# Neural Proposals, Symbolic Guarantees: Neuro-Symbolic Graph Generative Modeling

**Chuqin Geng**[1,2] **Li Zhang**[2] **Mark Zhang**[2] **Zhaoyue Wang**[1] **Haolin Ye**[1] **Xujie Si**[2]

[1]School of Computer Science, McGill University
[2]Department of Computer Science, University of Toronto

chuqin.geng@mail.mcgill.ca zli@cs.toronto.edu six@cs.toronto.edu

## Abstract

While deep generative models excel at capturing graph data distributions, they struggle to satisfy complex, hard constraints. In unconstrained settings, these models typically produce valid topologies; yet imposing strict compositional rules, like those in drug discovery, creates an out-of-distribution (OOD) setting where purely neural methods frequently fail. Because these neural approaches rely on soft conditioning and post-hoc filtering on such tasks, they cannot provide the formal guarantees needed for high-stakes domains. To address this, we introduce Neuro-Symbolic Graph Generative Modeling (NSGGM), a framework built on the principle of *Neural Proposals, Symbolic Guarantees*. NSGGM decouples generation: an autoregressive model proposes structural scaffolds, and a Satisfiability Modulo Theories (SMT) solver handles the final discrete assembly of the proposed substructures. Empirically, NSGGM is competitive with state-of-the-art methods on unconstrained tasks. To evaluate logical-constraint satisfaction inspired by real drug discovery workflows, we introduce MOLSAT, a benchmark for hard compositional rules. On MOLSAT, the evaluated neural baselines achieve 0% satisfaction on zero-training-support tasks, while NSGGM achieves >95% satisfaction on tasks with training support and 64–86% with zero training support.

## 1 Introduction

Deep generative models, ranging from autoregressive models to diffusion processes, have revolutionized the exploration of complex data distributions. In the context of graph-structured data, such as molecular discovery [19, 6] and electronic circuit design [7], these models exhibit a remarkable ability to capture high-dimensional features and propose novel topologies. Through implicit optimization, neural generators effectively handle soft objectives like property targets, distribution matching, and continuous conditioning. But these mechanisms provide no per-sample guarantees on hard logical constraints, and offer no transparency into whether any given sample actually satisfies a user's specification. In downstream tasks where users demand strict compositional rules, like medicinal-chemistry filters or scaffold requirements, the feasible set is often entirely out-of-distribution (OOD), and soft conditioning produces neither a checkable certificate of satisfaction nor a usable signal of infeasibility.

Existing approaches handle these constraints implicitly, biasing latent distributions toward seed examples [24] or using structural fragments as inpainting context [17]. The standard fallback is post-hoc filtering: generate many candidates, discard those that violate the specification. This generate-and-filter loop is wasteful when the constraint is rare in the training distribution, and gives the user no recourse when it is unsatisfiable, often returning no satisfiable examples after exhausting the sampling budget [24]. We first introduce MOLSAT, a benchmark designed to test compositional logical-constraint satisfaction across four distinct reasoning families. Grounded in real-world drug discovery pipelines, MOLSAT tasks range from simple filter cascades (*Family I*) to

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

complex diagnostic stress tests involving non-local logical coupling (*Family IV*). Unlike existing benchmarks, MOLSAT evaluates complex Boolean logic over substructure predicates, specifically probing the gap between distributional matching and logical reasoning across three difficulty tiers: *High-support, Low-support,* and *Zero-support*. On constraints with no satisfying GuacaMol training examples (Table 4), the tested neural baselines produce no satisfying samples within the evaluation budget. Zero support here concerns new combinations of retained motifs and slot variants, not unseen vocabulary primitives.

To close this gap, we introduce **N**euro-**S**ymbolic **G**raph **G**enerative **M**odeling (NSGGM), a neuro-symbolic generative model that separates the distribution learning of neural networks from logical reasoning. Our approach is built on the principle of *Neural Proposals, Symbolic Guarantees*. Rather than forcing a neural model to implicitly memorize structural patterns, we utilize a lightweight generator to capture the data-driven prior by proposing high-level subgraph tokens and interaction guidance. Simultaneously, we define explicit, symbolic assembly constraints over these components. By offloading the final discrete assembly to a Satisfiability Modulo Theories (SMT) solver [4], NSGGM enforces structural rules by construction. This symbolic architecture naturally supports user-specified constraints that are entirely out-of-distribution. Evaluated on MOLSAT, NSGGM successfully generalizes to constraints with zero training support. In summary, our contributions are:

- **A neuro-symbolic framework with verifiable guarantees:** We propose NSGGM, a two-stage pipeline in which a neural model proposes motif-level scaffolds and a SMT assembles the final graph. Chemical and user-specified structural rules are enforced as hard clauses.
- **Auditable, retraining-free control for high-stakes domains:** Because constraints are natively handled in the symbolic layer, domain experts can directly encode structural rules like scaffolds, exclusion patterns, compositional logic as hard specifications, allowing for auditable control for high-stakes domains under the explicitly encoded rules.
- **The MolSAT benchmark.** To evaluate this capability, we introduce MOLSAT, a benchmark of families for compositional logical-constraint satisfaction inspired by drug-discovery patterns, designed to evaluate whether generative models can reason over hard logical specifications rather than merely learning the training distribution.

## 2 Preliminaries

### 2.1 Graph Generation

The goal of graph generation is to learn the distribution underlying a dataset $\mathcal{G} = \{G_1, \ldots, G_n\}$ and synthesize novel samples from it. We consider attributed graphs $G = (V, E, \ell_V, \ell_E)$ with node-label map $\ell_V : V \to \mathcal{X}$ and edge-label map $\ell_E : E \to \mathcal{E}$ over discrete label sets, the standard setting in molecular design [32]. The task is to learn $P_\theta(G)$ approximating the data distribution and sample $G' \sim P_\theta(\cdot)$. Existing approaches fall broadly into autoregressive models that emit nodes, edges, or motifs sequentially [24, 10]; one-shot models, including diffusion and flow-based methods, that generate the full graph through iterative refinement of a noise prior [32, 36]; and fragment-based methods that assemble graphs from substructure vocabularies [14]. While these methods achieve strong generative quality on unconstrained benchmarks, they provide only soft control over output structure, lack transparency into why a particular graph was generated, and cannot give formal guarantees that generated graphs satisfy user-specified compositional rules.

### 2.2 SMT Solvers

A Satisfiability Modulo Theories (SMT) solver decides whether a quantifier-free first-order formula $\phi$ over a background theory (e.g., linear integer arithmetic, uninterpreted functions) admits a satisfying assignment. On `SAT`, it returns a model $M \models \phi$; on `UNSAT`, it reports that no such model exists, optionally accompanied by an *unsat core*: a subset of the input clauses whose conjunction is already infeasible; and on `UNKNOWN`, no verdict, typically due to timeout, incomplete theory reasoning, or numerical issues. Our implementation uses Z3 [8].

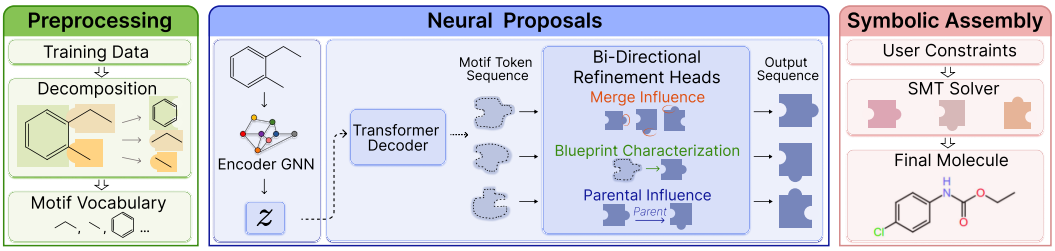



Figure 1: An overview of the NSGGM framework.

## 3 The NSGGM Framework

NSGGM treats graph generation as a compositional task. From the training set, we build a vocabulary of *motifs* and their *slot variants* record how each motif attaches at its atoms and shared edges (Section 3.1). A neural decoder then proposes a *blueprint* of motif–variant pairs with neural connection guidance, and an SMT solver assembles the corresponding tiles under hard structural, chemical, and user-specified constraints, treating the neural influence as a soft prior (Figure 1).

### 3.1 Decomposition & Vocabulary

We pre-process an input graph $G = (V, E, \ell_V, \ell_E)$ into a *blueprint* $S_G$, the interface the neural decoder (Section 3.2) emits and the SMT solver (Section 3.3) assembles. The construction consists of three steps: *tile decomposition, slot annotation, and vocabulary formation*.

**Definition 3.1** (Tile)**.** A *tile* of $G$ is a tuple $t = (V_t, E_t, S_t)$ with $V_t \subseteq V$, $E_t \subseteq E[V_t]$, and shared-edge set $S_t \subseteq E_t$. A tile inherits labels from $G$ via $\ell_V, \ell_E$.

**Tile decomposition.** We follow a junction-tree-style decomposition [14]: $E$ splits into bridges $E_{\text{br}}$ and ring edges $E_{\text{ring}} = E \setminus E_{\text{br}}$. Ring tiles are the cycles of a minimum cycle basis of $(V, E_{\text{ring}})$ [12]; acyclic tiles are produced on $(V, E_{\text{br}})$ by bounded edge-consumption BFS. We denote the tile multiset $\mathcal{T}(G) = \{t_i\}_{i=1}^{K}$ and linearize it via a randomized BFS over the *tile graph* $H(G) = (\mathcal{T}(G), E_H)$, in which two tiles are adjacent iff they share at least one slot atom or slot edge.

**Slot annotation.** Define $b_G(v)(\tau)$ as the count of bond type $\tau \in \mathcal{B} = \{\texttt{single}, \texttt{double}, \texttt{triple}, \texttt{aromatic}\}$ at $v$ in $G$, and $b_t(v)(\tau)$ as the same count restricted to tile $t$. The *node slot* of $v$ in $t$ is the multiset of bond-type stubs pointing outside the tile:

$$\sigma_v^t(\tau) \;=\; b_G(v)(\tau) - b_t(v)(\tau).$$

For each shared edge $e \in S_t$ we record its bond label $\sigma_e^t(e) := \ell_E(e)$. A *slot-annotated tile* is $\tilde{t} = (t, \sigma_v^t, \sigma_e^t)$, the unit the solver consumes: node slots determine which atoms across tiles can be fused, and edge slots determine which shared edges can be fused (Section 3.3).

**Vocabulary.** We identify each tile by two Weisfeiler–Lehman hashes [30]: a *structural* hash $h_{\text{struct}}(t)$ over $(V_t, E_t, \ell_V|_{V_t}, \ell_E|_{E_t})$, and a *metadata* hash $h_{\text{meta}}(\tilde{t})$ over $(V_t, E_t, \ell_V|_{V_t}, \ell_E|_{E_t}, \sigma_v^t, \sigma_e^t)$. We define a motif as an equivalence class under $h_{\text{struct}}$, and a slot variant as one under $h_{\text{meta}}$. We write $\mu$ for a motif identifier ($h_{\text{struct}}$) and $\nu$ for a slot-variant identifier ($h_{\text{meta}}$); $\nu$ refines $\mu$.

We form a vocabulary $\mathcal{V}$ by aggregating across the training set, retaining motifs and their slot variants by frequency. Let $\Omega(\mu)$ denote the retained slot variants for motif $\mu$, and let $\tilde{t}_{\mu,\nu}$ be the canonical slot-annotated tile for key $(\mu, \nu)$. The vocabulary is $\mathcal{V} = \{\tilde{t}_{\mu,\nu} : \mu \in \text{motifs},\ \nu \in \Omega(\mu)\}$.

**Blueprint.** Pre-processing $G$ yields a sequence of vocabulary keys:

$$S_G \;=\; \big((\mu_i,\, \nu_i)\big)_{i=1}^{K}, \qquad \mu_i = h_{\text{struct}}(t_i),\;\; \nu_i = h_{\text{meta}}(t_i),$$

which we call the *blueprint* of $G$. At inference, the neural component *proposes* a blueprint $\hat{S} = ((\hat{\mu}_i, \hat{\nu}_i))_{i=1}^{K}$, and the SMT solver (Section 3.3) retrieves the corresponding tiles $\{\tilde{t}_{\hat{\mu}_i,\hat{\nu}_i}\}$ from $\mathcal{V}$ and assembles them into a graph. Figure 2 gives a concrete example; decomposition details and proofs appear in Appendix A.

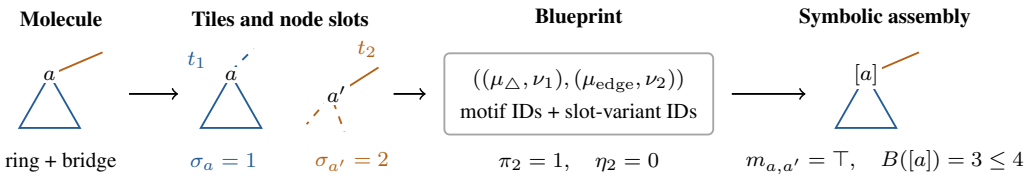



Figure 2: Worked decomposition and assembly of methylcyclopropane (implicit hydrogens; all bonds single). The ring tile and bridge tile share carbon $a$. Its node-slot counts are $\sigma_a^{t_1} = 1$ and $\sigma_{a'}^{t_2} = 2$. The blueprint records each motif and its slot variant; merging the two copies of $a$ restores three incident bonds, within carbon's valency cap.

### 3.2 Neural Proposals

The neural component proposes a blueprint $\hat{S}$ together with guidance signals that bias, but never *override*, the symbolic assembler. We model this with a GNN VAE encoder that maps input graphs into a latent $z$, then a causal Transformer decoder emits a motif sequence, and finally a bi-directional refinement network attaches per-position guidance.

**Tile-graph encoder.** We reuse the tile graph $H(G) = (\mathcal{T}(G), E_H)$ from Section 3.1. Each node carries $(\mu_i, \nu_i)$, and each edge is annotated with $r_{ij} \in \{0, 1\}$ indicating whether the overlap contains a shared edge. A stack of GINE-Conv layers [13] followed by mean-pooling produces a graph embedding $h_G$, which parametrizes a Gaussian variational posterior $q_\phi(z \mid G)$.

**Autoregressive decoder.** A causal Transformer decoder, conditioned on the encoder latent $z$, generates a motif sequence $\mathcal{M} = (\mu_1, \ldots, \mu_K, \texttt{<eos>})$ over the motif identifiers, factorizing causally over positions. Sequence length $K$ equals $|S_G|$ on training inputs and is determined by the `<eos>` token at inference.

**Bi-directional refinement.** A bidirectional Transformer $\phi_{\text{ref}}$ maps the decoder's per-position hidden states to refined states $\{\mathbf{h}_i\}_{i=1}^K$, deterministic in $(\mathcal{M}, z)$. Three conditionally independent heads read from $\{\mathbf{h}_i\}$: a *metadata head* predicting the slot-variant $\nu_i$ over $\Omega(\mu_i)$ via a masked softmax; a *parent-pointer head* predicting an attachment index $\pi_i \in \{1, \ldots, i-1\}$ via scaled bilinear attention over preceding refined states; and a *merge-type head* predicting a binary $\eta_i \in \{0, 1\}$ (node-merge vs. edge-merge) via sigmoid. Combined with $\mathcal{M}$, the metadata head completes the proposed blueprint $\hat{S} = ((\mu_i, \nu_i))_{i=1}^K$ that the solver consumes; the triple $(\nu_i, \pi_i, \eta_i)$ is the per-tile guidance the solver uses as a soft prior in Section 3.3.

**Training objective.** Let $y = (\mathcal{M}, \nu_{1:K}, \pi_{2:K}, \eta_{2:K})$ denote the supervised targets from $G \sim \mathcal{D}$. We optimize the standard ELBO:

$$\mathcal{L}(\theta, \phi) \;=\; \mathbb{E}_G \, \mathbb{E}_{z \sim q_\phi(z|G)} \big[\log p_\theta(y \mid z)\big] \;-\; \beta \, \mathrm{KL}\big(q_\phi(z \mid G) \,\|\, p(z)\big),$$

where $p_\theta(y \mid z)$ factorizes across the four supervised heads independently for tractability:

$$p_\theta(y \mid z) \;=\; p_\theta(\mathcal{M} \mid z) \prod_{i=1}^{K} p_\theta(\nu_i \mid \mathcal{M}, z) \prod_{i=2}^{K} p_\theta(\pi_i \mid \mathcal{M}, z) \, p_\theta(\eta_i \mid \mathcal{M}, z),$$

The full conditional factorization, and per-head loss weighting are given in Appendix B.

