ROUGE-L	HIV	BBBP	BACE	Avg. BA	Succ.%	CO_2	N_2	O_2	FFV	TC	Avg. MAE	Succ.%
Qwen3-8B (CRM)	1.000	0.123	0.220	0.392	0.539	0.495	0.514	0.516	34.6	1.143	1.044	0.898	0.031	0.077	0.639	82.7
Qwen3-8B (CRM-greedy)	1.000	0.118	0.217	0.390	0.497	0.555	0.501	0.517	37.0	1.110	1.024	0.906	0.028	0.091	0.632	73.7

forces $\mu(\mathrm{outtype}(\rho)) \leq 0$, which only terminal rules satisfy, so closure is guaranteed. Logits over disallowed rules are masked to $-\infty$ before sampling, so every decoded sequence is by construction both type-chained and closed within the cap.

F.2 Generating Valid Polymers via Star Counting

For polymer generation, we further impose exactly two $*$ symbols in the output. Each rule ρ contributes a non-negative count $\sigma(\rho) \in \mathbb{N}$ of $*$ atoms (the dangling attachment points marking the polymer's repeat-unit boundaries), and we fix the target $\sigma^\star = 2$. We lift the closure table to a joint state (t, s) pairing the pending intype t with the remaining star budget s, and redefine $\mu(t, s)$ as the smallest k such that some chain starting from intype t closes via a terminal rule while contributing *exactly* s stars. The BFS seeding generalizes: layer 1 contains every (t, s) admitting a terminal rule ρ with $\mathrm{intype}(\rho) = t$ and $\sigma(\rho) = s$; layer $k + 1$ contains every (t, s) from which some non-terminal ρ with $\mathrm{intype}(\rho) = t$, $\sigma(\rho) = d \leq s$ transitions into a layer-k state $(\mathrm{outtype}(\rho),\ s - d)$. States absent from all layers are infeasible — either overshooting the target or unable to hit it on time. At decoding step i with running star count $\sigma_i = \sum_{j<i} \sigma(\rho_j)$ and $K - i + 1$ slots remaining, we admit ρ iff $\mathrm{intype}(\rho) = t$ and (i) ρ is terminal with $\sigma_i + \sigma(\rho) = \sigma^\star$, or (ii) ρ is non-terminal with $\sigma_i + \sigma(\rho) \leq \sigma^\star$ and $\mu(\mathrm{outtype}(\rho),\ \sigma^\star - \sigma_i - \sigma(\rho)) \leq K - i$. Clause (i) forbids both overshoot and undershoot at termination; clause (ii) prunes any continuation that cannot reach the target within the remaining budget. As before, logits over disallowed rules are masked to $-\infty$, so every decoded polymer is simultaneously type-chained, length-capped, and contains exactly two $*$ atoms by construction.

F.3 Inference Complexity

We analyze the overhead added by type chaining and star counting on top of standard autoregressive decoding. Let $|R|$ denote the number of rules in the grammar, $|T|$ the number of distinct types appearing as intype or $\mathrm{outtype}$ of any rule, V the language model's vocabulary size, K the maximum rule-sequence length, and $\sigma^\star$ the target star count (taken as a constant, $\sigma^\star = 2$ for polymers; for the plain type-chaining setting we treat $\sigma^\star = 0$ throughout).

One-off precomputation. At model load time we build (i) a forward index $\mathrm{intype} \to \{\text{rules with this } \mathrm{intype}\}$ and (ii) a reverse index $\mathrm{outtype} \to \{\text{rules with this } \mathrm{outtype}\}$, both in $O(|R|)$ time and space. The reverse-BFS that fills the closure table μ then runs as follows. We maintain a frontier of newly-discovered states. Initialization scans every rule once, seeding $\mu(\mathrm{intype}(\rho), \sigma(\rho)) \leftarrow 1$ for each terminal ρ in $O(|R|)$. Propagation then pops each frontier state (t', s') and, using the reverse index, examines each rule ρ with $\mathrm{outtype}(\rho) = t'$; if $\sigma(\rho) + s' \leq \sigma^\star$ and $(\mathrm{intype}(\rho), s' + \sigma(\rho))$ is undiscovered, it is added to the next frontier with $\mu \leftarrow \mu(t', s') + 1$. Each rule ρ is examined at most $(\sigma^\star + 1)$ times across the entire BFS — once per value s' for which $(\mathrm{outtype}(\rho), s')$ is reachable — and each examination is $O(1)$. The frontier itself contains at most $|T| \cdot (\sigma^\star + 1)$ states, each popped at most once. The total precomputation cost is therefore

$$T_{\text{pre}} = O\big((|R| + |T|) \cdot (\sigma^\star + 1)\big),$$

with table size $|\mu| = O(|T| \cdot (\sigma^\star + 1))$. For our setting ($|R| = 9{,}937$, $|T| = 60$, $\sigma^\star \in \{0, 2\}$) this is a few tens of milliseconds and is amortized across all subsequent inference calls.

Per-step decoding overhead. At decoding step i with pending intype t and running star count σ_i, the logits processor (a) enumerates the at most $|R_t|$ rules with $\mathrm{intype} = t$ via the forward index in $O(|R_t|)$, (b) performs an $O(1)$ lookup in μ for each candidate's successor state, and (c) writes $-\infty$ into the V-dimensional logits vector for all positions outside the admitted set, in $O(V)$. The per-step overhead is therefore

$$T_{\text{step}} = O\big(|R_t| + V\big) = O\big(|R| + V\big)$$

in the worst case. Summing over the at most K rule positions in a single design block,

$$T_{\text{infer}}^{\text{grammar}} = O\big(K \cdot (|R| + V)\big).$$

Comparison with the language model cost. The autoregressive language model itself contributes $O(K \cdot L \cdot d^2)$ time to the design block, where L is the prefill plus generation length and d is the model's hidden dimension. With $K = 20$, $V \approx 1.6 \times 10^5$, $|R| \approx 10^4$, $L \in [10^2, 10^3]$, and $d = 4096$, the grammar overhead $O(K(|R| + V)) \sim 3 \times 10^6$ operations is negligible compared with the $O(KLd^2) \sim 10^{11}$ operations of the language model itself, contributing well under 0.1% of total wall-clock time. Both type chaining and star counting therefore impose no asymptotic burden on inference and a constant-factor overhead bounded above by $V/d^2 \approx 10^{-5}$ per generated rule.

Correctness within the budget. By induction on the remaining slots $K - i + 1$, every state (t, s) admitted at step i has $\mu(t, s) \leq K - i + 1$, so a closing chain of length at most K exists from the current frontier; in particular at $i = K$ only terminal rules satisfying $\sigma(\rho) = \sigma^\star - \sigma_K$ remain admissible. Thus closure within the cap, with the exact star budget when active, is guaranteed by construction.

F.4 Rule Vocabulary

GRAMOLE's rule vocabulary is the enumerated atomic rule set (Theorem 1; $50{,}082{,}289$ rules for our reasonableness constants) together with the composed rules the grammar learns. Every atomic rule is a member of the vocabulary, so any reasonable molecule tokenizes in-vocabulary (Corollary 1); we verified that every rule required to tokenize our benchmark corpora is present in this set. Embeddings are learned as rules are used during training, so the materialized embedding table scales with the rules actually encountered rather than the full dictionary.

G Compute Cost

SmileyLlama Benchmark. Training was a single epoch of LoRA SFT (rank 32, α=16, dropout 0.05, with the embedding and language-model-head rows added to the LoRA save set so the new rule-token rows train end-to-end), at effective batch size 64 (micro-batch 4, gradient accumulation 4, across 4 ranks), AdamW-8bit with peak learning rate 2×10^{-4} on a cosine schedule with 10 warmup steps, sequence length 2048 with sample packing. Wall-clock training took 6 hours and 10 minutes on 4x NVIDIA H100 80GB GPUs.

mCLM Benchmark. Training was 2 epochs of full-parameter SFT over the `mCLM_Pretrain_1k` corpus, at effective batch size 32 (per-device micro-batch 1, gradient accumulation 8, across 4 ranks), AdamW with peak learning rate 2×10^{-5} and warmup ratio 0.03, max gradient norm 1.0, sequence length 1024. We train in fp32 master-weight precision with TF32 matmuls and gradient checkpointing enabled. We found that bf16 caused gradient overflows at the freshly initialized rule-token embedding rows. In total, we trained for 96,258 optimizer steps. Wall-clock training took 49 hours and 45 minutes on 4x NVIDIA H100 80GB GPUs.

Llamole Benchmark. Training was four epochs of LoRA SFT (rank 8, α=16, dropout 0.0, applied to all attention and MLP projections, with `embed_tokens` and `lm_head` added to `modules_to_save` so the resized rows for the 9,937 added rule tokens train end-to-end), at effective batch size 12 (micro-batch 6, gradient accumulation 2, on a single rank), AdamW with peak learning rate 1×10^{-4} on a cosine schedule with 10% warmup, sequence length 2048 in pure bfloat16. The joint loss combined language-modelling cross-entropy and the frozen-graph-predictor retrosynthesis cross-entropy with equal weights. Wall-clock training took 14 hours and 20 minutes (38,464 optimizer steps) on a single NVIDIA H100 80GB GPU.

H Computational Complexity

We analyze the complexity of AlphaGrammar training and inference. Parser complexity is treated separately in App. C. GRAMOLE's inference complexity is discussed in App. F.

Three options for the per-step $R^\star$ feature. A central design decision is how to compute the cluster feature $f(\eta)$ that drives the potential $\phi_\eta(\theta)$ at each step. Because $R^\star$ absorbs terminal atoms across the rollout, its feature must change as contractions accumulate. We considered three options.

(O1) Aggregate cached atom features into the $R^\star$ singleton (cheapest). Run the GNN once on H_M and, at each step, pool the features of the original terminal atoms that have been absorbed into $R^\star$. This is cheap (one GNN forward, $O(|R^\star|)$ aggregation per step) but *trajectory-dependent*: two contraction sequences that arrive at the same hypergraph fragment can produce different $e_{R^\star}$, because the aggregate is taken over original atom indices rather than the current fragment's topology. This breaks the Markov assumption underlying the context-free grammar, where the intermediate state should depend only on the current fragment.

(O2) Re-run a fragment-aware GNN every step (ideal). Promote $R^\star$ to a first-class atom type, pretrain a GNN over hypergraph fragments that respects this type, and re-run it on the contracted hypergraph at every step. This restores the Markov property: $e_{R^\star}$ becomes a function of the current fragment alone. However, this pays a full GNN forward at every step, $O(|A_M| + |B_M|) = O(N)$, for a per-rollout cost of $O(N\,T)$ that is wall-clock-dominated by the GNN call (confirmed by our profiling). It also requires pretraining a fresh fragment GNN, which is costly in its own right. We did not pursue this route.

(O3) Cache GNN features once, project to each fragment (ours). We run the GNN once on the original terminal-only hypergraph H_M at the start of the rollout, in $O(|A_M| + |B_M|)$, and at every contraction step *project* these cached features onto the current fragment to obtain $f(\eta)$ for every η in the current junction tree. The projection reflects the fragment's actual topology while bootstrapping off the high-quality pretrained extractor: $e_{R^\star}$ depends on *which* atoms are currently inside $R^\star$ via the projection, not on the order in which they entered, so trajectory-dependence is removed up to the choice of projection. Per-step projection cost is $O(\tilde{N}_t)$, where $\tilde{N}_t$ is the number of unabsorbed clusters at step t. Compared with (O2), (O3) trades the $O(N)$-per-step GNN call (which dominates training in practice) for $O(\tilde{N}_t)$-per-step projection.

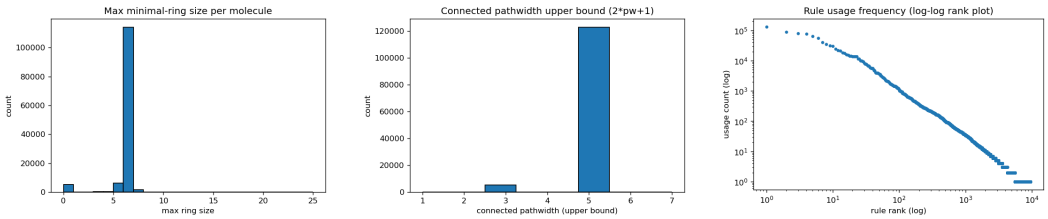
Rollout cost. Under (O3), a rollout has three components:

- **Initial GNN pass** on H_M plus cluster pooling produces $f(\eta)$ for every $\eta \in T_M$, in $O(|A_M| + |B_M|) = O(N)$, performed once.
- **Per-step attention projection.** At step t, the projection, rooted-tree masked attention, the upward DP $A[v] = \exp(u[v]) \prod_c (1 + A[c])$, and top-down constrained-Bernoulli sampling each cost $O(\tilde{N}_t)$. The attention is linear in $\tilde{N}_t$ rather than $\tilde{N}_t^2$ because the ancestor-only mask matches our tree graphical model, an optimization that aligns the neural and probabilistic structures.
- **Bookkeeping.** The union–find contraction, watershed update, and rule emission at step t touch only the newly absorbed region; amortized over the rollout each atom and bond is visited $O(1)$ times, contributing $O(N)$ total.

