UNSUPERVISED INDUCTION OF NEURAL GRAPH GRAMMARS

Anonymous authors
Paper under double-blind review

ABSTRACT

Probabilistic graph grammar provide a principled framework for modeling graphs through latent compositional processes, yet existing approaches have largely remained confined to statistical models and linguistic applications. We introduce neural graph grammars, establishing a flexible neural parameteriztaion while preserving exact inference over latent derivations. We show that structural encoding improves generalization through parameter sharing across structurally related rules. We further demonstrate the applicability of learned graph grammars beyond language, using them for graph generation and molecular property prediction. Together, these result show neural graph grammars as a compositional model for learning, generating, and analyzing graphs across domains.

1 INTRODUCTION

A fundamental goal in generative modeling of graphs is to learn a probability distribution $P(G)$ over discrete graph structures. This poses a challenging structural learning problem: the size of the space of possible graphs grows exponentially w.r.t. the number of vertices. A central strategy for addressing this combinatorial complexity is to decompose $P(G)$ into a set of tractable probabilistic decisions, such that the probability associated with each individual decision is simple enough to be learnable. Existing models instantiate this strategy through different choices of elementary decisions. You et al. (2018) and Liao et al. (2019) represent graph generation as a sequence of vertex- and edge-level decisions $a_{1:T}$, factorizing their joint probability as $p(a_{a:T}) = \sum_{t=1}^{T} p(a_t|a_{<t})$ and learning these decisions autoregressively. Niu et al. (2020) and Vignac et al. (2022) define a sequence of stochastic transitions that progressively transform a noisy graph into a structured graph sampled from the target distribution. Despite their different probabilistic formulations, these approaches share a common modeling principle: they factorize the generation of a complex graph into a sequence of elementary stochastic decisions. The elementary units may correspond to vertices, edges, or local graph-state transitions, but they are organized through a consective sequence of generative steps.

An alternative perspective to generate graphs, well established in theoretical computer science, is to construct complex structures compositionally, with elementary units organized hierarchically rather than sequential decisions. Context-free graph rewriting formalisms, such as hyperedge replacement grammars (HRGs) and node replacement grammars, provide a principled framework for this view (Engelfriet, 1997; Drewes et al., 1997). Informally, a graph grammar describes a graph using reusable structural pieces and rules for composing them. Starting from a simple structure, the grammar repeatedly expands parts of it into larger graph fragments. The expansions form a tree-structured derivation, which shows how the final graph is assembled from smaller components. Viewed in reverse, the same derivation provides a hierarchical decomposition of an observed graph into its constituent substructures. See Figure 1 for example. The compositional structure captures regularities in both recurring graph fragments and the ways in which they are combined. A probabilistic graph grammar assigns probabilities to rewriting rules, resulting in a proper distribution over derivations (D). The probability of a graph is then characterized over derivations, i.e. $P(G) = \sum_{\text{yield}(D)=G} P(D) = \sum_{\text{yield}(D)=G} \prod_{r \in D} P(r)$. Note that multiple derivations may yield the same graph.

While probabilistic graph grammars provide a structural factorization of $P(G)$, their statistical learning has been studied primarily in NLP, particularly for semantic graph modeling (Koller, 2015; Groschwitz et al., 2021; Gao & Sun, 2025). Traditional approaches typically use a scalar parameteriztaion, assigning an independently estimated probability to each rule. Although this allows

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

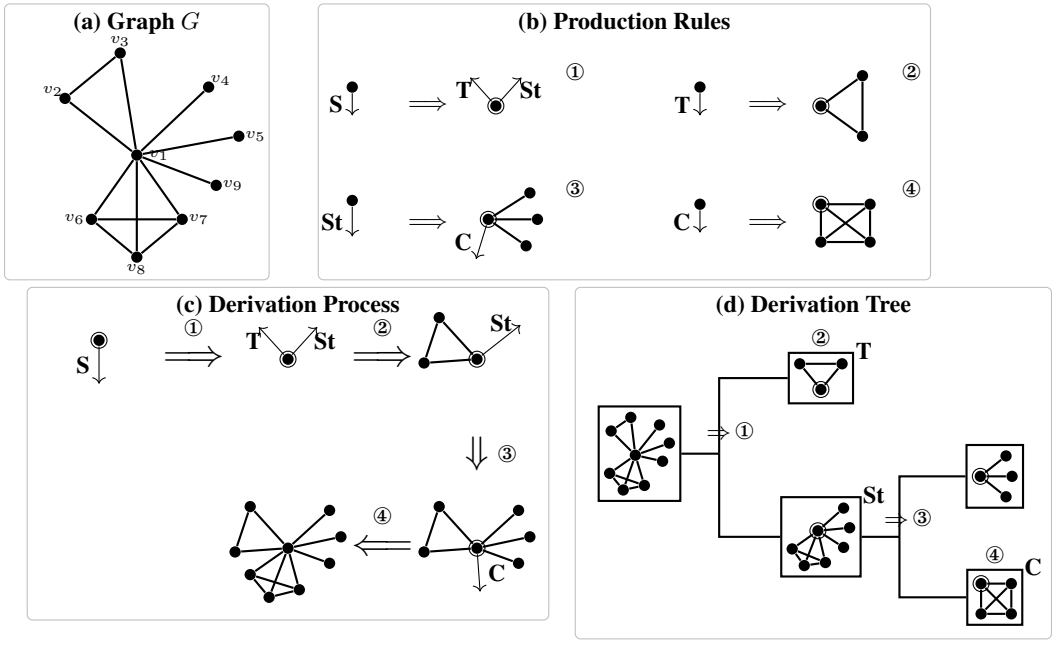

Figure 1: Illustration of graph-grammar derivation. (a) Graph G. (b) Hyperedge-rewriting rules ①–④, each replacing a nonterminal edge with a fragment whose external nodes (haloed) attach to the rewritten edge's endpoints. (c) A derivation of G from an S-labeled edge by applying the rules in turn. (d) The corresponding derivation tree, with each node showing the subgraph from its subderivation.

unsupervised maximum-likelihood estimation by marginalizing over latent derivations, it provides no mechanism for sharing statistical information across structurally related rules. Therefore, estimating probabilities for rare or unseen productions remains challenging. Moreover, the application of learned probabilistic graph grammars beyond linguistic semantic graphs remains largely unknown.

In this paper, we study neural probabilistic graph grammars for learning distributions over graph-structured data. Our contributions are twofold. First, we develop a neural reparameterization of HRGs, establishing an extensible framework for incorporating neural architectures into graph grammars while preserving exact inference over latent derivations. We demonstrate that explicitly encoding production rules' right-hand side graph fragments improves generalization by sharing information across structurally related productions. Second, we extend the study of learned probabilistic graph grammars to general-purpose graphs beyond linguistic semantic graphs. We evaluate their generative capabilities and their ability to infer latent compositional structures across 4 different graph domains, demonstrating their appilcability in both graph generation and downstream prediction.

