RULE-GUIDED GRAPH SYNTHESIS: INTEGRATING FIRST-ORDER LOGIC INTO DEEP GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing graph generative models produce graphs that are often quite realistic, but sometimes miss domain-specific patterns. Enhancing graph learning with domain knowledge is one of the current frontiers for neural models of graph data. In this paper, we propose a new approach to guiding deep graph generative models with *knowledge that is represented by first-order logic rules.* First-order logic provides an expressive formalism for representing interpretable causal knowledge about relational structures. Our conceptual contribution is a new first-order semantic loss function for training a graph generative model on relational data: maximize the model likelihood subject to a rule moment matching constraint, namely that the expected commuting matrix of each rule matches the observed one. Our algorithmic contribution is a novel matrix-multiplication-based method for computing the expected commuting matrix of a first-order rule for a Graph Variational Autoencoder model. Empirical evaluation across benchmark datasets shows that rule moment matching substantially improves the quality of generated graphs across multiple graph quality metrics.

1 INTRODUCTION

Graph generation requires modelling both individual connections and the relational patterns they form. Different enhancement methodologies are appropriate for different types of knowledge. In this paper, we consider leveraging knowledge in the form of a first-order logic (FOL) knowledge base, comprising a set of first-order formulas. FO formulas represent domain knowledge by specifying important patterns in a domain, which often take the form of if-then rules. An example rule would be "If person X works in city Y, then X lives in city Y (with probability p)." Such formulas describe interpretable relational regularities without requiring every instance to satisfy them.

Approach. Figure 1 shows our system components. FOL rules can and typically do encompass multiple node attributes and multiple edge attributes, which describe patterns in richly attributed graphs, known as heterogeneous information networks (Sun & Han, 2012), knowledge graphs, or heterogeneous graphs (HG) for short (Ling et al., 2021). Our approach aligns with the well-established metapath framework for heterogeneous graph analysis (Sun & Han, 2012). A metapath is a semantically meaningful path template that is specified by a user to represent their semantic understanding of a domain. For example, the metapath A–P–A describes a connection between two authors of the same paper. Logical formulas can describe metapaths; for instance, the formula $\mathrm{Author}(X, P), \mathrm{Paper}(P), \mathrm{Author}(Y, P)$ is equivalent to the A–P–A metapath. FOL expands the metapath approach in two directions: (1) expressive power—formulas can refine paths with additional conditions, such as a paper's topic; and (2) learnability—first-order rule learners can be leveraged to suggest rules to users in a mixed-initiative approach.

Our approach introduces a novel neuro-symbolic **semantic loss**, which encodes symbolic knowledge into the network's loss function (Kautz, 2022; Xu et al., 2018; Marra et al., 2019). Our semantic loss is based on **moment matching**, which requires that for each formula/metapath, *the expected instance count for a model should match the observed instance count in a training graph.* While in classical statistics, moment-matching (MM) is viewed as an alternative to maximizing likelihood (ML), we

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.

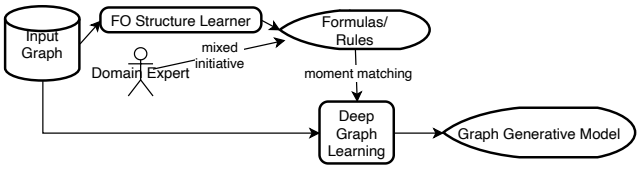

Figure 1: System Overview for Rule-Guided Graph Generation

develop a **hybrid objective** that combines training graph likelihood with moment matching. Intuitively, moment-matching enforces global semantic patterns, whereas graph likelihood measures local correlations (e.g., presence of specific edges).

We implement moment matching at the level of commuting matrices, which we call **commuting matrix moment matching** (MMM for short). Rather than a single instance count, for each selected rule a commuting matrix records the number of satisfying assignments conditional on its endpoint nodes (cf. (Sun et al., 2011)). To make moment-matching computationally feasible, we develop a *differentiable new matrix multiplication method* for computing observed and expected commuting matrices.

We develop MMM for graph-level GraphVAE (Simonovsky & Komodakis, 2018; Zahirnia et al., 2022), resulting in the rule-guided model **GraphVAE+RG**. Matrix operations allow gradients to propagate from relational statistics to the decoder. Once trained, the model generates graphs directly, without invoking a rule learner or symbolic inference procedure.

Evaluation. Our baseline model is GraphVAE (Simonovsky & Komodakis, 2018; Zahirnia et al., 2022). We evaluate on six datasets using standard graph statistics and precision-recall metrics of graph realism. GraphVAE+RG substantially improves GraphVAE, by up to two orders of magnitude on some datasets and metrics (e.g., the GRID orbit MMD improves from 0.7617 to 0.0078). Compared to DeFoG, a recent discrete graph flow-matching method (Qin et al., 2025), GraphVAE+RG is competitive. It also outperforms a recent latent space diffusion method for heterogeneous graph generation (Zhou et al., 2024) on most datasets and metrics.

Contributions Our main contributions can be summarized as follows.

- A new approach to graph generation guided by domain knowledge from first-order formulas viewed as generalized metapaths.
- A new semantic loss objective function: Maximize the data likelihood of a generative model, such that the expected commuting matrix of each rule matches the observed commuting matrix.
- A new matrix multiplication algorithm for computing the commuting matrix of a rule in a graph.
- A proof that the matrix multiplication algorithm also computes the expected commuting matrix exactly for a standard mixture model, under conditional independence given the latent vector. It can therefore be leveraged to compute the new semantic loss objective.

2 Related Work

Neuro-symbolic learning. Our work falls under the heading of *neuro-symbolic AI*, a cutting-edge field of AI that aims to combine symbolic formalisms, such as first-order logic, with neural network learning (Raedt et al., 2020; Garcez & Lamb, 2023; Kautz, 2022). Much of neuro-symbolic AI explores MCP-type approaches for leveraging *symbolic computations*, where the neural system (e.g., LLM) calls a symbolic program at test time. This paper focuses on leveraging *symbolic knowledge*, which is encoded into the network's semantic loss function; see Figure 2. The trained system is a standard NN model that does not utilize symbolic computations at test time. Compared to previous semantic loss approaches (Xu et al., 2018), our main innovation is that *we incorporate knowledge expressed in first-order logic, rather than the less powerful formalism of propositional logic*. For

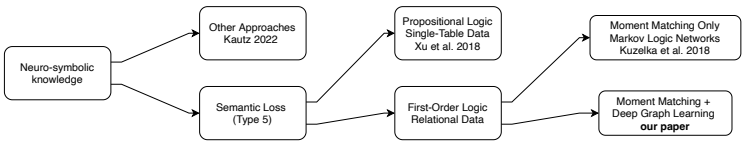

Figure 2: Within neuro-symbolic AI, we develop a new first-order semantic loss approach.

example, a propositional rule would be "if a movie is a horror movie, it is not likely to be a romance". A first-order rule could be "if a user rates a horror movie, the user is most likely to be a man". Since first-order rules incorporate relationships, *first-order logic (FOL) can leverage the full power of relational data.* Compared to previous FOL approaches (e.g., (Marra et al., 2019)), we use standard FOL semantics (not fuzzy logic), and our computations do not require as input the full Cartesian product grounding over all domain elements (nodes).

Statistical-Relational Learning (SRL) and Moment Matching. We use the same FO knowledge representation structure as the well-known **Markov Logic Network** (MLN) model, namely a set of FO formulas (Domingos & Lowd, 2019). Kuzelka et al. (2018) show that a distribution P over graphs maximizes entropy subject to moment matching if and only if P is defined by an MLN with maximum likelihood weights, which therefore captures global graph patterns represented by instance counts. However, the deep likelihood function can *in addition* capture local graph patterns. For example, matching the number of observed triangles in a graph is unlikely to capture community structure, or which nodes have special properties such as centrality.

Deep graph generation. Of the canonical deep generative models, we selected GraphVAE (Simonovsky & Komodakis, 2018) as our baseline for two reasons. 1) GraphVAE generates attributed graphs directly, and its probabilistic decoder supports differentiable rule statistics; 2) As we show in the paper, we can compute expected commuting matrices for a GraphVAE model exactly, under the conditional-independence assumption of Proposition 2. This keeps the computational overhead from moment matching small, and allows us to assess its import without approximation schemes as a confounding variable.

3 Background on First-Order Logic

Attributed Heterogeneous Graphs An attributed graph is a pair $G = (V, E)$ where V is a set of nodes of size $|V| = n$ and $E \subseteq V \times V$ is a set of edges. Node features are summarized in $n \times f$ matrix $\mathbf{X}$ and node labels in a $n \times c$ matrix $\mathbf{L}$ where the u-th row of $\mathbf{L}$ is a one-hot encoding of the label of node u. Different edge types are represented by a set of adjacency matrices $A = \{A_1, \ldots, A_K\}$. The notation $A_t[u, v] = 1$ indicates that there is a link $u \rightarrow_t v$ of type t from node u to node v. Figure 3(a) shows a small heterogeneous graph with node types Author and Paper.