Summing the per-step neural work, one rollout costs $O\big(N + \sum_{t=1}^{T} \tilde{N}_t\big)$. In the worst case (one cluster absorbed per step, $T = N$, $\tilde{N}_t = N - t$) this is $O(N^2)$. In practice the "at least one" Bernoulli pass typically commits many children of $R^\star$ in a single step, so the realized contraction depth T_{eff} is much smaller than N and the rollout cost is $O(N\,T_{\text{eff}})$ with $T_{\text{eff}} \ll N$ on the molecules in our datasets (App. I). The spanning-tree fallback for non-reasonable molecules (App. B) adds a one-off BFS per step and does not change the asymptotic cost.

Training. A training batch takes K rollouts ($K = \texttt{MCMC_size}$) over $|\mathcal{B}|$ molecules, so the sampling pass costs $O(K|\mathcal{B}|N\,T_{\text{eff}})$. We parallelize the contraction steps across a pool of workers. The main process then replays each rollout's per-step projection, attention, and DP to accumulate $\log p_\theta$ with gradient tracking, adding another $O(K|\mathcal{B}|N\,T_{\text{eff}})$ for the forward replay and the same for back-propagation through the (fixed-width) potential network. GRPO normalization and the per-step attribution loop of Sec. 3 touch each step once more. The ALPHAGRAMMAR-specific training cost is independent of the grammar size $|\mathcal{G}|$. Periodic bonuses $(\mathcal{M}_d, \mathcal{M}_e)$ run one additional rollout pass every J batches, amortized to $O(K|\mathcal{B}|N\,T_{\text{eff}}/J)$ per batch and is not a dominant term for $J > 1$. Parser calls used for in-batch metrics $(\mathcal{M}_a, \mathcal{M}_b)$ are independent of ALPHAGRAMMAR and analyzed in App. C.

Inference. Tokenizing a molecule M at inference runs exactly one rollout of the trained policy, costing $O(N_M\,T_{\text{eff}}(M))$ for the per-step neural work plus $O(N_M)$ for the initial GNN call and bookkeeping. Batch tokenization of a dataset $\mathcal{D}$ is therefore $O\big(\sum_{M\in\mathcal{D}} N_M\,T_{\text{eff}}(M)\big)$. No parser is invoked and the cost does not depend on $|\mathcal{G}|$.



(a) Max SSSR ring size per molecule.

(b) Connected pathwidth upper bound $2\,\mathrm{pw}+1$.

(c) Rule-usage log-log rank plot.

Figure 8: Empirical properties of the learned grammar.

I Empirical Analysis of Theoretical Assumptions

We measure whether the four assumptions underpinning AlphaGrammar's design hold on the MolQA corpus (128,140 molecules). (A1) molecules are *reasonable* in the sense of Def. A.7, (A2) the default SSSR+singleton clustering induces an acyclic junction tree, (A3) the enumerated atomic-rule set of Theorem 1 tightly covers the rules AlphaGrammar actually learns, and (A4) rule usage is heavy-tailed enough to warrant the compositionality bonus in our RL objective.

I.1 A1: Reasonableness is satisfied for >98% of the dataset

Def. A.7 requires bounded valence (V=6), bounded minimal ring size (R), and bounded connected pathwidth (C). Valence is trivially satisfied by the chemistry of the dataset. For ring size, only $1835/128{,}140$ (1.4%) of molecules contain a minimal ring larger than R=6, 0.2% larger than 7, and 0.1% larger than 8 (Fig. 8a). For connected pathwidth, our min-degree elimination + vertex-separation heuristic (an upper bound on pathwidth) yields $\mathrm{pw} \le 3$ for every molecule, with mean 1.96. Plugging into Dereniowski's bound (Dereniowski, 2012), $\mathrm{cpw}(G) \le 2\,\mathrm{pw}(G)+1$ gives $\mathrm{cpw} \le 7$ for *every* molecule in MolQA, $\mathrm{cpw} \le 5$ for $\sim 99.95\%$, and $\mathrm{cpw} \le 3$ for 4.1% (Fig. 8b). Taking $V = 6$ and C=7, we conclude that at least $\sim 98.6\%$ of MolQA satisfies Def. A.7; the heuristic upper bound likely emits more reasonable molecules.

I.2 A2: The default junction-tree initialization is almost always acyclic

The raw SSSR+singleton clustering already yields an acyclic connectivity graph for $126{,}312/128{,}140$ (98.6%) of molecules; after the single union–find cycle-fusion pass that `find_edges` (Sec. A) performs by default, 100% of the corpus has a valid junction tree. This validates our choice to initialize bags with minimal rings and non-ring singletons (App. A lines 368–372): the heuristic is tight for nearly all molecules and provably recoverable via one pass of cycle fusion for the rest.

I.3 A3: The learned grammar is a *tiny* fraction of the theoretical atomic set.

Theorem 1 with E=R=6 evaluates to the upper bound 3.03×10^{11}; enumerating the atomic rule set for these constants gives 50,082,289 rules (Remark 4), which Gramole embeds as its rule vocabulary (App. F.4). The vocabulary is complete over reasonable molecules, whereas the learned grammar *exercises* only a small subset on any given corpus (e.g. 9,635 atomic rules on MolQA (Table 11)) reflecting the sharply Zipfian usage of rules. The atomic rule support breakdown matches the theory:

- Category A (single atom): 497 rules; bound 48,384; ratio $\approx 97\times$.
- Category B (R^* external): 7,059 rules; bound 2.14×10^{11}.
- Category C (R^* inside ring): 2,034 rules; bound 8.93×10^{10}.

Per ring size, the support grows roughly proportionally with the bound only until r=5; for r=6 the support is 3,825 out of a theoretical 2.82×10^{11}, confirming that the $g(1)^r$ worst-case scaling identified in App. G dominates the looseness at large ring sizes while the chemistry of real molecules

is a minuscule subset. At the finest granularity, the $9{,}635$ rules fall into only 60 distinct type signatures (multisets of external bond types); the ten most frequent signatures cover 83% of the vocabulary and $\{1,1\},\{1\},\{1,1,1\}$ alone account for 42% (Tab. 13). The effective branching factor during contraction is therefore governed by a handful of dominant types, not by the raw vocabulary size.

Table 11: Rules the learned grammar *exercises* per corpus—a usage statistic, distinct from the rule vocabulary (the full atomic dictionary, App. F.4). |learned| = composed rules ALPHAGRAMMAR learns; |atomic| = atomic rules used on the corpus; ∩/∪ = their intersection/union. The previously reported $9{,}937$ (MOSES) and $19{,}026$ (mCLM) counts are *unions* of these usage counts, not vocabulary sizes.

dataset	|learned|	|atomic|	∩	∪
moses	2,074	9,635	1,772	9,937
mclm	18,622	694	290	19,026
molqa (base)	4,754	9,635	1,840	12,549
molqa (ecm)	2,282	9,635	2,281	9,636
molqa (rq)	2,343	9,635	2,337	9,641
molqa (crm)	2,409	9,635	2,403	9,641
smiley	28,798	102,453	1,509	129,742

The previously reported $9{,}937$ (MOSES) and $19{,}026$ (mCLM) counts are *unions* of the composed and atomic rules *exercised* on each corpus—usage statistics, not the vocabulary, which is the full atomic dictionary. Learning larger composed rules is not universally beneficial: ablating compositionality (encouraging only large rules) inflates parse success but hurts usage, recall, and quality (Table 4, `−avg_rule_usage`). Rule usage is sharply Zipfian in both the learned and atomic vocabularies (Table 12).

Table 12: Zipf (power-law) fits of rule usage vs. rank (> 0.9 R^2 in log-usage vs. log-rank), for learned and atomic vocabularies per corpus.

folder	vocab	#rules	α	R^2
moses	learned	2,074	1.399	0.963
moses	atomic	9,635	1.394	0.968
mclm	learned	18,622	0.681	0.919
mclm	atomic	694	1.104	0.684
molqa	learned	4,754	1.232	0.946
molqa	atomic	9,635	1.394	0.968
smiley	learned	28,798	0.526	0.849
smiley	atomic	102,453	1.546	0.972

I.4 A4: RULE USAGE IS HEAVY-TAILED, MOTIVATING COMPOSITIONALITY

Across the $127{,}768$ parsed derivations, $9{,}537$ of the atomic rules are exercised at least once. The usage distribution is strikingly Zipfian (Fig. 8c): the top-1 rule accounts for 10.1% of all rule applications, and the top-10 for 48.4%. This long-tailed structure is the empirical justification for the compositionality bonus in our objective: rules are recurrent enough that amortizing RL credit across derivations yields a useful signal, yet diverse enough (60 type signatures, 968 distinct ring-system topologies) that vocabulary coverage is non-trivial.

Table 13: Support over distinct rule type signatures.

bond multiset	1,1	1	1,1,1	1,2	1,1,2	1,2,2	2	2,2	1,1,1,1	1,1,2,2	1,1,1,2	()	2,2,2	1,2,2,2	2,2,2,2	1,1,1,1,1	1,1,1,2,2	1,1,2,2,2	1,1,1,1,2	1,2,2,2,2
rules	1,600	1,372	1,107	962	759	606	544	524	421	358	276	267	204	166	89	85	77	53	44	21
support	16.6%	14.2%	11.5%	10.0%	7.9%	6.3%	5.6%	5.4%	4.4%	3.7%	2.9%	2.8%	2.1%	1.7%	0.9%	0.9%	0.8%	0.6%	0.5%	0.2%

I.5 TAKEAWAYS

The combined picture is that reasonableness is an essentially harmless assumption on MOLQA (A1), the junction-tree heuristic works out of the box (A2), and the theoretical rule bound, although

conservative by $\sim 10^7\times$ due to worst-case free-valence assumptions and absence of chemical-validity pruning, is nevertheless a valid *finite* superset over which ALPHAGRAMMAR's learned grammar is a well-distributed $\mathcal{D}$-support (A3–A4). Together these results confirm that Theorem 1 (universal coverage) and Corollary 1 (no OOV for reasonable molecules) are not merely theoretical safeguards but bite on real-world chemistry datasets.

J TOKENIZER PROPERTIES

J.1 DEFINITIONS

Fix a corpus $\mathcal{D}$ of N molecules, a tokenizer T with vocabulary V_T, and a top-k shortlist $V_T^{(k)} \subseteq V_T$ ranked by corpus token frequency. For each molecule m, let $T_0(m)$ denote a single canonical tokenization and let $\mathcal{P}_T(m)$ denote the set of *distinct synonymous parses* of m (each a token sequence; specified per-baseline in J.3). Sampling for the parse-dependent metrics uses a random subset $\mathcal{S}$ of $\mathcal{D}$ restricted to molecules covered by $V_T^{(k)}$; see Section J.5 for handling timed-out parses.

Usage@k For each $t \in V_T^{(k)}$, let $f_t(m) = |\{\, p \in \mathcal{P}_T(m) : t \in p \,\}| \,/\, |\mathcal{P}_T(m)|$ be the fraction of m's parses containing t. Average across $\mathcal{S}$, then across the top-k:

$$\mathrm{Use}_T@k \;=\; \tfrac{1}{k} \sum_{t \in V_T^{(k)}} \tfrac{1}{|\mathcal{S}|} \sum_{m \in \mathcal{S}} f_t(m) \;\in [0,1].$$

We additionally report the histogram of the per-token averages $\frac{1}{|\mathcal{S}|}\sum_m f_t(m)$ over $V_T^{(k)}$.

Uniqueness@k Mean inverse parse count over the sample:

$$\mathrm{Uniq}_T@k \;=\; \tfrac{1}{|\mathcal{S}|} \sum_{m \in \mathcal{S}} \tfrac{1}{|\mathcal{P}_T(m)|} \;\in (0,1].$$

Higher is closer to a unique representation. In addition to $\mathrm{Uniq}_T@k$ we report the mean parse count $\mathbb{E}|\mathcal{P}_T(m)|$ and its distribution.

Quality@k For tokenizers whose tokens carry molecular structure, define $q(t)$ as the maximum Tanimoto similarity (Morgan-3 fingerprint, 2048 bits) of t (after stripping any dummy attachment atoms) against a fixed reference set of 768 functional groups (Ertl (2017)); then

$$\mathrm{Qual}_T@k \;=\; \tfrac{1}{k} \sum_{t \in V_T^{(k)}} q(t).$$

For tokenizers whose tokens are not chemically meaningful, this metric is N/A.