### 3.3 Symbolic Assembly

Given a neural blueprint proposal $\hat{S} = ((\mu_i, \nu_i))_{i=1}^K$, the symbolic assembler retrieves the corresponding tiles $\tilde{t}_i = \mathcal{V}[\mu_i, \nu_i]$ from the vocabulary and decides which slot atoms across tiles are identified, creating a final graph $G' = (V', E')$. This is encoded as an SMT problem $\Phi = \phi_{\text{hard}} \wedge \phi_{\text{user}}$, with the neural guidance $(\pi_i, \eta_i)_{i=2}^K$ as soft preferences in a Max-SMT layer. The hard constraints enforce the encoded structural and chemical rules by construction (Corollary C.6); the user layer enforces custom compositional rules on substructures; the soft layer transmits the neural model's distributional signal, which attachments are typical of the training data, into the search space, never overriding the hard or user constraints.

**Decision variables.** For each tile $\tilde{t}_i$, let $V_i^{\text{slot}} = \{v \in V_{t_i} : \sigma_v^{t_i}(\tau) > 0 \text{ for some } \tau \in \mathcal{B}\}$ be its slot atoms, and $V^{\text{slot}} = \bigcup_{i=1}^{K} V_i^{\text{slot}}$. We define $\text{sg}(u) \in \{1, \ldots, K\}$ as the tile that contains $u$, and $\text{elem}(u)$, $\text{val}(u)$ for its element type and prescribed valency. The candidate-merge set is:

$$\mathcal{C} \;=\; \big\{\{u, v\} \subseteq V^{\text{slot}} \;:\; \text{sg}(u) \neq \text{sg}(v),\; \text{elem}(u) = \text{elem}(v)\big\},$$

and for each $\{u, v\} \in \mathcal{C}$ we represent this as a Boolean merge variable $m_{u,v} \in \mathbb{B}$ with $m_{u,v} = \top$ iff $u$ and $v$ are merged in $G'$. We also introduce a tile-participation indicator $\text{in_mol}_i \in \mathbb{B}$ for each tile, and substructure indicators $p_u \in \mathbb{B}$ for the user layer. For each $u \in V^{\text{slot}}$, we write $[u]$ for the equivalence class of $u$ under the transitive closure of the merge relation $\{(u, v) : m_{u,v} = \top\}$.

**Hard constraints $\phi_{\text{hard}}$.** $\phi_{\text{hard}}$ is a conjunction of six clauses enforcing topological and chemical validity of $G'$.

*(i) Element consistency.* A merge is admissible only between atoms of equal element type: $m_{u,v} \Rightarrow \text{elem}(u) = \text{elem}(v)$.

*(ii) Transitivity.* The relation $\{m_{u,v}\}$ is transitively closed across pairwise distinct tiles, ensuring that the induced equivalence classes $[u]$ are well-defined.

*(iii) Subgraph integrity.* Distinct slot atoms of a common tile cannot be identified under merging, so the inclusion $\tilde{t}_i \hookrightarrow G'$ is injective for every $i$.

*(iv) Valency cap.* For every $u \in V^{\text{slot}}$, the total bond order incident to $[u]$ in $G'$ satisfies $B([u]) \leq \text{val}(u)$, where $B([u])$ aggregates within-tile bonds at each $w \in [u]$, bonds inherited through merge partners, and a de-duplication correction for shared edges identified under clause (v).

*(v) Edge-slot consistency.* For shared edges $e = (u, v) \in S_{t_i}$ and $e' = (u', v') \in S_{t_j}$ with $i \neq j$, endpoint identification in either orientation is admissible only under bond-label agreement:

$$(m_{u,u'} \wedge m_{v,v'}) \;\vee\; (m_{u,v'} \wedge m_{v,u'}) \;\Longrightarrow\; \sigma_e^{t_i}(e) = \sigma_e^{t_j}(e').$$

When the antecedent holds, the identified edges collapse to a single edge in $G'$, and clause (iv) accounts for this de-duplication in the valency totals of $[u]$ and $[v]$.

*(vi) Connectivity.* Active merges entail tile participation, and every participating tile must be reachable from the canonical anchor (the lowest-indexed active tile) under a propositional depth-bounded BFS over the merge graph, guaranteeing connectivity of $G'$.

The full encoding is given in Appendix C.1.

**User constraints $\phi_{\text{user}}$.** A user specification is a pair $(U, \psi)$: a finite set of named substructures $U = \{u_1, \ldots, u_\ell\}$, and a propositional formula $\psi$ over $U$ using the operators $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$. Each $u \in U$ is specified by a tile block $T(u) \subseteq \{1, \ldots, K\}$ and a fusion set $F(u)$ of prescribed atom-level merges. We compile each $u$ into an indicator $p_u \in \mathbb{B}$ defined by

$$p_u \;\Leftrightarrow\; \Big(\bigwedge_{i \in T(u)} \text{in_mol}_i\Big) \wedge \Big(\bigwedge_{\{a,b\} \in F(u)} m_{a,b}\Big) \wedge \Big(\bigwedge_{\substack{\{x,y\} \in \mathcal{C} \\ \text{sg}(x), \text{sg}(y) \in T(u) \\ \{x,y\} \notin F(u)}} \neg m_{x,y}\Big)$$

so that $p_u$ holds iff every tile in $T(u)$ participates and every merge in $F(u)$ is realized. We additionally enforce atomicity via $\phi_u^{\text{atom}} := \bigwedge_{a,b \in T(u)} \big(\text{in_mol}_a \leftrightarrow \text{in_mol}_b\big)$. We then add the hard clause $\phi_{\text{user}} := \psi[u \mapsto p_u] \wedge \bigwedge_{u \in U} \phi_u^{\text{atom}}$ to the solver, making logical combinations of substructure presence enforceable by construction. Appendix C.2 gives the propositional reduction of $p_u$.

**Soft guidance via Max-SMT.** The parent-pointer and merge-type predictions $(\pi_i, \eta_i)_{i=2}^{K}$ are encoded as soft clauses with weights $w_\pi, w_\eta$, biasing the solver toward attachments consistent with the neural prior without interfering with $\phi_{\text{hard}} \wedge \phi_{\text{user}}$. For each $i \geq 2$, a parent-influence clause rewards any merge between tile $i$ and its predicted parent $\pi_i$, and a merge-type clause rewards the predicted node- or edge-merge orientation. The slot-variant $\nu_i$ is consumed at the hard level (it determines which tile $\mathcal{V}[\mu_i, \nu_i]$ the solver receives) and does not appear in the soft layer. The solver maximizes cumulative satisfied soft-clause weight, falling back to alternative attachments only when the predicted parent or merge type conflicts with the hard constraints. The full encoding is given in Appendix C.3.

**Inference, guarantee, and transparency.** Z3 returns one of three outcomes: SAT, UNSAT, or UNKNOWN (the last covering timeouts, incomplete theory reasoning, and numerical issues). On SAT, the satisfying assignment $\{m_{u,v}\}$ yields $G'$ by quotienting along the equivalence classes induced by the merges; the same assignment is itself a witness that $G'$ satisfies $\phi_{\text{hard}} \wedge \phi_{\text{user}}$, so every output is paired with a verifiable certificate of validity. On UNSAT, the solver returns an unsat core: a subset of clauses whose conjunction is already infeasible. This identifies which constraints cannot be jointly met over $\hat{S}$, allowing for actionable diagnoses on failures. Both UNSAT and UNKNOWN count as failed generation attempts; only UNSAT establishes infeasibility for the sampled blueprint. An unsat core identifies conflicting clauses and is distinct from an emitted proof object. Soundness under $\phi_{\text{hard}} \wedge \phi_{\text{user}}$ (Theorem C.4) and completeness for retained, representable training graphs (Theorem C.5) are proved in Appendix C.4; pseudocode for assembly (Algorithm 2) and inference (Algorithm 1) appears in Appendix D.

# 4 Experiments

We evaluate NSGGM along two objectives: (i) distribution fidelity and competitive performance under unconstrained generation on small and large scale benchmarks, and (ii) MOLSAT, our new benchmark of logical drug-discovery inspired specifications, which tests inference-time logical constraint satisfaction.

## 4.1 Unconstrained Generation

**Setup.** We evaluate on QM9 [34], MOSES [28], and GuacaMol [5]; dataset details are in Appendix E. For MOSES and GuacaMol, we use the official training, validation, and test splits and 80/10/10 split for QM9. Unless stated otherwise, unconstrained results are averaged over three runs on a single-core CPU, each generating 10k molecules; we report mean and standard deviation. Additional constrained-output and retry analyses use the protocols in Appendix H.

For distribution fidelity, we report Fréchet ChemNet Distance (FCD) [29] and KL divergence (KL) as GuacaMol-normalized scores [5] in $[0, 100]$, with 100 indicating a perfect match. Raw FCD compares generated and reference ChemNet activation distributions; raw KL aggregates divergences over physicochemical descriptors. We also report RDKit-based *Validity* and *Uniqueness*. Baseline values come from the original publications, preserving their tuned configurations and normalized to our reporting standard.

**Small-scale unconstrained generation.** Table 1 reports QM9 validity, uniqueness, FCD, and KL.

Table 1: Unconditioned molecule generation on QM9 implicit hydrogens. — indicates that the metric is not reported in the original paper.

| Model | Valid↑ | Unique↑ | FCD↑ | KL↑ |
|---|---|---|---|---|
| JT-VAE [14] | 100.0 | 54.9 | 58.8 | 89.1 |
| MOLER [24] | 100.0 | 94.0 | 93.1 | 96.9 |
| LO-ARM-ST-SEP [33] | 99.9 | 98.9 | 95.3 | — |
| DIGRESS [32] | $99.0 \pm 0.1$ | $96.2 \pm 0.1$ | — | — |
| UNIGEM [9] | $95.0 \pm 3.1$ | 98.1 | — | — |
| MICAM [10] | 100.0 | 93.2 | 94.5 | 98.0 |
| DISCO-GT [36] | $99.3 \pm 0.6$ | 99.5 | — | — |
| NSGGM (ours) | $100.0 \pm 0.0$ | $96.1 \pm 0.2$ | $94.7 \pm 0.4$ | $94.0 \pm 1.0$ |

**Large-scale generation.** We evaluate on MOSES [28] and GuacaMol [5], reporting results in Tables 2 and 3. For MOSES we evaluate on the *TestSF* split, whose scaffolds are disjoint from those seen during training.

**Analysis.** Across all three benchmarks, NSGGM achieves $100\%$ validity. Its GuacaMol normalized FCD exceeds most baselines while preserving high novelty; MOSES results combine high filter pass rates with competitive FCD and SNN. Symbolic assembly thus retains competitive distributional fidelity.

Table 2: Unconditional de novo molecule generation on GuacaMol. FCD and KL are GuacaMol-normalized scores in $[0, 100]$, higher is better.

| Model | Valid ↑ | Unique ↑ | Novel ↑ | FCD ↑ | KL ↑ |
|---|---|---|---|---|---|
| MOLER [24] | 100.0 | 100.0 | 99.1 | 62.5 | 96.4 |
| MICAM [10] | 100.0 | 99.4 | 98.6 | 73.1 | 98.9 |
| DIGRESS [32] | 85.2 | 100.0 | 99.9 | 68.0 | 92.9 |
| LSTM | 95.9 | 100.0 | 91.2 | 91.3 | 99.1 |
| DISCO-GT [36] | 100.0 | 86.6 | 99.9 | 59.7 | 92.6 |
| NSGGM (ours) | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $98.9 \pm 0.0$ | $86.1 \pm 0.4$ | $95.5 \pm 0.1$ |

Table 3: Unconditional de novo molecule generation on MOSES (*TestSF* split). Filters: fraction passing MOSES chemical filters. SNN: Tanimoto similarity to nearest test molecule. Scaf: Bemis–Murcko scaffold overlap. FCD: GuacaMol-normalized in $[0, 100]$, higher is better.

| Model | Valid ↑ | Unique ↑ | Novel ↑ | Filters ↑ | FCD ↑ | SNN ↑ | Scaf ↑ |
|---|---|---|---|---|---|---|---|
| JT-VAE [14] | 100.0 | 100.0 | 99.9 | 97.8 | 81.8 | 0.53 | 10.0 |
| MOLER [24] | 100.0 | 99.9 | 97.1 | — | 85.2 | — | — |
| DIGRESS [32] | 85.7 | 100.0 | 95.0 | 97.1 | 78.8 | 0.52 | 14.8 |
| GRAPHINVENT [26] | 96.4 | 99.8 | — | 95.0 | 78.3 | 0.54 | 12.7 |
| DISCO-GT [36] | 88.3 | 100.0 | 97.7 | 95.6 | 74.9 | 0.50 | 15.1 |
| NSGGM (ours) | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $94.4 \pm 0.3$ | $97.1 \pm 0.1$ | $82.3 \pm 0.3$ | $0.5 \pm 0.0$ | $11.2 \pm 0.1$ |

### 4.2 Constrained Generation

**The MolSAT Benchmark.** MOLSAT organizes tasks into four constraint families defined on GUACAMOL, each capturing a distinct reasoning pattern relevant to drug-discovery workflows. *Family I* (filter cascade) encodes structural-alert pipelines in the style of PAINS [3], combining required scaffolds with exclusion of flagged substructures. *Family II* (disjunctive selection) encodes chemistry queries in the style of bioisosteric replacement at the functional-group level and scaffold-hopping at the heterocyclic-core level, where multiple substructures can satisfy the same role [25][31]. *Family III* (hybrid compound) tests compositional reasoning patterns with scaffold requirements, exclusive functional-group selections, and SAR-style conditional implications; we instantiate two distinct Low-support formulas ($\varphi_1$, $\varphi_2$) in this family to probe robustness across different hybrid structures at the same difficulty tier. Finally, *Family IV* (coupled XOR and IFF) provides a diagnostic stress test over non-local logical coupling between exclusive substructure choices. Each family is instantiated at three difficulty tiers *(High-support (High), Low-support (Low), Zero-support (Zero))* that vary the constraint's coverage in the training distribution, probing the gap between distributional matching and logical reasoning. Full details of the benchmark are in Appendix F.3.

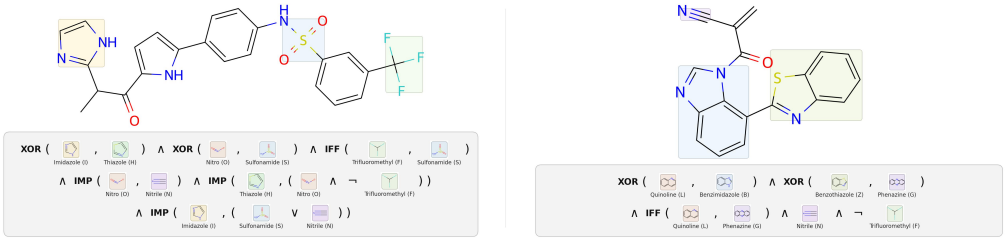



Figure 3: Two sample MOLSAT instances solved by NSGGM. **Left:** a low-support constraint $\varphi_2$ from Family III combining XOR, IFF, and IMP over substructure predicates. **Right:** a zero-support constraint from Family IV. In each panel, the highlighted substructures in the generated molecule correspond to the predicates referenced in the formula below, and the SMT solver returns a SAT certificate verifying the assignment.

**Baselines.** We compare against MOLER [24] and GENMOL [17] as two recent controllable generators that accept structural conditioning without task-specific finetuning.

All neural components are trained on GUACAMOL. As neither natively supports arbitrary logical constraints, we use a *generate-and-filter* evaluation strategy that gives each model its strongest native form of conditioning. On tasks with satisfying training examples, both are seeded with such molecules, where MOLER samples from a biased latent distribution fit to these seeds, and GENMOL uses them as inpainting context. We give a sampling budget of $\mathbf{10^6}$ molecule attempts for the neural baselines. For NSGGM, each MOLSAT constraint task is expressed directly as a user specification $(U, \psi)$ as specified in Section 3.3: the motif substructures become $U$, the task's logical formula becomes $\psi$, and the resulting $\phi_{user}$ is joined with $\phi_{hard}$ at inference. We perform no retraining or task-specific tuning and count both *UNSAT* and *UNKNOWN* outcomes as failed attempts. Hit% measures successful constraint-satisfying generations over all attempts, rather than compliance conditional on a returned assembly.