First-Order Logic We follow previous work in SRL (Schulte & Gholami, 2017; Kimmig et al., 2014). A **population** is a set of individuals of the same type (e.g., a set of $Authors$, a set of $Papers$). Individuals are denoted by constants (e.g., han and $kare$). An attributed graph specifies a set of individuals (nodes) for each type. A **node variable** ranges over a population, and is denoted in upper case such as X, Y, P, U, V. A unary **functor** maps an individual to a value, and corresponds to a node attribute/label. A binary functor maps an ordered pair of individuals to a value, and corresponds to an edge/edge type. Functors are denoted f, f' etc.

A **first-order** term (FOT) is of the form $f(\boldsymbol{U})$ where each node variable U_i is of the appropriate type. FOT examples are $\mathrm{Paper}(P)$ and $\mathrm{Author}(X, P)$. A FOT can be instantiated with individual constants, much like an index in a plate model (Kimmig et al., 2014). A **grounding** $\boldsymbol{U} = \boldsymbol{u}$ for a list of FOTs simultaneously replaces each node variable in the list by a constant. A ground term, Python-style, assigns individuals as argument to node variables, then applies the functor to return a value. Examples are $\mathrm{Paper}(P = kare)$, and $\mathrm{Author}(X = han, P = kare)$.

An FO **literal** is of the form $\alpha \equiv f(\boldsymbol{U}) = v$. A **conjunction** is a list of literals $\phi = \alpha_1, \ldots, \alpha_s$. We write $\phi(\boldsymbol{U})$ for an FO conjunction and $\phi(\boldsymbol{U} = \boldsymbol{u})$ for a ground conjunction. A graph G **satisfies** a

ground literal if the graph assigns value v to the ground term $f(\boldsymbol{U} = \boldsymbol{u})$, and satisfies the conjunction ϕ if it satisfies each ground literal in the conjunction. We say that a grounding is **valid** if it assigns different nodes to different node variables. For example, $U = u, V = v$ is a valid grounding for $R(U, V)$, but $U = u, V = u$ is not. Throughout the paper we consider only valid groundings.

We write ω for a grounding $\boldsymbol{U} = \boldsymbol{u}$, $\Omega_\phi(G)$ for the set of groundings of ϕ in G, and $G \models \phi[\omega]$ if G satisfies the ground conjunction $\phi[\omega]$. The **instance count** $n_\phi(G)$ in a graph G returns the number of ϕ-groundings satisfied by graph G.

A **probabilistic graph** $\tilde{G}$ assigns a probability $p_{\tilde{G}}(\alpha(\boldsymbol{U} = \boldsymbol{u}))$ to each ground literal. The **probabilistic instance count** of a conjunction (Kuzelka, 2023) is the probability product, summed over all conjunction groundings.

Examples. $\mathrm{Author}(X, P), \mathrm{Paper}(P), \mathrm{Author}(Y, P)$ is an FO conjunction. Its grounding $X = han, P = kare, Y = jiang$ is satisfied by the graph of Figure 3(a). In the probabilistic graph Figure 3(b), the probability of this conjunction is $0.9 \times 0.8 = 0.72$ (node types are observed). The graph has four satisfying valid groundings, one per paper for each order of the two authors, so the instance count is 4. *An FO conjunction generalizes a metapath*, and its instance count is the number of metapath instances (for non-path patterns such as cycles, the motif count (Ma et al., 2019); see Figure 3 for illustration).

Rule commuting matrices. We write ϕ_r for the r-th selected conjunction. For example,

$$\phi_r(X, P, Y) \equiv \big(\,\mathrm{Author}(X, P) = 1,\ \mathrm{Paper}(P) = 1,\ \mathrm{Author}(Y, P) = 1\big) \tag{1}$$

describes the A–P–A metapath of the introduction; adding $\mathrm{Venue}(P) = ICLR$ restricts the intermediate node. The values of its literals can specify absence as well as presence. Thus a family of conjunctions over the same functors can characterize both a probabilistic implication and its exceptions.

For a conjunction ϕ_r with two designated endpoint variables U_r, W_r, let $\Omega_r^{uv}(G)$ contain the groundings in $\Omega_{\phi_r}(G)$ that assign $U_r = u$ and $W_r = v$. Its commuting matrix records the instance count of the metapath represented by ϕ_r for each endpoint pair:

$$[T_r(G)]_{uv} = \sum_{\omega \in \Omega_r^{uv}(G)} \mathbf{1}\{G \models \phi_r[\omega]\}. \tag{2}$$

Here $\mathbf{1}$ is the indicator of a true grounded conjunction. Each entry records how many satisfying assignments of the internal variables connect its endpoints. Because every grounding assigns exactly one endpoint pair, summing all entries recovers the instance count, $n_{\phi_r}(G) = \sum_{u,v} [T_r(G)]_{uv}$. Different conjunctions over the same functors define different matrices. For the A–P–A conjunction with endpoints X, Y, $[T_r(G)]_{han,jiang} = 2$ in Figure 3(a), the number of papers the two authors wrote together; for metapaths, T_r is the commuting matrix of Sun et al. (2011). These endpoint-resolved statistics specify which relational configurations the generator is encouraged to reconstruct.

4 Rule-Guided Graph Generation

This section considers how to guide the training of a parametrized graph generative model (GGM) P_θ on a training graph G_i from a collection $\mathcal{D}_{\mathrm{tr}} = \{G_i\}_{i=1}^N$, with a list of conjunctions $\phi_1, \ldots, \phi_k$, indexed by $r \in \mathcal{R}$. Our *semantic loss objective* maximizes the data likelihood $P_\theta(G_i)$, subject to the FO moment matching constraint that $E_\theta[T_r] = T_r(G_i)$, where $T_r(G_i)$ is the **observed commuting matrix** of conjunction ϕ_r, and $E_\theta[T_r] \equiv \sum_G P_\theta(G)\, T_r(G)$ is the **expected commuting matrix** for the GGM. For a mixture model GGM, we derive the following Lagrangian evidence lower bound (ELBO).

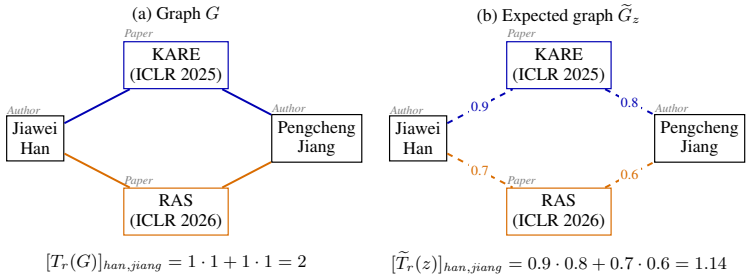

Figure 3: Running example: the A–P–A conjunction $\mathrm{Author}(X, P), \mathrm{Paper}(P), \mathrm{Author}(Y, P)$ with two real instances, as Jiawei Han and Pengcheng Jiang co-authored KARE (Jiang et al., 2025) and RAS (Jiang et al., 2026). (a) In graph G, their commuting-matrix entry is 2. (b) In an expected graph $\widetilde{G}_z$, with edge probabilities from the decoder, the soft entry is 1.14; moment matching moves it towards 2.

Proposition 1. *Suppose that* $P_\theta(G) = \int p(G|z)p(z)dz$ *is a mixture model. Then*

$$\ln P_\theta(G_i) - \lambda/k \sum_{r\in\mathcal{R}} \rho(T_r(G_i), E_\theta[T_r]) \geq \tag{3}$$

$$E_{z\sim q_\psi(z|G_i)}[\ln P_\theta(G_i|z) - KL\big(q_\psi(z|G_i)||p(z)\big) \tag{4}$$

$$-\lambda/k \sum_{r\in\mathcal{R}} \rho(T_r(G_i), E_\theta[T_r|z])], \tag{5}$$

where q_ψ *is the GraphVAE encoder (Section 4.1) and* $\rho(count_1, count_2) \geq 0$ *is a differentiable count distance metric convex in* $E_\theta[T_r|z]$.

Proposition 1 says that the constrained likelihood equation 3 can be approximated by our new **moment matching variational ELBO objective** equation 4.

To compare an expected commuting matrix to an observed one, our experiments use

$$\rho\big(T_r(G_i), E_\theta[T_r|z_i]\big) = -\frac{1}{|\mathcal{I}_r|} \sum_{a\in\mathcal{I}_r} \ln \mathcal{N}\big(T_r(G_i)_a \,\big|\, \widetilde{T}_r(z_i)_a, \sigma_r^2\big), \tag{6}$$

where $\widetilde{T}_r(z_i) = E_\theta[T_r|z_i]$ is the soft commuting matrix for $z_i \sim q_\psi(z|G_i)$, defined in equation 9 (Proposition 2), and $\mathcal{I}_r$ indexes the matrix entries compared for rule r. We estimate σ_r as the root-mean-square residual of rule r in the current minibatch of B graphs, plus a small constant $\sigma_0 > 0$ that prevents a vanishing scale:

$$\sigma_r = \sigma_0 + \Big(\frac{1}{B|\mathcal{I}_r|} \sum_{j=1}^{B} \sum_{a\in\mathcal{I}_r} \big[T_r(G_j)_a - \widetilde{T}_r(z_j)_a\big]^2\Big)^{1/2}. \tag{7}$$

Because commuting matrices can differ considerably in magnitude, this per-rule scale keeps rules with large counts from dominating the loss; σ_r is recomputed for every minibatch and treated as a constant when computing gradients, so that equation 6 is convex in $\widetilde{T}_r(z_i)$, as required by Proposition 1. The classic moment-matching loss (Hansen, 1982) also uses the (unnormalized) squared distance between observed and expected moments.