J.2 PARSE ESTIMATION FOR INTRACTABLE TOKENIZERS

For most of the tokenizers below, the population $\mathcal{P}_T(m)$ is far too large to enumerate exhaustively: even plain SMILES admits exponentially many synonymous strings per molecule, and the count grows factorially in the presence of rings (each ring breaks at any of its bonds, multiplying the number of valid spanning-tree DFS traversals). Existing SMILES/SELFIES data-augmentation packages are *sampling* routines. For SMILES, prior works like Bjerrum (2017) use RDKit's atom-renumbering randomization. For SELFIES, we adopt the TYCHE spanning-tree shuffling algorithm Nigam et al. (2026). They produce one synonymous string at a time, with no built-in support to report the population size.

To convert sampling into a count, we use a coupon-collector species-richness estimator. After drawing k randomized parses and observing u distinct ones, we ask: what total N would best explain the data? Modeling each draw as i.i.d. uniform with replacement, $\mathbb{E}[u] = N\big(1 - e^{-k/N}\big)$, which we invert numerically (Brent's method) for $\hat{N}(m)$. Intuitively, if nearly every new draw adds a previously-unseen parse we have barely scratched a large pool ($\hat{N} \gg u$); if most new draws repeat ones already seen we have nearly enumerated it ($\hat{N} \approx u$).

The estimator is *consistent* — it converges to the true $|\mathcal{P}_T(m)|$ as $k \to \infty$ — but *not unbiased* at finite k, and it strictly assumes uniform sampling. TYCHE's randomization is non-uniform over distinct parses (common synonymous strings are produced more often than rare ones), so $\hat{N}$ typically *underestimates* $|\mathcal{P}_T(m)|$ and Uniqueness$_T$@k is correspondingly biased high. ALPHAGRAMMAR (exact Bolinas parse-tree count) and BRICS (closed-form multinomial) provide unbiased reference points against which the bias of the sampled tokenizers can be calibrated. Under exact enumeration ($\tau = 600$s), timeouts are frequent for the non-ALPHAGRAMMAR tokenizers—SMILES and SELFIES (char and BPE) 99.7%, GroupSELFIES and mCLM 100%, BRICS 24.0%—whereas ALPHAGRAMMAR times out on only 3.1%.

J.3 PARSE ENUMERATION PER BASELINE

The set $\mathcal{P}_T(m)$ — the synonymous parses we enumerate or estimate — is implemented as follows.

- **SMILES (char)** (Weininger (1988)): distinct tokenizations of TYCHE-randomized Nigam et al. (2026) SMILES of m; tokens are characters.
- **SMILES (BPE)** (Chithrananda et al. (2020)): same string source, tokenized with the pretrained ChemBERTa BPE.
- **SELFIES (char)** (Krenn et al. (2020)): TYCHE-randomized SMILES, each piped through `selfies.encoder`; tokens are characters of the SELFIES string.
- **SELFIES (BPE)** (Yüksel et al. (2023)): same encoding, tokenized with SELFormer's pretrained BPE.
- **GroupSELFIES** (Cheng et al. (2023)): TYCHE-randomized SMILES, each encoded via a fixed group grammar checkpoint (most useful 30 fragments merged with the essential chirality set, $|V| = 53$); tokens are bracketed units (`[C]`, `[Ring1]`, `[:0frag12]`,...).
- **BRICS** (Degen et al. (2008)): $\mathcal{P}_T(m)$ is the set of distinct serial orderings of the BRICS fragment multiset of m (obtained via `BRICS.BreakBRICSBonds` + `Chem.GetMolFrags`). The cardinality is the closed-form multinomial $n!/\prod_i m_i!$, since BRICS fragment composition is order-independent. Tokens are fragment SMILES carrying the BRICS attachment-point dummies (`[1*]`, `[3*]`,...).
- **mCLM** (Edwards et al. (2025)): TYCHE-randomized SMILES, each passed through their get_blocks function to obtain forward and reverse block strings; both contribute to $\mathcal{P}_T(m)$. Tokens are individual blocks (split on `.` and `^`).
- **ALPHAGRAMMAR (ours)**: $\mathcal{P}_T(m)$ is the set of distinct parse trees (rule sequences) produced by the Bolinas parser on a grammar instance restricted to the rules in $V_T^{(k)}$. Tokens are integer rule rank-IDs from the precomputed parse cache.

J.4 RULE QUALITY PER FRAGMENT-BASED BASELINE

To measure similarity between formal rules and functional groups (molecular structures), we add the following conversion logic for the fragment-based baselines and ALPHAGRAMMAR:

- **BRICS** and **mCLM** tokens are RDKit-parsed, dummy-stripped, sanitized, and scored as above definition.
- **ALPHAGRAMMAR (ours)** rules are scored by having the RHS hypergraph is decomposed (nonterminal edges removed, attachment nodes stubbed with carbons, components mapped to fragments).
- **SMILES**, **SELFIES**, and **Group-SELFIES** tokens have no canonical molecular interpretation and are scored N/A.

J.5 WORST/BEST BOUNDS FOR SAMPLED PARSES

For ALPHAGRAMMAR at large k the Bolinas parser can exceed a wall-clock budget (we use a 120 s limit per molecule). A timed-out molecule contributes an unknown value in $[0, 1]$ to Usage and Uniqueness. We report two bounds: worst $= \frac{S+0\cdot t}{n_{\text{ok}}+t}$, best $= \frac{S+1\cdot t}{n_{\text{ok}}+t}$, where S is the contribution of successfully-parsed mols, n_{ok} the count of those, and t the number of timed-out mols. The two coincide when no mols time out. Per-token Usage is bounded analogously per-token before averaging across $V_T^{(k)}$. Coverage and Quality are deterministic and unbounded.

K Interpretability Evaluation

To evaluate whether the trained potential function has learned chemically meaningful contraction preferences, we run a full-enumeration rollout over the molecules in the MOSES test dataset. At every contraction step t for a molecule m, rather than sampling a single action, we enumerate the complete set of valid action lists $\mathcal{A}_m^{(t)} = \{a_1, a_2, \ldots, a_{N_t}\}$ and record the potential function's probability $\pi_\theta(a_i \mid s_t)$ for each candidate, along with the production rule $\rho(a_i)$ it would emit. The MCMC chain still advances by selecting one action per step (sampled proportionally to π_θ).

Each production rule ρ is scored by the RQ oracle, which returns 1 if the contracted fragment exactly matches a functional group in the bank (Ertl (2017)). Let $\mathcal{E}_m^{(t)} \subseteq \mathcal{A}_m^{(t)}$ denote the set of exact-matching actions at step t. Actions at each step are sorted in descending order of $\pi_\theta(a_i \mid s_t)$, assigning rank $r_i \in \{1, \ldots, N_t\}$ to action a_i (rank 1 = highest probability). We compute the following metrics over the exact-matching actions.

K.1 Metric Definitions

Average percentile. For each exact action at rank r, its percentile is

$$\text{pct}(r) = 1 - \frac{r-1}{N_t - 1}, \tag{5}$$

normalized to $[0, 1]$ with 1 indicating rank 1. The step-level average percentile is

$$\overline{\text{pct}}^{(t)} = \frac{1}{|\mathcal{E}_m^{(t)}|} \sum_{a_i \in \mathcal{E}_m^{(t)}} \text{pct}(r_i). \tag{6}$$

Steps with $N_t = 1$ are excluded (percentile is undefined).

Hit@k, Precision@k, Recall@k, AP@k. For a cutoff k (default $k = 5$), restricted to steps with $N_t \geq k$:

$$\text{Hit@}k = \mathbf{1}\left[\min_{a_i \in \mathcal{E}_m^{(t)}} r_i \leq k\right], \tag{7}$$

$$\text{Prec@}k = \frac{|\{a_i \in \mathcal{E}_m^{(t)} : r_i \leq k\}|}{k}, \tag{8}$$

$$\text{Rec@}k = \frac{|\{a_i \in \mathcal{E}_m^{(t)} : r_i \leq k\}|}{|\mathcal{E}_m^{(t)}|}, \tag{9}$$

$$\text{AP@}k = \frac{1}{\min(k,\, |\mathcal{E}_m^{(t)}|)} \sum_{i=1}^{k} \frac{|\{a_j \in \mathcal{E}_m^{(t)} : r_j \leq i\}|}{i} \cdot \mathbf{1}[r_i \leq k \text{ and } a_i \in \mathcal{E}_m^{(t)}]. \tag{10}$$

Each metric is averaged over eligible steps within a molecule, then over molecules to yield a dataset-level score.

K.2 Statistical Test

To assess statistical significance, we compare each potential function's observed test statistic T_{obs} against a random-logits null distribution. For each of $B = 1000$ null samples, a fresh parameter $p_{\text{null}} \sim \text{Uniform}(0, 1)$ is drawn per step, and actions are re-ranked by the Bernoulli likelihood score

$$s_{\text{null}}(a_i) = p_{\text{null}}^{n_i} \cdot (1 - p_{\text{null}})^{N_t - n_i}, \tag{11}$$

where $n_i = \sum \mathbf{a}_i$ is the number of tree nodes selected by action a_i. Uniform random noise is added for tiebreaking. The null test statistic T_b is computed from the re-ranked actions identically to the observed statistic. The two-sided Monte Carlo p-value is

$$p = \frac{1 + \sum_{b=1}^{B} \mathbf{1}[T_b \geq T_{\text{obs}}]}{B + 1}. \tag{12}$$

We evaluate three settings for training θ. **Base** uses rule usage and success as local rewards, CRM uses CRM in addition to the base rewards, and **ECM** uses ECM in addition to the base (see D for rule definitions). All four checkpoints are evaluated on MOSES test, and the $\overline{\text{pct}}$, Hit@k, Prec@k, Rec@k, AP@k scores and associated p-values are reported over 900 randomly sampled molecules in the dataset. Higher scores in each metric indicate that the potential function systematically assigns greater probability mass to contractions that correspond to exact functional groups, demonstrating chemically interpretable behavior.

L Benchmarking Details

L.1 SmileyLlama Cavanagh et al. (2024)

Background. To assess whether AlphaGrammar tokenization carries over to the supervised drug-design setting, we extended the benchmark introduced by Cavanagh et al. (2024) (henceforth *SmileyLlama*). SmileyLlama is a Llama-3.1-8B-Instruct checkpoint LoRA-supervised-fine-tuned on a corpus of drug-like SMILES drawn from ChEMBL, with an optional DPO refinement stage that improves prompt-following accuracy on per-property generation tasks. The authors evaluate on two benchmarks. The first is the GuacaMol distribution-learning suite (Brown et al., 2019) (Validity, Uniqueness, Novelty, KL-divergence to a held-out reference set, and the Fréchet ChemNet Distance variant FCD-Guacamol), measured on a single pool of 10,000 unconditional samples drawn at sampling temperature T=1.0. The second is a property-following benchmark of 387 natural-language prompts that span 17 task families: counts of hydrogen-bond donors and acceptors, upper bounds on molecular weight, LogP, TPSA, rotatable bonds, fraction of sp^3 carbons, presence or absence of macrocycles, presence or absence of PAINS substructures and named covalent-warhead groups, Enamine substructure inclusion, and the Lipinski rule-of-five and rule-of-three composite constraints. For each prompt the model is asked to generate 1,000 valid SMILES at each of $T \in \{0.7, 1.0\}$, and an RDKit-based property checker scores how often the generated molecule satisfies the requested constraint.

Gramole Extension. Our extension replaces SmileyLlama's SMILES tokenization with AlphaGrammar's rule-based tokenization while keeping the same base LLM (Llama-3.1-8B-Instruct) and the same supervised-fine-tuning recipe (LoRA, alpaca prompt format). Concretely, every SMILES string in the SFT data is parsed into a sequence of indices into the AlphaGrammar production-rule vocabulary, and we extend the Llama tokenizer with one new special token per rule, with the embedding and language-model-head rows added to the LoRA adapter so that the new tokens train end-to-end. For property-following tasks whose prompt contains a SMILES literal – specifically, the "substructure of $[SMILES]$" family — we re-parse that literal into the same rule-token form so the model sees a consistent vocabulary at training and inference. At inference, the type-chained logits processor described in App. F enforces the AlphaGrammar type-chaining constraint at every generation step, as described in App. F.

Evaluation Protocol. To reproduce SmileyLlama's two evaluations, the decoded SMILES files are drop-in inputs for SmileyLlama's existing GuacaMol and property-following analysis pipelines. For Table 16, we draw a single pool of 10,000 samples at T=1.0. For Table 2 we report property-following accuracy at both T=0.7 and T=1.0 as in the original paper, but exclude three of the 17 task families from the AlphaGrammar comparison that favor SMILES tokenizers. The first, "Enamine substructures," contains a SMILES literal in its prompt; the SMILES tokenizer can read this directly while the rule-based tokenizer must re-encode it, which makes the input format different. The remaining two, "presence of covalent warheads" and the "has bad SMARTS" half of the bad-SMARTS family, are scored by SMARTS substructure matching and reward the model for emitting specific SMARTS patterns. Because SmileyLlama emits SMILES tokens that share syntax with SMARTS while AlphaGrammar must plan a rule sequence that decodes into a specific SMARTS pattern, these two task families confound modality with property-following ability. The retained 14 families test descriptor counts, numeric ranges, ring topology, and Lipinski composite constraints, where neither modality enjoys an advantage in the modality format, and the comparison better measures property-following ability.