Table 4: Constraint satisfaction rate (Hit%) across 13 benchmarks in 4 families (I–IV), with tiers **H** = High, **L** = Low, and **Z** = Zero based on training coverage. MoLeR and GenMol receive a $10^6$-attempt budget per benchmark. Both neural baselines produce zero satisfying samples on all Zero-tier benchmarks. Best per column in bold.

| | I | | | II | | | III | | | | IV | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | H | L | Z | H | L | Z | H | L $\varphi_1$ | L $\varphi_2$ | Z | H | L | Z |
| MoLeR | 14.1 | 7.1 | 0.0 | 23.0 | 4.3 | 0.0 | 13.3 | 10.9 | 5.9 | 0.0 | 12.5 | 14.2 | 0.0 |
| GenMol | 64.4 | 37.6 | 0.0 | 51.0 | 48.6 | 0.0 | 59.2 | 31.5 | 28.3 | 0.0 | 61.5 | 48.5 | 0.0 |
| NSGGM (Ours) | **98.9** | **99.9** | **79.7** | **100.0** | **99.9** | **84.1** | **96.7** | **99.7** | **96.0** | **63.9** | **99.6** | **98.9** | **85.6** |

**Results.** From Table 4, across all four families, NSGGM achieves $\geq \mathbf{96}\%$ Hit on every *High*- and *Low*-support tier and is the only method that produces any satisfying molecule on Zero-support, where both neural baselines exhaust the sampling budget without a hit; Figure 3 shows a representative NSGGM output certified against Family III low-support $\varphi_2$ and zero-support Family IV. The gap supports the benefit of explicit assembly-time constraints in this evaluation. SAT assignments witness satisfaction of the encoded specification; UNSAT cores diagnose infeasibility over a particular proposed blueprint. Post-hoc filters can also record predicate outcomes, but do not establish that the blueprint admits no feasible assembly. Appendix F.3 reports benchmark definitions and constrained solver scaling.

### 4.3 Quality and Reliability under Constraints

**Diversity and distributional quality.** Constraint satisfaction alone does not establish useful sample diversity. Table 5 summarizes the quality of MolSAT-satisfying outputs: uniqueness is 99.2–100%, with high internal and scaffold diversity across all 13 tasks, providing evidence against severe collapse. This does not establish drug-likeness or conditional distribution matching: raw conditional FCD ranges from 14.37 to 34.61 on supported tasks, with small condition-matched reference sets limiting interpretation. At zero support, this training-reference FCD is undefined. Full per-task diversity, SNN, and FCD results appear in Appendix H.1.

Table 5: Quality of constraint-satisfying outputs. Each interval is the minimum–maximum across tasks in that tier, not a confidence interval.

| Tier | Tasks | Uniqueness (%) | Internal diversity | Scaffold diversity |
|---|---|---|---|---|
| High | 4 | 99.9–100.0 | 0.819–0.861 | 0.924–0.999 |
| Low | 5 | 99.2–100.0 | 0.758–0.829 | 0.828–0.997 |
| Zero | 4 | 100.0–100.0 | 0.798–0.826 | 0.975–1.000 |

**Failure diagnosis and blueprint resampling.** Table 6 separates infeasible blueprints (UNSAT) from 30 s solver timeouts (UNKNOWN), conditional on failed attempts. Timeouts dominate II-Zero failures; infeasibility dominates I/III/IV-Zero. In a separate experiment with 1,000 trials per task, we resample a fresh blueprint after either failure, stopping at the first success or after five total attempts, each with the same 30 s timeout. Observed cumulative Hit@5 reaches 98.5–100%, while every returned assembly remains subject to the same hard constraints. These are empirical success rates, not a guarantee of eventual success. A longer timeout may help UNKNOWN cases but cannot repair an UNSAT blueprint. Appendix H.2 gives all retry budgets and a worked failure example.

Table 6: Zero-support failure diagnosis and cumulative retry success. UNSAT and UNKNOWN percentages are *among failed single attempts*; Hit@$k$ is the fraction of 1,000 retry trials succeeding within $k$ total attempts.

| Task | UNSAT (%) | UNKNOWN (%) | Hit@2 (%) | Hit@5 (%) |
|---|---|---|---|---|
| I-Zero | 68.62 | 31.38 | 95.9 | 100.0 |
| II-Zero | 14.39 | 85.61 | 96.8 | 100.0 |
| III-Zero | 87.55 | 12.45 | 85.3 | 98.5 |
| IV-Zero | 59.62 | 40.38 | 96.8 | 100.0 |

### 4.4 Scalability Analysis and Ablation Study

Table 7: Solver scaling on unconstrained GuacaMol generation (single-core CPU), binned by proposed motif count $k$. $v$: mean merge-variable count; #HardClauses: formula size; Neural: inference time per molecule; Solver med/95p: median/95th-percentile solve time.

| #Motifs (k) | #Mols | #MergeVars (v) | #HardClauses | Neural (ms) | Solver med (ms) | Solver 95p (ms) | SAT Rate |
|---|---|---|---|---|---|---|---|
| 2–5 | 519 | 16.5 | 26.3 | 52.6 | 3.1 | 5.2 | 1.000 |
| 6–10 | 6,294 | 62.3 | 98.0 | 79.9 | 10.5 | 28.5 | 1.000 |
| 11–15 | 2,832 | 150.7 | 242.9 | 113.3 | 40.7 | 107.6 | 1.000 |
| 16–20 | 322 | 332.1 | 644.1 | 154.3 | 154.2 | 367.5 | 0.997 |
| 21+ | 35 | 665.7 | 1,744.6 | 202.1 | 546.4 | 830.9 | 1.000 |
| **Overall** | **10,004** | **95.7** | **158.6** | **90.7** | **15.1** | **99.2** | **1.000** |

**Scaling.** SMT solving has worst-case exponential complexity. Table 7 measures empirical scaling on unconstrained GUACAMOL generation: median solve time is 15.1 ms overall and remains sub-second for 21+ motifs, with SAT rates near 100%. Solve time scales approximately as $v^{1.4}$ in average merge-variable count and $k^{2.7}$ in motif count. Motif-level variables reduce the assembly problem relative to atom-level encodings, while neural guidance narrows candidate merges. Under additional MOLSAT constraints, the observed relationship is approximately $v^{1.45}$. These trends characterize the evaluated instances, not a general complexity guarantee; Appendix F.3 provides the constrained breakdown.

Table 8: Neural guidance ablation on GUACAMOL (10k molecules). Connectivity is relaxed; other constraints are fixed. Without blueprint characterization, each motif uses its most frequent variant. *Frag. %*: disconnected assemblies. *Tokens Lost %*: proposed tokens outside the largest connected component.

| Variant | FCD ↑ | KL ↑ | Frag. % ↓ | Tokens Lost % ↓ |
|---|---|---|---|---|
| Full | **86.1** | **95.5** | **0.0** | **0.0** |
| W/o parent pred. | 49.2 | 87.6 | 62.5 | 23.4 |
| W/o merge-type pred. | 63.0 | 88.3 | 22.8 | 8.4 |
| W/o blueprint char. | 62.4 | 88.9 | 0.0 | 0.0 |

**Ablation.** Table 8 isolates the contribution of each neural guidance signal in a diagnostic setting with hard connectivity relaxed and the remaining constraints held fixed. Parent prediction is the dominant connectivity signal: removing it leaves $62.5\%$ of assembled graphs disconnected and $23.4\%$ of proposed tokens unmerged, since the solver loses its primary cue for attachments. Merge-type prediction provides complementary structural guidance where ablating it yields less severe fragmentation ($22.8\%$) while still degrading distributional match. Blueprint characterization preserves connectivity ($0\%$ fragmentation) but lowers FCD, showing its role in selecting interface slots rather than driving structural assembly. In all, neither component alone suffices; the symbolic layer is necessary for hard-constraint satisfaction, while neural guidance is necessary for distributional fidelity.

## 5 Related Work

A wide selection of neural generative models have been applied to molecular graph generation, including autoregressive node, edge, and motif emitters [18, 26, 24], continuous and discrete diffusion models [27, 16, 32, 11], VAEs, GANs, and normalizing flows [21, 20, 37], and fragment-based methods that assemble graphs from a learned substructure vocabulary [14, 15, 10]. Although these methods differ in how they parametrize the joint over nodes and edges, they share a similar perspective of constraints: structural and chemical rules are absorbed implicitly into the learned distribution and enforced primarily through reconstruction loss, soft penalties, or downstream filtering. The generated molecules reflect the learned distribution rather than a guarantee that can be verified.

Existing approaches handle user constraints through soft mechanisms, latent biasing toward seed molecules [24], fragment inpainting in a discrete diffusion process [17], and property-aware guidance applied late in a denoising schedule [9], all of which work when the target is a single scaffold or fragment but cannot natively express coupled, non-local logical specifications (conjunctions, exclusions, exclusive disjunctions, conditional implications over substructure presence), and offer no transparency when a specification is unsatisfiable. Generate-and-filter can be costly when satisfying samples have low probability under the learned generator. NSGGM closes both gaps by compiling chemical rules and arbitrary user-specified Boolean predicates into a Satisfiability Modulo Theories (SMT) solver [4] that reasons over predicates on top of neural guidance, so every output is either certified by a SAT model or the proposed blueprint is diagnosed as infeasible, with an UNSAT core identifying conflicting clauses. Appendix H.5 discusses complementary neuro-symbolic frameworks and constraint-aware training.

## 6 Conclusion, Limitations, and Future Work

We introduce NSGGM, which pairs neural motif and interaction proposals with SMT-based assembly, and MOLSAT, a benchmark of compositional logical specifications. The framework combines competitive unconstrained generation with inference-time control, inspectable SAT assignments, and blueprint-level infeasibility diagnoses. Additional constrained-output analyses show high diversity and 98.5–100% observed cumulative success within five attempts on zero-support tasks.

**Scope and limitations.** Guarantees apply to the explicitly encoded structural and chemical rules. Translating medicinal-chemistry knowledge into hard clauses remains a modeling challenge: under-specified requirements may yield brittle constraints. The current user language supports propositional combinations of explicitly grounded binary indicators, but not arbitrary occurrence quantification, unbounded counting, overlapping predicate blocks, or continuous properties. Zero-support generation recombines retained vocabulary primitives; users must supply the predicates and formulas. Exact satisfaction does not establish conditional distribution matching or medicinal utility.

The molecular experiments concern 2D graphs, with validity assessed through RDKit sanitization. Richer electronic or stereochemical models, and 3D extensions with coordinate variables and geometric predicates, require new encodings and compatible proposals; their tractability and fidelity remain untested. Constraint-aware proposal learning and parallel assembly are complementary future directions. Appendix H details expressivity, vocabulary coverage, baseline scope, and deployment limitations.

## Acknowledgments and Disclosures

**Funding.** This work was partially supported by the ProML project, funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the French National Research Agency (ANR), under grant reference ANR-25-CE23-6715.

**Competing Interests.** The authors declare no competing interests.

## References

[1] K. Ahmed, S. Teso, K.-W. Chang, G. Van den Broeck, and A. Vergari. Semantic probabilistic layers for neuro-symbolic learning, 2022. URL https://arxiv.org/abs/2206.00426.

[2] S. Badreddine, A. d'Avila Garcez, L. Serafini, and M. Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022. doi: 10.1016/j.artint.2021.103649.

[3] J. B. Baell and G. A. Holloway. New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *Journal of Medicinal Chemistry*, 53 (7):2719–2740, 2010. doi: 10.1021/jm901137j.

[4] C. Barrett and C. Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pages 305–343. Springer, 2018.

[5] N. Brown, M. Fiscato, M. H. S. Segler, and A. C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019. doi: 10.1021/acs.jcim.8b00839. URL https://pubs.acs.org/doi/10.1021/acs.jcim.8b00839.

[6] N. D. Cao and T. Kipf. Molgan: An implicit generative model for small molecular graphs. *CoRR*, abs/1805.11973, 2018. URL http://arxiv.org/abs/1805.11973.

[7] C. Chang, Y. Shen, S. Fan, J. Li, S. Zhang, N. Cao, Y. Chen, and X. Zhang. Lamagic: Language-model-based topology generation for analog integrated circuits. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=MjGCD8wk1k.

[8] L. M. de Moura and N. S. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008. doi: 10.1007/978-3-540-78800-3_24. URL https://doi.org/10.1007/978-3-540-78800-3_24.

[9] S. Feng, Y. Ni, L. yan, Z.-M. Ma, W.-Y. Ma, and Y. Lan. UniGEM: A unified approach to generation and property prediction for molecules. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Lb91pXwZMR.

[10] Z. Geng, S. Xie, Y. Xia, L. Wu, T. Qin, J. Wang, Y. Zhang, F. Wu, and T.-Y. Liu. De novo molecular generation via connection-aware motif mining. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Q_Jexl8-qDi.

[11] K. K. Haefeli, K. Martinkus, N. Perraudin, and R. Wattenhofer. Diffusion models for graphs benefit from discrete state spaces, 2023. URL https://arxiv.org/abs/2210.01549.

[12] J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, 1987. doi: 10.1137/0216026. URL https://doi.org/10.1137/0216026.

[13] W. Hu*, B. Liu*, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

[14] W. Jin, R. Barzilay, and T. S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2328–2337. PMLR, 2018. URL http://proceedings.mlr.press/v80/jin18a.html.

[15] W. Jin, R. Barzilay, and T. Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. URL https://arxiv.org/abs/2002.03230.

[16] J. Jo, S. Lee, and S. J. Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. 2022. URL https://arxiv.org/abs/2202.02514.

[17] S. Lee, K. Kreis, S. P. Veccham, M. Liu, D. Reidenbach, Y. Peng, S. Paliwal, W. Nie, and A. Vahdat. Genmol: A drug discovery generalist with discrete diffusion. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://openreview.net/pdf?id=KM7pXWG1xj.

[18] R. Liao, Y. Li, Y. Song, S. Wang, C. Nash, W. L. Hamilton, D. Duvenaud, R. Urtasun, and R. S. Zemel. Efficient graph generation with graph recurrent attention networks. 2019. URL https://arxiv.org/abs/1910.00760.

[19] Q. Liu, M. Allamanis, M. Brockschmidt, and A. L. Gaunt. Constrained graph variational autoencoders for molecule design. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7806–7815, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/b8a03c5c15fcfa8dae0b03351eb1742f-Abstract.html.

[20] Y. Luo, K. Yan, and S. Ji. Graphdf: A discrete flow model for molecular graph generation. 2021. URL https://arxiv.org/abs/2102.01189.

[21] K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe. Graphnvp: An invertible flow model for generating molecular graphs. 2019. URL https://arxiv.org/abs/1905.11600.

[22] R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/dc5d637ed5e62c36ecb73b654b05ba2a-Abstract.html.

[23] K. Martinkus, A. Loukas, N. Perraudin, and R. Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15159–15202. PMLR, 2022. URL https://proceedings.mlr.press/v162/martinkus22a.html.

[24] K. Maziarz, H. R. Jackson-Flux, P. Cameron, F. Sirockin, N. Schneider, N. Stiefl, M. Segler, and M. Brockschmidt. Learning to extend molecular scaffolds with structural motifs. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=nS6YyqpTT6y.

[25] N. A. Meanwell. Synopsis of some recent tactical application of bioisosteres in drug design. *Journal of Medicinal Chemistry*, 54(8):2529–2591, 2011. doi: 10.1021/jm1013693. URL https://doi.org/10.1021/jm1013693. PMID: 21413808.

[26] R. Mercado, T. Rastemo, E. Lindelöf, G. Klambauer, O. Engkvist, H. Chen, and E. J. Bjerrum. Graph networks for molecular design. *Mach. Learn. Sci. Technol.*, 2(2):25023, 2021. doi: 10.1088/2632-2153/ABCF91. URL https://doi.org/10.1088/2632-2153/abcf91.

[27] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon. Permutation invariant graph generation via score-based generative modeling. 2020. URL https://arxiv.org/abs/2003.00638.

[28] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, A. Kadurin, S. Johansson, H. Chen, S. Nikolenko, A. Aspuru-Guzik, and A. Zhavoronkov. Molecular sets (moses): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11:565644, 2020. doi: 10.3389/fphar.2020.565644. URL https://www.frontiersin.org/articles/10.3389/fphar.2020.565644.

[29] K. Preuer, P. Renz, T. Unterthiner, S. Hochreiter, and G. Klambauer. Fréchet chemnet distance: A metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 58(9), 2018. doi: 10.1021/acs.jcim.8b00234. URL https://doi.org/10.1021/acs.jcim.8b00234. PMID: 30118593.