We can interpret equation 3 as the sum of two log-likelihood functions: 1) the standard graph model likelihood, which is based on local features and adjacencies, and 2) a multi-variate Gaussian likelihood for the entries of the rule commuting matrices. Our training ELBO is the sum of the ELBOs for 1) and 2). The training objective, averaged over the graphs of a minibatch, is

$$\mathcal{L}_{\mathrm{RG}} = \mathcal{L}_{\mathrm{GV}} + \lambda/k \sum_{r\in\mathcal{R}} \rho\big(T_r(G_i), \widetilde{T}_r(z_i)\big), \tag{8}$$

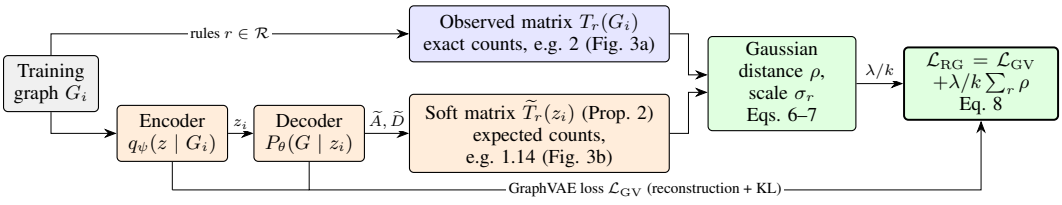

Figure 4: Computing the moment matching objective. For each rule r, the observed commuting matrix $T_r(G_i)$ (top) and the soft commuting matrix $\widetilde{T}_r(z_i)$ from the decoder's probabilities (bottom) are computed by the same matrix product (Section 4.3); their Gaussian distance ρ is added to the GraphVAE loss.

where $\mathcal{L}_{\mathrm{GV}}$ is the GraphVAE loss (Section 4.1) and $k = |\mathcal{R}|$ is the number of selected rules, which all contribute equally. The hyperparameter λ controls the importance of moment matching and can be interpreted as a Lagrange multiplier. Setting $\lambda = 0$ removes the rule objective. The Gaussian term is a reconstruction penalty; Proposition 1 gives the sense in which equation 8 approximates a moment-matching-constrained likelihood objective.

4.1 Implementing the Moment Matching ELBO

GraphVAE represents each graph by a latent vector z. At a high level, a graph encoder $q_\psi(z|G_i)$ maps a training graph G_i to a latent vector z_i. A graph MLP decoder maps a latent vector z to a distribution $P_\theta(G|z)$ over attributed graphs, given by edge and attribute probabilities. The training loss is an ELBO approximation to the data log-likelihood. The decoder provides probabilities for connectivity and available categorical attributes. We write its base training objective as $\mathcal{L}_{\mathrm{GV}}$: graph and attribute reconstruction, Kullback–Leibler regularization toward the prior. Only attributes present in a dataset contribute reconstruction targets.

Section 4.3 specifies the computation of $\widetilde{T}_r(z_i)$. We train the decoder to match $\widetilde{T}_r(z_i)$ to $T_r(G_i)$. Figure 4 shows how the resulting moment matching objective is computed.

Training differentiates equation 8 through the matrix statistics and decoder. Generation draws $z \sim p(z)$ and uses the decoder without running the rule learner or counter.

4.2 Computing Expected Commuting Matrices

Given a latent vector z, the **expected graph** $\tilde{G}_z$ is a probabilistic graph that assigns a probability to each ground literal by applying the decoder to the relevant links/node features/edge types. For examples see Figure 3(b) and Figure 6. For a grounding ω, we assume that no two literals of $\phi_r[\omega]$ refer to the same ground literal (e.g., the same edge), so that their indicators are distinct random variables. Then

$$[\widetilde{T}_r(z)]_{uv} = \sum_{\omega \in \Omega_r^{uv}} \prod_{i=1}^{s} p(\tilde{G} \models \alpha_i(\omega)). \tag{9}$$

Here Ω_r^{uv} contains the groundings of ϕ_r with endpoints u, v; it depends only on the node set. This is the endpoint-resolved version of the probabilistic instance count (Section 3). In Figure 3(b), $[\widetilde{T}_r(z)]_{han,jiang} = 0.9 \cdot 0.8 + 0.7 \cdot 0.6 = 1.14$; the loss equation 6 moves it towards the observed value 2.

Proposition 2. *For a fixed node domain and grounding restrictions, if the distinct graph-state variables in each grounding are conditionally independent given z, then the expected commuting matrix given a latent vector can be computed as the commuting matrix of the expected graph:* $[\widetilde{T}_r(z)]_{uv} = \mathbb{E}[T_r(G)_{uv} \mid z]$.

The proof is given in Appendix A.

Outside these conditions, the product of probability matrices is a differentiable surrogate (Appendix A).

The upshot is that *FO moment matching can be implemented by performing (probabilistic) instance counting in a single graph*.

4.3 Computing Commuting Matrices With Matrix Multiplication

To compute the observed and expected commuting matrix for each rule, we develop a novel matrix multiplication method, that is differentiable and applies to both discrete and probabilistic graphs. Our method maps each conjunction to a sequence of matrix multiplications, such that *the conjunction's commuting matrix can be found by executing the matrix multiplications.*

To illustrate the basic idea, consider the A–P–A conjunction $\mathrm{Author}(X,P)$, $\mathrm{Paper}(P)$, $\mathrm{Author}(Y,P)$ of Figure 3, corresponding to a metapath X–P–Y (reading $\mathrm{Author}(Y,P)$ as an edge from P to Y). This is an example of a **chain conjunction** of binary literals, of the form $\phi = \alpha_1(U_1,V_1),\ldots,\alpha_L(U_L,V_L)$ where $V_i = U_{i+1}$ for every i. *If every node variable occurs only once in a chain conjunction*, except consecutively, then we can use the metapath method of multiplying the corresponding adjacency matrices (Sun et al., 2011). In our example conjunction, the commuting matrix can be computed as $A_{\mathrm{AP}}A_{\mathrm{AP}}^{\top}$ where A_{AP} is the author–paper adjacency matrix (the matrix A_t of edge type Author), with the diagonal set to zero because $X = Y$ is not a valid grounding; applied to the soft adjacency matrix, the same product gives $\widetilde{T}_r$ (Figure 3(b)). The method extends to node features, which are mapped to diagonal matrices, and to chain conjunctions whose repeated node variables do not cross; the full method is described in Appendix B.

4.4 Rule Learning and Rule Selection

Rule selection determines the conjunctions in $\mathcal{R}$; they remain fixed during generator training. The method therefore separates acquiring relational knowledge from using it as guidance. Candidate rules can be supplied by a domain expert or proposed by a first-order rule learner. We utilize FactorBase (Schulte & Qian, 2015) as a SOTA Markov Logic structure learner. FactorBase outputs rules of the form $\boldsymbol{B} \rightarrow H$ where the body $\boldsymbol{B}$ is an FO conjunction and the head H is a single FO literal. As FactorBase can output many rules, we prune them with a BIC-type rule quality metric and keep all rules with a positive score.

Rule pruning reduces the number of rules for scalability, and also simulates the impact of a domain expert selecting the most important rules. Appendix F provides more details on rule learning.

5 Evaluation

5.1 Questions and datasets

We examine two questions: does rule moment-matching improve the GraphVAE backbone, and is the graph quality of the improved backbone competitive with recent graph generation methods? We use six datasets: three synthetic graph families (GRID, triangular grid, LOBSTER) and three attributed molecular and protein datasets (PTC, PROTEINS, QM9); full descriptions are given in Appendix C.

5.2 Compared models

We compare GraphVAE without the rule objective, rule-guided graph generation GraphVAE+RG (**our method**), and DeFoG (Qin et al., 2025). DeFoG is a SOTA graph generation method based on discrete flow matching. We extended it to graphs with general node attributes following the authors' suggestions.[1] DeFoG is not a direct competitor to GraphVAE as it is in a different model class; we report DeFoG metrics to benchmark the rule-guided model against a strong recent graph generator. We also compare with LGD (Zhou et al., 2024), a latent-space diffusion model for heterogeneous graph generation.