2 Probabilistic Hyperedge Replacement Grammars

We use HRGs (Drewes et al., 1997) to define a generative process in which graphs are constructed through recursive applications of production rules. HRGs have been used for graph-structured prediction in NLP, including machine translation (Jones et al., 2012), semantic parsing (Chiang et al., 2013; Peng et al., 2015; Chen et al., 2018; Chen & Sun, 2020) and meaning-to-text generation (Ye & Sun, 2020; Gao & Sun, 2025). Importantly, many inference and learning algorithms for HRGs operate over the tree-structured derivations, and can often be adapted to other compositional graph formalisms, such as node replacement grammars and apply-modify (AM) graph algebra (Groschwitz et al., 2017).

2.1 Hyperedge Replacement Grammars

An HRG is a tuple $\mathcal{G} = \langle N, T, P, S \rangle$, with nonterminal labels N, terminal labels T, start symbol $S \in N$, and production rule set P. N, T and P are all finite. Each label $a \in N \cup T$ has fixed

arity $\mathrm{arity}(a) \in \mathbb{N}$, specifying the number of vertices incident to a hyperedge carrying this label. A production is $A \to R$, where $A \in N$, R is a finite hypergraph over $N \cup T$ with an ordered set of distinct external vertices Ex_R, and $|Ex_R| = \mathrm{arity}(A)$. These external vertices define the interface through which R connects to the surrounding graph. Terminal edges remain in the generated graph; nonterminal edges mark expansion sites. Let (G) contain a nonterminal hyperedge e labeled A, with ordered attachment tuple $\mathrm{att}(e) = (u_1, \ldots, u_k)$. Applying $A \to R$ removes e, inserts a fresh copy of R, and identifies Ex_R with the nodes of $\mathrm{att}(e)$. We write the resulting rewrite as $G \Rightarrow G'$.

A rewriting sequence $G_0 \Rightarrow \cdots \Rightarrow G_n$ starts from a single S-labeled nonterminal edge and is complete when G_n contains no nonterminal edges. We represent a complete derivation D by its derivation tree, whose nodes correspond to rule applications and whose subtrees generate subgraphs. Its $\mathrm{yield}(D) = G_n$ is the resulting terminal graph. Rewriting sequences that differ only in the order of independent rule applications correspond to the same derivation tree. Figure 1 illustrates a rewriting sequence and its corresponding derivation tree.

2.2 Probabilistic Grammars

A probabilistic HRG assigns a multinomial distribution $p_\theta(\cdot|A)$ over the rules $\mathcal{R}_A = \{r \in \mathcal{R} : \mathrm{lhs}(r) = A\}$ associated with each nonterminal A. Under the context-free assumption, each production is selected according to its left-hand-side nonterminal, independently of the other rule choices. The probability of a complete derivation D therefore factorizes as

$$p_\theta(D) = \prod_{r \in D} p_\theta(r|\,\mathrm{lhs}(r)), \tag{1}$$

where the product is taken over all rule applications in D. These rule probabilities define a distribution $P_\theta^{\mathrm{deriv}}$ over complete derivations. Since each complete derivation determines a terminal graph, the distribution over derivations induces a distribution $P_\theta^{\mathrm{graph}}$ over graphs. To simplify notation, superscripts are omitted when they can be inferred unambiguously.

2.3 Marginalization and Structural Inference

An observed graph may admit multiple derivations, each corresponding to a different decomposition into graph fragments. Since these derivations are unobserved, the probability of a graph is obtained by summing over all compatible derivations,

$$p_\theta(G) = \sum_{D:\mathrm{yield}(D)=G} p_\theta(D). \tag{2}$$

This marginalization accounts for every possible way the grammar can generate G.

The induced graph distribution also allows us to infer the latent compositional structure of an observed graph. Given G, the posterior distribution $P_\theta(\cdot|G)$ over compatible derivations assigns each derivation D a probability

$$p_\theta(D|G) = \frac{\mathbb{1}[\mathrm{yield}(D) = G] p_\theta(D)}{p_\theta(G)}. \tag{3}$$

This posterior assigns probabilities to alternative structural decompositions of G and can be used to compute quantities such as the expected counts of applications of a rule. Let $c_r(D)$ denote the number of times rule r is applied in derivation D. Its posterior expected count is $\mathbb{E}_{D \sim P_\theta(\cdot|G)}\big[c_r(D)\big] = \sum_{D:\,\mathrm{yield}(D)=G} p_\theta(D|G)\, c_r(D)$. Directly computing the marginal and posterior distributions may require enumerating exponentially many derivations. The context-freeness of HRG allows these computations to be shared across derivations through dynamic programming.

2.4 Exact Inference with Packed Derivation Forests

The sum in Eq. 2 may contain exponentially many derivations, but these derivations often share substantial substructure. We represent this shared computation with a packed derivation forest, following Chiang et al. (2013) and Ye & Sun (2020). A partial derivation is identified by the nonterminal being expanded, the subgraph it generates, and its external node interface. Partial derivations with the same (A, H, Ex) can therefore be represented by a single state, because the

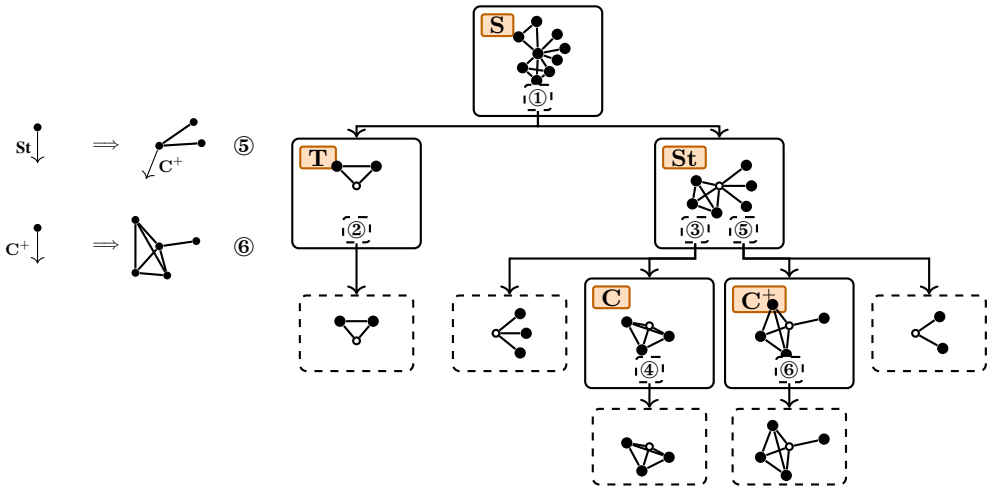

Figure 2: Packed derivation forest for G (Fig. 1) under a grammar extended by productions ⑤ and ⑥ (left). OR-nodes represent subgraphs and dashed AND-nodes their alternative parses; hollow nodes mark external interfaces. The two analyses of **St** share one OR-node, as do all common subgraphs.

surrounding derivation depends only on the generated subgraph and its boundary, not how that subgraph was derived internally.

The resulting forest is an acyclic AND/OR graph, illustrated in Figure 2. An OR-node represents a subgraph that may admit several derivations, while each AND-child represents one production application together with the subderivations required by its nonterminal edges. A complete derivation is recovered by selecting one AND-child at each visited OR-node This representation allows shared subproblems to be computed once.