[30] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12(null):2539–2561, Nov. 2011. ISSN 1532-4435.

[31] H. Sun, G. Tawa, and A. Wallqvist. Classification of scaffold-hopping approaches. *Drug Discovery Today*, 17(7):310–324, 2012. ISSN 1359-6446. doi: https://doi.org/10.1016/j.drudis.2011.10.024. URL https://www.sciencedirect.com/science/article/pii/S1359644611003813.

[32] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

[33] Z. Wang, J. Shi, N. Heess, A. Gretton, and M. Titsias. Learning-order autoregressive models with application to molecular graph generation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=EY6pXIDi3G.

[34] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. Moleculenet: A benchmark for molecular machine learning, 2018. URL https://arxiv.org/abs/1703.00564.

[35] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5502–5511, 2018. URL https://proceedings.mlr.press/v80/xu18h.html.

[36] Z. Xu, R. Qiu, Y. Chen, H. Chen, X. Fan, M. Pan, Z. Zeng, M. Das, and H. Tong. Discrete-state continuous-time diffusion for graph generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=YkSKZEhIYt.

[37] Y. Zhu, Y. Du, Y. Wang, Y. Xu, J. Zhang, Q. Liu, and S. Wu. A survey on deep graph generation: Methods and applications. 2022. URL https://arxiv.org/abs/2203.06714.

# A Decomposition and Vocabulary Details

This section expands on Section 3.1: the full tile-decomposition procedure, vocabulary thresholds, and the slot-consistency lemma showing that decomposition followed by reassembly recovers the input graph.

## A.1 Full Tile Decomposition

Bridges are defined as $E_{\mathrm{br}} = \{e : G \setminus e \text{ disconnects}\}$. Ring tiles are the cycles of a minimum cycle basis of $(V, E_{\mathrm{ring}})$ [12]. An edge shared by multiple basis cycles has one copy in each corresponding ring tile and is recorded in the shared-edge set of each such tile.

Acyclic tiles are produced on $(V, E_{\mathrm{br}})$ by edge-consumption BFS with depth bound $H = 2$: starting from an arbitrary uncovered vertex, we expand outward up to depth $H$ along bridge edges, marking each consumed edge as covered, and emit the induced subgraph as a tile. Vertices not covered by any ring or BFS tile are emitted as singleton tiles (isolated atoms with empty edge sets).

The tile graph $H(G) = (\mathcal{T}(G), E_H)$ is rooted at a uniformly sampled tile, and BFS traversal from this root yields the linear order $(t_1, \ldots, t_K)$, similar to the scaffold-traversal scheme of Maziarz et al. [24].

## A.2 Vocabulary Construction

We retain all ring motifs observed in the training set and the top-$N_{\mathrm{acyc}}$ most frequent acyclic motifs by occurrence count. For each retained motif $\mu$, we keep the top-$N_{\mathrm{slot}}$ slot variants $\nu$ by frequency, yielding $\Omega(\mu)$. The canonical representative $\tilde{t}_{\mu,\nu}$ is the first slot-annotated tile encountered with key $(\mu, \nu)$ during a fixed-seed traversal of the training set.

## A.3 Slot Consistency Lemma

**Lemma A.1** (Cover)**.** *The tile decomposition $\mathcal{T}(G) = \{t_i\}_{i=1}^{K}$ of Section 3.1 satisfies $\bigcup_i V_{t_i} = V$ and $\bigcup_i E_{t_i} = E$. Moreover, for every edge $e \in E$:*

- *if $e \in E_{\mathrm{br}}$, then $e \in E_{t_i}$ for exactly one $i$;*
- *if $e \in E_{\mathrm{ring}}$ and $e$ is not fused, then $e \in E_{t_i}$ for exactly one $i$;*
- *if $e \in E_{\mathrm{ring}}$ belongs to multiple basis cycles, it appears in every corresponding ring tile and is recorded in the shared-edge set of each such tile.*

*Proof.* For $V$: every vertex $v \in V$ either lies in a ring tile (if $v$ is incident to a ring edge), in a BFS tile (if $v$ is incident to a bridge edge), or is emitted as a singleton tile (otherwise). For $E$: bridges $E_{\mathrm{br}}$ are covered by edge-consumption BFS, which terminates with every bridge consumed; ring edges $E_{\mathrm{ring}}$ are covered by the minimum cycle basis. No edge is shared between a ring tile and a BFS tile, or between two BFS tiles, following from $E_{\mathrm{br}} \cap E_{\mathrm{ring}} = \emptyset$. Thus, multiplicities follow from construction: BFS consumes each bridge once and assigns it to one tile; cycle-basis cycles share an edge precisely when it is fused, and the fused-bond duplication step records this in both $S_{t_i}$ and $S_{t_j}$. □

**Lemma A.2** (Slot Consistency)**.** *Let $G = (V, E, \ell_V, \ell_E)$ be an attributed graph and let $\mathcal{T}(G) = \{t_i\}_{i=1}^{K}$ be its tile decomposition with slot annotations $\{\sigma_v^{t_i}, \sigma_e^{t_i}\}$. Let $G^\star$ be the graph obtained from the disjoint union $\bigsqcup_i t_i$ by (a) identifying, for each $v \in V_{\mathrm{sh}}$ where $V_{\mathrm{sh}}$ is the set of multi-tile vertices, all tile-local copies of $v$, and (b) identifying, for each $e \in E_{\mathrm{sh}}$ where $E_{\mathrm{sh}}$ is the set of all fused edges, the two copies of $e$ in $S_{t_i}$ and $S_{t_j}$. Then:*

1. *$G^\star \cong G$ as attributed graphs (vertex set, edge set, and labels);*
2. *for every $v \in V_{t_i}$ and bond type $\tau \in \mathcal{B}$,*

$$\sigma_v^{t_i}(\tau) \;=\; \big|\{e \in E \setminus E_{t_i} : e \text{ incident to } v,\ \ell_E(e) = \tau\}\big|,$$

*i.e., $\sigma_v^{t_i}(\tau)$ counts exactly the bond-type-$\tau$ edges incident to $v$ in $G$ that are not contained in tile $t_i$.*

*Proof. (1)* The identifications (a) and (b) are well-defined: tile-local copies of $v \in V_{\text{sh}}$ are in bijection with the occurrences of $v$ across $\{V_{t_i}\}$ (and $v \in V \setminus V_{sh}$ are in exactly one tile by definition), and each fused edge has exactly two copies by Lemma A.1. By Lemma A.1, $\bigcup_i V_{t_i} = V$ and $\bigcup_i E_{t_i} = E$. The vertex set of $G^\star$ is the quotient of $\bigsqcup_i V_{t_i}$ under (a), which has one equivalence class per $v \in V$; thus $V(G^\star) = V$. The edge set of $G^\star$ is the quotient of $\bigsqcup_i E_{t_i}$ under (b): each non-fused edge of $G$ contributes exactly one copy (Lemma A.1), and each fused edge contributes two copies that are identified under (b), giving one edge per element of $E$; thus $E(G^\star) = E$. Labels $\ell_V, \ell_E$ descend to the quotient because every tile-local copy of $v$ carries label $\ell_V(v)$ (tiles inherit labels from $G$ by definition), and analogously for edges; the quotient is therefore well-defined on labels. Hence $G^\star \cong G$.

*(2)* By definition (Section 3.1), $\sigma_v^{t_i}(\tau) := b_G(v)(\tau) - b_{t_i}(v)(\tau)$, where $b_G(v)(\tau)$ counts bond-type-$\tau$ edges incident to $v$ in $G$ and $b_{t_i}(v)(\tau)$ counts those in $t_i$. By Lemma A.1, $E_{t_i} \subseteq E$, so every bond-type-$\tau$ edge incident to $v$ in $t_i$ is also incident to $v$ in $G$. Hence

$$b_G(v)(\tau) - b_{t_i}(v)(\tau) \;=\; \big|\{e \in E \setminus E_{t_i} : e \text{ incident to } v,\ \ell_E(e) = \tau\}\big|.$$

By Lemma A.1, these are exactly the edges incident to $v$ that lie in some other tile or are inter-tile attachments which is the exact bonds the SMT assembler must reconstruct via inter-tile merges (Section 3.3). □

# B Neural Component Details

This appendix expands on Section 3.2: the parametrization of the three guidance heads, the full conditional factorization of $p_\theta(y \mid z)$, and the training loss with per-head weighting.

## B.1 Guidance Head Parametrizations

Let $\{\mathbf{h}_i\}_{i=1}^K \subset \mathbb{R}^d$ be the refined hidden states output by the bidirectional Transformer $\phi_{\text{ref}}$. The three heads are:

**Metadata head.**

$$P_\theta(\nu_i \mid \mathcal{M}, z) \;=\; \text{softmax}\big(\mathbf{W}_\nu \mathbf{h}_i + \text{mask}(\mu_i)\big),$$

where $\mathbf{W}_\nu \in \mathbb{R}^{|\mathcal{V}_\nu| \times d}$ projects to the global slot-variant vocabulary and $\text{mask}(\mu_i)$ sets logits outside $\Omega(\mu_i)$ to $-\infty$, restricting the support to slot variants observed for motif $\mu_i$ in training.

**Parent-pointer head.**

$$P_\theta(\pi_i = j \mid \mathcal{M}, z) \;\propto\; \exp\big(d^{-1/2}\,(\mathbf{h}_i \mathbf{W}_Q)(\mathbf{h}_j \mathbf{W}_K)^\top\big), \qquad 1 \le j < i,$$

with $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d}$. Self-parenting and forward references are masked out.

**Merge-type head.**

$$P_\theta(\eta_i \mid \mathcal{M}, z) \;=\; \text{sigmoid}\big(\mathbf{w}_\eta^\top [\mathbf{h}_i; \mathbf{h}_{\pi_i}]\big),$$

with $\mathbf{w}_\eta \in \mathbb{R}^{2d}$, conditioned on the predicted parent index $\pi_i$.

The three heads factorize as conditionally independent given $\{\mathbf{h}_i\}$.

## B.2 Full Conditional Factorization

The conditional likelihood of the supervised targets $y = (\mathcal{M}, \nu_{1:K}, \pi_{2:K}, \eta_{2:K})$ given the latent $z$ factorizes as

$$p_\theta(y \mid z) \;=\; p_\theta(\mathcal{M} \mid z) \prod_{i=1}^{K} p_\theta(\nu_i \mid \mathcal{M}, z) \prod_{i=2}^{K} p_\theta(\pi_i \mid \mathcal{M}, z)\, p_\theta(\eta_i \mid \mathcal{M}, z),$$

where the motif-sequence likelihood is itself causal:

$$P_\theta(\mathcal{M} \mid z) \;=\; \prod_{i=1}^{K+1} P_\theta(\mu_i \mid \mu_{<i}, z), \qquad \mu_{K+1} \equiv \texttt{<eos>}.$$

*Remark* B.1 (No distribution shift on training support). From Theorem C.5, the symbolic assembler depends on the support of the training distribution: every training graph $G$ is recoverable as $G' \cong G$ from some blueprint $S_G$. Thus, distributional fidelity at inference reduces to a neural-component learning objective on the blueprint distribution; the symbolic step includes no further additional distributional shifts for in-support non-hard constraints violating samples.

### B.3 Training Loss

The ELBO of Section 3.2 expands into a weighted reconstruction term plus a KL regularizer:

$$\begin{aligned}\mathcal{L}(\theta,\phi) \;=\; \mathbb{E}_G\,\mathbb{E}_{z\sim q_\phi(z|G)}\Big[&\lambda_{\mathcal{M}}\,\log p_\theta(\mathcal{M}\mid z)+\lambda_\nu\,\log p_\theta(\nu_{1:K}\mid \mathcal{M},z)\\ &+\lambda_\pi\,\log p_\theta(\pi_{2:K}\mid \mathcal{M},z)+\lambda_\eta\,\log p_\theta(\eta_{2:K}\mid \mathcal{M},z)\Big]\\ &-\beta\,\mathrm{KL}\big(q_\phi(z\mid G)\,\|\,p(z)\big).\end{aligned}$$

with prior $p(z) = \mathcal{N}(0, I)$. Loss weights $\lambda_{\mathcal{M}}, \lambda_\nu, \lambda_\pi, \lambda_\eta$ and KL coefficient $\beta$ (with optional warm-up) are tuned on the validation set.

### B.4 Training Hyperparameters

**Architecture.** Embedding, hidden, and latent dimensions are all 512. The encoder is a two-layer GINECONV network with mean-pooling. The decoder is a 4-layer, 4-head Transformer with max length 128, conditioned on $z$ via a linear projection to $L_{\mathrm{mem}} = 8$ cross-attention memory tokens. A 2-layer, 4-head bidirectional Transformer refiner produces the metadata, edge-type, and parent-pointer heads. The tokenizer uses subgraph radius $r = 2$ and metadata top-$K_\nu = 15$.

**Loss weights.** $\lambda_{\mathcal{M}} = \lambda_\nu = \lambda_\pi = 2$, $\lambda_\eta = 1$. $\beta$ is linearly warmed from 0 to $\beta_{\max} = 0.5$ over the first 1,000 steps and held constant after, with prior $p(z) = \mathcal{N}(0, I)$.

**Optimisation.** Adam ($(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, no weight decay), constant learning rate $10^{-4}$, global gradient-norm clipping at 1.0, mini-batch size 1024, up to 1,500 epochs with early stopping on validation loss (patience = 10). Each training sample is presented with a freshly drawn random topological order of its blueprint tree per epoch; validation uses a fixed BFS order. Seed 0, `cudnn.deterministic=True`.

**Hardware.** 4× NVIDIA L40S (48 GB) GPUs (16 CPU cores, 128 GB RAM). Software: Python 3.10.13, CUDA 12.2, PyTorch + PyTorch Geometric, RDKit 2024.03.5.

## C Symbolic Assembly Details

This section provides the full SMT encoding of the hard constraints (C.1), the user-constraint layer (C.2), and the soundness proof for the assembly procedure (C.4). Refer to Algorithm 2 for the full assembly algorithm.

### C.1 Full Hard Constraints

We give the full propositional encoding of the six clauses summarized in Section 3.3. Recall that for each candidate $\{u, v\} \in \mathcal{C}$ we have a Boolean merge variable $m_{u,v}$; that $\mathrm{in_mol}_i$ indicates whether tile $\tilde{t}_i$ participates in $G'$; and that $[u]$ denotes the equivalence class of $u$ under transitive closure of the active merges.

**(i) Element consistency.** $\mathcal{C}$ is constructed to contain only same-element pairs, so the implication $m_{u,v} \Rightarrow \mathrm{elem}(u) = \mathrm{elem}(v)$ holds by construction.

**(ii) Transitivity.** For every triple $u, v, w \in V^{\mathrm{slot}}$ in pairwise distinct tiles,

$$m_{u,v} \wedge m_{v,w} \;\Rightarrow\; m_{u,w}.$$

When $u$ and $w$ lie in the same tile (so the variable $m_{u,w}$ does not exist in $\mathcal{C}$), the conjunction is forbidden outright: $\neg(m_{u,v} \wedge m_{v,w})$. This guarantees the relation $\{(u, v) : m_{u,v} = \top\}$ is transitive on $V^{\mathrm{slot}}$, so equivalence classes $[u]$ are well-defined.

**(iii) Subgraph integrity.** For every tile $\tilde{t}_i$ and every pair of distinct slot atoms $u, v \in V_i^{\text{slot}}$, no merge variable $m_{u,v}$ exists ($\mathcal{C}$ excludes intra-tile pairs). Combined with transitivity, this prevents any chain of merges from collapsing two distinct atoms of the same tile, ensuring vertex injectivity $V_{t_i} \hookrightarrow V'$. Edge injectivity $E_{t_i} \hookrightarrow E'$ then follows: edges of $\tilde{t}_i$ are pairwise distinct in $E_{t_i}$, and clause (v) only identifies shared edges *across* distinct tiles, never within a single tile.