[1] https://github.com/manuelmlmadeira/DeFoG

Table 1: Statistics-Based Evaluation. Entries report squared MMD across five structural descriptors, given as mean ± sample SD across generator seeds. Lower is better. Bold indicates the smallest mean within a dataset/metric.

Dataset	Method	Degree ↓	Clustering ↓	Orbit ↓	Spectral ↓	Diameter ↓
GRID	GraphVAE	0.1359 ± 0.0667	0.1207 ± 0.0276	0.7617 ± 0.1655	0.0215 ± 0.0042	0.2071 ± 0.0709
	GraphVAE+RG	**0.0033 ± 0.0019**	**0.0030 ± 0.0023**	**0.0078 ± 0.0114**	**0.0103 ± 0.0013**	**0.1278 ± 0.0261**
	DeFoG	0.0094 ± 0.0121	0.0138 ± 0.0137	0.0194 ± 0.0202	0.0228 ± 0.0027	0.4560 ± 0.3616
	LGD	0.2797 ± 0.2084	0.9774 ± 0.3149	0.5651 ± 0.4610	0.0512 ± 0.0218	1.0572 ± 0.1326
Triangular grid	GraphVAE	0.0089 ± 0.0082	0.1376 ± 0.0743	0.0777 ± 0.0571	0.0187 ± 0.0032	0.0814 ± 0.0277
	GraphVAE+RG	**0.0017 ± 0.0012**	**0.0815 ± 0.0291**	**0.0139 ± 0.0071**	**0.0165 ± 0.0008**	**0.0314 ± 0.0235**
	DeFoG	0.0137 ± 0.0092	0.9965 ± 0.1794	0.0839 ± 0.1214	0.0309 ± 0.0031	0.2143 ± 0.0240
	LGD	–	–	–	–	–
LOBSTER	GraphVAE	0.0588 ± 0.0279	0.4181 ± 0.1349	0.1846 ± 0.0663	0.0272 ± 0.0082	0.0969 ± 0.0093
	GraphVAE+RG	0.0112 ± 0.0057	0.0450 ± 0.0286	0.0251 ± 0.0075	0.0191 ± 0.0048	**0.0479 ± 0.0346**
	DeFoG	**0.0006 ± 0.0002**	**0.0030 ± 0.0028**	**0.0172 ± 0.0089**	**0.0078 ± 0.0019**	0.2056 ± 0.0333
	LGD	0.0277 ± 0.0035	0.0338 ± 0.0156	0.0194 ± 0.0118	0.0224 ± 0.0063	0.1235 ± 0.0741
PTC	GraphVAE	0.0498 ± 0.0042	0.2429 ± 0.0890	0.0249 ± 0.0123	0.0211 ± 0.0033	0.1243 ± 0.0083
	GraphVAE+RG	**0.0098 ± 0.0024**	**0.0265 ± 0.0097**	**0.0008 ± 0.0003**	**0.0114 ± 0.0013**	**0.0485 ± 0.0283**
	DeFoG	0.0145 ± 0.0136	0.0928 ± 0.1278	0.0149 ± 0.0219	0.0151 ± 0.0058	0.0879 ± 0.0579
	LGD	0.0346 ± 0.0137	0.0584 ± 0.0455	0.0051 ± 0.0049	0.0218 ± 0.0059	0.0628 ± 0.0339
PROTEINS	GraphVAE	0.0375 ± 0.0056	0.0259 ± 0.0010	0.0245 ± 0.0042	0.0216 ± 0.0014	0.0974 ± 0.0103
	GraphVAE+RG	0.0219 ± 0.0030	0.0216 ± 0.0014	**0.0107 ± 0.0028**	0.0133 ± 0.0022	0.0421 ± 0.0219
	DeFoG	**0.0116 ± 0.0070**	**0.0160 ± 0.0025**	0.0240 ± 0.0197	**0.0054 ± 0.0017**	**0.0290 ± 0.0173**
	LGD	0.0698 ± 0.0359	0.0466 ± 0.0128	0.0900 ± 0.0147	0.0376 ± 0.0065	0.1348 ± 0.0283
QM9	GraphVAE	0.0195 ± 0.0003	0.1505 ± 0.0094	0.0031 ± 0.0000	0.0033 ± 0.0005	0.0022 ± 0.0017
	GraphVAE+RG	0.0188 ± 0.0009	0.1001 ± 0.0054	0.0009 ± 0.0002	0.0038 ± 0.0002	0.0040 ± 0.0006
	DeFoG	**0.0002 ± 0.0001**	**0.0030 ± 0.0012**	**0.0001 ± 0.0001**	**0.0011 ± 0.0002**	**0.0008 ± 0.0008**
	LGD	0.0021 ± 0.0004	0.0211 ± 0.0005	0.0024 ± 0.0001	0.0017 ± 0.0004	0.0080 ± 0.0022

5.3 Statistics-Based Metrics

As in previous work on graph generation (O'Bray et al., 2022), we measure discrepancies between reference and generated collections through five graph descriptors: degree distributions, local clustering-coefficient distributions, graphlet-orbit counts, normalized-Laplacian spectra, and diameter. An orbit identifies a node's structural position within a small connected subgraph; its descriptor summarizes the average number of occurrences per graph node.

Each descriptor distribution is compared using the squared empirical maximum mean discrepancy (MMD), with the same descriptor-specific kernel and normalization across the compared methods. Smaller values indicate that the generator reproduces a specific structural aspect more faithfully.

5.4 Random-GIN precision–recall

Following Thompson et al. (2022), we also compare distributions of graph embeddings obtained from an untrained random graph neural network. Precision measures generated samples lying within reference neighbourhoods; recall measures reference samples covered by generated neighbourhoods. F1 is their harmonic mean. For random-GIN metrics, ten evaluator initializations are averaged within each generator seed before computing the across-seed mean and sample standard deviation. For details on the random-GIN metrics see Appendix E. Both metric families assess graph topology; native node and edge attributes are excluded from the evaluator inputs, even when a generator models them. The resulting scores describe these processed graphs rather than all properties of the raw generated output. Configuration details are given in Appendix D (Table 3).

5.5 Results

To obtain standard deviations, we initialized training with three different seeds for graph generation.

Improvement over GraphVAE. Table 2 shows **higher mean F1-PR for GraphVAE+RG on all six selected datasets.** GRID increases from 0.3928 to 0.9135, a 132.57% relative gain. As Table 1 shows, graph statistics **improve on almost all datasets**, except for QM9, where spectral and diameter discrepancies worsen slightly. Rule supervision therefore changes the balance of generated properties rather than improving every descriptor automatically.

Table 2: Random-GIN F1-PR. Entries report F1-PR as mean ± sample SD across generator seeds, for GraphVAE, GraphVAE+RG, DeFoG and LGD. The last three columns give the relative change of GraphVAE+RG with respect to each baseline. Higher is better. Bold indicates the largest mean within a dataset.

Dataset	GraphVAE ↑	GraphVAE+RG ↑	DeFoG ↑	LGD ↑	RG vs. GV	RG vs. DeFoG	RG vs. LGD
GRID	0.3928 ± 0.1426	$\mathbf{0.9135 \pm 0.0309}$	0.9002 ± 0.1059	0.3459 ± 0.1957	+132.57%	+1.48%	+164.09%
Triangular grid	0.9035 ± 0.0230	$\mathbf{0.9274 \pm 0.0395}$	0.7946 ± 0.0701	–	+2.65%	+16.72%	–
LOBSTER	0.7525 ± 0.0713	$\mathbf{0.9754 \pm 0.0107}$	0.9709 ± 0.0245	0.9592 ± 0.0141	+29.62%	+0.46%	+1.69%
PTC	0.7896 ± 0.0434	$\mathbf{0.9374 \pm 0.0148}$	0.8927 ± 0.0944	0.8615 ± 0.0309	+18.73%	+5.02%	+8.81%
PROTEINS	0.8968 ± 0.0518	0.9407 ± 0.0077	$\mathbf{0.9712 \pm 0.0076}$	0.8792 ± 0.0412	+4.90%	-3.14%	+6.99%
QM9	0.7814 ± 0.0056	0.8009 ± 0.0022	$\mathbf{0.9804 \pm 0.0021}$	0.9394 ± 0.0150	+2.49%	-18.31%	-14.74%

Comparison with DeFoG. On PTC, GraphVAE+RG has the lowest five structural MMDs and the highest F1-PR. The GRID and triangular-grid summaries show the same ordering on these metrics. LOBSTER is more mixed: GraphVAE+RG better matches diameter, while DeFoG better matches the other displayed structural descriptors. Their F1-PR means are close (0.9754 versus 0.9709), with a difference smaller than either reported between-seed SD. Overall on F1-PR, GraphVAE+RG scores more highly than DeFoG on 4 out of 6 datasets, and substantially worse on QM9. The graph statistics show a mixed, competitive pattern of performance between GraphVAE+RG and DeFoG.