Let $\alpha(v)$ denote the inside probability of an OR-node v, i.e., the total probability of all derivations represented below v. For a node with outgoing AND-edges e, $\alpha(v) = \sum_{e \in \mathrm{out}(v)} p_\theta(r(e)|A_v) \prod_{u \in \mathrm{ch}(e)} \alpha(u)$, where $r(e)$ is the rule application associated with e and $\mathrm{ch}(e)$ denotes its child OR-nodes. The sum accounts for alternative derivations of the same subgraph, while the product follows from the factorization of rule choices in Eq. 1. At the root of the forest, $p_\theta(G) = \alpha(v_S)$.

Although $p_\theta(G)$ is defined as a sum over exponentially many derivations, it can be computed exactly by dynamic programming over the packed forest. Let V_F and E_F denote the OR-nodes and AND-edges of the forest, repectively, and let $\mathrm{ch}(e)$ be the child occurrences of AND-edge e. Given precomputed rule probabilities, both inside and inside-outside computation require $\mathcal{O}\big(|V_F| + |E_F| + \sum_{e \in E_F} |\,\mathrm{ch}(e)|\big)$ operations. Detailed forest construction, outside recursions, and complexity analysis are described in Appendix B.

3 Neural Parameterization and Learning

The probabilistic formalization in Section 2.1 does not prescribe how rule probabilities are parameterized. We introduce a neural parameterization of HRGs that replaces independently estimated rule probabilities with a shared scoring function, following neural PCFGs (Kim et al., 2019a). We consider both learned rule embeddings and structural encodings of graph fragments, the latter allowing information to be shared explicitly across structurally related productions. We further include a compound extension with per-graph latent variables that condition rule probabilities, allowing dependencies between rule choices within a graph. These parameterizations retain the context-free factorization required for exact inference over derivations. Figure 3 summarizes our design.

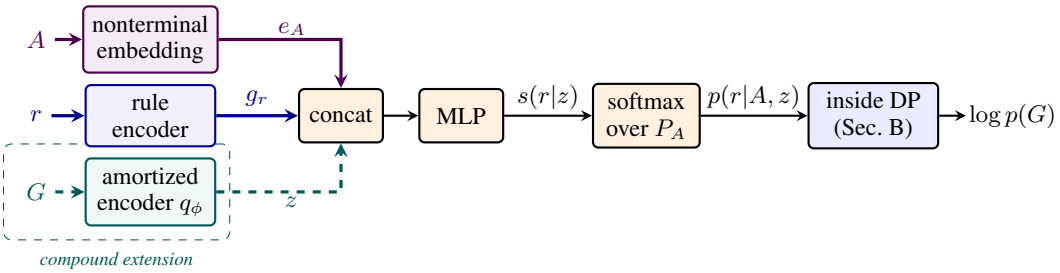

Figure 3: Neural parameterization architecture. We embed the left-hand-side symbol A and the right-hand side graph fragment of rule r, concatenate, and pass through an MLP and softmax over rules with shared left-hand side to score $p(r|A)$; a traversal over the packed forest gives $\log p(G)$. The right-hand-side encoder swaps between a lookup by rule identity and a structural encoder over the hypergraph. The dashed teal panel adds the compound extension: an amortized encoder produces a per-graph latent z that joins the concatenation, conditioning rule probabilities on the input graph.

3.1 Neural Rule Distributions

In a traditional probabilistic HRG, each rule is assigned an independently estimated probability, subject to normalization over rules with the same left-hand-side nonterminal. Following neuralized HMMs and PCFGs (Chiu & Rush, 2020; Kim et al., 2019a), we introduce a neural scoring function that shares parameters across rules.

For each nonterminal A, let $e_A \in \mathbb{R}^d$ be a learned embedding, and let $g_r \in \mathbb{R}^d$ represent production r. We compute a production score using a shared neural network,

$$s_\theta(r, A) = \mathrm{MLP}_\theta([e_A; g_r]), \tag{4}$$

where $[;]$ denotes concatenation. The scores are normalized over rules with the same left-hand-side nonterminal,

$$p_\theta(r|A) = \frac{\exp s_\theta(r, A)}{\sum_{\mathrm{lhs}(r')=A} \exp s_\theta(r', A)}. \tag{5}$$

This parameterization replaces independently learned probabilities with a shared scoring function while preserving the factorization of derivation probabilities. The exact inside-outside inference procedure remains applicable.

The representation g_r may be a learned embedding associated with an individual production or an encoding of its graph structure. The former allows parameter sharing through the scoring function but still treats each rule as an independent object. The latter additionally exploits structural similarities between rules, allowing information to be shared explicitly across related graph fragments.

3.2 Structural Encoding

Although neural scoring allows parameters to be shared across productions, learned rule embeddings still treat each production as an independent object. To exploit similarities between productions, we instead construct their representations from the graph fragments they generate. For a production $r : A \to R$, we define

$$g_r = \mathrm{Enc}_\theta(R), \tag{6}$$

where Enc_θ is a shared structural encoder. The encoder represents R as a labeled hypergraph, applying message passing over its vertices and hyperedges to capture their structural relationships. The resulting representations are pooled to obtain a representation of the fragment's internal structure and combined with a separate encoding of its ordered external interface. This allows the rule representation to capture both the structure of the generated fragment and how it connects to the surrounding graph.

3.3 Unsupervised Learning from Observed Graphs

Given a dataset $\mathcal{D} = \{G_1, \ldots, G_n\}$ of observed graphs, we estimate the model parameters by maximizing their marginal log-likelihood. The training objective marginalizes over all compatible

derivations, $\mathcal{L}(\theta) = \sum_{i=1}^{n} \log p_\theta(G_i) = \sum_{i=1}^{n} \log \sum_{D:\text{yield}(D)=G_i} p_\theta(D)$. For each graph, the inside algorithm computes the marginal likelihood exactly under the context-free factorization. Because the inside recursion consists of differentiable operations on production probabilities, we can optimize the neural parameters directly by backpropagating through dynamic programming.

The gradient of the marginal log-likelihood can also be expressed as an expectation under the posterior distribution over derivations,

$$\nabla_\theta \log p_\theta(G) = \mathbb{E}_{D \sim p_\theta(D|G)} \left[\sum_{r \in D} \nabla_\theta \log p_\theta(r \mid \text{lhs}(r)) \right]. \tag{7}$$

This applies to both independently parameterized and neural rule distributions. For the former, the same marginal-likelihood objective admits an expectation-maximization (EM) algorithm with closed-form updates (Gao & Sun, 2025). For neural parameterizations, we optimize by gradient descent. The derivation of Eq. 7, EM updates, and implementation details are provided in Appendix C.

4 Learning Rule Distributions

4.1 Recovering Latent Rule Probabilities

We first study under what conditions maximum-likelihood estimation can recover rule probabilities when the underlying derivations are unobserved. To evaluate this directly, we construct synthetic grammars with known probabilistic distributions. We construct a complete grammar within a bounded rule space by enumerating every graph topology with fewer than $n_{\max}$ vertices and all possible assignments of terminal and nonterminal labels. We then sample the ground-truth rule probabilities from symmetric Dirichlet distributions and generate graphs through recursive rule applications. The generating derivations are discarded and each observed graph is reparsed with the complete grammar, allowing multiple compatible derivations for each graph. Details of grammar construction are provided in Appendix D.1. We evaluate parameter recovery by comparing the estimated rule probabilities with their ground-truth values, using Kullback-Leibler (KL; Kullback & Leibler, 1951) divergence and Pearson correlation (Pearson, 1895).