**(iv) Class-level valency cap.** For every atom $u \in V^{\text{slot}}$, let $\text{iv}(u) = \sum_{e \in E_{t_i}, u \in e} \ell_E(e)$ be its within-tile bond-order degree (with $i = \text{sg}(u)$). The total bond order incident to the class $[u]$ in $G'$ is

$$B([u]) \;=\; \Big( \sum_{w \in [u]} \text{iv}(w) \Big) \;-\; \Delta_{\text{share}}([u]),$$

where $\Delta_{\text{share}}([u])$ corrects for shared edges $e \in S_{t_i}$ that are duplicated across merge partners. Concretely, the edge-slot consistency clause (v) below identifies shared-edge copies $e \in S_{t_i}, e' \in S_{t_j}$ (in either endpoint orientation) under bond-label agreement; closing this identification under transitivity partitions the shared-edge copies into *fused-edge classes*, each realizing a single edge in $G'$. Within every fused-edge class we designate one copy as *canonical*—the copy in the lowest-indexed tile, with a fixed tiebreaker on $V^{\text{slot}}$—and call the remaining copies *non-canonical*. We then set

$$\Delta_{\text{share}}([u]) = \sum \{\ell_E(e) : e \text{ is a non-canonical copy with an endpoint in } [u]\}\,.$$

The hard constraint is

$$B([u]) \;\le\; \text{val}(u) \qquad \forall u \in V^{\text{slot}}.$$

By transitivity of merges, $B([u])$ is well-defined regardless of which representative of the class is chosen, and the bound is enforced once per class.

**(v) Edge-slot consistency.** For shared edges $e = (u, v) \in S_{t_i}$ and $e' = (u', v') \in S_{t_j}$ with $i \neq j$:

$$\big(m_{u,u'} \wedge m_{v,v'}\big) \vee \big(m_{u,v'} \wedge m_{v,u'}\big) \implies \sigma_e^{t_i}(e) = \sigma_e^{t_j}(e').$$

When the antecedent holds, $e$ and $e'$ are identified as a single edge in $G'$. The Boolean indicator $\text{fuse}_{i,j} \in \mathbb{B}$ records whether the realized merge between tiles $i$ and $j$ is an edge fusion (used by the soft layer):

$$\text{fuse}_{i,j} \iff \bigvee_{e \in S_{t_i},\, e' \in S_{t_j}} \big[(m_{u,u'} \wedge m_{v,v'}) \vee (m_{u,v'} \wedge m_{v,u'})\big].$$

**(vi) Connectivity.** For each tile $\tilde{t}_i$, $\text{in_mol}_i \in \mathbb{B}$ indicates participation in $G'$. Active merges entail participation:

$$m_{u,v} \;\Rightarrow\; \text{in_mol}_{\text{sg}(u)} \wedge \text{in_mol}_{\text{sg}(v)}.$$

Connectivity is encoded via a propositional depth-bounded BFS from the canonical anchor (the lowest-indexed tile with $\text{in_mol}_i = \top$): a Boolean variable $\text{reach}_i^{(d)}$ records reachability within $d$ hops of the anchor, and we require $\text{in_mol}_i \Rightarrow \text{reach}_i^{(K-1)}$, where $K$ is the number of tiles in the blueprint (every simple path in the tile graph has length at most $K-1$, so this depth bound suffices). The recursive definition is:

$$\text{reach}_i^{(0)} \;\Leftrightarrow\; \mathbb{1}[i = \text{anchor}], \qquad \text{reach}_i^{(d+1)} \;\Leftrightarrow\; \text{reach}_i^{(d)} \;\vee\; \bigvee_{j \neq i} \big(M_{i,j} \wedge \text{reach}_j^{(d)}\big),$$

where $M_{i,j} = \bigvee_{\{u,v\} \in \mathcal{C},\, \text{sg}(u)=i,\, \text{sg}(v)=j} m_{u,v}$ is the tile-level merge indicator.

This reachability requirement is enforced in the full assembler and relaxed in the diagnostic ablation of Table 8.

**Tile participation.** By default we require $\text{in_mol}_i = \top$ for every $i \in \{1, \ldots, K\}$, so every neurally proposed tile must be incorporated into $G'$. This forecloses silent token discard. For tiles in user-substructure blocks, participation is governed jointly by $\psi$ and the atomicity clauses of Appendix C.2, which force the tiles of each $T(u)$ to share a common $\text{in_mol}$ value.

### C.2 Full User Constraints

This appendix expands on the user-constraint encoding of Section 3.3: the structure of each substructure specification, the propositional reduction of the indicator $p_u$, and worked examples of common user formulas $\psi$.

**Substructure specification.** Each $u \in U$ is specified by:

- a *tile block* $T(u) \subseteq \{1, \ldots, K\}$, the indices of tiles that must be present in $G'$ for $u$ to be realized;
- a *fusion set* $F(u)$ of prescribed atom-level merges, each of the form $\{a, b\}$ with $a, b$ ranging over the slot atoms of tiles in $T(u)$.

We require tile-group disjointness: $T(u) \cap T(u') = \emptyset$ for distinct $u, u' \in U$. This ensures the per-substructure clauses (introduced below) compose without imposing contradictory constraints on a shared tile. Single-tile substructures (e.g., a ring motif on its own) have $F(u) = \emptyset$ and reduce to a tile-presence check; multi-tile substructures additionally pin the inter-tile bonds in $F(u)$ that glue the constituent tiles into the intended fragment.

**Block clause.** We take $F(u)$ to be transitively closed within $T(u)$: if $\{a, b\}, \{b, c\} \in F(u)$ with $\mathrm{sg}(a), \mathrm{sg}(b), \mathrm{sg}(c)$ pairwise distinct, then $\{a, c\} \in F(u)$. We add $F(u)$ pairs to $\mathcal{C}$ so they participate in $\phi_{\mathrm{hard}}$; the transitive closure ensures this addition is well-defined. Any cross-tile node-merge inside $T(u)$ that is not prescribed by $F(u)$ is forbidden:

$$\phi_u^{\mathrm{block}} \;:=\; \bigwedge_{\substack{\{x,y\}\in\mathcal{C}\\ \mathrm{sg}(x),\mathrm{sg}(y)\in T(u)\\ \{x,y\}\notin F(u)}} \neg\, m_{x,y}.$$

**Atomicity clause.** The tiles of $T(u)$ are forced to share a common participation value:

$$\phi_u^{\mathrm{atom}} \;:=\; \bigwedge_{a,b\in T(u)} \big(\mathrm{in_mol}_a \leftrightarrow \mathrm{in_mol}_b\big).$$

**Indicator reduction.** The compilation rule

$$p_u \iff \Big(\bigwedge_{i\in T(u)} \mathrm{in_mol}_i\Big) \wedge \Big(\bigwedge_{\{a,b\}\in F(u)} m_{a,b}\Big) \wedge \Big(\bigwedge_{\substack{\{x,y\}\in\mathcal{C}\\ \mathrm{sg}(x),\mathrm{sg}(y)\in T(u)\\ \{x,y\}\notin F(u)}} \neg m_{x,y}\Big)$$

introduced in Section 3.3 is added to the SMT instance for every $u \in U$, pinning each $p_u$ as a derived Boolean of the merge variables and tile-participation indicators.

**Logical formula.** The user formula $\psi$, written over the substructure symbols $u_1, \ldots, u_\ell$, is lifted to the SMT layer by substitution and added as the hard clause $\phi_{\mathrm{user}} := \psi[u \mapsto p_u] \wedge \bigwedge_{u\in U} \phi_u^{\mathrm{atom}}$. Z3 handles arbitrary propositional combinations of the resulting indicators natively.

Here $p_u$ denotes realization of the designated tile block, rather than arbitrary substructure occurrence anywhere in the output. Consequently, $\neg p_u$ does not exclude another occurrence of the same substructure elsewhere, including within a larger motif or across other tiles.

### C.3 Soft Guidance via Max-SMT

This section expands on the soft-clause encoding of Section 3.3: the parent-pointer prediction $\pi_i$ and merge-type prediction $\eta_i$ are lifted into weighted Max-SMT clauses that bias the solver toward the neural prior without overriding $\phi_{\mathrm{hard}} \wedge \phi_{\mathrm{user}}$.

**Parent guidance.** For each tile $i \in \{2, \ldots, K\}$, the parent-pointer head provides the top-1 prediction

$$\pi_i \;:=\; \arg\max_{j\neq i} P(\pi_i = j \mid \mathcal{M}, z).$$

We add the soft clause

$$\phi_i^{\pi} \;:=\; M_{i,\pi_i} \;\equiv \bigvee_{\substack{u \in V_i^{\mathrm{slot}},\, v \in V_{\pi_i}^{\mathrm{slot}} \\ \{u,v\} \in \mathcal{C}}} m_{u,v},$$

weighted by $w_\pi$. This rewards *any* atom-level merge between tile $i$ and its predicted parent; the choice of which atoms to merge is left to $\phi_{\mathrm{hard}}$.

**Merge-type guidance.** For the same parent-child pair $(i, \pi_i)$, the merge-type head provides $\eta_i \in \{0, 1\}$ ($\eta_i = 1$ for an edge merge, $\eta_i = 0$ for a node merge). Using the edge-fusion indicator $\mathrm{fuse}_{i,j}$ from clause (v) of Appendix C.1, we set

$$\phi_i^{\eta} \;:=\; \begin{cases} \mathrm{fuse}_{i,\,\pi_i} & \text{if } \eta_i = 1, \\ \neg\, \mathrm{fuse}_{i,\,\pi_i} & \text{if } \eta_i = 0, \end{cases}$$

weighted by $w_\eta$. The clause is attached only at the predicted parent; the merge-type signal is undefined for non-realized parents.

**Composition and guarantee.** The Max-SMT problem solved by the assembler is

$$\max \sum_{i=2}^{K} \Big( w_\pi \cdot \mathbb{1}[\phi_i^{\pi}] + w_\eta \cdot \mathbb{1}[\phi_i^{\eta}] \Big) \quad \text{subject to} \quad \phi_{\mathrm{hard}} \wedge \phi_{\mathrm{user}}.$$

The soft layer enters only as the objective: SAT/UNSAT outcomes of $\Phi$ are unaffected, unsat cores remain intact, and any model returned by the solver still satisfies every hard and user clause. Empirically we set $w_\pi \gg w_\eta$, representing that mis-attaching to the wrong tile is a more severe error than choosing the wrong merge type at the right tile (although this can be tuned / configured for specific objectives).

**Note for $\nu_i$.** The slot-variant prediction $\nu_i$ is consumed at the hard level: it determines which slot-annotated tile $\tilde{t}_i = \mathcal{V}[\mu_i, \nu_i]$ the solver receives, and so does not appear as a soft clause.

### C.4 Soundness and Procedure

This subsection proves correctness of the assembler under $\phi_{\mathrm{hard}} \wedge \phi_{\mathrm{user}}$. We first create three structural lemmas: class well-definedness (Lemma C.1), tile injection (Lemma C.2), and connectivity (Lemma C.3). The soundness theorem (Theorem C.4) and its converse over training graphs (Theorem C.5) follow.

**Lemma C.1** (Class well-defined)**.** *Let $M = \{m_{u,v}\}$ satisfy $\phi_{\mathrm{hard}}$, and let $[\,\cdot\,]$ denote the equivalence relation on $V^{\mathrm{slot}}$ obtained as the transitive closure of $\{(u, v) : m_{u,v} = \top\}$. For every class $[u]$ we have:*

1. *all atoms $w \in [u]$ share a common element type, so $\mathrm{elem}([u])$ and $\mathrm{val}([u])$ are well-defined;*
2. *the canonical/non-canonical labelling of shared-edge copies introduced in clause (iv) of Appendix C.1 partitions the shared-edge copies into fused-edge classes deterministically;*
3. *the class-level total $B([u])$ defined in clause (iv) is independent of the choice of representative $w \in [u]$.*

*Proof. (1)* Clause (i) gives $m_{u,v} \Rightarrow \mathrm{elem}(u) = \mathrm{elem}(v)$. Clause (ii) closes the merge relation under transitivity, so any two atoms in $[u]$ are connected by a chain of active merges, each step preserving the element type; hence elem is constant on $[u]$, and so is val.

*(2)* Clause (v) implies that every pair of merged endpoints between shared edges $e \in S_{t_i}$ and $e' \in S_{t_j}$ $(i \neq j)$ agrees on bond label. Closing this identification under transitivity (clause (ii)) partitions the union $\bigsqcup_i S_{t_i}$ into *fused-edge classes*, each realizing one edge of $G'$. Within every such class, the canonical copy is selected by the lowest tile index, with a fixed tiebreaker on $V^{\mathrm{slot}}$; both rules depend only on the assignment $M$, so the canonical/non-canonical split is a deterministic function of $M$.

*(3)* The first term $\sum_{w\in[u]} \mathrm{iv}(w)$ ranges over the elements of $[u]$ and is therefore a function of the class. The correction $\Delta_{\mathrm{share}}([u])$ sums $\ell_E(e)$ over non-canonical copies whose endpoint lies in $[u]$; by (2) the canonical/non-canonical labels do not depend on which $w$ one starts from, and by transitivity (clause (ii)) the predicate "endpoint in $[u]$" is a property of the class. Hence

$$B([u]) \;=\; \sum_{w\in[u]} \mathrm{iv}(w) \;-\; \Delta_{\mathrm{share}}([u])$$

depends only on $[u]$. □

**Lemma C.2** (Tile injection)**.** *Let $M$ satisfy $\phi_{\mathrm{hard}}$. For every tile $\tilde{t}_i$ with $\mathrm{in_mol}_i = \top$, the natural maps*

$$\iota_V : V_{t_i} \to V', \quad u \mapsto [u], \qquad\qquad \iota_E : E_{t_i} \to E'$$

*induced by the quotient construction are both injective.*

*Proof. Vertex injectivity.* Take distinct $u, v \in V_{t_i}$. If either of them lies outside $V_i^{\mathrm{slot}}$, that atom has no incident merge variable in $\mathcal{C}$, so its equivalence class is a singleton and cannot coincide with $[u]$ or $[v]$ for any other atom. If both lie in $V_i^{\mathrm{slot}}$, clause (iii) of Appendix C.1 (combined with transitivity, clause (ii)) forbids any equivalence class from containing two distinct slot atoms of the same tile. Hence $[u] \neq [v]$ in every case. *Edge injectivity.* Take distinct $e_1, e_2 \in E_{t_i}$. Edge identification in the quotient comes only from clause (v), which acts on shared-edge pairs $(e, e') \in S_{t_{i'}} \times S_{t_{j'}}$ with $i' \neq j'$; it never identifies two edges within a single tile. Any implicit identification of $e_1$ and $e_2$ via the vertex quotient would require their endpoint pairs to coincide under $\iota_V$, but $\iota_V$ is injective by the previous paragraph, so this is not possible. □

**Lemma C.3** (Connectivity from reach)**.** *Let $M = \{m_{u,v}\}$, $\{\mathrm{in_mol}_i\}$, and $\{\mathrm{reach}_i^{(d)}\}_{0\le d\le K-1}$ satisfy $\phi_{\mathrm{hard}}$, and:*

$$V_M := \{i : \mathrm{in_mol}_i = \top\}, \qquad E_M := \{\{i,j\} : i \neq j,\ M_{i,j} = \top\}$$

*for the active-merge graph $\mathcal{G}_M = (V_M, E_M)$. Assume $V_M \neq \emptyset$ and let $i^* := \min V_M$ represent the canonical anchor (so $\mathrm{anchor} = i^*$ in the recurrence of clause (vi)). Then:*

1. *for every $d \in \{0, \dots, K-1\}$ and $i \in \{1, \dots, K\}$, $\mathrm{reach}_i^{(d)}$ holds iff there is a path in $\mathcal{G}_M$ from $i^*$ to $i$ of length at most $d$;*
2. *$\mathcal{G}_M$ is connected.*

*Proof. (1)* Induction on $d$. Base $d = 0$: $\mathrm{reach}_i^{(0)} \Leftrightarrow \mathbb{1}[i = i^*]$ holds iff $i = i^*$, which is the unique zero-length path. Inductive step: assuming the claim holds at depth $d$, the recurrence

$$\mathrm{reach}_i^{(d+1)} \;\Leftrightarrow\; \mathrm{reach}_i^{(d)} \;\vee\; \bigvee_{j\neq i} \big(M_{i,j} \wedge \mathrm{reach}_j^{(d)}\big)$$

holds iff either $i$ is reachable in $\le d$ hops or some neighbour $j$ of $i$ in $\mathcal{G}_M$ (i.e. $M_{i,j} = \top$) is reachable in $\le d$ hops, in which case extending by the edge $\{i, j\}$ gives a path of length $\le d + 1$ from $i^*$ to $i$. The converse follows analogously.