Comparison with LGD. GraphVAE+RG has a higher F1-PR than LGD on four of the five datasets on which LGD was run, all except QM9 (Table 2). On the graph statistics, it has the lower MMD in 20 of the 25 dataset–descriptor pairs; LGD is better on three QM9 descriptors and on the LOBSTER clustering and orbit statistics (Table 1).

Summary. Our experiments provide strong evidence that rule-guided GraphVAE generates more realistic graphs than baseline GraphVAE. Compared with our SOTA flow-matching benchmark, rule guidance boosts the GraphVAE backbone towards competitive performance.

6 Conclusion, Limitations, and Future Work

We proposed a semantic loss objective function for training a deep graph generative model to incorporate FO domain knowledge expressed by logical formulas: maximize the data likelihood subject to a moment matching constraint, which requires the expected commuting matrix of each rule under a model to match the observed commuting matrix. Our main algorithmic contribution is a differentiable matrix multiplication method for computing both observed and expected commuting matrices. The six-dataset comparison shows clearly improved metrics of graph realism over the GraphVAE backbone, competitiveness with DeFoG, and higher F1-PR than LGD on four of the five datasets where LGD was evaluated.

Limitations. Scaling the matrix multiplication algorithm for expected counts is a challenge. A possible direction is approximation algorithms from the related problem of weighted model counting (van Bremen & Kuzelka, 2020). An incorrect or incomplete set of rules limits the effectiveness of the semantic loss function; we did not explore methods for validating the knowledge expressed in formulas, such as human-in-the-loop. Our expected commuting matrix computation is exact only for mixture models under the conditional-independence assumption of Proposition 2; otherwise it is a differentiable surrogate. Using rule guidance to train other graph generation models is a fruitful avenue for future research.

In sum, commuting matrix moment matching (MMM) presents a novel semantic loss approach to neuro-symbolic AI that combines logical rules with deep graph learning. Our experiments show great potential for guiding deep graph generative models with rule-based knowledge.

AI use statement

We did not use generative AI to generate research ideas, create datasets, or assist with proving claims. We used generative AI to improve the clarity and presentation of our writing and scripts, in addition to searching for related works. All edits were reviewed by at least three contributors.

Ethics statement

This work develops a training objective for graph generative models and evaluates it on public synthetic graph families and on public molecular and protein benchmark datasets (TUDataset and QM9). These datasets contain no personal or sensitive information, and no human subjects were involved. The running example uses only publicly available bibliographic facts. As with other generative models for molecules, generated graphs could in principle be misused, for example to propose harmful compounds; our method does not target specific chemical properties and adds no capability beyond existing graph generators. Rules learned from data may encode biases present in the training graphs, so rule sets used for guidance should be inspected by domain experts. We are not aware of other ethical concerns.

Reproducibility statement

The appendix includes Appendix D on hyperparameter choices. Generated graph preprocessing for evaluation removes self-loops and isolates and retains the largest connected component where applicable.

References

Pedro Domingos and Daniel Lowd. Unifying logical and statistical ai with markov logic. *Communications of the ACM*, 62(7):74–83, 2019.

Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning*. MIT Press, 2007.

Artur d'Avila Garcez and Luis C Lamb. Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, pp. 1–20, 2023.

Lars Peter Hansen. Large sample properties of generalized method of moments estimators. *Econometrica: Journal of the econometric society*, pp. 1029–1054, 1982.

Pengcheng Jiang, Cao Xiao, Minhao Jiang, Parminder Bhatia, Taha Kass-Hout, Jimeng Sun, and Jiawei Han. Reasoning-enhanced healthcare predictions with knowledge graph community retrieval. In *ICLR*, 2025.

Pengcheng Jiang, Lang Cao, Ruike Zhu, Minhao Jiang, Yunyi Zhang, Jiaming Shen, Jimeng Sun, and Jiawei Han. RAS: Retrieval-and-structuring for knowledge-intensive LLM generation. In *ICLR*, 2026.

Henry Kautz. The third AI summer: AAAI Robert S. Engelmore Memorial Lecture. *AI magazine*, 43(1):105–125, 2022.

Seyed Mehran Kazemi, David Buchman, Kristian Kersting, Sriraam Natarajan, and David Poole. Relational logistic regression. In *International Conference on the Principles of Knowledge Representation and Reasoning, KRR 2014*, 2014.

Hassan Khosravi, Oliver Schulte, Jianfeng Hu, and Tianxing Gao. Learning compact Markov logic networks with decision trees. *Machine Learning*, 89(3):257–277, 2012.

Angelika Kimmig, Lilyana Mihalkova, and Lise Getoor. Lifted graphical models: a survey. *Machine Learning*, 99(1):1–45, 2014.

Ondrej Kuzelka. Counting and sampling models in first-order logic. In *International Joint Conference on Artificial Intelligence, IJCAI*, pp. 7020–7025, 2023.

Ondrej Kuzelka, Yuyi Wang, Jesse Davis, and Steven Schockaert. Relational marginal problems: Theory and estimation. In *AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 1–8, 2018.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019. URL https://arxiv.org/abs/1904.06991.

Chen Ling, Carl Yang, and Liang Zhao. Deep generation of heterogeneous networks. In *2021 IEEE international conference on data mining (ICDM)*, pp. 379–388. IEEE, 2021.

Chenhao Ma, Reynold Cheng, Laks VS Lakshmanan, Tobias Grubenmann, Yixiang Fang, and Xiaodong Li. Linc: a motif counting algorithm for uncertain graphs. *VLDB Endowment*, 13(2): 155–168, 2019.

Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Lyrics: A general interface layer to integrate logic inference and deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD*, pp. 283–298, 2019.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020. URL https://arxiv.org/abs/2007.08663.

Leslie O'Bray, Max Horn, Bastian Rieck, and Karsten Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. In *International Conference on Learning Representations, ICLR*, 2022.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge university press, 2000.

Zhensong Qian and Oliver Schulte. Factorbase: Multi-relational model learning with sql all the way. In *Data Science and Advanced Analytics (DSAA)*, pp. 1–10, 2015.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *ICML*, 2025. URL https://arxiv.org/abs/2410.04263.

Luc de Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. In *International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4943–4950, 2020.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014. URL https://www.nature.com/articles/sdata201422.

Oliver Schulte and Sajjad Gholami. Locally consistent Bayesian network scores for multi-relational data. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2693–2700, 2017.

Oliver Schulte and Zhensong Qian. FactorBase: SQL for learning a multi-relational graphical model. arXiv preprint arXiv:1508.02428, 2015.

Oliver Schulte and Zhensong Qian. FACTORBASE: multi-relational structure learning with SQL all the way. *International Journal of Data Science and Analytics*, 7(4):1–21, 2018.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018*, pp. 412–422. Springer, 2018.

Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*, volume 3. Morgan & Claypool Publishers, 2012.

Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011. doi: 10.14778/3402707.3402736.

Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W Taylor. On evaluation metrics for graph generative models. In *International Conference on Learning Representations, ICLR*, 2022.

Timothy van Bremen and Ondrej Kuzelka. Approximate weighted first-order model counting: Exploiting fast approximate model counters and symmetry. In *International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4252–4258, 7 2020.

Daisy Zhe Wang, Eirinaios Michelakis, Minos Garofalakis, and Joseph M Hellerstein. Bayesstore: managing large, uncertain data repositories with probabilistic graphical models. In *VLDB*, pp. 340–351, 2008.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International Conference on Machine Learning, ICML*, volume 80, pp. 5502–5511, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019. URL https://arxiv.org/abs/1810.00826.

Kiarash Zahirnia, Oliver Schulte, Parmis Naddaf, and Ke Li. Micro and macro level graph modeling for graph variational auto-encoders. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=zUbMHIxszNp.

Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion. In *Advances in Neural Information Processing Systems*, volume 37, pp. 61963–61999, 2024.

A PROOF THAT EXPECTED INSTANCE COUNTS EQUAL INSTANCE COUNTS IN THE EXPECTED GRAPH

The output of the GraphVAE decoder is an expected graph like Figure 3(b) and Figure 6. The following proposition states that the expected counts can be computed based on an expected graph.

Preliminaries We state the proposition in terms of instance counts rather than the commuting matrix, which provides greater generality and notational simplicity.

We write ω for a grounding $\boldsymbol{U} = \boldsymbol{u}$ and $\Omega_\phi(G)$ for the set of groundings of ϕ in G, so that **the instance count** formula ϕ in graph G is given by

$$n_\phi(G) = \sum_{\omega \in \Omega_\phi(G)} \mathbf{1}\{G \models \phi[\omega]\}. \tag{10}$$

The **probabilistic instance count** of a conjunction (Kuzelka, 2023) in an expected graph is the probability product, summed over all conjunction groundings:

$$n_\phi(\tilde{G}) = \sum_{\omega \in \Omega_\phi(G)} \prod_{i=1}^{s} p(\tilde{G} \models \alpha_i(\omega)) \text{ for } \phi = \alpha_1, \ldots, \alpha_s \tag{11}$$

We restate the definition of first-order chain conjunctions such that only consecutive node variables are equal, and other equalities are enforced by explicit equality constraints. Probabilistically, this means that we can view each variable as an independent selection, and equalities as constraints on groundings.