Experiment	Condition	KL ↓	Pearson ρ ↑
Training-set size	$n = 100$	0.081 ± 0.042	0.648 ± 0.138
	$n = 500$	0.037 ± 0.018	0.868 ± 0.076
	$n = 2000$	0.024 ± 0.015	0.909 ± 0.075
	$n = 5000$	0.022 ± 0.016	0.911 ± 0.078
Derivational ambiguity	Low (1.0)	0.0013 ± 0.0002	0.999 ± 0.0010
	Medium (2.2)	0.0011 ± 0.0002	0.999 ± 0.0004
	High (4.3)	0.0013 ± 0.0009	0.998 ± 0.0010

Table 1: Results across training-set size, derivational ambiguity, and Dirichlet concentration.

We study three factors, training set size, derivational ambiguity, and the concentration of the ground-truth distribution. As shown in Table 1, recovery improves with increasing training data size, with diminishing improvements beyond $2,000$ graphs. To study recovery under different levels of derivational ambiguity, we construct three grammars with different average number of compatible derivations per graph. Recovery remains nearly unchanged across the ambiguity levels, indicating that multiple compatible derivations does not necessarily impair estimation in this setting. We additionally vary the Dirichlet concentration to examine recovery under different ground-truth rule distributions. As the distributions become more uniform, KL divergence generally decreases while correlation tends to decline, reflecting the reduced variation among ground-truth rule probabilities (results see Appendix D). Together these results show that accurate recovery is achievable across the tested conditions.

4.2 Generalization through Structural Parameter Sharing

We next test how different paramterization affects performance on natural data. Hauser et al. (2002) hypothesized that recursion is the defining, uniquely human component of the faculty of language

in the narrow sense. Under this hypothesis, linguistic expressions are constructed by recursively combining smaller units into larger structures, making language inherently compositional. This makes language a natural setting for evaluating graph grammars.

We use the LinGO Redwoods Treebank (Oepen et al., 2002), a widely used benchmark resource for semantic parsing in NLP. Representations derived from Redwoods and the English Resource Grammar have been used in several major shared tasks, including SemEval 2014 and 2015 on broad-coverage semantic dependency parsing (Oepen et al., 2014; 2015), as well as the CoNLL 2019 and 2020 shared tasks on Cross-Framework Meaning Representation Parsing (Oepen et al., 2019; 2020). We extract the semantic representations as hypergraphs, whose vertices represent semantic entities and hyperedges encode predicate-argument relationships. Unlike the synthetic setting, these graphs exhibit naturally occurring compositional structures and variation in linguistic complexity.

Our task is to learn a probability distribution over these semantic graphs without observing their derivations. We compare four parameterizations of the same grammar: A uniform HRG (U-HRG), a multinomial HRG (EM-HRG) estimated using EM, Lookup neural HRG (L-NHRG), and a structural neural HRG (S-NHRG). We evaluate using graph log-likelihood, generation Bilingual Evaluation Understudy (BLEU; Papineni et al., 2002), and posterior derivation entropy. LL measures the probability assigned to graphs, BLEU compares sentences regenerated from semantic graphs with their reference sentences; and entropy characterizes uncertainty over alternative derivations. Complete details of model configuration and training are provided in Appendix E.

Method	#Param.	LL ↑	Entropy	BLEU ↑
U-HRG	NA	-74.96	4.696	0.4107
EM-HRG	14,319	-95.21	1.694	0.6195
L-NHRG	955,585	-92.80	1.609	0.6155
S-NHRG	390,192	$\mathbf{-60.58}$	1.639	$\mathbf{0.6543}$

Table 2: Results of HRG models on Redwoods.

Unseen rules	L-NHRG	S-NHRG
0	0.5522	0.5616
1–5	0.6059	0.6483
6–20	0.5947	0.7202

Table 3: BLEU by # of unseen rules.

Table 2 shows that structural encoding improves both LL and generation BLEU over the alternative parameterizations. Notably, introducing neural scoring with rule-specific embeddings alone provides only limited improvement over EM, whereas including graph structure produces a substantial improvement. This suggests that the benefit comes not simply from replacing parameters with a neural network, but from incorporating structural information to the representation of grammar rules.

To investigate generalization to unseen rules specifically, we stratify held-out graphs by the number of rules in their derivation trees that were unseen during training. As shown in Table 3, when all rules have been observed during training, the two models achieve comparable performance. S-NHRG performs better when graphs contain unseen rules, with the difference increasing as the number of unseen rules increases. The rule level analysis shows that, for productions absent from training but required by held-out data, S-NHRG obtains a LL of -11.88 compared to -17.84 of L-NHRG. S-NHRG also achieves higher scores across every observed rule-frequency group. These results are consistent with structural representations enabling statistical information to be shared across rules, including those without direct training signal.

5 Evaluating Graph Generation and Graph parsing

Once the rule probabilities have been learned, the same probabilistic HRG supports two operations in the opposite directions:

Graph Generation: Sample a complete derivation from the learned distribution and construct the graph it generates,

$$D \sim P_\theta^{\mathrm{deriv}}, G = \mathrm{yield}(D).$$

Graph Parsing: Given an observed graph G, infer the posterior distribution over its compatible derivations,

$$p_\theta(D|G) = \frac{\mathbb{1}[\mathrm{yield}(D) = G] p_\theta(D)}{p_\theta(G)}.$$

We evaluate both directions on naturalist data from different domains, demonstrating the applicability of learned graph grammars to both graph generation and structural analysis across domains.

5.1 Graph Generation from Learned Grammars

We first study whether the learned rule distribution can generate graphs that reproduce the structural properties of observed data. We sample complete derivations from our S-NHRG) and compare the resulting graphs with those generated by two established models, GraphRNN (You et al., 2018) and DiGress (Vignac et al., 2022). We assess the generated graphs against held-out data using divergence in degree distributions, clustering coefficients, graph spectra, and node-label distributions.

Model	Degree ↓	Clustering ↓	Spectrum ↓	Node-label TV ↓	Connected
Held-out (floor)	1.48×10^{-4}	4.09×10^{-4}	3.37×10^{-4}	0.0095	0.980
HRG	7.40×10^{-4}	$\mathbf{5.49\times10^{-4}}$	9.91×10^{-4}	**0.0078**	1.000
GraphRNN + labels	0.0015	0.0011	0.0023	0.0170	0.999
DiGress	$\mathbf{3.69\times10^{-4}}$	7.67×10^{-4}	$\mathbf{3.91\times10^{-4}}$	0.0129	0.936

Table 4: Structural fidelity of generated graphs compared with GraphRNN and DiGress.

As shown in Table 4, HRG achieves the lowest clustering and node-label discrepancies among the generative models, with clustering close to the held-out reference. DiGress achieves better performance on degree distributions and graph spectra. All graphs generated by our grammar are connected. These results show that a grammar can reproduce important properties of observed graphs through compositional generation. Full details are provided in the appendix F.