*(2)* The merge-implies-participation clause $m_{u,v} \Rightarrow \mathrm{in_mol}_{\mathrm{sg}(u)} \wedge \mathrm{in_mol}_{\mathrm{sg}(v)}$ ensures every edge of $\mathcal{G}_M$ has both endpoints in $V_M$. Clause (vi) requires $\mathrm{in_mol}_i \Rightarrow \mathrm{reach}_i^{(K-1)}$, so by (1) every $i \in V_M$ admits a path from $i^*$ in $\mathcal{G}_M$ of length at most $K - 1$. Since $|V_M| \le K$, every simple path has at most $K - 1$ edges, so this depth captures full reachability. Hence every $i \in V_M$ lies in the connected component of $i^*$, and $\mathcal{G}_M$ is connected. □

**Theorem C.4** (Soundness of assembly)**.** *Let $\hat{S} = ((\mu_i, \nu_i))_{i=1}^K$ be a proposed blueprint and let $M$ be a satisfying assignment to $\phi_{\mathrm{hard}} \wedge \phi_{\mathrm{user}}$. Let $G' = (V', E')$ be the quotient graph obtained by identifying each equivalence class $[u]$ to a single vertex of $V'$ and de-duplicating shared edges according to clause (v). Then:*

1. *$G'$ is connected;*

2. *every vertex of $G'$ has well-defined element type and prescribed valency, with* $\mathrm{elem}([u]) = \mathrm{elem}(u)$ *for every $u$ in the class;*

3. $B([u]) \le \mathrm{val}([u])$ *for every vertex* $[u] \in V'$*;*

4. *every tile $\tilde{t}_i$ with* $\mathrm{in_mol}_i = \top$ *injects into $G'$ on both vertices and edges;*

5. *for every $u \in U$ with $p_u = \top$ in $M$, the substructure $u$ is realized in $G'$ exactly as specified: every tile in $T(u)$ is active, every merge in $F(u)$ is realized, and no cross-tile merge in $T(u)$ outside $F(u)$ is active.*

*Proof. (1)* Each tile $\tilde{t}_i$ is connected as a graph: ring tiles are simple cycles; acyclic tiles are connected BFS components; singleton tiles are trivially connected (Appendix A). $G'$ is the disjoint union $\bigsqcup_i \tilde{t}_i$ quotiented under the merge relation, and two tiles share a vertex of $V'$ iff $M_{i,j} = \top$. Hence the connectivity of $G'$ reduces to the connectivity of the active-merge graph $\mathcal{G}_M$, which Lemma C.3 establishes (with $V_M = \{1, \dots, K\}$ by the default tile-participation requirement of Appendix C.1).

*(2)* Follows from Lemma C.1(1). *(3)* Clause (iv) of $\phi_{\mathrm{hard}}$ asserts $B([u]) \le \mathrm{val}(u)$ for every $u \in V^{\mathrm{slot}}$. Lemma C.1 gives $\mathrm{val}([u]) = \mathrm{val}(u)$ and the well-definedness of $B([u])$, so the bound holds class-wise.

*(4)* From Lemma C.2.

*(5)* Fix $u \in U$ with $p_u = \top$. The indicator reduction of Appendix C.2 forces (a) $\mathrm{in_mol}_j = \top$ for every $j \in T(u)$, (b) $m_{a,b} = \top$ for every $\{a, b\} \in F(u)$, and (c) $m_{x,y} = \bot$ for every cross-tile pair $\{x, y\} \in \mathcal{C}$ with $\mathrm{sg}(x), \mathrm{sg}(y) \in T(u)$ and $\{x, y\} \notin F(u)$ (the block clause $\phi_u^{\mathrm{block}}$). The atomicity clause $\phi_u^{\mathrm{atom}}$ enforces (a) consistently across $T(u)$. Together (a)+(b)+(c) realize $u$ in $G'$ with exactly the prescribed tiles active and the prescribed merges, and nothing more. □

**Theorem C.5** (Completeness for retained, representable training graphs)**.** *Let $G = (V, E, \ell_V, \ell_E)$ be any training graph with blueprint $S_G = ((\mu_i, \nu_i))_{i=1}^K$ produced by the decomposition from Appendix A. There exists an assignment to $\{m_{u,v}\}_{\{u,v\} \in \mathcal{C}}$, $\{\mathrm{in_mol}_i\}_{i=1}^K$, and $\{\mathrm{reach}_i^{(d)}\}$ that satisfies $\phi_{\mathrm{hard}}$, with quotient graph $G' \cong G$ as attributed graphs.*

*Proof.* By Lemma A.2, the disjoint-union-and-quotient $G^\star := \bigsqcup_i \tilde{t}_i \,/\, \sim$ is isomorphic to $G$, where $\sim$ identifies tile-local copies of each shared atom and pairs of fused-edge copies. We use $\sim$ to construct the witness:

- $\mathrm{in_mol}_i := \top$ for every $i \in \{1, \dots, K\}$;
- $m_{u,v} := \top$ for $\{u, v\} \in \mathcal{C}$ iff $u \sim v$ (i.e. $u, v$ are tile-local copies of the same global atom of $G$);
- the reach variables $\mathrm{reach}_i^{(d)}$ are determined by the clause-(vi) recurrence once the $m$'s are fixed.

We verify each clause of $\phi_{\mathrm{hard}}$: *(i) Element consistency.* $m_{u,v} = \top$ implies $u \sim v$, hence $u, v$ refer to the same global atom and share the element type. *(ii) Transitivity.* Each tile contains at most one copy of any global atom (tiles are subgraphs of $G$), so $\sim$ is intra-tile-free; closure under transitivity is automatic for an equivalence relation.

*(iii) Subgraph integrity.* Distinct slot atoms of a single tile correspond to distinct global atoms of $G$ (again because tiles are subgraphs), so are never identified by $\sim$.

*(iv) Valency cap.* For every $u \in V^{\mathrm{slot}}$ with $u$ a copy of $v \in V$, the class $[u]$ collects all tile-local copies of $v$. By Lemma A.1 and Lemma A.2, the total bond order incident to $[u]$ in $G^\star$ equals the total bond order incident to $v$ in $G$, which is at most $\mathrm{val}(v) = \mathrm{val}(u)$ since $G$ is itself a valid graph. Hence $B([u]) \le \mathrm{val}(u)$. *(v) Edge-slot consistency.* Two shared-edge copies $e \in S_{t_i}$, $e' \in S_{t_j}$ with $i \ne j$ have endpoints simultaneously merged under $\sim$ iff they are the two copies of the same fused edge of $G$ (Lemma A.1); in that case $\sigma_e^{t_i}(e) = \ell_E(e) = \sigma_{e'}^{t_j}(e')$.

*(vi) Connectivity.* Training graphs are connected, so the active-merge graph $\mathcal{G}_M$ on $\{1, \dots, K\}$ is connected: any two tiles are joined by a path along shared atoms. Lemma C.3 then gives

$\text{in_mol}_i \Rightarrow \text{reach}_i^{(K-1)}$ for every $i$. Therefore, by construction the quotient graph of this assignment is $G^\star \cong G$. □

**Corollary C.6** (Correctness by construction)**.** NSGGM *assembler is sound (Theorem C.4) and complete for retained, representable training graphs (Theorem C.5): every SAT assembly satisfies* $\phi_{\text{hard}} \wedge \phi_{\text{user}}$*, and every retained, representable training graph admits a satisfying assignment whose quotient is isomorphic to itself. In particular, infeasibility detected by the solver (*UNSAT*) is a formal certificate that no graph over the proposed blueprint satisfies the joint specification.*

# D NSGGM Algorithms

**Algorithm 1:** NSGGM inference: end-to-end generation with neural proposals and symbolic assembly.

**Input:** Trained decoder $p_\theta$ (autoregressive head, refinement net $\phi_{\text{ref}}$, metadata/parent/merge-type heads); vocabulary $\mathcal{V}$; optional user specification $(U, \psi)$ with abstract substructure patterns; soft weights $(w_\pi, w_\eta)$; solver timeout $\tau$.

**Output:** Assembled graph $G'$ together with its certificate $\mathcal{A}$ on SAT; unsat core $\mathcal{K} \subseteq \Phi$ on UNSAT; UNKNOWN on timeout/incompleteness.

**Function** NSGGMGENERATE $(p_\theta,\ \mathcal{V},\ (U, \psi),\ (w_\pi, w_\eta),\ \tau)$

```
// 1. Neural proposal (Section 3.2).
Sample z ~ p(z) = N(0, I)
M ← AUTOREGRESSIVEDECODE(p_θ, z)        // motif sequence (μ_1, ..., μ_K, <eos>); K determined by <eos>
{h_i}_{i=1}^K ← φ_ref(M, z)             // refined hidden states, deterministic in (M, z)
for i = 1 to K do
    ν_i ← METADATAHEAD(h_i; μ_i)                    // ν_i ∈ Ω(μ_i) via masked softmax
for i = 2 to K do
    π_i ← PARENTHEAD({h_j}_{j<i}; h_i)
    η_i ← MERGETYPEHEAD(h_i, h_{π_i})
Ŝ ← ((μ_i, ν_i))_{i=1}^K                                  // proposed blueprint
// 2. Vocabulary lookup (Section 3.1).
for i = 1 to K do
    t̃_i ← V[μ_i, ν_i]                   // slot-annotated tile; well-defined since ν_i ∈ Ω(μ_i)
// 3. User-spec compilation: map abstract patterns in U to positional tile
   blocks T(u) and fusion sets F(u) over Ŝ (App. C.2).
if (U, ψ) provided then
    {(T(u), F(u))}_{u∈U} ← COMPILEUSERSPEC(U, Ŝ, V)
else
    U ← ∅;  ψ ← ⊤
// 4. Symbolic assembly (Algorithm 2).
r ← ASSEMBLESMT(Ŝ, (U, ψ, {(T(u), F(u))}), (π, η), (w_π, w_η), τ)
// 5. Outcome handling (Section 3.3).
if r = SAT with assignment A then
    G' ← QUOTIENT({t̃_i}_{i=1}^K, A)          // collapse equivalence classes induced by {m_{u,v} = ⊤}
    return (G', A)                          // A is a verifiable certificate of G' ⊨ Φ
else if r = UNSAT then
    K ← UNSATCORE(Φ)      // infeasible subset; isolates conflicting hard/user clauses
    return (UNSAT, K)
else
    return UNKNOWN    // timeout/incomplete theory; soft failure (Section 3.3)
```

**Algorithm 2:** SMT-based assembly of a blueprint

**Input:** Blueprint $\hat{S} = ((\mu_i, \nu_i))_{i=1}^{K}$; user specification $(U, \psi)$ with tile blocks $\{T(u)\}_{u \in U}$ and fusion sets $\{F(u)\}_{u \in U}$; neural guidance $(\pi_i, \eta_i)_{i=2}^{K}$ with soft weights $(w_\pi, w_\eta)$.

**Output:** Satisfying assignment $\mathcal{A}$ over $\{m_{u,v}\}_{\{u,v\} \in \mathcal{C}} \cup \{\text{in_mol}_i\}_{i=1}^{K} \cup \{p_u\}_{u \in U}$ for $\Phi = \phi_{\text{hard}} \wedge \phi_{\text{user}}$; otherwise UNSAT (with unsat core) or UNKNOWN.

**Function** ASSEMBLESMT $(\hat{S},\ (U, \psi),\ (\pi, \eta),\ (w_\pi, w_\eta))$

$\mathcal{O} \leftarrow$ Z3OPTIMIZE()

`// Retrieve slot-annotated tiles from the vocabulary (Section 3.1).`

**for** $i = 1$ **to** $K$ **do**

  $\tilde{t}_i \leftarrow \mathcal{V}[\mu_i,\ \nu_i]$

`// Decision variables (Section 3.3, App. C.1).`

$V^{\text{slot}} \leftarrow \bigcup_{i=1}^{K} V_i^{\text{slot}}$

$\mathcal{C} \leftarrow \big\{\{u, v\} \subseteq V^{\text{slot}} : \text{sg}(u) \neq \text{sg}(v),\ \text{elem}(u) = \text{elem}(v)\big\}$

**foreach** $\{u, v\} \in \mathcal{C}$ **do**

  Introduce $m_{u,v} \in \mathbb{B}$

**for** $i = 1$ **to** $K$ **do**

  Introduce $\text{in_mol}_i \in \mathbb{B}$

`// Hard validity: clauses (i)-(vi) (App. C.1): element consistency, transitivity, subgraph integrity, valency cap, edge-slot consistency, connectivity from canonical anchor.`

$\mathcal{O}.\text{add}(\phi_{\text{hard}})$

`// User substructure indicators (App. C.2).`

**foreach** $u \in U$ **do**

  Introduce $p_u \in \mathbb{B}$

  $$\mathcal{O}.\text{add}\left(p_u \ \Leftrightarrow\ \Big[\bigwedge_{i \in T(u)} \text{in_mol}_i\Big] \wedge \Big[\bigwedge_{\{a,b\} \in F(u)} m_{a,b}\Big] \wedge \Big[\bigwedge_{\substack{\{x,y\} \in \mathcal{C} \\ \text{sg}(x),\, \text{sg}(y) \in T(u) \\ \{x,y\} \notin F(u)}} \neg\, m_{x,y}\Big]\right)$$

  $\mathcal{O}.\text{add}(\phi_u^{\text{atom}})$

`// User formula` $\psi$ `over` $\{p_u\}_{u \in U}$ `using` $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$.

**if** $\psi$ *provided* **then**

  $\mathcal{O}.\text{add}\big(\psi[u \mapsto p_u]\big)$

`// Soft preferences via Max-SMT (parent-pointer and merge-type).`

**for** $i = 2$ **to** $K$ **do**

  $\mathcal{O}.\text{add_soft}\big(\phi_i^{\pi},\ w_\pi\big)$ `//` $\phi_i^{\pi} := M_{i,\pi_i} = \bigvee_{\substack{\{u,v\} \in \mathcal{C} \\ \text{sg}(u)=i,\, \text{sg}(v)=\pi_i}} m_{u,v}$

  $\mathcal{O}.\text{add_soft}\big(\phi_i^{\eta},\ w_\eta\big)$ `//` $\phi_i^{\eta} := \text{fuse}_{i,\pi_i}$ `if` $\eta_i$`=1, else` $\neg\text{fuse}_{i,\pi_i}$

`// Solve.`

$r \leftarrow \mathcal{O}.\text{check}()$

**if** $r =$ SAT **then**

  **return** $\mathcal{O}.\text{model}()$ `// certificate witnessing` $\Phi$

**else**

  **return** $r$ `// UNSAT (return unsat core) or UNKNOWN`

# E Dataset details

## E.1 Molecular graph datasets

QM9 [34], MOSES and GuacaMol are molecular datasets, where nodes represent atoms and edges correspond to bonds. Planar dataset [23] is a non-molecular graph.

**QM9.** QM9 is a supervised, property-rich benchmark of small organic molecules (up to nine heavy atoms among {C,N,O,F}), each paired with a single low-energy 3D conformation and a suite of quantum-chemical targets (e.g., dipole moment, polarizability, energies). Its compact chemical space and dense labels make it ideal for studying message passing, geometry-aware encoders, and multi-task

regression. QM9's strength is label depth (node/edge/3D information and many targets), not breadth of chemotypes; it is less suitable for evaluating large-scale distribution learning.

**MOSES.** MOSES is a large, cleaned corpus for generative modeling drawn from drug-like ZINC subsets with standardized filters (e.g., MW, logP, allowed atom types). It ships with train/test splits and a unified metric suite (validity, uniqueness, FCD, KL, internal diversity), encouraging apples-to-apples comparison across unconditional generators. Unlike QM9, MOSES prioritizes scale and distributional realism over supervised labels; molecules are provided primarily as SMILES without per-atom/bond targets.

**GuacaMol.** GuacaMol is a broad, medicinal-chemistry benchmark that evaluates both distribution learning and goal-directed generation (e.g., multi-objective property optimization, rediscovery). It emphasizes realistic scaffold diversity and challenge tasks rather than supervised labels, providing standardized splits, metric implementations, and leaderboards. Practically, the dataset includes many chemically complex structures.

# F Experimental Details

## F.1 Unconstrained Generation Details.

**Hardware.** Unconstrained generation experiments were run on CPU-based compute only. Specifically, a *Google Colab CPU runtime* (Google Compute Engine VM) with an AMD EPYC 7B12 CPU (8 vCPUs) and 50 GB RAM, running Linux 6.6.105.