A **conjunction** ϕ comprises three elements:

1. A list $\alpha_1(U_1, V_1), \ldots, \alpha_L(U_L, V_L)$ comprising L relationship literals where U_i and V_i are node variables.

2. A list $\alpha_1(W_1), \ldots, \alpha_Q(W_Q)$ comprising Q unary literals, where W_i is a node variable.

3. A set of equality constraints EQ of the form $D_k = E_k$ for any two node variables D_k and E_k that appear in the list of relationship literals or unary literals. Formally, EQ is a set of unordered pairs of node variables.

Proposition 2. *The expected conjunction count given a latent vector can be computed as the conjunction count in the expected graph:* $E_{G \sim P(G|z)}[n_\phi(G)] = n_\phi(\tilde{G}_z)$

Proof. Without loss of generality, assume the conjunction is of the form $\alpha_1(U_1 = u_1, V_1 = v_1), \ldots, \alpha_L(U_L = u_L, V_L = v_L), \alpha_1(W_1 = w_1), \ldots, \alpha_Q(W_Q = w_Q)$ with equality constraints EQ. Define the random variables $l_i^{uv}, i = 1, \ldots, L$ to return the indicator value $I_G(\alpha_i(U_i = u, V_i = v)$ and $l_j^w, j = 1, \ldots, Q$ to return the indicator value $I_G(\alpha_j(W_j = w))$. Then

$$
\begin{aligned}
&E_{G\sim P(G|z)}[n_\phi(G)] \\
&= E[\sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} \prod_{i=1}^{L} I_G(\alpha_i(U_i = u_i, V_i = v_i)) \\
&\prod_{j=1}^{Q} I_G(\alpha_j(W_j = w_j))] \\
&= E[\sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} \prod_{i=1}^{L} l_i^{\boldsymbol{u}_i\boldsymbol{v}_i} \prod_{j=1}^{Q} l_j^{\boldsymbol{w}_j}] \\
&= \sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} E[\prod_{i=1}^{L} l_i^{\boldsymbol{u}_i\boldsymbol{v}_i} \prod_{j=1}^{Q} l_j^{\boldsymbol{w}_j}] \qquad (12) \\
&= \sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} \prod_{i=1}^{L} E[l_i^{\boldsymbol{u}_i\boldsymbol{v}_i}] \prod_{j=1}^{Q} E[l_j^{\boldsymbol{w}_j}] \\
&= \sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} \prod_{i=1}^{L} p_{\tilde{G}_z}(\alpha_i(U_i = \boldsymbol{u}_i, V_i = \boldsymbol{v}_i)) \\
&\prod_{j=1}^{Q} p_{\tilde{G}_z}(\alpha_j(W_j = \boldsymbol{w}_j)) = n_\phi(\tilde{G}_z)
\end{aligned}
$$

Equation (12) follows because the expectation of a product of independent random variables is the product of their expectations. The random variables $l_i^{\boldsymbol{u}_i\boldsymbol{v}_i}$ and $l_j^{\boldsymbol{w}_j}$ are independent because the (in)equality constraints ensure that in a valid grounding, no two different literals are ground to the same ground literal. And conditional on the latent vector z, any two different ground literals are independent. □

Commuting matrices. Applying the same argument to the groundings with fixed endpoints (u, v) gives Proposition 2 entrywise for the commuting matrix equation 2.

Tied variables. The proposition does not justify multiplying probabilities of tied variables. For an undirected Bernoulli edge with probability p, traversing it in both directions involves the same random variable, whose squared indicator has expectation p, not p^2. A naive product of probability matrices therefore requires grounding restrictions or corrections before it has the exact interpretation in equation 9. Outside these conditions, it remains a differentiable surrogate. Edge-type probabilities must likewise respect their dependence on edge existence.

B Full Description of Matrix Multiplication Algorithm for Formula Instance Counting

We develop a novel matrix multiplication method for instance counting with conjunctive formulas, that is differentiable and applies to both discrete and probabilistic graphs. For commuting-matrix supervision, the output retains the endpoints of each grounding, as defined in equation 2. To illustrate the basic idea, consider the conjunction $R(U_1, V_1), R(V_1, V_2), R(V_2, U_1)$, whose instance count is six times the number of triangles in an undirected graph represented by an adjacency matrix A. It is well-known that this count is given by $\sum_{u=1}^{n} A^3_{u,u}$, the trace of the third power of the adjacency matrix. We generalize this approach to a large class of logical formulas. A **chain conjunction** of binary literals is of the form $\phi = \alpha_1(U_1, V_1), \ldots, \alpha_L(U_L, V_L)$ where $V_i = U_{i+1}$ for every i. The matrix product equation 13 maps each chain conjunction to a sequence of adjacency matrix multiplications, such that the conjunction's instance count can be found by executing the matrix multiplications.

Supported chain conjunctions. If a node variable occurs repeatedly and non-consecutively, simple matrix multiplication does not suffice. We show that matrix multiplication can be extended to the following large class of chain conjunctions.

1. The conjunction can include any number of node features (e.g., $C(U)$).
2. The binary relationship literals can contain any number of repeated node variables that do not *cross* each other. An example of crossing repetitions is shown by the formula $R(U_1, V_1), R(V_1, U_1), R(U_1, V_1)$.

Generally, let ϕ be a chain conjunction with repeated occurrences of two node variables $U_{k_1} = V_{k_2}$ and $U_{k_3} = V_{k_4}$ in positions $k_i, i = 1, \ldots, 4$. The repetitions **cross** if $k_1 < k_3 < k_2$ and $k_2 < k_4$. Our generalized matrix multiplication method covers a formula like $R(U_1, V_1), R(V_1, V_2), R(V_2, U_1)$ which features a non-consecutive repetition of U_1. Node feature constraints that occur in a formula are mapped to diagonal matrices that select the nodes satisfying the feature constraint.

For two relations R, S, let $A^{(R)}$ and $A^{(S)}$ be their binary adjacency matrices (the matrices A_t of their edge types). For a categorical node attribute x_u, define the diagonal indicator $[D_a]_{uu} = \mathbf{1}\{x_u = a\}$ and define D_b, D_c analogously for states b, c. The rule specifying a relation-R edge followed by a relation-S edge, with node states a, b, c, has matrix

$$T_r = D_a A^{(R)} D_b A^{(S)} D_c. \tag{13}$$

This expression sums over the intermediate node while preserving the endpoints. It applies to the grounding domain admitting these assignments; any additional inequality constraints must also be imposed. Longer chains extend the product, and a relation between exposed endpoints can be imposed through an elementwise mask.

Example. Consider the chain conjunction $AdvisedBy(Student, Professor)$, $Teaches(Professor, Course)$, $TakesCourse(Course, Student)$. Figure 5 shows the corresponding sequence of matrix multiplications in a sample graph. Figure 7 illustrates the matrix multiplication method for the expected graph in our example.

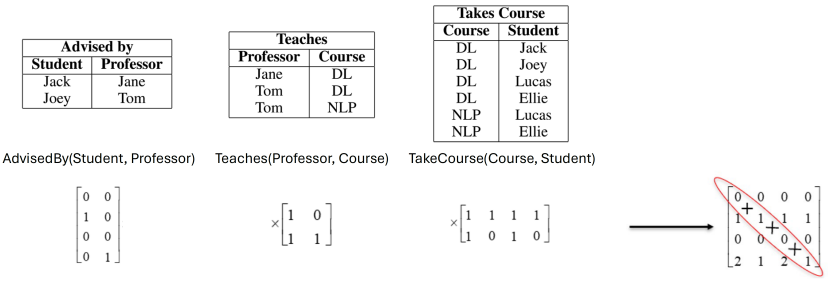

Figure 5: The matrix multiplication sequence for our example conjunction and sample graph data. The final result is 2, which is the number of satisfying groundings in the input graph.

Extensions. Unary literals are mapped to diagonal indicator matrices D_a that select the nodes satisfying them, as in equation 13. Probabilistic instance counts can be obtained by using the soft matrices $\widetilde{A}, \widetilde{D}_a$; see Proposition 2.

Correctness. The next proposition shows that the instance count for the chain conjunction can be obtained through summing over the entries in the constructed matrix product.