5.2 Graph Parsing for Molecular Prediction

Given an observed graph G, a learned grammar allows us to infer its latent compositional structure through the posterior distribution $P_\theta(\cdot|G)$. We use this posterior to construct graph representations that account for multiple possible structural decompositions. We evaluate whether these representations capture information useful for downstream prediction, using molecular graphs as a test case.

Representation	ESOL ↓	FreeSolv ↓	BACE ↑	BBBP ↑
Atom counts	1.1199	3.3850	0.7663	0.8882
Fragments	1.2900	2.8734	**0.8496**	0.8952
ECFP4 (Rogers & Hahn, 2010)	1.5540	5.2359	0.8481	0.8830
MACCS (Durant et al., 2002)	1.4346	**2.4188**	0.8200	0.9044
RDKit descriptors (Landrum, 2026)	**0.9651**	2.6517	0.8018	0.9178
ChemBERTa (Chithrananda et al., 2020)	1.5433	3.6194	0.8057	**0.9540**

Table 5: Molecular property prediction. All representations are evaluated on the same five scaffold splits and on molecules admitted by the grammar. Values are means across the five splits.

We train the model on $20,000$ unlabeled molecules from the ZINC dataset (Irwin & Shoichet, 2005). Given an observed graph G, we extract the structural features from its derivation forest, including posterior expected rule counts, $\phi_r(G) = \mathbb{E}_{D\sim P_\theta(\cdot|G)}\left[c_r(D)\right]$. and posterior-weighted fragments. These representations summarize the compositional structures of a graph while accounting for uncertainty over its possible derivations.

We evaluate the representations on four MoleculeNet (Wu et al., 2018) datasets: ESOL (Delaney, 2004) and FreeSolv (Mobley & Guthrie, 2014) for regression, and BACE (Subramanian et al., 2016) and BBBP (Martins et al., 2012) for classification. we compare against conventional molecular fingerprints, chemical descriptors, and pretrained ChemBERTa representations, using the same random-forect predictor for all methods. We use five 80/10/10 Bemmis-Murcko scaffold splits (Bemis & Murcko, 1996) to partition molecules according to their core molecular frameworks, and evaluate using root mean squared error (RMSE) for regression and area under the ROC curve (AUC) for classification. Full details are provided in Appendix G.

As shown in Table 5, representations derived from graph decompositions achieve competitive predictive performance against established molecular fingerprints, chemical descriptors, and pretrained neural representations across the four tasks. The posterior fragment achieves the highest AUC on BACE among the evaluated methods. Although its performance on BBBP is below several alternatives, the results demonstrate that grammar-derived representations can capture molecular information relevant to distinct prediction tasks. Notably, these representations are obtained without molecular property labels, and the same posterior-derived features can be used across different prediction tasks. These results demonstrate the potential of probabilistic graph parsing to provide reusable structural representations beyond graph generation and density estimation.

6 Related Work

Graph Generative Models Graph generative modeling has a long history in probabilistic network analysis. Classical approaches define graph distributions through assumptions about edge formation, including independent edge probabilities in Erdős–Rényi models (Erdős et al., 1960), community structure (Holland et al., 1983; Peixoto, 2017), small-world connectivity (Watts & Strogatz, 1998), preferential attachment (Barabási & Albert, 1999), and expected degree sequences (Chung & Lu, 2002). Neural approaches instead learn graph distributions from data, adopting different representations and factorizations of the generative process. Autoregressive models construct graphs through sequences of conditional node and edge decisions (You et al., 2018; Li et al., 2018), with subsequent approaches improving efficiency through blockwise generation (Liao et al., 2019), or hierarchical representations of sparse adjacency matrices (Dai et al., 2018). Latent-variable models learn graph distributions through continuous representations, ranging from direct graph decoding (Simonovsky & Komodakis, 2018) to hierarchical generation from molecular substructures (Jin et al., 2018; 2020).

Other approaches define graph distributions through adversarial learning (Bojchevski et al., 2018), invertible transformations (Zang & Wang, 2020; Shi et al., 2020), or global spectral conditioning (Martinkus et al., 2022). Score-based and diffusion models learn stochastic generative processes that transform noise into graph structure, progressing from permutation-equivariant score estimation (Niu et al., 2020) to continuous-state graph diffusion (Jo et al., 2022) and discrete diffusion over node and edge attributes (Vignac et al., 2022). Recent work has also adapted discrete diffusion to efficient generation of large, sparse graphs (Chen et al., 2023).

Our approach instead places a distribution over recursive graph derivations, with the probability of an observed graph obtained by marginalizing over all its compatible derivations. This makes the compositional structure explicit and supports both graph generation and inference within a single framework. Its closest methodological connections are therefore to probabilistic graph grammars.

Probabilistic and Neural grammars Probabilistic grammars provide a well-established framework for defining distributions over structured objects. On strings, neural PCFGs replace the per-rule probability table with a learned scorer (Yang et al., 2021; Collins & Cohen, 2012), and compound PCFGs draw a per-sentence latent that modulates rule scores via amortized variational inference, leaving the inside algorithm exact (Kim et al., 2019b). We carry both moves to hyperedge replacement grammars, where derivations build a graph rather than a string. HRGs and related context-free graph rewriting have a long parsing-theoretic literature (Drewes et al., 1997) and underpin grammar-based work in semantic parsing and translation (Jones et al., 2012; Chiang et al., 2013; Peng et al., 2015; Chen et al., 2018). Two recent contributions are direct antecedents: Ye & Sun (2020) make exact graph parsing efficient and Gao & Sun (2025) estimate rule probabilities by EM on semantic graphs.

7 Limitations

Our framework has several limitations. First, it learns rule probabilities over a fixed rule set rather than inducing the symbolic components of a grammar; full grammar identification remains open, though accurate rule parameterization is a necessary component of that problem. Second, neural HRGs incur overhead from explicit derivation forests, dynamic programming, neural rule scoring, and variational inference, so scaling to larger graphs and grammars will require more efficient pruning, batching, and amortized scoring. Third, our experiments cover linguistic semantic graphs and molecular graphs, but broader generalization remains open.

AI STATEMENT

Generative AI was used to edit the manuscript and structure the presentation of the paper. They were also used in adapting and updating existing baseline implementations for graph generation and molecular graph experiments, where the original implementations used outdated software. It was also used to develop experiment drivers for hyperparameter search and debug code. The authors fully designed the methodology, experimental design, hyperparameter space, evaluation, and conclusions. The authors take full responsibility for the final manuscript and reported results.

REFERENCES

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286 (5439):509–512, 1999.

Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *International conference on machine learning*, pp. 610–619. PMLR, 2018.

Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111*, 2023.

Yufei Chen and Weiwei Sun. Parsing into variable-in-situ logico-semantic graphs. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6772–6782, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.605. URL https://aclanthology.org/2020.acl-main.605/.

Yufei Chen, Weiwei Sun, and Xiaojun Wan. Accurate SHRG-based semantic parsing. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 408–418, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1038. URL https://aclanthology.org/P18-1038/.

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

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

Justin Chiu and Alexander Rush. Scaling hidden Markov language models. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1341–1349, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.103. URL https://aclanthology.org/2020.emnlp-main.103/.

Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

Michael Collins and Shay Cohen. Tensor decomposition for fast parsing with latent-variable pcfgs. *Advances in Neural Information Processing Systems*, 25, 2012.

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

John S Delaney. Esol: estimating aqueous solubility directly from molecular structure. *Journal of chemical information and computer sciences*, 44(3):1000–1005, 2004.

F. Drewes, H.-J. Kreowski, and A. Habel. *Hyperedge Replacement Graph Grammars*, pp. 95–162. WORLD SCIENTIFIC, February 1997. ISBN 978-981-02-2884-2 978-981-238-472-0. doi: 10.1142/9789812384720_0002.

Joseph L Durant, Burton A Leland, Douglas R Henry, and James G Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and computer sciences*, 42(6): 1273–1280, 2002.

Joost Engelfriet. Context-free graph grammars. In *Handbook of formal languages: volume 3 beyond words*, pp. 125–213. Springer, 1997.

Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publications of the*, 1960.

Yuan Gao and Weiwei Sun. A computational simulation of language production in first language acquisition. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10992–11006, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.557.

Jonas Groschwitz, Meaghan Fowlie, Mark Johnson, and Alexander Koller. A constrained graph algebra for semantic parsing with AMRs. In Claire Gardent and Christian Retoré (eds.), *Proceedings of the 12th International Conference on Computational Semantics (IWCS) — Long papers*, 2017. URL https://aclanthology.org/W17-6810/.

Jonas Groschwitz, Meaghan Fowlie, and Alexander Koller. Learning compositional structures for semantic graph parsing. In Zornitsa Kozareva, Sujith Ravi, Andreas Vlachos, Priyanka Agrawal, and André Martins (eds.), *Proceedings of the 5th Workshop on Structured Prediction for NLP (SPNLP 2021)*, pp. 22–36, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.spnlp-1.3. URL https://aclanthology.org/2021.spnlp-1.3/.

Marc D Hauser, Noam Chomsky, and W Tecumseh Fitch. The faculty of language: what is it, who has it, and how did it evolve? *science*, 298(5598):1569–1579, 2002.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pp. 4839–4848. PMLR, 2020.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pp. 10362–10383. PMLR, 2022.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING 2012*, pp. 1359–1376, 2012.

Yoon Kim, Chris Dyer, and Alexander Rush. Compound probabilistic context-free grammars for grammar induction. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2369–2385, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1228. URL https://aclanthology.org/P19-1228/.

Yoon Kim, Chris Dyer, and Alexander M Rush. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2369–2385, 2019b.

Alexander Koller. Semantic construction with graph grammars. In Matthew Purver, Mehrnoosh Sadrzadeh, and Matthew Stone (eds.), *Proceedings of the 11th International Conference on Computational Semantics*, pp. 228–238, London, UK, April 2015. Association for Computational Linguistics. URL https://aclanthology.org/W15-0127/.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Greg Landrum. Rdkit: Open-source cheminformatics software. https://www.rdkit.org, 2026.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pp. 15159–15179. PMLR, 2022.

Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52 (6):1686–1697, 2012.

David L Mobley and J Peter Guthrie. Freesolv: a database of experimental and calculated hydration free energies, with input files. *Journal of computer-aided molecular design*, 28(7):711–720, 2014.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International conference on artificial intelligence and statistics*, pp. 4474–4484. PMLR, 2020.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. The LinGO redwoods treebank: Motivation and preliminary applications. In *COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes*, 2002.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pp. 63–72, Dublin, Ireland, 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2008. URL https://aclanthology.org/S14-2008/.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 915–926, Denver, Colorado, 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2153. URL https://aclanthology.org/S15-2153/.

Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. MRP 2019: Cross-framework meaning representation parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pp. 1–27, Hong Kong, 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-2001. URL https://aclanthology.org/K19-2001/.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pp. 1–22, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.conll-shared.1. URL https://aclanthology.org/2020.conll-shared.1/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A Method for Automatic Evaluation of Machine Translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135.

Karl Pearson. Note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58:240–242, 1895.

Tiago P Peixoto. Nonparametric bayesian inference of the microcanonical stochastic block model. *Physical Review E*, 95(1):012317, 2017.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. A synchronous hyperedge replacement grammar based approach for amr parsing. In *Proceedings of the nineteenth conference on computational natural language learning*, pp. 32–41, 2015.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pp. 412–422. Springer, 2018.

Govindan Subramanian, Bharath Ramsundar, Vijay Pande, and Rajiah Aldrin Denny. Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches. *Journal of chemical information and modeling*, 56(10):1936–1949, 2016.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 01 2018. ISSN 2041-6520. doi: 10.1039/c7sc02664a. URL https://doi.org/10.1039/c7sc02664a.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. Pcfgs can do better: Inducing probabilistic context-free grammars with many symbols. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1487–1498, 2021.

Yajie Ye and Weiwei Sun. Exact yet Efficient Graph Parsing, Bi-directional Locality and the Constructivist Hypothesis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4100–4110, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.377.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 617–626, 2020.

A Formal Properties of Probabilistic HRG

A.1 Hyperedge Replacement Grammar

Let Σ be a finite alphabet. A *hypergraph* over Σ is $H = \langle V, E, l \rangle$, where V is a finite node set, E is a finite set of hyperedges, each $e \in E$ is a non-empty non-repeating sequence of nodes, and $l : E \to \Sigma$ labels edges. The arity of e is its length $|e|$; ordinary directed graphs are the case $|e| = 2$.

A *fragment* is a hypergraph with an ordered interface, $G = \langle V, E, l, Ex \rangle$, where Ex is a non-repeating sequence of external nodes. External nodes are visible to composition; all other nodes are internal. We view hypergraphs as fragments with $Ex = \emptyset$.

An HRG is $\mathcal{G} = \langle N, T, P, S \rangle$, with nonterminal labels N, terminal labels T, start symbol $S \in N$, and productions P. Each label $a \in N \cup T$ has fixed arity $\mathrm{arity}(a)$. A production is $A \to R$, where $A \in N$, R is a fragment over $N \cup T$, and $|Ex_R| = \mathrm{arity}(A)$. Terminal edges remain in the generated graph; nonterminal edges mark expansion sites. Let $\mathcal{D}$ denote the set of finite complete derivation trees and $\mathcal{G}$ the set of terminal graph isomorphism classes. For $G \in \mathcal{G}$, define

$$\mathcal{D}_G = \{\, D \in \mathcal{D} : \mathrm{yield}(D) \cong G \,\}.$$

The sets $\mathcal{D}_G : G \in \mathcal{G}$ partition the set of complete derivations. A graph may have multiple compatible derivations because different applications of the grammar can produce isomophic terminal structures. This ambiguity is explicitly represented in the probability model.

B Dynamic Programming on the Packed Forest

The packed forest $\mathcal{F}(G)$ reduces marginalization over the exponentially many derivations of an observed graph to two recursions whose cost is linear in the size of the forest. We give the recursions, the posteriors derived from them, and the log-space form used in practice.