L.2 mCLM Edwards et al. (2025)

Background. To assess whether AlphaGrammar tokenization generalizes to the molecular optimization setting, we extended the benchmark introduced by Edwards et al. (2025) (henceforth *mCLM*). mCLM is a chemical language model that interleaves natural language with a vocabulary of molecular building blocks, where each molecule is decomposed via a two-stage tokenizer: a *synthesis-guaranteed* tokenizer that disconnects only at amide coupling, Suzuki–Miyaura coupling, and Buchwald–Hartwig coupling bonds (the reactions compatible with state-of-the-art automated synthesis platforms), with a fallback rule-based BRICS-style tokenizer that ensures full reconstructibility of any molecule. The model is initialized from Qwen2.5-3B and pretrained on the released `mCLM_Pretrain_1k` corpus (1.66M instruction–response pairs constructed from approximately 8M molecules), where the molecular vocabulary is filtered to the 1,000 most frequent blocks (957 unique end and mid blocks in the released checkpoint). At each molecule position the model emits a sequence of block tokens, and a graph neural network produces an embedding for each block which is concatenated with the language-model token embedding before the transformer.

Gramole Extension. Our extension replaces mCLM's block tokenization with AlphaGrammar's rule-based tokenization while keeping the same base LLM (Qwen2.5-3B) and the same training corpus (`mCLM_Pretrain_1k`). Concretely, we substitute every `[MOL]`*block sequence*`[/MOL]` span in mCLM's training data with `[MOL]`<rule_i_1>...<rule_i_k>`[/MOL]`, where the indices index the production rules of our learned grammar, and we extend Qwen's tokenizer with one new special token per used rule plus the four structural markers `[MOL]`, `[/MOL]`, `<SMILES>`, `</SMILES>`. The model is fine-tuned with the standard cross-entropy objective; no auxiliary GNN encoder is required because the rule tokens already carry compositional structure. At inference we apply a `GrammarChainLogitsProcessor` that enforces the type-chain property of the production grammar via a BFS-precomputed reachability table: between an opening `[MOL]` and its closing `[/MOL]`, the processor masks any rule token whose input type does not match the previous rule's output type, and forces `[/MOL]` once a terminal rule has been emitted.

Evaluation Protocol. mCLM's drug-editing evaluation (their Section 3.2) selects FDA-approved drugs whose synthesis-guaranteed tokenization yields full coverage, yielding their reported set of 122 molecules, of which 120 contain at least one block outside the 1,000-block training vocabulary (153 unseen blocks total). For each drug, mCLM is prompted to improve one of six ADMET endpoints (AMES mutagenicity, BBB permeability, CYP3A4 inhibition, drug-induced liver injury, human intestinal absorption, and P-glycoprotein substrate likelihood; see Therapeutics Data Commons (Huang et al., 2021) for the underlying datasets), and the resulting molecules are scored by an unreleased ensemble oracle that concatenates FARM, ChemBERTa-2, and GNN features and is trained on the TDC task splits. The reported metric is *Avg. Improv. (%)*, the mean across the six properties of the direction-signed relative change in oracle score versus the FDA baseline.

Note on Table 1. For faithful comparison to the original paper, we obtained mCLM's unreleased 122-drug subset and in-house oracle predictor through private correspondence with the authors. The top section (top nine rows) of Table 1 is directly from Edwards et al. (2025). To eliminate any barriers to future reproducibility, we conducted our own reproduction of the FDA drug improvement case study using public substitutes. For the drug subset, we take the 2,845 DrugBank-approved structures in the `admet_ai` reference set (Swanson et al., 2024) and apply our own faithful reproduction of the synthesis-guaranteed tokenizer — restricting BRICS to the three documented reactions (amide, Suzuki–Miyaura, Buchwald–Hartwig) with N-only environs on amide and aryl-amine bonds — yielding 84 drugs that pass the synth-coverage check. For the oracle, we use (1) ADMET-AI Chemprop-RDKit ensemble (Swanson et al., 2024) (which is trained on the same TDC ADMET splits that mCLM's ensemble uses but produces a different probability calibration) and (2) ADMETlab 3.0 (Fu et al., 2024), which has a different training corpus, different architecture, and different institution from ADMET-AI. The middle and bottom section of Table 1 uses this publicly reproducible setup. Per-endpoint saturation histograms for CYP3A4 and PGP are in Table 18.

L.3 LLamole Liu et al. (2024)

Background. Llamole (Liu et al., 2024) is a multimodal language–graph model for molecular design and retrosynthesis. Their training data, MolQA, comprises ∼135k question–answer pairs, each prompting a target molecule from a natural-language specification (drug-like prop-

erties or polymer-permeability targets) and answering with a designed SMILES followed by a multi-step retrosynthesis trace. Architecturally, Llamole couples a base LLM (Mistral-7B-Instruct-v0.3, Qwen2-7B-Instruct, or Llama-3.1-8B-Instruct) with three frozen graph modules: a graph encoder for embedding input molecules into the token stream, a graph diffusion decoder for de novo design, and a graph predictor that, conditioned on the LM's hidden states, selects retrosynthetic templates over a 301k-template vocabulary. The assistant response is structured by special tokens: "<design_start><design_body>...<design_end><molecule><rollback_start>{SMILES}<rollback_end>" for design, where the eight <design_body> learned-query tokens drive the diffusion decoder and the literal SMILES is exposed via a <rollback> block; and "<retro_start><retro_body>...<retro_end>" for each retrosynthetic step, where the eight <retro_body> tokens' mean hidden state is projected into the graph predictor's text-conditioning space by a learned MLP. Training jointly optimises a language-modelling loss, a graph-design diffusion loss, and a retrosynthesis cross-entropy over templates; LoRA adapters and resized embedding/output layers are the only trainable components. At inference, design generation is greedy/sampled from the LM, and retrosynthesis is delegated to the molstar A* planner of Liu et al. (2024), which iteratively expands a search tree by querying the graph predictor for top-k template candidates and applying `rdchiral` to compute reactant sets.

GRAMOLE Extension. We replace Llamole's <design_body>/<rollback> design representation with our ALPHAGRAMMAR rule-token vocabulary while preserving the retrosynthesis pipeline unchanged. Concretely, in the assistant response the design block becomes "<design_start><design_end><rule_i_1>...<rule_i_n>", with the rule tokens emitted directly into the LM stream rather than threaded through a separate diffusion decoder; this removes the <design_body> learned queries and the <rollback> SMILES echo entirely, and obviates the graph diffusion decoder and its loss term. Each rule token is added to the tokenizer as a regular vocabulary entry (not a special token); Llamole's nine structural special tokens are unaffected. We base our model on Qwen3-8B, trained with LoRA (rank 8 over all attention and MLP projections, with `lm_head` and `embed_tokens` added to `modules_to_save` so the new rule embeddings are learnable). The training loss reduces to the LM cross-entropy plus the retrosynthesis cross-entropy of Liu et al. (2024): $\mathcal{L} = \mathcal{L}_{\text{lm}} + \mathcal{L}_{\text{retro}}$, with the graph predictor and its 768-dimensional text-conditioning interface frozen at the weights released by Liu et al. (2024). We train on the full MolQA split for four epochs at learning rate 1×10^{-4} with cosine schedule and 10% warmup, in pure bfloat16, with effective batch size 12. Because rule tokens form a context-free derivation rather than free text, we apply a custom `GrammarChainLogitsProcessor` during inference that enforces (i) type-chaining—after the empty <design_start><design_end> marker pair, only rules whose *in-type* matches the current derivation frontier are admissible, with the final emitted rule constrained to a terminal—and (ii) an optional cumulative wildcard-atom budget that tracks the number of $\star$ atoms introduced by each rule and forbids rule choices that overshoot a target count or that lead to states from which the target cannot be reached within the remaining slot budget. For polymer evaluation we set this target to 2, ensuring every generated molecule is a valid two-endpoint repeat unit; for drug evaluation the budget is disabled.

Evaluation Protocol. For evaluation we follow Liu et al. (2024) exactly: their `molqa_drug` split (9,986 prompts) and `molqa_material` split (750 prompts), with structural metrics (validity, Tanimoto similarity to the ground-truth designed molecule), text metrics (BLEU-4 and ROUGE-L over the response), property metrics (binding-affinity classifier scores for HIV/BBBP/BACE on drugs, and per-property MAE for CO_2/N_2/O_2 permeability, fractional free volume, and thermal conductivity on materials), and retrosynthesis success rate as defined in Liu et al. (2024). At inference we use the released graph predictor with their molstar A* planner, top-k template expansion, and 30-second per-target planning budget. Since our design path emits a deterministic rule-token derivation rather than a SMILES, we decode the rule sequence to obtain the final SMILES that is then handed to the retrosynthesis stage; this decoding step is the only modification to Llamole's evaluation harness.

M ADDITIONAL RESULTS

M.1 (RQ1) TOKENIZER EVALUATION

Results. Fig. 5 uses the results in Table 14 at $|V| = 100$.

Table 14: Each random seed (3 total), we sample 100 molecules from ChemBl, compute or estimate parse sets, and evaluate metrics. For ALPHAGRAMMAR, we provide worst and best-case bounds on metrics that require the parse forest due to fixed-budget timeouts of the Bolinas parser (complexity in App. C; timeout rates in App. J.2). For non-ALPHAGRAMMAR tokenizers, we resort to a coupon-collector estimator (App. J.2), not an exact count, due to intractability of brute-force parsing. Atoms/Tok is mean heavy atoms per token; E$|P(m)|$@100 is the mean parse count (* = exact enumeration for ALPHAGRAMMAR and BRICS). Uniq is reported as a worst–best interval wherever exact enumeration is intractable (all methods except BRICS's closed-form count). Details are in App. J.

Method	Atoms/Tok	E$\|P(m)\|$@100	Use@10	Use@100	Use@1000	Uniq@10	Uniq@100	Uniq@1000	Qual@10	Qual@100	Qual@1000
SMILES (char)	0.53	2.28e4±5.58e3	0.678±0.0311	0.37±0.0113	0.366±0.0113	0.0112±0.0131	0.0029±0.0018–0.0041±0.0028	0.00338±0.00272	N/A	N/A	N/A
SMILES (BPE: ChemBERTa)	0.78	8.37e3±1.59e3	N/A	0.157±0.00566	0.118±0.00636	N/A	0.0015±0.0015–0.0068±0.0376	0.00332±0.0026	N/A	N/A	N/A
SELFIES (char)	0.13	2.43e4±4.62e3	N/A	0.686±0.00141	0.683±0.00636	N/A	0.0028±0.0017–0.0039±0.0027	0.00351±0.00292	N/A	N/A	N/A
SELFIES (BPE: SELFormer)	0.39	2.42e4±3.59e3	0.875±0.0629	0.486±0.00495	0.487±0.00566	0.00602±0.00661	0.0029±0.0019–0.0040±0.0027	0.00342±0.00267	N/A	N/A	N/A
GroupSELFIES	0.74	4.53e3±3.51e3	0.683±0.218	0.196±0.00566	0.194±0.00424	0.0397±0.0531	0.0027±0.0014–0.0063±0.0074	0.00462±0.0038	N/A	N/A	N/A
BRICS	4.34	1.82e4*	0.25	0.0425±0.00933	0.0238±0.00339	0.183	0.135±0.101	0.101±0.0557	0.541±0.0926	0.334±0.0304	0.303±0.0262
mCLM	11.84	1.4	0.225	0.0166±0.000566	0.0102±0.000863	0.25	0.620±0.013–0.718±0.033	0.592	0.201±0.00919	0.162±0.017	0.168±0.0212
ALPHAGRAMMAR	2.04	253*	N/A	0.074±0.00693–0.199±0.184	0.0215±0.00785–0.289±0.254	N/A	0.412±0.216–0.537±0.039	0.0365±0.0313–0.304±0.231	0.597±0.0191	0.533±0.0445	0.49±0.053

M.2 (RQ2) GRAMMAR SAMPLING FOR MOLECULAR GENERATION

We see in Table 15 that chaining random rules from the grammar (equivalent to GRAMOLE predicting random logits) ensures validity; conceptually, ALPHAGRAMMAR and JTN-VAE both use junction tree to constrain for validity. ALPHAGRAMMAR attains the most diverse set of molecules, and is the sole method with 100% Novelty. It's worth noting Ngram and Combinatorial are analogs of ALPHAGRAMMAR, but for character and BRICS tokens. Together, these random token sampling baselines are a litmus test for the MGT itself; an ideal MGT should have the highest "expressivity ceiling" possible, before model development and distribution learning. This shows the ALPHAGRAMMAR tokenizer, separate from model training, is sufficiently expressive on its own.