## F.2 Unconstrained Generation Examples.

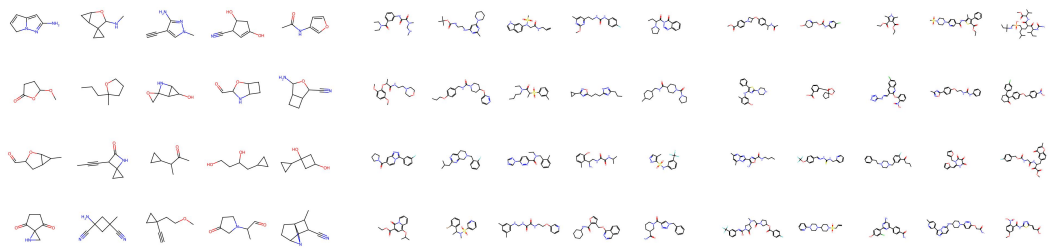


Figure 4: Example outputs from QM9 implicit (left), MOSES (middle), and GuacaMol (right). Samples were randomly chosen.

## F.3 MolSAT Benchmark

Table 9: Natural prevalence of each logical constraint in the GuacaMol training set.

| | I | | | II | | | III | | | | IV | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | H | L | Z | H | L | Z | H | L $\varphi_1$ | L $\varphi_2$ | Z | H | L | Z |
| Train hits | 12,370 | 1,276 | 0 | 7,215 | 2,448 | 0 | 26,704 | 605 | 387 | 0 | 17,275 | 5,784 | 0 |
| Train freq (%) | 0.972 | 0.100 | 0.000 | 0.567 | 0.192 | 0.000 | 2.098 | 0.048 | 0.030 | 0.000 | 1.357 | 0.454 | 0.000 |

**Coverage.** The three tiers measure **training-set prevalence** on GUACAMOL, quantifying difficulty for generate-and-filter baselines. *HIGH* constraints have $\geq 0.5\%$ training support; *LOW* have $< 0.5\%$ but $> 0$; *ZERO* have no satisfying training examples while remaining logically satisfiable. Note that tiers do not directly reflect logical complexity or solver difficulty. We report exact training hits in Table 9.

**Predicate definitions.** We define 8 core scaffolds of Single- and fused-ring heteroaromatic systems, specified in Table 10 and 4 functional motifs, reported in Table 11.

Each predicate $P_i$ corresponds to some Boolean atom $x_i$ interpreted as the output molecule containing substructure $s_i$ as a subgraph match. For a candidate molecule $m$, we define $x_i(m) = 1$ iff RDKit substructure match returns *True* for $s_i$ in $m$. Molecule $m$ satisfies a formula $\varphi \iff \varphi(x_1(m), \ldots, x_n(m))$ evaluates to *True* under standard propositional semantics.

**Logical operators.**
The base operators $\neg$, $\wedge$, $\vee$ are propositional negation, conjunction, and disjunction. Three extended operators are used throughout:

$IMP(a, b) := \neg a \vee b$
$XOR(a, b) := (a \vee b) \wedge \neg(a \wedge b)$
$IFF(a, b) := IMP(a, b) \wedge IMP(b, a)$

Table 10: Core scaffold predicates. Each predicate $P_i$ corresponds to a Boolean atom $x_i$ that is true iff the candidate molecule contains the SMILES substructure as a subgraph match.

| Symbol | Name | SMILES | Description |
|---|---|---|---|
| Pyr | Pyridine | `C1=CC=NC=C1` | Six-membered ring, one N |
| Pym | Pyrimidine | `C1=CN=CN=C1` | Six-membered ring, two N (1,3) |
| Imi | Imidazole | `C1=CN=CN1` | Five-membered ring, two N (1,3) |
| Thz | Thiazole | `C1=CSC=N1` | Five-membered ring, N + S |
| Qui | Quinoline | `C1=CC=C2C(=C1)C=CC=N2` | Fused benzene + pyridine |
| BzI | Benzimidazole | `C1=CC=C2C(=C1)NC=N2` | Fused benzene + imidazole |
| BzT | Benzothiazole | `C1=CC=C2C(=C1)N=CS2` | Fused benzene + thiazole |
| Phz | Phenazine | `C1=CC=C2C(=C1)N=C3C=CC=CC3=N2` | Tricyclic, two ring N |

Table 11: Functional motif predicates. Compact substituent groups used as electron-withdrawing or polar functional handles.

| Symbol | Name | SMILES | Description |
|---|---|---|---|
| Ntr | Nitrile | `C#N` | Cyano group |
| Nto | Nitro | `[N+](=O)[O-]` | Nitro group |
| $TF_3$ | Trifluoromethyl | `C(F)(F)F` | $CF_3$ group |
| Sul | Sulfonamide | `S(=O)(=O)N` | Sulfonamide group |

**Constraint Families.** The four families isolate distinct logical patterns inspired by medicinal-chemistry workflows. We define the explicit formulations for each benchmark below:

**Family I: Filter cascade.** Form: $C \wedge G \wedge \neg B_1 \wedge \neg B_2 \wedge \ldots$ where $C$ is a required scaffold, $G$ a required group, and $\{B_i\}$ forbidden substructures. Every deployed filter set (PAINS, Brenk, NIBR, Lilly MedChem) is structurally a conjunction of negated predicates.

$$\varphi_{\text{I-HIGH}} = \text{Pyr} \wedge \text{Ntr} \wedge \neg\text{Nto} \wedge \neg\text{Sul}$$
$$\varphi_{\text{I-LOW}} = \text{Qui} \wedge \text{TF}_3 \wedge \neg\text{Nto} \wedge \neg\text{Sul}$$
$$\varphi_{\text{I-ZERO}} = \text{BzT} \wedge \text{Phz} \wedge \text{Ntr} \wedge \neg\text{TF}_3$$

**Family II: Disjunctive selection.** Form: $C \wedge (G_1 \vee G_2 \vee \ldots) \wedge \neg B$. Reflects pharmacophore models where a position is satisfied by any of several alternatives.

$$\varphi_{\text{II-HIGH}} = \text{Thz} \wedge (\text{Ntr} \vee \text{Sul} \vee \text{TF}_3) \wedge \neg\text{Nto}$$
$$\varphi_{\text{II-LOW}} = \text{Qui} \wedge (\text{Ntr} \vee \text{TF}_3) \wedge \neg\text{Sul}$$
$$\varphi_{\text{II-ZERO}} = \text{Phz} \wedge (\text{BzI} \vee \text{BzT}) \wedge \neg\text{TF}_3$$

**Family III: Hybrid compound.** Combines XOR selections, IFF couplings, and implications with composite antecedents or consequents.

$$\begin{aligned}\varphi_{\text{III-HIGH}} &= (\text{Pyr} \vee \text{Pym}) \wedge \text{XOR}(\text{Ntr}, \text{Sul}) \\ &\quad \wedge \text{IMP}(\text{Pyr}, \neg\text{Nto}) \wedge \text{IMP}(\text{Pym}, \text{TF}_3)\end{aligned}$$

$$\begin{aligned}\varphi_{\text{III-LOW-}\varphi_1} &= \text{XOR}(\text{Pyr}, \text{Pym}) \wedge \text{XOR}(\text{Ntr}, \text{TF}_3) \wedge \text{XOR}(\text{Nto}, \text{Sul}) \\ &\quad \wedge \text{IMP}(\text{Pyr}, \ \text{Ntr} \vee (\text{TF}_3 \wedge \text{Nto})) \\ &\quad \wedge \text{IMP}(\text{Pym}, \ \text{TF}_3 \vee (\text{Ntr} \wedge \text{Sul})) \\ &\quad \wedge \text{IMP}(\text{Nto}, \neg\text{TF}_3) \wedge \text{IMP}(\text{Sul}, \neg\text{Ntr})\end{aligned}$$

$$\begin{aligned}\varphi_{\text{III-LOW-}\varphi_2} &= \text{XOR}(\text{Imi}, \text{Thz}) \wedge \text{XOR}(\text{Nto}, \text{Sul}) \wedge \text{IFF}(\text{TF}_3, \text{Sul}) \\ &\quad \wedge \text{IMP}(\text{Nto}, \text{Ntr}) \wedge \text{IMP}(\text{Thz}, \text{Nto} \wedge \neg\text{TF}_3) \\ &\quad \wedge \text{IMP}(\text{Imi}, \text{Sul} \vee \text{Ntr})\end{aligned}$$

$$\begin{aligned}\varphi_{\text{III-ZERO}} &= (\text{Pyr} \wedge \text{Pym} \wedge \text{Thz}) \wedge \text{XOR}(\text{Ntr}, \text{TF}_3) \wedge \text{XOR}(\text{Nto}, \text{Sul}) \\ &\quad \wedge \text{IFF}(\text{Nto}, \text{Ntr}) \wedge \text{IFF}(\text{Sul}, \text{TF}_3)\end{aligned}$$

**Family IV: Coupled XOR + IFF.** Form: $\text{XOR}(s_1, s_2) \wedge \text{XOR}(g_1, g_2) \wedge \text{IFF}(s_1, g_1)$. The IFF forces the two XORs to co-vary: choosing $s_1$ forces $g_1$, choosing $s_2$ forces $g_2$.

$$\begin{aligned}\varphi_{\text{IV-HIGH}} &= \text{XOR}(\text{Pyr}, \text{Pym}) \wedge \text{XOR}(\text{Ntr}, \text{Sul}) \wedge \text{IFF}(\text{Pyr}, \text{Ntr}) \\ \varphi_{\text{IV-LOW}} &= \text{XOR}(\text{Imi}, \text{Thz}) \wedge \text{XOR}(\text{Ntr}, \text{TF}_3) \wedge \text{IFF}(\text{Imi}, \text{Ntr}) \\ \varphi_{\text{IV-ZERO}} &= \text{XOR}(\text{Qui}, \text{BzI}) \wedge \text{XOR}(\text{BzT}, \text{Phz}) \wedge \text{IFF}(\text{Qui}, \text{Phz}) \\ &\quad \wedge \text{Ntr} \wedge \neg\text{TF}_3\end{aligned}$$

Table 12: NSGGM SMT solver scaling across logical-constraint benchmarks. *#Scaffolds* = distinct substructure predicates in the formula. *#CNF Clauses* = grounded user formula clauses after CNF conversion. *Merge Vars* = average Boolean merge variables ($z$-vars) per instance. *SAT Rate* = fraction of solver calls returning SAT. These solver statistics were collected in a separate run from the evaluation reported in Table 4

| Benchmark | #Scaffolds | #CNF Clauses | Merge Vars (k) | Solver med (ms) | Solver 95p (ms) | SAT Rate |
|---|---|---|---|---|---|---|
| I-HIGH | 4 | 4 | 164.3 | 692 | 1,172 | 100.0% |
| I-LOW | 4 | 4 | 184.8 | 732 | 1,310 | 99.9% |
| I-ZERO | 4 | 4 | 350.0 | 1,939 | 18,583 | 79.9% |
| II-HIGH | 5 | 3 | 114.6 | 502 | 891 | 100.0% |
| II-LOW | 4 | 3 | 216.1 | 913 | 1,648 | 100.0% |
| II-ZERO | 4 | 3 | 439.7 | 2,626 | 21,384 | 84.2% |
| III-HIGH | 6 | 5 | 245.9 | 2,009 | 21,768 | 97.1% |
| III-LOW-PHI1 | 6 | 12 | 135.8 | 538 | 944 | 99.7% |
| III-LOW-PHI2 | 6 | 10 | 134.7 | 816 | 1,837 | 100.0% |
| III-ZERO | 7 | 11 | 197.2 | 2,174 | 27,041 | 63.9% |
| IV-HIGH | 4 | 6 | 137.7 | 532 | 1,004 | 100.0% |
| IV-LOW | 4 | 6 | 172.9 | 902 | 1,441 | 100.0% |
| IV-ZERO | 6 | 8 | 625.6 | 5,073 | 29,639 | 85.9% |

**Solver scaling.** Table 12 reports per-instance solver statistics across the MOLSAT benchmark. Note that tiers do not reflect the formula complexity or expected solver difficulty; those are determined primarily by the merge-variable count, which scales with the size of the candidate slot space the solver must search over. Empirically, the solver time scales sub-quadratically with number of merge variables. Specifically, $t \approx 0.47 \cdot v^{1.45}$ on SAT instances, closely matching the $\sim v^{1.41}$ scaling in unconstrained generation. We cap individual solver calls at 30s; recorded as `UNKNOWN` and counted as unsuccessful generation attempts and clamped at the cap in the median/95p columns.

Table 13: Timing on MolSAT (seconds per valid molecule, as reported). Dashes denote zero satisfying outputs within the $10^6$-attempt budget; they do not estimate an infinite expected cost. Hardware and batching differ across methods.

| | I | | | II | | | III | | | | IV | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | H | L | Z | H | L | Z | H | L $\varphi_1$ | L $\varphi_2$ | Z | H | L | Z |
| NSGGM (Ours) | 0.769 | 0.803 | 4.605 | 0.536 | 1.006 | 5.468 | 4.898 | 0.585 | 0.969 | 5.907 | 0.666 | 0.951 | 8.933 |
| MoLeR | 0.895 | 1.446 | — | 0.506 | 2.571 | — | 0.890 | 1.167 | 2.259 | — | 0.970 | 0.892 | — |
| GenMol | 0.039 | 0.068 | — | 0.047 | 0.054 | — | 0.039 | 0.078 | 0.090 | — | 0.038 | 0.052 | — |

Table 14: Unconditional generation on SBM and planar graphs. V.U.N.: valid, unique & novel graphs.

| **Model** | **Deg** $\downarrow$ | **Clus** $\downarrow$ | **Orb** $\downarrow$ | **V.U.N.** $\uparrow$ |
|---|---|---|---|---|
| *Stochastic block model* | | | | |
| GraphRNN | 6.9 | 1.7 | 3.1 | 5% |
| GRAN | 14.1 | 1.7 | 2.1 | 25% |
| GG-GAN | 4.4 | 2.1 | 2.3 | 25% |
| SPECTRE | 1.9 | 1.6 | **1.6** | 53% |
| ConGress | 34.1 | 3.1 | 4.5 | 0% |
| DiGress | 1.6 | **1.5** | 1.7 | 74% |
| NSGGM (ours) | **1.3** | **1.5** | 1.9 | **77%** |
| *Planar graphs* | | | | |
| GraphRNN | 24.5 | 9.0 | 2508 | 0% |
| GRAN | 3.5 | 1.4 | 1.8 | 0% |
| SPECTRE | 2.5 | 2.5 | 2.4 | 25% |
| ConGress | 23.8 | 8.8 | 2590 | 0% |
| DiGress | 1.4 | 1.2 | **1.7** | **75%** |
| NSGGM (ours) | **1.3** | **1.1** | 2.0 | 72% |

**Timing comparison.** Table 13 reports the per-valid-molecule wall clock time across all the MOLSAT tasks. GENMOL is the fastest generator on *High*- and *Low*-support, where its forward pass saturates an L40S GPU in parallel. As NSGGM runs sequentially, this is not a hardware-equalized comparison: neural proposals can be batched by GPU, however, the per-molecule cost is dominated by the sequential SMT solver. Parallelizing the solver across cores or solver instances is a natural extension we leave to future work. Even under this bias, NSGGM is the only method with finite cost on every Zero-support tier, where both baselines exhaust their $10^6$-attempt budget without producing a single satisfying molecule. This overhead reflects a tradeoff between constraint-enforcing search and generation speed. A filter can record which predicates hold for a sample; the symbolic layer additionally diagnoses infeasibility over the proposed blueprint. Appendix H.6 discusses parallel throughput and per-instance latency.

## G Non-Molecular Graph Dataset

The non-molecular benchmark of Martinkus et al. [23] consists of two datasets of 200 graphs: (a) stochastic block models (SBM; up to 200 nodes) and (b) planar graphs (64 nodes). Following the protocol, we assess how well generated graphs match degree distributions (Deg), clustering coefficients (Clus), orbit counts (Orb), and the proportion of valid, unique, and novel graphs (V.U.N.).

NSGGM is used in an unconditional setting. The sampler proposes discrete construction sequences, which the symbolic solver assembles under task-specific constraints (e.g., simple connectivity for SBM and planarity for planar graphs). On *SBM*, NSGGM attains the lowest Deg error (1.3) and ties the best Clus (1.5), while remaining competitive on Orb (1.9), yielding the highest V.U.N. (77%). On *Planar*, NSGGM achieves the best Deg (1.3) and Clus (1.1) and competitive Orb (2.0), with V.U.N. 72%, close to DiGress (75%), indicating strong fidelity to planar structure with minimal loss in novelty. These outcomes illustrate that explicit, verifiable constraints during assembly can improve structural metrics without sacrificing diversity.