B.1 Inside Recursion

For an OR-node v, let $\alpha(v)$ denote the total probability of all subderivations rooted at v. For an AND-node e with $\mathrm{par}(e) = v$, let $r(e)$ be its rule and $\mathrm{ch}(e)$ its child OR-nodes. The inside recursion is

$$\alpha(v) \;=\; \sum_{e\,:\,\mathrm{par}(e)=v} p\big(r(e) \mid \mathrm{lhs}(r(e))\big) \prod_{u \in \mathrm{ch}(e)} \alpha(u),$$

with $\alpha(v) = 1$ at every leaf OR-node, those whose only AND-child applies a rule with no nonterminal edges in its right-hand side. Each summand is one local decomposition of the subgraph at v, weighted by its rule probability and by the inside values that aggregate the probabilities of independently deriving the child subproblems. Because $\mathcal{F}(G)$ is acyclic, the recursion is evaluated bottom-up. The marginal of the observed graph is the inside value at the root,

$$p(G) \;=\; \alpha(v_{\mathrm{root}}).$$

B.2 Outside Recursion

Inside values alone do not give posterior usage of a rule application, since they do not account for the derivational context above the local subproblem. For an OR-node v, let $\beta(v)$ denote the total probability mass of all surrounding contexts that connect v to a complete derivation of G, with the convention $\beta(v_{\mathrm{root}}) = 1$. For a non-root OR-node u,

$$\beta(u) \;=\; \sum_{e\,:\,u \in \mathrm{ch}(e)} \beta\big(\mathrm{par}(e)\big)\, p\big(r(e) \mid \mathrm{lhs}(r(e))\big) \prod_{w \in \mathrm{ch}(e) \setminus \{u\}} \alpha(w),$$

where the sum runs over every AND-node in which u appears as a child. The three factors record, in order, the context above the parent subproblem, the rule that introduces u, and the independent derivation of the sibling subproblems. The recursion is evaluated top-down on the acyclic forest.

B.3 Posterior Rule Probabilities and Expected Counts

Combining inside and outside values gives the posterior probability that an AND-node e is used in a derivation of G,

$$p(e \mid G) \;=\; \frac{\beta\big(\mathrm{par}(e)\big)\, p\big(r(e) \mid \mathrm{lhs}(r(e))\big) \prod_{u \in \mathrm{ch}(e)} \alpha(u)}{p(G)}.$$

The numerator is the total mass of complete derivations that pass through e, and dividing by $p(G)$ normalizes within the posterior over derivations of the observed graph. The posterior expected count of a rule r is then

$$\mathbb{E}[c_r \mid G] \;=\; \sum_{e \,:\, r(e) = r} p(e \mid G),$$

the sum of posterior probabilities over all AND-nodes in $\mathcal{F}(G)$ that instantiate r. Corpus-level expected counts are obtained by summing $\mathbb{E}[c_r \mid G^{(i)}]$ over the dataset and form the sufficient statistics used in the multinomial M-step of Section C.1.

B.4 Log-Space Computation and Differentiability

Each derivation probability is a product of many local rule probabilities, so $\alpha(v)$ underflows in floating point on graphs of practical size. We evaluate both recursions in log space, replacing products with sums and summations with the log-sum-exp operation. The inside recursion becomes

$$\log \alpha(v) \;=\; \mathrm{logsumexp}_{e \,:\, \mathrm{par}(e) = v} \left[\log p\big(r(e) \mid \mathrm{lhs}(r(e))\big) + \sum_{u \in \mathrm{ch}(e)} \log \alpha(u) \right],$$

and the outside recursion is treated identically. This computation is differentiable in the rule log-probabilities, so $\nabla_\theta \log p(G)$ for any differentiable parameterization of $p(r \mid A)$ is obtained by automatic differentiation through the inside pass; the same procedure therefore supports both the closed-form M-step of Section C.1 and the gradient-based training of Sections C.2 and C.3.

B.5 Complexity

Let $|V_{\mathcal{F}}|$ and $|E_{\mathcal{F}}|$ be the numbers of OR- and AND-nodes in $\mathcal{F}(G)$. Each recursion visits each AND-node once and each OR-node once, so both run in time $\mathcal{O}(|V_{\mathcal{F}}| + |E_{\mathcal{F}}|)$ and space $\mathcal{O}(|V_{\mathcal{F}}|)$. The cost of exact inference therefore scales with the size of the packed forest, which by the bound in Section **??** is polynomial in $|V_G|$ for grammars of bounded treewidth and inputs of bounded degree, despite the derivation set being in general exponential.

C Learning

C.1 Expectation–Maximization

The multinomial parameterization admits a closed-form EM procedure, which we recapitulate from Gao & Sun (2025). The E-step uses the inside and outside passes over the packed forest to compute, for each rule r and graph $G^{(i)}$, the posterior expected count $\mathbb{E}[c_r \mid G^{(i)}]$ — the expected number of times r is used in derivations of $G^{(i)}$ — in time linear in the size of the forest (Appendix B). The M-step normalizes the aggregated counts within each left-hand-side class,

$$p(r \mid A) \;=\; \frac{\sum_{i=1}^{N} \mathbb{E}[c_r \mid G^{(i)}]}{\sum_{r' \in P_A} \sum_{i=1}^{N} \mathbb{E}[c_{r'} \mid G^{(i)}]}, \qquad r \in P_A.$$

The two steps alternate until $\mathcal{L}(\theta)$ converges. EM applies only to the multinomial parameterization: the closed-form M-step relies on each $p(r \mid A)$ being a free per-rule scalar, which neither the neural nor the compound parameterization provides.

C.2 Gradient Descent

The marginal log-likelihood of can be optimized directly by gradient descent. We reparameterize each rule by an unconstrained logit $s_r \in \mathbb{R}$ and take a grouped softmax within each left-hand-side class,

$$p(r \mid A) \;=\; \frac{\exp(s_r)}{\sum_{r' \in P_A} \exp(s_{r'})}, \qquad r \in P_A.$$

The inside recursion is evaluated in log-space and is differentiable in the rule probabilities, so automatic differentiation through $\log p(G^{(i)}; \theta) = \log \alpha(v^{(i)}_{\text{root}})$ yields the exact gradient of $\mathcal{L}$ with respect to the logits. On the multinomial parameterization this defines the same probabilistic model as EM and offers no statistical advantage over the closed-form M-step; its value is that the logits need not be free per-rule scalars and may instead be produced by a neural scorer, where parameter sharing across rules precludes a closed-form M-step.

C.3 Amortized Variational Inference

The compound parameterization makes the marginal $p(G) = \int p(z)\, p(G \mid z)\, dz$ intractable, and we replace it by the evidence lower bound (ELBO). A diagonal Gaussian variational posterior $q_\phi(z \mid G) = \mathcal{N}(z;\, \mu_\phi(G), \mathrm{diag}(\sigma_\phi(G)^2))$ produces μ_ϕ and σ_ϕ from an amortized encoder shared across graphs (the encoder branch of Figure 3). The reparameterization $z = \mu_\phi(G) + \sigma_\phi(G) \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$ lets gradients flow through samples, and the scorer and encoder parameters are trained jointly on

$$\mathcal{L}_{\text{ELBO}} \;=\; \sum_{i=1}^{N} \Big[\mathbb{E}_{q_\phi(z \mid G^{(i)})}[\log p(G^{(i)} \mid z)] - \mathrm{KL}\Big(q_\phi(z \mid G^{(i)}) \,\|\, p(z) \Big) \Big].$$

The conditional likelihood $\log p(G^{(i)} \mid z)$ inside the expectation is computed by the inside recursion with rule probabilities $p(r \mid A, z)$, so the variational approximation enters only the outer integral over z, not the marginalization over derivations.