Table 15: We benchmark ALPHAGRAMMAR on MOSES. Each random seed (3 total), we sample 30000 type-chained rule sequences and decode into valid molecules.

Model	Valid (↑)	Unique@1k (↑)	Unique@10k (↑)	IntDiv (↑)	IntDiv2 (↑)	Novelty (↑)
Train	1.000	1.000	1.000	0.857	0.851	1.000
HMM	0.076±0.032	0.623±0.122	0.567±0.142	0.847±0.040	0.810±0.051	0.999±0.001
NGram	0.238±0.003	0.974±0.011	0.922±0.002	0.874±0.000	0.864±0.000	0.969±0.001
Combinatorial	1.000±0.000	0.998±0.002	0.991±0.001	0.873±0.000	0.867±0.000	0.988±0.001
CharRNN	0.975±0.026	**1.000±0.000**	0.999±0.000	0.856±0.001	0.850±0.001	0.842±0.051
AAE	0.937±0.034	**1.000±0.000**	0.997±0.002	0.856±0.003	0.850±0.003	0.793±0.029
VAE	0.977±0.001	**1.000±0.000**	0.998±0.001	0.856±0.000	0.850±0.000	0.695±0.007
JTN-VAE	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	0.855±0.003	0.849±0.004	0.914±0.006
LatentGAN	0.897±0.003	**1.000±0.000**	0.997±0.000	0.857±0.001	0.851±0.001	0.950±0.001
ALPHAGRAMMAR	**1.000±0.000**	0.986±0.001	0.972±0.001	**0.884±0.000**	**0.877±0.000**	**1.000±0.000**

M.3 (RQ3, RQ5) UNCONDITIONAL SAMPLING AND PROMPT FOLLOWING

In Table 16 (which extends SmileyLlama's benchmark), we evaluate post-SFT GRAMOLE on the unconditional generation benchmark GuacaMol. We compare with SMILES-only unconditional CLMs and matched-assumption SmileyLlama. GRAMOLE leads on validity (only 100%-valid) and novelty-at-realism: LSTM's top FCD (0.901) has the worst novelty (0.848) vs GRAMOLE's 0.977. SmileyLlama benefits from positive transfer of Llama pretraining on Internet SMILES data during pretraining, while GRAMOLE is learning grammar from scratch.

LSTM, GPT, and S4 are strong baselines as they are single-modality (SMILES) CLMs pretrained specifically for unconditional sampling. They lack native instruction-following ability, and therefore are not benchmarked in Table 17. In Table 17, we show GRAMOLE can also be prompted to generate molecules with specified properties, ranging from molecular structure, drug-like properties, chemical characteristics, and absence/presence of certain substructures. Together, this suggests GRAMOLE's single-stage SFT makes up for any positive transfer SMILES-based LLMs may have, accomplishing multiple tasks at once: learning $\mathcal{V}_r$ embedding, aligning distributionally to ChEMBL, and learning to follow instructions. This shows extending the vocabulary does not require continued pretraining or full finetuning. ALPHAGRAMMAR tokenizer integration at SFT time is feasible, and rule token embeddings can be learned jointly with natural language context via LoRA-style finetuning.

Decoding modes. We see high Validity, Uniqueness, and Novelty but a regression in distributional (KL, FCD) metrics when applying the same inference mode as MOSES (a hard type-chaining mask), since it constrains valid-but-unrealistic rules that move the distribution into low-mass regimes. Post-hoc repair of non-chaining steps did not help either: six strategies all stall at FCD 0.607–0.613. What worked is in-loop interjection: sample freely, but on a non-chaining rule substitute the

top-probability chaining rule and feed it back, recovering FCD $0.601 \rightarrow 0.746$ at 100% validity, surpassing SmileyLlama.

Table 16: We compare GRAMOLE to other CLMs trained on ChEMBL via GuacaMol benchmark. Metrics include validity, uniqueness, novelty to training set, and distribution similarity via KL and FCD. All metrics are in $[0, 1]$; higher (↑) is better. We report three decoding modes for GRAMOLE: the default strict type-chaining mask, best post-hoc repair, and a non-chaining in-loop interjection (sample freely, substitute the top-probability chaining rule, feed back) that recovers FCD to 0.746 at 100% validity, surpassing SmileyLlama.

Benchmark	Validity	Uniqueness	Novelty	KL div	FCDGuac
LSTM Özçelik et al. (2024)	0.983	0.999	0.848	0.993	0.901
GPT Özçelik et al. (2024)	0.915	1.000	0.978	0.977	0.826
S4 Özçelik et al. (2024)	0.971	0.997	0.961	0.994	0.853
Llama zero-shot	0.688	0.457	0.635	0.736	0.002
Llama twenty-shot	0.465	0.999	0.949	0.913	0.079
SmileyLlama Cavanagh et al. (2024)	0.958	1.000	0.987	0.967	0.686
GRAMOLE (strict mask)	1.000	0.994	0.977	0.954	0.601
GRAMOLE (post-hoc repair)	1.000	0.994	0.973	0.963	0.607
GRAMOLE (in-loop interject., T=0.9)	1.000	0.993	0.976	0.929	0.746

Table 17: We follow Cavanagh et al. (2024). For all tasks, we generate 1000 molecules from a prompt requesting some property, and score the result based on the proportion of molecules which are valid, distinct, and satisfy the properties requested in the prompt. More details in App. L.

	SmileyLlama SFT		GRAMOLE SFT			SmileyLlama SFT		GRAMOLE SFT	
Present in Trained Prompt	T=0.7	T=1.0	T=0.7	T=1.0	Absence of warheads	96.4%	94.2%	98.8%	97.3%
$\leq$ k H-bond donors	96.6%	94.4%	96.4%	96.2%	Lipinski rule-of-five	89.0%	81.7%	88.3%	84.0%
$\leq$ k H-bond acceptors	96.3%	90.7%	92.9%	90.4%	Presence of macrocycle	38.8%	44.7%	29.7%	37.1%
$\leq$ k Molecular weight	90.1%	84.3%	96.0%	92.6%	Average (11 properties)	88.0%	84.9%	86.1%	84.5%
$\leq$ k ClogP	89.2%	85.4%	91.3%	86.1%	Absent in Trained Prompt	T=0.7	T=1.0	T=0.7	T=1.0
Rotatable bonds in range	90.0%	86.7%	78.4%	82.2%	Rule-of-three	77.5%	62.0%	84.4%	69.5%
Fraction sp3 in range	87.6%	85.0%	85.5%	79.0%	Exactly k H-bond donors	23.8%	22.4%	15.8%	18.9%
TPSA in range	95.9%	91.1%	94.5%	94.5%	Exactly k H-bond acceptors	17.9%	15.0%	18.5%	19.2%
Absence of macrocycle	98.4%	95.3%	94.9%	90.6%	Average (3 properties)	39.7%	33.1%	39.5%	35.9%

M.4 (RQ4) FDA-DRUG OPTIMIZATION

Per-endpoint predicted-value histograms for the saturation-prone endpoints (CYP3A4, PGP) referenced in Table 1. Under ADMET-AI both methods compress against the predictor floor (bin $[0, 1)$); ADMETlab spreads the distribution and PGP flips to a GRAMOLE win.

Table 18: Per-endpoint predicted-value histograms for the saturation-prone endpoints (CYP3A4, PGP), fine bins at the low end (n=84 under ADMETlab, n=504 under ADMET-AI). ADMET-AI compresses both methods against the predictor floor (bin $[0, 1)$); ADMETlab spreads the distribution and shows PGP flips to a Gramole win.

Endpoint	Method	Oracle	mean	$[0, 1)$	$[1, 5)$	$[5, 10)$	$[10, 25)$	$[25, 50)$	$[50, 100]$	n
CYP3A4	mCLM	ADMET-AI	1.06	447	28	17	5	6	1	504
CYP3A4	mCLM	ADMETlab 3.0	1.98	66	5	7	6	0	0	84
CYP3A4	Gramole	ADMET-AI	1.78	407	58	16	16	5	2	504
CYP3A4	Gramole	ADMETlab 3.0	6.48	48	19	3	4	7	3	84
PGP	mCLM	ADMET-AI	0.60	458	34	1	9	2	0	504
PGP	mCLM	ADMETlab 3.0	25.02	8	7	28	12	20	9	84
PGP	Gramole	ADMET-AI	2.66	365	84	19	17	16	3	504
PGP	Gramole	ADMETlab 3.0	8.65	26	45	2	4	1	6	84

M.5 (RQ6) INVERSE DESIGN WITH RETROSYNTHESIS

Table 19: We follow Liu et al. (2024)'s multi-conditional molecular design protocol. Responses include (i) a designed molecule, (ii) the proposed synthesis route, and (iii) textual analysis for the design and synthesis: (i) measures validity, similarity, and deviation from desired properties; (ii) measures success of LLM-guided A^*-search; (iii) measures response coherence and similarity. GRAMOLE (CRM, MolQA default) RS is reported as mean ± std over 3 seeds; following mCLM/Llamole (neither reported seeds), other rows are single runs (multi-seed training is prohibitively expensive, so we seed the settings our claims rest on: tokenizer-property evaluation, MOSES, and MolQA-CRM). The temperature/variant sweep is deferred to App. M.5 (Table 23). Best global result is **bolded**; best category result is *italicized*.

Base LLM or Method	Structure (↑)		Text (↑)		Drug (BA ↑)				Drug (RS ↑)		Material (MAE ↓)					Material (RS ↑)
	Validity	Similarity	BLEU-4	ROUGE-L	HIV	BBBP	BACE	Avg. BA	Success %	CO_2Perm	N_2Perm	O_2Perm	FFV	TC	Avg. MAE	Success %
GraphGA	*0.885*	*0.112*	NA	NA	*0.536*	0.515	0.560	0.537	*29.9*	0.847	*1.536*	0.747	*0.020*	*0.042*	*0.642*	14.3
DiGress	0.375	0.046	NA	NA	0.515	*0.522*	*0.580*	*0.539*	18.0	*0.655*	1.884	*0.680*	*0.020*	0.049	0.658	*15.4*
In-Context Learning																
Llama-2-7B	0.167	0.024	0.030	0.141	0.051	0.060	0.053	0.055	0.1	5.463	3.982	4.943	0.308	0.199	2.979	0.3
Mistral-7B	0.251	0.044	0.066	0.203	0.163	0.153	0.200	0.172	0.2	5.062	3.824	4.657	0.289	0.186	2.804	0.4
Qwen2-7B	0.180	0.012	0.030	0.147	0.089	0.091	0.085	0.088	0.0	5.552	4.251	5.068	0.322	0.211	3.081	0.0
Qwen3-8B	0.186	0.023	0.017	0.091	*0.561*	*0.500*	*0.603*	*0.554*	0.1	*0.894*	*2.493*	*0.565*	*0.017*	*0.029*	*0.800*	0
Llama-3-8B	0.656	0.112	0.155	0.307	0.471	0.473	0.562	0.502	*5.5*	3.233	3.106	2.924	0.171	0.123	1.911	*4.8*
Flan-T5-XXL	0.570	0.094	*0.226*	*0.388*	0.329	0.333	0.403	0.355	0.4	2.869	3.039	2.799	0.165	0.120	1.798	0.8
Granite-13B	0.498	0.079	0.170	0.326	0.260	0.293	0.285	0.279	0.6	2.994	3.165	2.993	0.180	0.123	1.891	1.6
Llama-2-13B	0.346	0.058	0.121	0.279	0.236	0.250	0.259	0.248	1.2	5.031	4.285	4.816	0.291	0.184	2.921	1.2
Mistral-8x7B	0.546	0.094	0.181	0.345	0.345	0.346	0.388	0.360	1.6	3.695	3.150	3.440	0.191	0.138	2.123	1.7
Llama-2-70B	0.299	0.045	0.099	0.222	0.237	0.242	0.274	0.251	1.0	5.368	4.336	5.017	0.319	0.202	3.048	0.8
Llama-3-70B	*0.706*	*0.124*	0.210	0.367	0.415	0.403	0.484	0.434	-	2.659	2.848	2.421	0.135	0.099	1.632	-
Supervised Fine-tuning																
Mistral-7B	0.718	0.125	0.105	0.216	0.460	0.483	0.515	0.486	1.5	3.269	3.094	2.985	0.184	0.128	1.932	0.8
Qwen2-7B	0.768	0.133	0.221	0.377	0.436	0.457	0.457	0.450	0.2	2.691	2.562	2.721	0.147	0.106	1.645	0.1
Qwen3-8B	*0.983*	*0.186*	**0.357**	**0.495**	*0.529*	*0.553*	*0.624*	*0.569*	*6.1*	*0.872*	*1.618*	*0.791*	**0.018**	*0.038*	*0.667*	*4.7*
Llama-3-8B	0.797	0.136	0.093	0.206	0.426	0.445	0.440	0.437	0.6	2.222	2.322	2.119	0.110	0.086	1.372	0.7
Llama-3.1-8B	0.692	0.121	0.121	0.250	0.417	0.432	0.433	0.427	0.8	3.210	2.991	2.974	0.179	0.122	1.895	0.8
Llamole																
Mistral-7B	0.900	0.139	*0.262*	*0.434*	0.596	0.617	0.740	0.651	29.9	**0.593**	1.409	0.565	0.021	0.028	0.523	14.3
Qwen2-7B	0.888	0.135	0.261	0.432	0.600	**0.639**	**0.746**	**0.662**	33.7	0.645	1.452	0.581	0.021	**0.026**	0.545	17.9
Qwen3-8B	0.903	0.118	0.226	0.393	0.610	0.587	0.742	0.646	*36.7*	0.748	**1.054**	0.666	*0.020*	**0.026**	**0.503**	*25.3*
Llama-3.1-8B	*0.913*	**0.142**	0.254	0.427	**0.623**	0.629	0.713	0.655	35.1	0.653	1.344	**0.549**	0.021	0.030	0.519	17.6
Gramole																
Qwen3-8B (CRM, MolQA Default)	**1.000±0**	0.107±0.015	0.210±0.012	0.380±0.016	0.519±0.017	0.491±0.031	0.510±0.008	0.507±0.011	**48.1±11.8**	1.115±0.029	1.207±0.158	0.869±0.043	0.027±0.004	0.079±0.003	0.659±0.018	**78.7±4.8**

Results. Fig. 7 summarizes Table 19; we select the Pareto-efficient methods per category (ICL, SFT, Llamole, GRAMOLE).