Proposition 3. *Let ϕ_r be a chain conjunction with $L \geq 1$ relation factors $A^{(R_1)}, \ldots, A^{(R_L)}$ and diagonal state indicators $D_{a_0}, D_{a_1}, \ldots, D_{a_L}$ on its $L + 1$ nodes, and let $T_r = D_{a_0} A^{(R_1)} D_{a_1} A^{(R_2)} \cdots A^{(R_L)} D_{a_L}$. The product also counts groundings that assign the same node to two different variables; entries that correspond to such invalid groundings are masked (e.g., the*

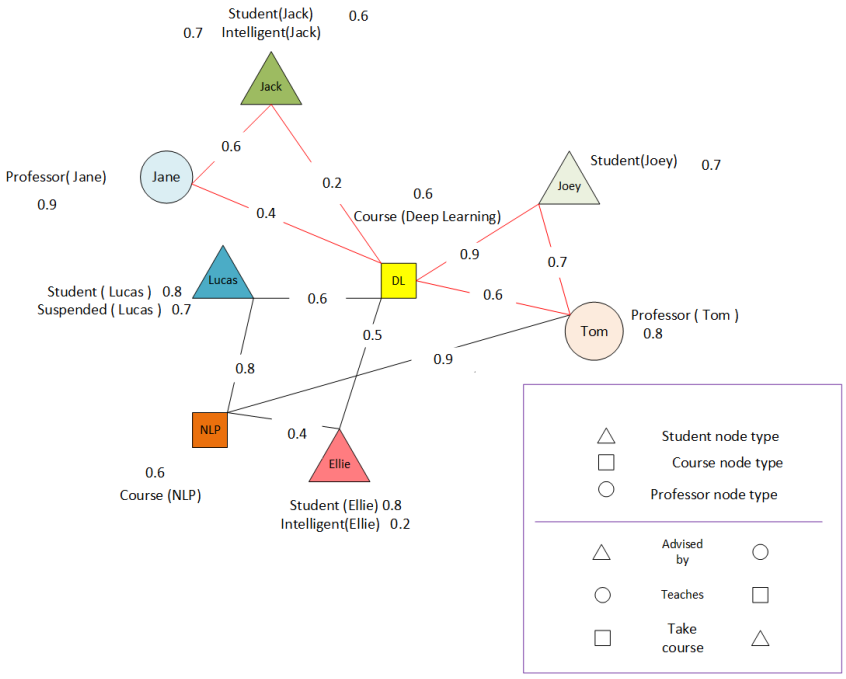

Figure 6: An example expected graph corresponding to Figure 5.

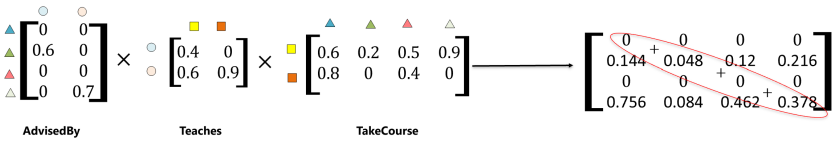

Figure 7: The matrix multiplication sequence for our example conjunction and Figure 6 graph.

diagonal is set to zero when the two endpoints must differ, as for the A–P–A metapath in Section 4.3), so that only valid groundings are counted. For longer chains, groundings that revisit an intermediate node are not removed by an endpoint mask and require additional corrections.

1. *For an input graph G, the (u, v)-th entry of T_r counts the number of groundings of ϕ_r in G where $U_1 = u$ and $V_L = v$. Therefore $n_{\phi_r}(G) = \sum(T_r)$.*

2. *For an expected graph $\tilde{G}_z$, under the conditions of Proposition 2 and when no grounding constrains the same graph-state variable twice, the (u, v)-th entry of T_r computed from the soft matrices counts the expected number of groundings of ϕ_r where $U_1 = u$ and $V_L = v$. Therefore $n_{\phi_r}(\tilde{G}_z) = \sum(T_r)$.*

Proof. Part 1, by induction on L. For $L = 1$, $[T_r]_{uv} = [D_{a_0}]_{uu}[A^{(R_1)}]_{uv}[D_{a_1}]_{vv}$, which is 1 exactly when u has state a_0, v has state a_1, and an R_1-edge connects u to v – precisely the single grounding admitted by ϕ_r for that endpoint pair, matching equation 2.

For the inductive step, assume the claim holds for a length-L prefix chain, so $[T_r^{(L)}]_{uw}$ counts groundings of the prefix from u to intermediate node w. Extending by one relation factor gives $T_r^{(L+1)} = T_r^{(L)} A^{(R_{L+1})} D_{a_{L+1}}$, so $[T_r^{(L+1)}]_{uv} = \sum_w [T_r^{(L)}]_{uw}[A^{(R_{L+1})}]_{wv}[D_{a_{L+1}}]_{vv}$. Each term in this sum counts the groundings of the length-L prefix ending at w that can be extended by an R_{L+1}-edge from w to v with v in state a_{L+1}; summing over w counts every length-$(L + 1)$ grounding from u to v exactly once, since the intermediate node of a grounding is uniquely determined once u, v and the grounding itself are fixed. This establishes the claim for $L + 1$.

Part 2. The same expansion writes each entry of the soft product as a sum over groundings of the product of the soft entries along the grounding. When no graph-state variable is constrained twice, this product is $\prod_{i=1}^{s} p(\tilde{G} \models \alpha_i(\omega))$ so the entry equals equation 9, which is the conditional expectation by Proposition 2. □

Computational Complexity The matrix product equation 13 translates a logical formula into a sequence of matrix multiplications in time linear in the length of the formula. The number of binary literals is small enough to be treated as a constant $L \leq 5$. The bottleneck is scaling an L-fold adjacency matrix product to large graphs, especially the dense expected adjacency matrices. The computational scope is supported relational chains and compatible endpoint constraints, not arbitrary first-order formulas. With n nodes and L relation factors, straightforward dense computation costs $O(Ln^3)$ time and $O(n^2)$ output storage per commuting matrix. Matrix computation avoids enumerating each path assignment explicitly, but many retained states can still be costly.

C Datasets

The six datasets span regular lattices, branching structures, and attributed molecular and protein graphs (which are heterogeneous):

- **GRID** contains lattice graphs with regular connectivity and long-range distance structure.
- **Triangular grid** adds triangular connectivity, testing local closure alongside lattice organization.
- **LOBSTER** contains tree-like graphs formed from a backbone and short branches.
- **PTC** contains molecular graphs, represented here by connectivity and a 19-state categorical node label. The split contains 240 training, 34 validation, and 70 test graphs; this representation has no edge-attribute target.
- **PROTEINS** contains protein graphs with a three-state categorical node label. Following graph-size filtering, evaluation uses 209 nonempty reference graphs.
- **QM9** contains molecular graphs, with node attributes encoding atom type and hydrogen-count category. Evaluation uses 512 generated and 512 held-out reference graphs per seed.

The three synthetic datasets have no native node or edge attributes. PTC and PROTEINS are available through TUDataset (Morris et al., 2020); QM9 originates from Ramakrishnan et al. (2014).

D Hyperparameters and Resources

Hyperparameters We used the ADAM optimizer for training, with default hyperparameter values. For *graph generation*, GraphVAE and GraphVAE+RG share the same backbone and training settings on each dataset; the weights of the adjacency and KL terms of the GraphVAE loss were set to 1, similar to (Simonovsky & Komodakis, 2018). On datasets with node attributes (PTC, PROTEINS, QM9) the attribute reconstruction terms were also weighted by 1; the synthetic datasets (GRID, triangular grid, LOBSTER) are topology-only, so no attribute terms were used. The rule guidance weight λ was set empirically per dataset ($\lambda = 0.1$ on all datasets except PROTEINS, where $\lambda = 0.03$). The rule term uses the commuting matrix with the Gaussian distance, computed over the rules retained by data-driven pruning of the FactorBase conditional-probability tables. Models were trained for 20,000 epochs on the synthetic datasets, PTC and PROTEINS and 250 epochs on QM9, and, except for the fixed-epoch QM9 checkpoint, the checkpoint with the best validation MMD was used for evaluation. Table 3 lists the rule guidance weight λ in equation 8, the latent dimension of z, and the numbers of generated and reference graphs used for evaluation.

Computing Resources The experiments were conducted on a combination of lab workstations and a shared GPU cluster. Workstation experiments used NVIDIA GeForce GTX 1080 Ti (11GB VRAM), NVIDIA TITAN X (12GB VRAM), and NVIDIA TITAN RTX (24GB VRAM) GPUs, on machines with 32–64GB RAM and 12–20 CPU threads. Cluster experiments used NVIDIA A40 GPUs with 48GB VRAM, 64GB RAM, and 8 CPU threads per job.

Table 3: Per-dataset hyperparameters and evaluation collections. #Graphs is both the number of generated graphs per seed and the number of reference graphs.

Dataset	λ	Latent dim.	#Graphs	Reference selection
GRID	0.1	128	20	Full test collection
Triangular grid	0.1	128	20	Full test collection
LOBSTER	0.1	128	20	Full test collection
PTC	0.1	128	70	Full test collection
PROTEINS	0.03	128	209	All usable test graphs after preprocessing
QM9	0.1	128	512	First 512 graphs of the fixed test order

E GNN-Based Graph Generation Metrics

The evaluator is a graph isomorphism network (GIN; Xu et al., 2019) with randomly initialized, frozen parameters: it aggregates neighbour representations over several message-passing layers and sums them into a single graph embedding. The same frozen network embeds both collections, and coordinate standardization fitted on reference embeddings is applied to both. For random-GIN metrics, ten evaluator initializations are averaged within each generator seed before computing the across-seed mean and sample standard deviation.