D Synthetic Grammar Construction and Robustness

Our synthetic experiments are designed to isolate the estimation of latent probabilities from assumptions specific to any natural graph domain. We construct grammars by exhaustively enumerating label assignments over a predefined set of graph topologies. The resulting rule set is complete with respect to these topologies and labels. The topology set and the grammar's arity constraints determine the family of graphs under consideration, while the rule probabilities are sampled independently of any particular dataset.

D.1 Formal Construction

Let $\mathcal{T}$ be a finite set of right-hand-side graph topologies, and let N and Σ be finite set of nonterminal and terminal labels, respectively. For each topology $t \in \mathcal{T}$, we enumerate all assignments of of labels to its edges, respecting the interfaces. Together, these rules define an HRG $\mathcal{G} = (N, \Sigma, R, S)$.

We partition the rule set by left-hand-side and arity,

$$R = \bigsqcup_{A,k} R_{A,k},$$

where $R_{A,k}$ contains productions rewriting an A-labeled nonterminal with k external nodes.

Ground-truth distribution. For each nonterminal A, we independently sample

$$\boldsymbol{\theta}^*_A \sim \mathrm{Dirichlet}\left(\alpha \mathbf{1}_{|R_A|}\right),$$

where R_A is the set of productions with left-hand side A. Within each arity class, we normalize the corresponding restriction:

$$\theta^*_{r|A,k} = \frac{\theta^*_{A,r}}{\sum_{r' \in R_{A,k}} \theta^*_{A,r'}}, \qquad r \in R_{A,k}.$$

The resulting distributions have the same product-of-Dirichelets form as the independently normalized rule distributions estimated by the multinomial HRG.

Graph generation. Given the sampled parameters θ^*, we recursively expand the start symbol S. At each expansion site (A, k), a production is sampled according to

$$r \sim \text{Categorical}\left(\boldsymbol{\theta}^*_{A,k}\right)$$

The rule's external nodes are identified with the attachment nodes of the rewritten hyperedge, while its internal nodes are freshly allocated. Rule choices at distinct expansion sites are conditionally independent.

For a complete derivation d, let $c_r(d)$ be the number of applications of production r. Its probability is

$$P_{\theta^*}(d) = \prod_{r \in R} (\theta^*_{r|A(r),k(r)})^{c_r(d)}$$

where $A(r)$ and $k(r)$ identify the rule's normalization class. Each observed graph his the yield of an accepted derivation, considered up to node renaming.

Latent derivations. After generation, we discard the sampled derivation and retain only its graph G. We parse G again using the complete grammar, constructing the packed forest

$$\mathcal{F}(G) = \{d' : \text{yield}(d') = G\}\,.$$

Assuming exhaustive parsing, the forest contains the generating derivation as well as every other compatible derivation.

Learning uses the same marginal likelihood objective as in the natural data experiments,

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log \sum_{d \in \mathcal{F}(G_i)} P_\theta(d).$$

The generating parameters θ^* are used only to evaluate recovery and are never provided to the learning algorithm.

E HRG Model Configuration

We use 4000 graphs from the Redwoods dataset, split into 3198 trianing and 802 held-out graphs. We deterministically extract a SHRG grammar from the gold training derivations, yielding 14319 rules with 181 distinct LHS labels and 352 LHS and arity groups. The extracted grammar is shared by all models. Although gold derivations are used to construct the grammar, model training is unsupervised.

Model Configurations We compare four parameterizations of the same grammar. EM-HRG estimates using batched inside-outside inference, with 10^{-10} smoothing, for 200 iterations until convergence.

Both neural models use a 64 dimensional group embedding and a 64-dimensional rule representation, concatenated and passed through a two-layer MLP with 128 hidden units and ReLU activation. Rule scores are normalized by a softmax within each LHS and arity group.

Training and model selection. We train both neural models with Adam at a learning rate of 0.05 for 600 optimization steps, using gradient-norm clipping at 1.0 and six random seeds.

F Graph Generation Experiments

We use the entire Redwoods corpus except for the news portion, comprising $44,887$ delexicalized graphs. We split the corpus by sentence into $35,851$ training and 9.036 test graphs.

We extract an HRG from the corpus, with $66,231$ rules in 634 left-hand-side and arity groups, spanning 277 distinct left-hand-side symbols.

GraphRNN The graph-sequence bandwidth is set to $M = 31$, estimated from the data, whose maximum observed step is 56. The model uses a hidden dimension of 128, four GRU layers, an edge hidden dimension of 16, and a batch size of 32. We train for the full 3000 epochs, with learning-rate milestones at epochs 400 and 1000 and no early stopping. We use the best checkpoint for sampling.

DiGress DiGress model uses marginal transition matrices, a cosine noise schedule, and auxiliary structural features comprising 3, 4, 5-cycle counts and Laplacian spectral features. The denoising network is a graph transformer with FiLM conditioning. We use 500 diffusion steps, 8 transformer layers, a node embedding dimension of 256, and approximately 4.52 million parameters. Training runs for 300 epochs.

G Molecular Grammar Construction and Evaluation

G.1 Molecular Grammar Construction

We construct a corpus-independent HRG by automatically enumerating production rules from chemical valence constraints. Rather than extracting recurring molecular motifs or manually specifying individual production rules, we define a small set of general rule templates and instantiate them using a chemical valence table.

Grammar representation. We represent molecules as graphs whose vertices are atoms and whose terminal edges are chemical bonds. Each nonterminal represents a fragment's attachment interface, specifying the outstanding valences of its boundary atoms. We express these outstanding valences as debts in half-units, where single, double, and triple bonds contribute 2, 4, and 6 units. A nonterminal $N_{d_1,\ldots,d_k}$ has arity k with d_i denoting the outstanding valence of its i-th boundary atom.

Given a vocabulary of atom types and their permitted valences, we automatically enumerate three families of production rules:

1. Lexical rules introduce individual atoms.
2. Join rules combine two existing fragments through a chemical bond
3. Ring-closure rules introduce a bond between two boundary atoms of the same fragment, allowing cyclic structures.

Grammar size and coverage. Our final grammar uses arity-two nonterminals, even-valued debts, yielding $2,971$ automatically enumerated rules. Increasing the maximum nonterminal arity expands the class of molecular structures, but also substantially increases computational cost. An arity-three grammar contains $79,947$ rules and can produe a packed derivation forest with approximately 13.5 million hyperedges for a 97-atom molecule. We therefore use the arity-two grammar throughout our experiments to balance structural coverage and tractability. The construction requires no manually selected molecular fragments.

G.2 Evaluation

We evaluate the learned representation on BACE (binary -secretase 1 inhibition), BBBP (binary blood-brain barrier penetration), ESOL (measured aqueous solubility), FreeSolv (experimental hydration free energy). Classification tasks are evaluated using ROC-AUC, while regression tasks are evaluated using root mean squared error (RMSE).