Further Ablations. We toggle on the graph encoder, a key component in the Llamole architecture, where a <molecule> token is appended to each rule sequence, which is substituted with graph embedding (+Encoder), after each rule sequence, and observe a modest bump in textual coherence but no consistent improvement in any other dimension, suggesting useful auxiliary features for textual analysis but comparable performance for design and retrosynthesis. This shows rule sequence alone is not only lossless in representation, but sufficient for performance; adding graph adapters bloat the system without tangible gains. **CRM-Template Ablation.** We test whether GRAMOLE's synthesis-planning gain depends on the coupling reactions its CRM reward targets, by removing all CRM-matching templates from the retro planner (Table 20). Dropping ∼40% of templates hurts both methods, but GRAMOLE generalizes beyond the coupling reactions it was rewarded on: its drug-RS gap versus Llamole flips from -2.1 to $+5.0$ pp, and it keeps a large material-RS lead.

Table 20: Retrosynthesis success with the planner's CRM-matching templates removed (301,245 full → 179,014 non-CRM), comparing GRAMOLE-CRM with Llamole-Qwen3-8B. Dropping ∼40% of templates hurts both, but GRAMOLE generalizes beyond the coupling reactions it was rewarded on: the drug-RS gap flips from -2.1 to $+5.0$ pp, and GRAMOLE keeps a large material-RS lead. (Single-seed; headline numbers are 3-seed, Table 19.)

# Templates	Method	Drug RS (%, ↑)	Material RS (%, ↑)
301,245 (full)	GRAMOLE (CRM)	34.6	**82.7**
	Llamole-Qwen3	36.7	25.3
179,014 (non-CRM)	GRAMOLE (CRM)	**25.7**	**69.3**
	Llamole-Qwen3	20.7	18.7

Retrosynthesis Operating Curves. To rule out size/similarity confounds, we bin retrosynthesis success by heavy-atom count (Table 21) and by Tanimoto-to-target against property-target compliance (Table 22). GRAMOLE leads Llamole in every heavy-atom bin up to 24 atoms (54% of designs) and by $1.9\times$ over the full distribution after size-matching, and remains far ahead in the low-similarity bins where the confound is most severe.

Temperature/Variant Sweep. Table 23 reports the full GRAMOLE temperature and reward-variant sweep on MolQA; all headline numbers (Table 19) are read off the CRM default at the default temperature. In Fig. 9, the frontier is an ablation showing the achievable trade-off.

Table 21: Retrosynthesis success rate binned by heavy-atom count (drug and material weighted equally per bin, matching the paper's aggregate). GRAMOLE-CRM leads Llamole-Qwen3-8B in every bin from 0–24 heavy atoms (54% of designs, $+22.5$ to $+30.0$ pp); the $1.9\times$ full-distribution advantage (58.7 vs. 31.0) survives size-matching.

Heavy atoms	N (Llamole)	RS % (Llamole)	N (GRAMOLE)	RS % (GRAMOLE)
invalid gen	51	0.0	0	—
0–14	43	45.7	45	**68.2**
15–19	76	43.2	93	**70.1**
20–24	213	29.8	159	**59.8**
25–29	82	29.3	109	**33.0**
30–34	31	10.7	75	12.7
35+	43	**8.0**	58	0.9
Total	537	31.0	537	**58.7**

Table 22: RS success (%) binned by Tanimoto-to-target (columns) and property-target compliance (rows; high $=\geq$67% drug BA labels correct or $\leq$33% material MRE), drug/material equal weight. Down the Total rows, GRAMOLE far exceeds Llamole in every low-similarity bin (where the confound bites hardest); the advantage is not from simpler/less-similar molecules nor at the cost of property fidelity.

Llamole (Qwen3-8B)							
compliance \ sim	[0.00,0.05)	[0.05,0.10)	[0.10,0.15)	[0.15,0.20)	[0.20,0.30)	[0.30,1.00]	N
low	0.0	17.5	14.6	32.5	42.5	0.0	42
med	0.0	21.1	15.7	71.1	68.2	100.0	48
high	23.1	22.4	27.5	36.1	63.0	83.3	194
Total	8.3	21.7	24.0	42.1	52.5	62.5	284
GRAMOLE *(CRM)*							
compliance \ sim	[0.00,0.05)	[0.05,0.10)	[0.10,0.15)	[0.15,0.20)	[0.20,0.30)	[0.30,1.00]	N
low	63.8	61.8	61.9	27.5	41.7	—	74
med	80.0	61.1	44.8	40.4	0.0	—	74
high	76.7	64.0	51.0	53.4	78.6	0.0	166
Total	**71.0**	**58.2**	**52.0**	44.3	62.5	0.0	314

M.6 (RQ7) FUNCTIONAL GROUP ENRICHMENT

N ATOMIC RULE ENUMERATION

An *atomic rule* is a production rule whose RHS contains exactly one cluster (terminal atom or ring).

There atomic rules may be divided into three categories:

- **Category A:** Rules with a single atom.
- **Category B:** Rules with R^* bonded to a minimal ring via a single atom.
- **Category C:** Rules with R^* replacing an atom within the ring.

For each rule, any atom and R^* may further have extra bonds (external nodes) not accounted for by the base topology. These are represented by a multiset of bond types, and must satisfy the per-atom-symbol valence constraints and total bond-order constraints. Any rule may further be either an *initial rule* (no external nodes) or a *terminating rule* (no R^*).

We bound the number of atomic rules subject to the following constraints:

```
ATOM_VALENCES     = {'C': 4, 'N': 3, 'O': 2, 'S': 6, 'F': 1, 'Cl': 1, 'Br': 1}

VALID_VALENCES    = {'C': [4], 'N': [3], 'O': [2], 'S': [2, 4, 6],
                     'F': [1], 'Cl': [1], 'Br': [1]}
```

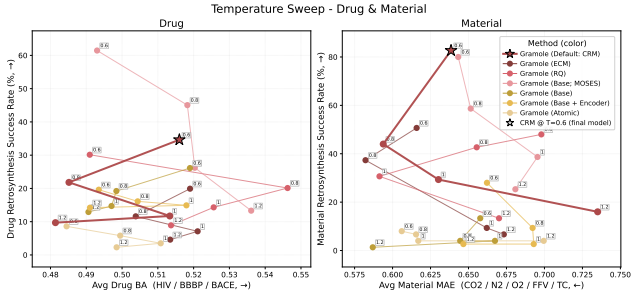


Figure 9: MolQA drug-RS / material-RS Pareto frontier under a decoding-temperature sweep at the CRM default, with 3-seed error bars. Per-configuration numbers are in Table 23.

Table 23: Full GRAMOLE temperature and reward-variant sweep on MolQA (temperatures 0.6/0.8/1.0/1.2 per setting). Headline Table 19 numbers use the CRM default at T=0.6; this sweep shows the achievable trade-off, not a source of best-per-metric claims.

Base LLM or Method	Structure (↑)		Text (↑)		Drug (BA ↑)				Drug (RS ↑)	Material (MAE ↓)						Material (RS ↑)
	Validity	Similarity	BLEU-4	ROUGE-L	HIV	BBBP	BACE	Avg. BA	Success %	CO_2Perm	N_2Perm	O_2Perm	FFV	TC	Avg. MAE	Success %
Gramole (T=0.6, 0.8, 1, 1.2)																
Qwen3-8B (CRM, MolQA Default)	1.000	0.123	0.220	0.392	0.539	0.495	0.514	0.516	34.6	1.143	1.044	0.898	0.031	0.077	0.639	82.7
	0.980	0.124	0.221	0.386	0.483	0.479	0.494	0.485	21.8	1.104	0.981	0.784	0.028	0.073	0.594	44.0
	0.970	0.126	0.213	0.373	0.488	0.517	0.535	0.513	11.7	1.054	1.137	0.886	0.022	0.052	0.630	29.3
	0.953	0.118	0.204	0.361	0.515	0.476	0.454	0.482	9.7	1.280	1.325	1.002	0.018	0.050	0.735	16.0
Qwen3-8B (Base)	0.999	0.117	0.119	0.258	0.529	0.515	0.513	0.519	26.1	1.164	1.097	0.924	0.028	0.077	0.658	13.3
	0.989	0.125	0.111	0.247	0.515	0.490	0.491	0.499	19.2	1.179	1.071	0.884	0.023	0.065	0.644	4.0
	0.990	0.120	0.102	0.234	0.500	0.483	0.509	0.497	14.7	1.162	1.182	0.919	0.021	0.052	0.667	4.0
	0.954	0.113	0.087	0.212	0.494	0.495	0.483	0.491	12.9	1.161	1.043	0.670	0.020	0.041	0.587	1.3
Qwen3-8B (Base) + Encoder	1.000	0.116	0.220	0.397	0.477	0.500	0.504	0.494	19.6	1.170	1.108	0.925	0.029	0.079	0.662	28.0
	0.999	0.125	0.220	0.386	0.524	0.496	0.493	0.504	16.1	1.198	1.199	0.981	0.025	0.058	0.692	9.3
	0.998	0.124	0.207	0.369	0.534	0.492	0.528	0.518	14.9	1.200	1.218	0.980	0.021	0.046	0.693	2.7
	0.977	0.119	0.194	0.353	0.510	0.497	0.466	0.491	14.2	1.159	1.142	0.883	0.016	0.034	0.647	2.7
Qwen3-8B (Atomic)	0.981	0.091	0.189	0.339	0.411	0.534	0.509	0.485	8.6	0.958	1.142	0.836	0.029	0.066	0.606	8.0
	0.982	0.097	0.182	0.331	0.483	0.508	0.507	0.499	5.8	0.957	1.211	0.820	0.028	0.061	0.615	6.7
	0.935	0.091	0.172	0.316	0.499	0.520	0.514	0.511	3.5	0.958	1.111	0.942	0.027	0.047	0.617	4.0
	0.917	0.094	0.161	0.299	0.475	0.479	0.542	0.499	2.3	1.155	1.267	0.985	0.029	0.062	0.700	4.0
Qwen3-8B (ECM)	0.947	0.115	0.218	0.381	0.522	0.503	0.531	0.519	19.9	1.070	1.012	0.907	0.022	0.069	0.616	50.7
	0.952	0.119	0.216	0.383	0.508	0.509	0.494	0.504	11.6	1.151	0.893	0.779	0.023	0.065	0.582	37.3
	0.958	0.125	0.210	0.369	0.498	0.505	0.560	0.521	7.1	1.168	1.169	0.907	0.020	0.046	0.662	9.3
	0.963	0.121	0.196	0.350	0.510	0.502	0.529	0.514	4.6	1.125	1.233	0.944	0.021	0.043	0.673	6.7
Qwen3-8B (RQ)	0.953	0.108	0.215	0.390	0.465	0.506	0.502	0.491	30.1	1.151	1.278	0.956	0.027	0.079	0.698	48.0
	0.946	0.117	0.217	0.386	0.563	0.551	0.525	0.546	20.1	1.094	1.179	0.917	0.024	0.063	0.655	42.7
	0.944	0.122	0.209	0.375	0.546	0.516	0.514	0.525	14.3	1.141	0.947	0.787	0.024	0.058	0.591	30.7
	0.950	0.120	0.203	0.357	0.514	0.514	0.513	0.514	8.9	1.218	1.126	0.929	0.021	0.056	0.670	13.3
Qwen3-8B (Base; MOSES)	0.980	0.110	0.210	0.384	0.487	0.496	0.497	0.493	61.5	1.202	0.934	0.960	0.028	0.092	0.643	80.0
	0.985	0.113	0.200	0.367	0.504	0.544	0.507	0.518	45.0	1.093	1.161	0.899	0.025	0.079	0.651	58.7
	0.971	0.122	0.199	0.364	0.520	0.497	0.544	0.520	26.2	1.218	1.193	0.969	0.027	0.070	0.695	38.7
	0.978	0.124	0.195	0.348	0.556	0.483	0.570	0.536	13.3	1.201	1.191	0.935	0.023	0.054	0.681	25.3

Table 24: Enrichment of positives in θ-parameterized contractions vs the Null. Details in App. K.