For one evaluator, let $r_1, \ldots, r_n$ be standardized reference embeddings and $y_1, \ldots, y_m$ the generated embeddings. Let $\rho_R(r_i)$ be the distance to the kth nearest other reference embedding, and define $\rho_Y(y_j)$ analogously within the generated collection. With $k = 5$, distributional precision and recall are

$$P = \frac{1}{m}\sum_{j=1}^{m} \mathbf{1}\{\exists i : \|y_j - r_i\|_2 \le \rho_R(r_i)\}, \qquad R = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\exists j : \|r_i - y_j\|_2 \le \rho_Y(y_j)\}. \quad (14)$$

Precision measures generated samples lying within reference neighbourhoods; recall measures reference samples covered by generated neighbourhoods. The harmonic mean

$$F_{1,\mathrm{PR}} = \frac{2PR}{P+R} \quad (15)$$

is defined as zero when both terms vanish. This is a distributional fidelity-and-coverage measure, not classification accuracy. The neighbourhood construction follows Kynkäänniemi et al. (2019).

F Rule Learning

In principle, any Markov Logic Network structure learning method can be used with our moment-matching training objective. We deployed the FactorBase system (Qian & Schulte, 2015) for several reasons.

1. Interpretability: The first phase of FactorBase uses Bayesian network structure learning method to discover a *causal graph* that provides a perspicuous visual representation of all associations learned from the data.
2. Scalability: FactorBase scales to larger and more complex datasets than other MLN structure learning methods, on the order of 100K nodes.
3. Quality: Research has shown that the formulas discovered by FactorBase support high quality inferences in relational prediction tasks.

FactorBase has three main phases.

1. Causal Graph Discovery.
2. Rule Extraction from the learned causal graph.

3. Moralization: convert learned rules to conjunctions.

Both rule extraction and moralization are computationally straightforward linear-time operations. Moralization is a method for converting learned rules to conjunctions. Previous research in statistical-relational learning (Kazemi et al., 2014; Khosravi et al., 2012) has shown that moralization is a strong method for converting learned rules to useful features for generative graph models. Note that therefore *our matrix multiplication algorithm can be applied to any set of learned rules after moralization*. This means that our algorithm for rule enhancement can be applied to any set of learned rules, not only those obtained from MLN structure learning. We next explain the phases of FactorBase formula discovery and illustrate the results on our datasets.

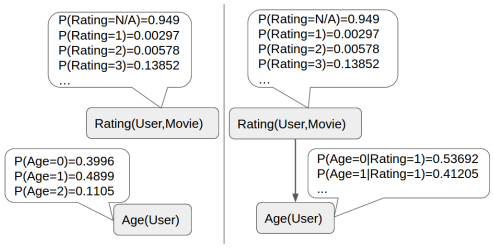

Figure 8: Example First-Order Causal Graphs: left = B_1 with graph G_1, right = B_1^+ with graph G_1^+.

Causal Graph Discovery A **causal graph (CG) structure** is a directed acyclic graph G (DAG) whose nodes comprise a set of random variables (Pearl, 2000). The interpretation is that the parents of a node represent its direct causes (Pearl, 2000). A **causal Bayesian network** B is a structure G together with a set of parameter values of the form $P(child_value|parent_values)$, which specify the distribution of a child node given an assignment of values to its parent node. A first-order causal graph (Wang et al., 2008; Kimmig et al., 2014) (FOCG) is a CG whose nodes are first-order terms. The CG parameters specify the distribution of a child node given an assignment of values to its parent node. Figure 8 shows two parametrized FOCGs. The right FOCG connects the rating of a movie to the age of the rater. In the left FOCG, ratings are independent of ages. The process of structure learning searches for statistically significant connections between first-order terms to introduce or remove edges (Schulte & Qian, 2018; Schulte & Gholami, 2017).

We show the causal graphs learned by FactorBase for two real-world datasets, QM9 (Figure 9) and PTC (Figure 10).

Rule Extraction from Causal Graphs A **rule** is of the form $\boldsymbol{B} \rightarrow H$ where the body $\boldsymbol{B}$ is an FO conjunction and the head H is a single FO literal. An example rule is

$$Age(User) = 1 \rightarrow Rating(User, Movie) = 4\,.$$

This rule expresses the knowledge that the age level of a user influences their ratings.

A single rule $\boldsymbol{B} \rightarrow H$ is defined by a combination $parent_values \rightarrow child_value$, for each child node value and possible combination of parents values (Khosravi et al., 2012). For example, the causal graph of Figure 8(right) defines the rule

$$Rating(User, Movie) = 3 \rightarrow Age(User) = 1.$$

Given 6 possible rating values and 5 age levels, the link $Rating(User, Movie) \rightarrow Age(User)$ defines $6 \times 5 = 30$ rules.

Rule Pruning As the FactorBase system can output many rules, we use

$$2n(\boldsymbol{B}, H) \ln(n(H|\boldsymbol{B})/n(H)) - \ln n(H)$$

as a *rule quality metric* for pruning. The conditional count is given by $n(H|\boldsymbol{B}) = n(H, \boldsymbol{B})/n(\boldsymbol{B})$. This metric computes the increase in the log-probability of the head given the body, relative to the prior probability of the head, together with a BIC-type correction for sample size (Schulte & Gholami, 2017). The quality metric can be interpreted as a positive local BIC model selection score. We keep all rules whose quality metric is above 0.

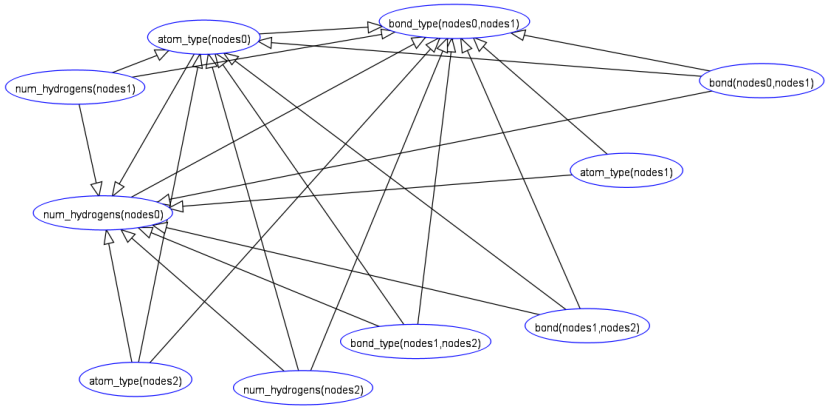

Figure 9: The causal graph for the QM9 dataset learned by FactorBase. The graph shows a complex set of dependencies between the types of linked atoms and the type of their bond. The number of hydrogen atoms plays an important role in the learned rules.

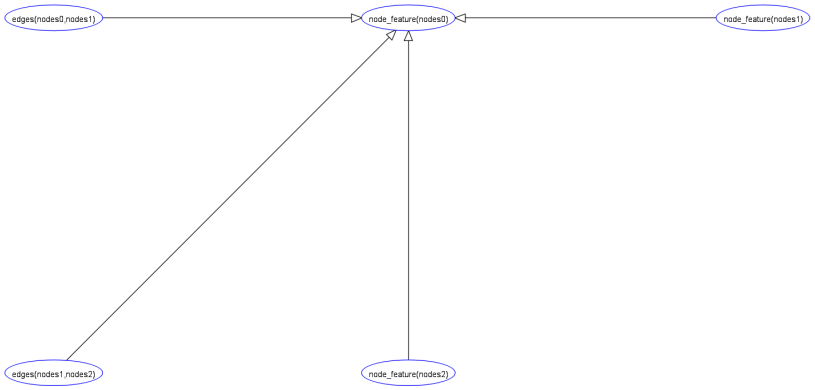

Figure 10: The causal graph for the PTC dataset learned by FactorBase. The graphs in this dataset do not have many features so the FactorBase rules capture mainly basic correlations between the node features linked by metapaths.

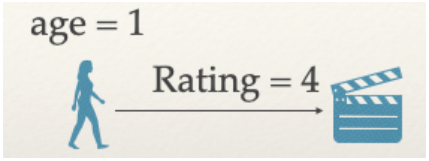

Figure 11: The FO conjunction $Age(User) = 1, Rating(User, Movie) = 4$ can be viewed as a two-node metapath with feature annotations. The conjunction is obtained from the rule $Age(User) = 1 \rightarrow Rating(User, Movie) = 4$ by moralization.

Moralization: From rules to conjunctions Most rule learners are based on *discriminative* learning, building on classification techniques to search for bodies that predict the head. A question researched in SRL is how to convert a set of rules to a set of graph features/statistics that support *generative* graph modelling. The recommended answer is *moralization*: convert each rule to a conjunction $\phi = (\boldsymbol{B}, H)$ (Domingos & Richardson, 2007, 12.5.3), (Kazemi et al., 2014). Figure 11 illustrates how moralization converts a rule to a conjunction, and how a conjunction represents a metapath.