# H Additional Constrained-Generation Analysis

## H.1 Quality of MolSAT-Satisfying Outputs

We evaluate the diversity and distributional properties of the outputs that satisfy each complete MolSAT formula. Table 15 reports all 13 tasks. The summary in Table 5 takes the minimum and maximum across tasks within each support tier; these ranges measure between-task variation and are not uncertainty estimates. Uniqueness measures distinct generated molecules, while internal and scaffold diversity summarize variation in molecular structure and scaffolds, respectively. These additional analyses are reported as point estimates, separately from the three-run unconstrained-generation results.

Table 15: Diversity of MolSAT-satisfying outputs across all tasks. These measurements provide evidence against severe output collapse; they do not establish biological activity, safety, or synthetic accessibility.

| Task | Uniqueness (%) | Internal diversity | Scaffold diversity |
|---|---|---|---|
| I-High | 100.0 | 0.819 | 0.989 |
| I-Low | 100.0 | 0.802 | 0.986 |
| I-Zero | 100.0 | 0.798 | 0.975 |
| II-High | 99.9 | 0.821 | 0.924 |
| II-Low | 100.0 | 0.758 | 0.975 |
| II-Zero | 100.0 | 0.819 | 0.997 |
| III-High | 100.0 | 0.823 | 0.999 |
| III-Low-$\varphi_1$ | 99.2 | 0.800 | 0.828 |
| III-Low-$\varphi_2$ | 99.7 | 0.784 | 0.958 |
| III-Zero | 100.0 | 0.823 | 1.000 |
| IV-High | 99.9 | 0.861 | 0.966 |
| IV-Low | 100.0 | 0.829 | 0.997 |
| IV-Zero | 100.0 | 0.826 | 0.979 |

Table 16: Distributional analysis of MolSAT-satisfying outputs. SNN uses the full GuacaMol training set as reference. Conditional FCD is the *raw distance* (lower is better), using the satisfying subset of the training set; it differs from the normalized FCD scores in the unconstrained tables. Dashes indicate an empty condition-matched training reference set.

| Task | SNN to full training set | Raw conditional FCD ↓ |
|---|---|---|
| I-High | 0.340 | 16.43 |
| I-Low | 0.367 | 14.37 |
| I-Zero | 0.313 | — |
| II-High | 0.322 | 24.91 |
| II-Low | 0.403 | 23.89 |
| II-Zero | 0.327 | — |
| III-High | 0.307 | 24.39 |
| III-Low-$\varphi_1$ | 0.347 | 28.27 |
| III-Low-$\varphi_2$ | 0.311 | 34.61 |
| III-Zero | 0.290 | — |
| IV-High | 0.352 | 14.41 |
| IV-Low | 0.386 | 20.60 |
| IV-Zero | 0.329 | — |

**Interpretation.** The near-complete uniqueness and substantial internal and scaffold diversity indicate that satisfying outputs are not concentrated on a few repeated structures under these measurements. However, restricting an unconditional proposal distribution at inference does not necessarily recover the data distribution conditioned on a formula. Table 16 makes this distinction explicit. Conditional FCD must also be interpreted with the reference-set size in mind [29]: the Low-support reference sets contain 387–5,784 satisfying training molecules (Table 9). At zero training support, FCD against that empty reference set cannot be computed. SNN remains measurable against the full training set but is not a condition-matched fidelity measure. These statistics do not constitute

a drug-likeness assay; neither medicinal utility nor exact conditional distribution matching follows from logical satisfaction alone.

### H.2 Failure Outcomes and Success under Resampling

**Denominators and outcomes.** A generation attempt fails if the solver returns UNSAT or UNKNOWN. UNSAT means that no assembly of the sampled motif/slot blueprint satisfies the encoded hard and user constraints. It does not establish that the user formula is infeasible for all possible blueprints. UNKNOWN denotes the 30 s timeout in the reported experiments and leaves feasibility unresolved. The failure percentages below use the number of failed single attempts as their denominator:

$$\text{UNSAT share} = \frac{N_{\text{UNSAT}}}{N_{\text{UNSAT}} + N_{\text{UNKNOWN}}}, \qquad \text{UNKNOWN share} = \frac{N_{\text{UNKNOWN}}}{N_{\text{UNSAT}} + N_{\text{UNKNOWN}}}.$$

They should not be read as unconditional failure rates or directly substituted for the single-attempt Hit rates in Table 4.

Table 17: Zero-support failure breakdown, conditional on a failed single attempt. UNKNOWN corresponds to the 30 s timeout.

| Task | UNSAT (%) | UNKNOWN (%) |
|---|---|---|
| I-Zero | 68.62 | 31.38 |
| II-Zero | 14.39 | 85.61 |
| III-Zero | 87.55 | 12.45 |
| IV-Zero | 59.62 | 40.38 |

**Retry protocol.** In an additional experiment, we run 1,000 trials for each zero-support task. A trial starts with a fresh neural blueprint. After either UNSAT or UNKNOWN, we sample a new blueprint and invoke the solver again with the same formula and a 30 s per-call timeout. A trial stops at its first successful assembly or after five total attempts. Hit@$k$ is the fraction of trials with at least one success within those $k$ attempts. Table 18 reports the measured cumulative rates; these are not values extrapolated from an independence assumption or from the original single-attempt evaluation.

Table 18: Observed cumulative retry success over 1,000 trials per zero-support task. Budgets count total attempts, including the first attempt.

| Task | Hit@2 (%) | Hit@3 (%) | Hit@4 (%) | Hit@5 (%) |
|---|---|---|---|---|
| I-Zero | 95.9 | 99.4 | 99.8 | 100.0 |
| II-Zero | 96.8 | 99.4 | 100.0 | 100.0 |
| III-Zero | 85.3 | 93.6 | 96.7 | 98.5 |
| IV-Zero | 96.8 | 99.6 | 99.9 | 100.0 |

**What the failures imply.** The majority of failures on I/III/IV-Zero are certified blueprint infeasibilities. Giving the same infeasible blueprint more time cannot create an assembly; changing the proposal can. II-Zero failures are mostly timeouts, so a longer timeout could help some instances, but no timeout-sweep result is reported. The larger candidate space (439.7 average merge variables) and disjunctive alternatives offer a possible explanation, not an isolated causal measurement. Resampling addresses both failure types by changing the instance. The observed 98.5–100% Hit@5 does not imply guaranteed eventual success, eliminate vocabulary limitations, or relax any hard constraints.

**Worked infeasibility example.** Consider an illustrative specification $\psi = p_A \wedge p_B$ requiring two disjoint tile blocks. Suppose the sampled slot variants admit only one cross-block candidate merge $m$, and both participating carbon copies already have within-tile bond order three, with no shared edge available for de-duplication. Connectivity requires $m$, while merging would give class bond order six, violating the carbon valency cap of four; the hard valency rule therefore entails $\neg m$. Together with the two required block-participation clauses, these requirements are inconsistent. This schematic

example illustrates how required motifs can be present but their sampled interfaces incompatible; it is not a recovered experimental solver trace. A different slot-variant proposal may admit an assembly. Empirically, incompatible motif/slot combinations were the common UNSAT failure mechanism described by inspection of failed proposals.

**Hard proposals versus soft guidance.** The proposed motifs and slot variants define the hard assembly instance. Poor choices can make that instance infeasible. Parent pointers and merge-type predictions instead enter as soft Max-SMT preferences: they cannot change the feasibility of the hard formula, because alternative attachments remain available. Incorrect preferences can nevertheless increase search or optimization time and contribute to UNKNOWN under a fixed timeout. An UNSAT core identifies an inconsistent subset of constraints; extracting that core is distinct from exporting an independently checkable proof object.

### H.3 Unconstrained Attempt-Level SAT Rates

Validity of a returned molecule and success of a raw solver call have different denominators. For the reported checks of 10,000 raw unconstrained solver calls per dataset, QM9 and MOSES had no UNSAT or UNKNOWN outcomes. One GuacaMol run had one UNSAT proposal and no UNKNOWN outcomes, giving 99.99% SAT for that run. This rare failure was a sampled combination of individually valid motif/slot variants that was not jointly assemblable under the hard chemistry constraints. Thus the observed proposal overhead was negligible in these unconstrained runs. These attempt-level checks should not be conflated with the separately collected, rounded scaling statistics in Table 7.

### H.4 Expressivity, Zero Support, and Vocabulary Coverage

**Current user language.** The implemented user layer supports arbitrary propositional combinations of a finite set of explicitly grounded binary substructure indicators, using negation, conjunction, disjunction, implication, equivalence, and XOR. Its current scope does not include quantification over arbitrary motif occurrences, unbounded counting, continuous-valued constraints, or overlapping predicate blocks. Fixed cardinality requirements over a finite list of grounded indicators can be compiled into propositional clauses. For example, "at least two among $p_1, \ldots, p_m$" can be written as $\bigvee_{i<j}(p_i \wedge p_j)$, at the cost of additional formula size. Counting actual motif occurrences throughout a molecule would require grounding the candidate occurrences or extending the encoding with auxiliary integer variables or richer SMT theories. The current guarantees apply to the supplied grounding and encoded predicates.

**Meaning of zero training support.** Zero support means that no molecule in the GuacaMol training set satisfies the complete formula. It does not mean the component motifs or slot variants are unseen: generation composes primitives from the retained vocabulary. Predicates and formulas are provided in advance by the user; NSGGM enforces this expert specification but does not discover the domain knowledge. A formula that requires an unavailable primitive or interface may be infeasible within the retained vocabulary even if a satisfying molecule exists outside it.

**Vocabulary retention.** The acyclic threshold is $N_{\mathrm{acyc}} = 10{,}000$, while GuacaMol contains only 4,706 observed acyclic motifs under the decomposition, so this threshold retains all observed acyclic motifs; all observed ring motifs are also retained. Keeping the top 5, 10, or 15 slot variants per motif retains 94.3%, 98.5%, or 99.5% of slot-variant occurrence mass, respectively. These are vocabulary-coverage statistics, not a full retraining and generation sensitivity study. At the default $N_{\mathrm{slot}} = 15$, all motifs and slot variants required by MolSAT are retained. Consequently, omission of those required variants from the vocabulary does not explain the reported MolSAT failures, although sampling an incompatible combination can still cause UNSAT. More generally, constraint-aware vocabulary auditing can identify required variants before training; retaining them may require a larger threshold.

**Vocabulary size and solver cost.** Vocabulary pruning limits the available primitives and attachment interfaces, but retained components can still be recombined into previously unseen graphs. The solver operates on a sampled blueprint rather than searching the entire vocabulary. Its cost is therefore driven more directly by the number of proposed motifs, candidate merges, and constraints in that instance.

Larger vocabularies can alter the proposal distribution and the resulting instances, so occurrence-mass coverage alone does not establish a speed/quality tradeoff.

**Beyond two-dimensional molecular graphs.** The evaluated molecular setting is 2D and guarantees only explicitly encoded structural and chemical rules. Extending the system to coordinates or continuous properties would require compatible neural proposals, additional decision variables, and appropriate geometric or arithmetic predicates. Although SMT supports richer theories, the tractability and fidelity of such extensions are untested here. Neither 3D conformation quality nor biological activity follows from the present guarantees.

### H.5 Baseline Choice and Constraint-Aware Training

**Evaluation regime.** Our comparison studies inference-time constraints without task-specific retraining. MoLeR and GenMol are selected for their native molecular controllability. For tasks with satisfying training molecules, MoLeR uses a latent bias fitted to those seeds, whereas GenMol uses satisfying molecules as inpainting context. These mechanisms provide structural conditioning beyond unconditional generate-and-filter, but neither natively accepts the complete MolSAT Boolean formula. The comparison does not establish dominance over all possible constrained generators or formula-trained alternatives.

**Why chemical validity differs from MolSAT success.** MoLeR's 100% unconstrained validity concerns chemical validity, while MolSAT requires satisfaction of the complete Boolean specification. Even High-support tasks have only 0.567–2.098% satisfying training molecules, and their formulas involve conjunctions, exclusions, disjunctions, XOR, implications, or equivalence. A chemically valid output may therefore fail MolSAT. GenMol's stronger in-support results also reflect a different conditioning mechanism: inpainting starts from a satisfying molecular context, while MoLeR must recover the required combination from a biased latent sample. The observed gap should not be interpreted as a failure of MoLeR's chemical-validity mechanism.

**Other neuro-symbolic approaches.** DeepProbLog [22] and Logic Tensor Networks [2] are general learning and reasoning frameworks; using them as molecular generation baselines would require specifying and implementing a corresponding generative model. Semantic Loss [35] provides a way to use logical constraints during learning and could complement the proposal model. Semantic Probabilistic Layers [1] also provide hard constraint guarantees for structured outputs, so hard guarantees are not unique to our approach across neuro-symbolic learning. Adapting such layers to variable-length motif blueprints and their assembly space would require a suitable representation and inference procedure; no such adaptation is evaluated here.

**Why the evaluated constraints enter at inference.** Keeping the neural prior independent of the target formula permits new specifications without formula-specific optimization. This separation can sacrifice conditional fidelity or proposal efficiency. An exact semantic-loss objective for this proposal/assembly formulation would need to account for the probability mass of compatible variable-length blueprints and feasible assemblies; a practical approximation is another possibility. Reward-based RL or GFlowNet variants could likewise optimize satisfaction or diversity, but would introduce additional training choices and a different evaluation protocol. We do not claim that these alternatives are impossible or inherently incapable of guarantees. Formula-conditioned training across many tasks, or task-specific optimization when appropriate, are complementary directions not evaluated in this work.

### H.6 Runtime and Deployment Scope

Reported solver experiments use sequential, single-core CPU solving. The cross-model constrained timings are not hardware-equalized: GenMol batches generation on an L40S GPU, whereas NSGGM's cost is dominated by per-molecule SMT solving. The reported zero-support per-valid-molecule times range from 4.605 to 8.933 s. The baseline budget was exhausted without a satisfying output on those tasks, so no empirical per-success runtime can be estimated for those runs. The retry experiment measures cumulative success under an attempt budget; it does not measure an end-to-end latency distribution or speedup.

Independent attempts could be queued after batched neural inference and distributed across CPU cores or machines. This could improve throughput without changing the hard constraints, but would not necessarily reduce the latency of an individual difficult instance. Distributed solving has not been implemented or benchmarked in the reported experiments. Increasing timeout and resampling are also different interventions: the former spends more time on the same instance, while the latter changes the blueprint after failure.

# I Broader Impacts

Explicit constraints and inspectable assembly decisions can make generated molecular graphs easier to audit and can help users diagnose incompatible specifications. The guarantees are limited to the encoding: they do not establish biological safety, therapeutic benefit, synthesizability, or correctness of the user's domain assumptions. Molecular generation also has dual-use potential, including the design of harmful compounds. This study does not evaluate downstream biological use or introduce a deployment safeguard; any practical application would require domain-specific validation and appropriate oversight. The present results concern graph generation and logical control, not approval for clinical or laboratory use.

## NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction summarize the core contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations in the Conclusion section.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: The proofs are provided in Appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [No]

   Justification: The architecture, training, constraint encoding, benchmark formulas, and retry protocol are provided in the paper and appendices. Complete reproduction of the additional quality analysis and zero-support baseline conditioning requires further evaluation details.

   Guidelines:

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

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

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

Answer: [No]

Justification: The paper cites the source datasets and specifies the MolSAT formulas. This version does not provide a public implementation or checkpoint link.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: The Appendix provides all necessary training and test details, including data splits, hyperparameters, etc.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: Unconstrained results include means and standard deviations across three runs. The additional constrained-quality and retry analyses report point estimates; retry success is measured over 1,000 trials per task, without across-run error bars.

Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: Training hardware, unconstrained-generation hardware, and per-instance solver timings are reported. Aggregate compute for all experiments and the retry experiment is not reported.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research presented in this paper complies with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Broader impacts are discussed in Appendix I.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [No]

    Justification: The work does not introduce a deployment safeguard or release a high-risk pretrained model in this manuscript. Encoded chemical validity does not establish biological safety; potential misuse and deployment limitations are discussed in Appendix I.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [No]

    Justification: Existing datasets and methods are cited. This manuscript does not enumerate all asset licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [No]

    Justification: MolSAT is a new benchmark contribution: its predicates, formulas, and training-set prevalence are specified in Appendix F.3. A separately packaged release, license, and data card are not provided in this manuscript.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

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

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

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

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology of this research does not involve the use of LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.