	$\text{Percentile}_{\text{avg}}$	p_{avg}	hit@5	$p_{\text{hit@5}}$	precision@5	$p_{\text{prec@5}}$	recall@5	$p_{\text{rec@5}}$
Base	0.779	< 0.001	0.768	< 0.001	0.337	< 0.001	0.470	< 0.001
CRM	0.839	< 0.001	**0.909**	< 0.001	**0.402**	< 0.001	**0.542**	< 0.001
ECM	**0.847**	< 0.001	0.886	< 0.001	0.401	< 0.001	0.540	< 0.001
Null	0.536	N/A	0.486	N/A	0.186	N/A	0.296	N/A

```
ALL_ATOM_TYPES     = ['C', 'N', 'O', 'S', 'F', 'Cl', 'Br']
RING_ATOM_TYPES    = ['C', 'N', 'O', 'S']
BOND_TYPES         = [1, 2]
SINGLE_ATOM_BOND_TYPES = [1, 2, 3]

MAX_EXT_NODES      = 6
MAX_RSTAR_DEGREE   = 6
MAX_RSTAR_EXT      = 4   # max bridge bonds (ext nodes in R* but not in any atom)
MIN_RING_SIZE      = 3
MAX_RING_SIZE      = 6
```

We use variables $A = 8$ atomic configurations (7 atomic symbols, with 1 or 2 explicit Hs for N), $E = 6$ external node budget, max ring size $R = 6$, for our analytic bounds. We assume hard-coded constants: S has the worst case valence of 6, 3 bond types (single, double, triple) for non-ring atoms, and 2 bond types (single, double) for ring atoms (kekulized form).

Theorem 3. Let $G = (V_G, E_G)$ be a connected molecular graph, where vertices are atoms and edges are bonds. Assume G is pathwidth bounded by k and maximal node degree V:

$$\text{pw}(G) \leq k \qquad \text{and} \qquad \Delta(G) \leq V$$

Then there exists a connected growth process

$$R_1^* \subseteq R_2^* \subseteq \cdots \subseteq R_t^* = V_G$$

such that each induced subgraph $G[R_i^*]$ is connected, and for every i, the number of external bonds of R_i^*,

$$\delta(R_i^*) := \{uv \in E_G : u \in R_i^*,\ v \notin R_i^*\},$$

is at most

$$|\delta(R_i^*)| \leq V(2k+2).$$

In particular, the number of external bonds of G is at most $V(2k+2)$.

Proof. Since $\mathrm{pw}(G) \leq k$, and connected pathwidth satisfies

$$\mathrm{cpw}(G) \leq 2\ \mathrm{pw}(G) + 1,$$

Dereniowski (2012) there exists a connected path decomposition

$$X_1, \ldots, X_m$$

of G with

$$|X_j| \leq 2k+2 \qquad \text{for all } j.$$

Define

$$R_i^* := \bigcup_{j=1}^{i} X_j.$$

Because the decomposition is connected, each $G[R_i^*]$ is connected.

Now let

$$\partial_V(R_i^*) := \{v \in R_i^* : \exists\, w \notin R_i^* \text{ such that } vw \in E_G\}$$

be the set of boundary atoms of R_i^*. We claim that

$$\partial_V(R_i^*) \subseteq X_i.$$

Indeed, let $v \in \partial_V(R_i^*)$. Since $v \in R_i^*$, there exists some $a \leq i$ such that $v \in X_a$. Also, v has a neighbor $w \notin R_i^*$. Because $vw \in E_G$, by the edge-cover property of a path decomposition there exists some bag X_b containing both v and w. Since $w \notin R_i^*$, we must have $b > i$. Therefore v appears in some bag X_a with $a \leq i$ and in some bag X_b with $b > i$. By the interval property of path decompositions, v must belong to every bag between X_a and X_b, in particular to X_i. This proves the claim. Hence,

$$|\partial_V(R_i^*)| \leq |X_i| \leq 2k+2.$$

Finally, each boundary atom is incident to at most V external bonds, since $\Delta(G) \leq V$. Therefore

$$|\delta(R_i^*)| \leq V\,|\partial_V(R_i^*)| \leq V(2k+2)$$

This holds for every i, so the connected growth process $R_1^*, \ldots, R_m^*$ has at most $V(2k+2)$ external bonds at every step. □

Remark 3. In Def. A.7, the reasonableness constants V (valence) and C (connected pathwidth) bound the constant E = MAX_EXT_NODES via the relation $E \leq V \cdot (C+1)$. For enumeration to be practical, we enumerate atomic rules up to the constant $E = 6$. We also restrict to MAX_RSTAR_EXT of the external bonds to be on R^*, and the total degree of R^* (external bonds and bonds connected to terminal atoms) to MAX_RSTAR_DEGREE.

N.1 Category A

A rule is determined by its atomic symbol, atom bond split, and external bond-type multiset.

In the following, we focus on non-terminal rules. Terminal rules may be turned into (disconnected) non-terminal rules by adding an R^* that is disconnected from the terminal atom. In this way, our bound will also include all terminal rules.

Bounding terminal atom bond splits.

The terminal atom's bond multiset is partitioned into external and R^* subsets. If it has n_1 single bonds, n_2 double bonds, and n_3 triple bonds, then the number of such splits is

$$(n_1 + 1)(n_2 + 1)(n_3 + 1).$$

If the atom has allowed valences in S, the numbers n_1, n_2, n_3 are subject to the constraint

$$n_1 + 2n_2 + 3n_3 \in S.$$

Thus, the number of splits of such an atom is

$$\sum_{(n_1,n_2,n_3)\,:\,n_1+2n_2+3n_3\in S} (n_1 + 1)(n_2 + 1)(n_3 + 1).$$

In the worst case, for the multivalent atom S, the allowed sums are

$$S = \{2, 4, 6\},$$

and the number of splits is 72.

Bounding bridge bonds.

All external bonds are either at R^* or the terminal atom. A terminal atom bond split thus also determines which external bonds are at R^*.

Bounding external bond-type multisets.

The number of external bond-type multisets over `SINGLE_ATOM_BOND_TYPES` of size at most E is

$$\mathrm{MS}(\leq E, T) := \binom{E + T}{T},$$

where $T = 3$ denotes the number of bond-types. This follows by the stars-and-bars argument (E stars, T bars).

Total count. The resulting total count is

$$\boxed{8 \cdot 72 \cdot \binom{E + 3}{3}.}$$

N.2 Category B

Bounding ring topologies. For counting ring topologies, we use the necklace formula, which counts all choices of atom types and bond types around a cycle, modulo rotational and reflective symmetry. Let A_r denote the number of ring-atom choices and T_r the number of ring-bond choices. We assume

$$|A_r| = 5 \qquad \text{and} \qquad |T_r| = 2,$$

corresponding to ring atoms C, O, S, N with N allowed to have one or two explicit hydrogens, and ring bonds being single or double.

Let D_r be the dihedral symmetry group, with r rotations and r reflections. Let $\mathrm{Fix}(g)$ be the set of ring configurations that are preserved by symmetry operation g. For any symmetry $g \in D_r$, we have $|\mathrm{Fix}(g)| = q^{c(g)}$, where $c(g)$ is the cycle count of g. If $g \in D_r$ is not the identity, then $c(g) \leq \lceil r/2 \rceil + 1$ with the upper bound achieved for the vertex-axis reflection of even r. Substituting into Burnside's lemma gives the closed-form upper bound $\frac{q^r + (2r-1)q^{\lceil r/2 \rceil + 1}}{2r}$, where $q = |A_r| \cdot |T_r| = 10$.

Bounding external node assignments. For a given ring topology of r atoms, each atom has free valence F after subtracting its two ring bonds from `ATOM_VALENCES`. To obtain a worst case bound, we assume that both of its ring bonds are single bonds. It can fill the remaining valence using single or double bonds. For multivalent atoms (S), only the valences in $[2, 4, 6]$ are allowed, as specified by `VALID_VALENCES`.

For example, a carbon atom (C) in the ring has $F = 2$, so the extra bonds can have total order in

$$\{0, 1, 2\}.$$

Table 25: We count bond-type multisets that satisfy valence and bond order constraints for varying external node budgets.

k (ext nodes used)	valid multisets from $\{1,2\}$ with sum $\in \{0,2,4\}$	count
0	$\emptyset \Rightarrow$ sum 0	1
1	(2) $\Rightarrow$ sum 2	1
2	(1,1) $\Rightarrow$ sum 2; (2,2) $\Rightarrow$ sum 4	2
3	(1,1,2) $\Rightarrow$ sum 4	1
4	(1,1,1,1) $\Rightarrow$ sum 4	1

The only bond-type multisets (with $1 =$ single and $2 =$ double) allowed with k extra bonds are

$$\{\emptyset\} \quad (k=0), \qquad \{\{1\},\{2\}\} \quad (k=1), \qquad \{\{1,1\}\} \quad (k=2),$$

so there are 1, 2, and 1 possibilities for $k = 0, 1, 2$, respectively. We represent these counts by the generating function

$$g_C(x) = 1 + 2x + x^2.$$

For S, we have $F = 4$, but its allowed extra valences are only $[0, 2, 4]$, so we can count the bond-type multisets as in Table 25.

The generating function. The generating function is

$$g_S(x) = 1 + x + 2x^2 + x^3 + x^4 = (x^2+1)(x^2+x+1).$$

We use $g_S(x)$ for every ring atom to obtain the worst-case bound.

For r atoms, the total external-assignment count with budget E is bounded by

$$G(r, E) := \sum_{j=0}^{E} [x^j]\, g_S(x)^r,$$

that is, the sum of the coefficients of $x^0, \ldots, x^E$ in $g_S(x)^r$.

Bounding R^* choices. Now that we have accounted for ring choices, we account for the choices made by R^*.

If we have a terminal rule, then there is no choice to be made, and $G(r, E)$ keeps track of the external assignments. Otherwise, R^* connects to exactly one of the r ring atoms, and we may use $G(r-1, E)$ to keep track of the external assignments for the other $r-1$ ring atoms.

The connected ring atom has external-node assignment constrained by the fact that at least one of its bonds is sent to R^*. If it has n_1 single bonds and n_2 double bonds, the number of splits is

$$(n_1+1)(n_2+1).$$

If the atom has free valence F, then the number of such assignments is

$$\sum_{(n_1,n_2)\,:\,n_1+2n_2\le F} ((n_1+1)(n_2+1)-1).$$

For the multivalent worst case S, we have $F = 4$, but the allowed sums are

$$\{0, 2, 4\},$$

so

$$\sum_{(n_1,n_2)\,:\,n_1+2n_2\in\{0,2,4\}} ((n_1+1)(n_2+1)-1) = 14.$$

Total count for Category B. Thus the total count is

$$\boxed{\sum_{r=3}^{R} \frac{10^r + (2r-1)\cdot 10^{\lceil r/2\rceil+1}}{2r} \binom{E+2}{2} \Big[G(r,E) + r\cdot 14\cdot G(r-1,E)\Big].}$$

Table 26: Ring topology counts for $r = 3, 4, 5, 6$.

r	$(10^r + (2r-1)\cdot 10^{\lceil r/2\rceil+1})/(2r)$	$G(r, E)$	$G(r-1, E)$	$\mathrm{Cat}_B + \mathrm{Cat}_C$
3	1000	127	33	58,996,000
4	2,125	370	127	626,535,000
5	19,000	892	370	20,158,544,000
6	92,500	1,883	892	282,110,570,000

N.3 Category C

The number of ring topologies is the same. Now R^* replaces one of the atoms, giving r choices. All of its extra bonds are external nodes, giving $g_S(1) = 6$ choices, together with

$$\binom{E+2}{2}$$

bond-type multisets. The remaining $r-1$ atoms use the same generating function, giving the assignment count $G(r-1, E)$.

Total count for Category C. Hence the total count is

$$\boxed{\sum_{r=3}^{R} \frac{10^r + (2r-1)\cdot 10^{\lceil r/2\rceil+1}}{2r} \cdot r \cdot 6 \cdot \binom{E+2}{2} \cdot G(r-1, E).}$$

N.4 Combined bound

Thus, the combined count is bounded by

$$\boxed{576 \cdot \binom{E+3}{3} + \sum_{r=3}^{R} \frac{10^r + (2r-1)\cdot 10^{\lceil r/2\rceil+1}}{2r} \cdot \binom{E+2}{2} \cdot \Big[G(r, E) + 20 \cdot r \cdot G(r-1, E)\Big].}$$

N.5 Numerical evaluation

We evaluate the bound for $E = 6$ and different values of r in Table 26.

$$\mathrm{Cat}_A = 8 \cdot 72 \cdot \binom{6+3}{3} = 48{,}384.$$

In particular, in the case $r = 6$, we have

$$\mathrm{Total} \le 48{,}384 + 302{,}954{,}645{,}000 \approx 3.0 \times 10^{11}.$$

Remark 4. Using a program, we enumerated 50,082,289 atomic rules (6,049× fewer than the upper bound) which GRAMOLE embeds as its rule vocabulary.

The main sources of the looseness of this bound are:

(1) **Worst-case free valence for all ring atoms.** The generating function $g(x) = 1+x+2x^2+x^3+x^4$ with $g(1) = 6$ assumes every ring atom is sulfur (free valence $F = V - 2 = 4$, valid sums $\{0, 2, 4\}$). In reality most ring atoms are carbon ($g_C(1) = 4$), nitrogen ($g_N(1) \in \{1, 2\}$), or oxygen ($g_O(1) = 1$). Since the per-structure count scales as $g(1)^r$, using g_S instead of a weighted mixture inflates the bound by a factor of roughly $(6/3)^r$ for a ring of size r.

(2) **Ring structure overcount.** The closed-form Burnside-based UB $\frac{q^r+(2r-1)q^{\lceil r/2\rceil+1}}{2r}$ caps every non-identity element of D_r at the worst-case fixed-point count (vertex-axis reflection of even r); this is loose by 5–10% versus the exact Burnside count, and additionally ignores chemical validity (e.g. O cannot have two double ring bonds, adjacent triple bonds violate valence). The actual number of valid ring structures is $\sim 10\times$ smaller than the bound for $r = 6$.

(3) **Independent bridge-bond factor.** The bound multiplies by $\binom{E+T_r}{T_r}$ for bridge bonds at every assignment, assuming the full budget E is always available. In reality the remaining budget is $\min(E - n_{\text{ext}},\ D - n_{\text{rstar}})$, which is often zero once the atoms consume their share. The $\binom{8}{2} = 28$ bridge multisets are mostly vacuous.
(4) **Worst-case rstar atom.** Phase 2 of Category B uses $h = P_r - g(1) = 14$, again assuming the R^* atom is S. A carbon R^* atom would contribute only $h_C = 3$, so the factor of $r \cdot h \cdot G(r-1, E)$ overcounts by roughly $14/3 \approx 5\times$ for typical rings.
(5) **No symmetry deduplication in Phase 2.** The bound multiplies by r for the choice of which atom connects to R^*, even though symmetric ring positions collapse under the rotation-based equivalence used in the enumeration itself.
(6) **No constraint coupling between categories.** The bridge and per-atom budgets interact (both draw from E and D), but the bound treats the bridge count as an independent multiplicative factor, ignoring that large bridge multisets leave little room for atom bonds.

O Case Study: AMES Improvement

In the example Fig. 6, the optimized molecule is derived from the scaffold of Emedastine, an H1 antihistamine used to treat allergic conjunctivitis. The molecule contains a heteroaromatic core linked to an amine-rich side chain. Predictive models can assign elevated AMES mutagenicity risk to molecules with aromatic heterocycles and amine functionalities. We thus instruct GRAMOLE to make a limited number of targeted structural edits to reduce the mutagenicity risk, which refers to the ability of the compound to induce potentially carcinogenic changes in DNA. The generated molecule differs from the original through two localized structural changes. First, the molecule omits the eighth rule from the original molecule parse. This shortens the terminal substituent on the ether side chain from an ethyl group to a methyl group, reducing steric bulk and slightly decreasing lipophilicity. This can limit cytochrome-mediated activation at the terminal carbon and reduce the formation of reactive intermediates.

Second, the left-hand amine-containing ring system is simplified from a seven-membered ring with two nitrogen atoms to a six-membered ring containing a single nitrogen, corresponding to switching out the second rule from the input molecule parse. This modification decreases heteroatom density and basicity, reducing the likelihood of forming highly protonated species that interact strongly with DNA. This also removes a potential site for activation such as N-oxidation. Contracting the ring from seven-membered to six-membered further restricts conformational flexibility, which could reduce the molecule's ability to adopt geometries associated with mutagenic interactions such as intercalation.

These edits arise from the underlying grammar representation, where the model selectively modifies a small subset of rules within this sequence. In this case, removing one rule and substituting another is sufficient to induce the observed change in mutagenicity. This enables controlled exploration by focusing on interpretable changes to the generative process. We note that this approach is limited in that it does not guarantee that the structural changes do not weaken the molecule's original intended activity. In particular, removing the additional tertiary amine could disrupt binding interactions, so the optimized molecule may no longer retain the same efficacy as the original design.

As a further remark, the rule sequences above construct the molecule not strictly along a single DFS-like traversal path. The parse first builds the left amine-containing ring (rule 2), then the right aromatic ring (rule 3), before the central heteroatom ring (rule 4) which connects these regions. This allows the grammar to represent chemically meaningful substructures as separate units before resolving how they are joined together. The nonlinear construction order gives the representation additional flexibility. In this example, the side-chain amine ring and aromatic ring rules can each be modified sequentially before the rule adding the middle heterocycle, as this can be treated as a later decision that organizes their relationship. This can make it easier to perform localized edits on outer moieties, such as the example provided. We also argue that it better represents the graph-like structure of small molecules in cases of central symmetry (see Fig. 1).

P Case Study: Rule Interpretability

In addition to the quantitative evaluation of rule contraction probabilities (see 24), we select a few rules here to examine from a rule interpretability perspective. We demonstrate through the following

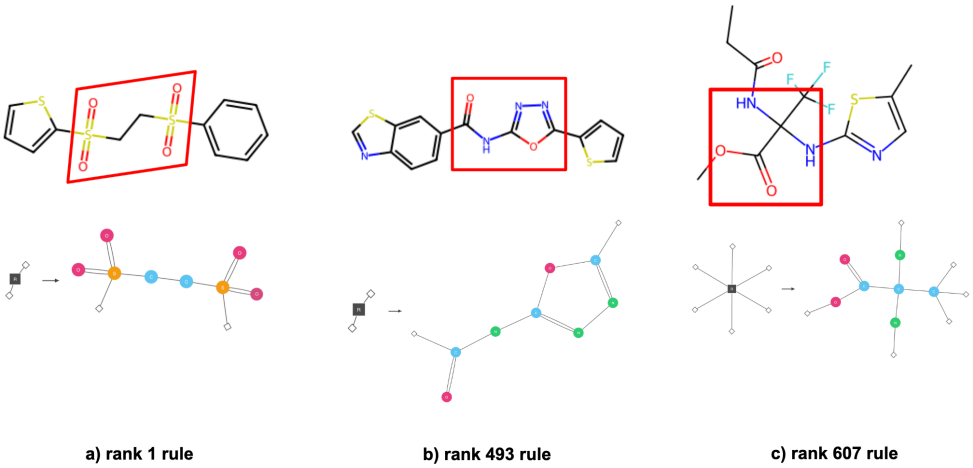


Figure 10: Selected highly ranked rules by fractional difference between CRM potential function sampling probability and null

case studies that ALPHAGRAMMAR's potential function favors sampling interpretable and chemically meaningful rules. Here, we use the CRM potential function and rank the top rules by fractional difference from null, in which node sampling probabilities are fixed at 0.5, across 9300 randomly sampled molecules. The following are some associated ranks and contraction probabilities relative to null.

Rule 1: 170% more likely to be selected than null Rule 1 is a bisulfone linker which appears in bioactive molecules as a polarity-controlling scaffold. Sulfone groups are strongly electron-withdrawing. They frequently serve as linker motifs in medicinal chemistry due to their ability to increase aqueous solubility, modulate lipophilicity, and enforce characteristic three-dimensional conformations. The paired sulfones define a coherent pharmacophoric unit that controls the spatial separation of the two aromatic systems. From a grammar perspective, preserving the entire bis-sulfone linker as a single fragment is substantially more interpretable than decomposing it into atom-level substructures, since the motif encodes a reusable higher-order chemical abstraction.

Rule 493: 69% more likely to be selected than null This rule corresponds to a carbamoyl–oxadiazole linker motif, consisting of an amide directly connected to a 1,3,4-oxadiazole heterocycle. This simultaneously controls hydrogen-bonding behavior, aromaticity, rigidity, and electronic distribution. In particular, 1,3,4-oxadiazoles are widely used as bioisosteres for esters and amides due to their metabolic stability, planarity, and strong electron-withdrawing character, while adjacent amide groups provide directional hydrogen-bond donor and acceptor functionality important for protein binding. Closely related motifs appear in numerous bioactive molecules, such as furamizole derivatives. It is reasonable for the potential function to treat the amide–heteroazole linkage as a unit because it rigidifies molecular geometry while preserving polar interactions.

Rule 607: 67% more likely to be selected than null This rule represents a quaternary carbon that serves as a rigid branching hub that connects a trifluoromethyl-containing substituent, a heterocycle, and an amide. Here, the central carbon defines the core topology of the surrounding scaffold by organizing multiple pharmacophoric groups around a single stereochemically constrained center. It is analogous to quaternary-carbon-containing cores found in many bioactive compounds, including kinase inhibitors and CNS-active heterocyclic drugs where rigid carbon branching points are used to optimize binding geometry. Additionally, note that this contraction would have been rewarded during training of the CRM potential function since the upper carbonyl-containing branch can be produced from amide coupling.

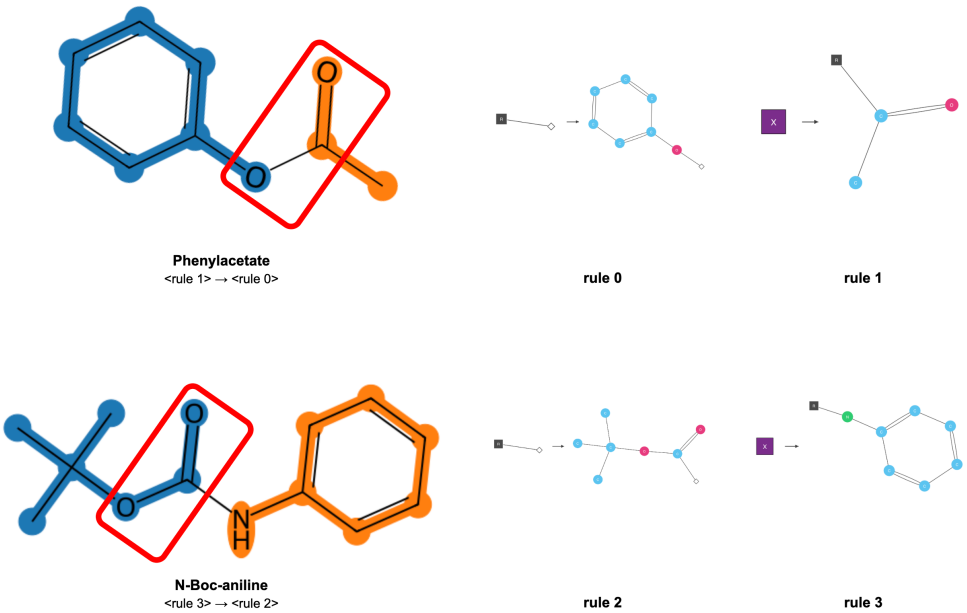


Figure 11: Comparison of phenylacetate and N-Boc-aniline parse

Q Case Study: Context-Aware Contraction

AlphaGrammar's learnable potential function makes contraction decisions that incorporate surrounding context, allowing discovery of chemically meaningful moieties.

For phenylacetate (11), the parse separates the O...C=O bond. The application of the second rule can be intuitively interpreted as the attachment of an acetyl group to phenol through an esterification reaction.

Now consider the parse for N-Boc-aniline. Here, the left part (tert-butyloxycarbonyl) is the Boc protecting group, a widely-used protecting group in synthetic chemistry that is installed or removed from NH2 under mild conditions. This parse does not separate the O...C=O bond. Instead, the contraction separates HN...C=O bond, which corresponds to the removal of the protecting group.

Comparing these two cases, the parser makes a different contraction decision at the O...C=O bond due to the surrounding molecular environment. By comparison, deterministic heuristic-based approaches do not natively support the nuance to adapt contraction decisions based on context. For instance, mCLM's (Edwards et al. (2025)) atomic fallback enforces rigid rules such as always cleaving at amide bonds or between benzene rings. The result is that they may either over-fragment chemically coherent groups or fail to factor chains into smaller, reusable units.