ZERO-SHOT LOGICAL REASONING FOR ARBITRARILY STRUCTURED QUERIES ON KNOWLEDGE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world complex logical query answering (CLQA) requires reasoning over arbitrarily structured existential first-order (EFO) queries on unseen knowledge graphs (KGs). However, existing methods are often restricted to tree-shaped queries or fixed KGs. Moreover, they often rely on link predictors designed to predict a missing entity for a single triplet, making them suboptimal for queries involving multiple atoms that form complex structures. We propose VASIL, the first foundation model for CLQA designed to directly answer EFO queries with arbitrary structures in a zero-shot manner. Given an EFO query on a KG, VASIL autoregressively infers probability distributions of the variables while jointly predicting mutually dependent variables. At each step, it labels the entities and relations in the KG based on the atoms involving the variables being resolved and performs message passing over the labeled KG. The resulting representations are used to resolve the variables, yielding predictions conditioned on both the KG and the query. To train VASIL across diverse query structures, we construct queries by merging multiple random walks, a procedure that theoretically covers all conjunctive EFO query structures. Because variables are predicted based solely on the structures of the query and the labeled KG, VASIL can perform CLQA on KGs containing unseen entities and relations. Experiments on 37 benchmarks show that VASIL outperforms baselines in answering queries requiring reasoning beyond simple link prediction, including non-tree-shaped or multi-free-variable queries.

1 INTRODUCTION

Knowledge graphs (KGs) organize human knowledge as a set of triplets, each of which is composed of a head entity, a relation, and a tail entity. Complex logical query answering (CLQA) is the task of answering Existential First-Order (EFO) queries over KGs. Figure 1 shows an example query, $?x\,.\,\exists y_1 y_2 : Major(y_1,Physics) \wedge Spouse(y_1,y_2) \wedge Win(y_1,x) \wedge Win(y_2,x)$, which asks: Which award was won by a person who majored in physics and their spouse? Since KGs are often incomplete, CLQA requires inferring missing knowledge rather than simply traversing a KG.

Early CLQA approaches (Hamilton et al., 2018; Ren et al., 2020; Ren & Leskovec, 2020) directly learn KG-specific query embeddings by training on a predefined set of query structures, making them transductive and limiting their applicability to those predefined query structures. Query graphs have been proposed to support a broader range of query structures under a transductive setting, including non-tree queries (Wang et al., 2023; Yin et al., 2024) and queries with multiple free variables (Fei et al., 2026). While other methods utilize inductive link predictors (Galkin et al., 2022; Cao et al., 2025) or foundation models for link prediction to handle unseen entities or relations (Galkin et al., 2024b), these methods are confined to tree-shaped queries with a single free variable. Consequently, as shown in Table 1, no existing CLQA method can answer queries with non-tree structures or multiple free variables over KGs containing unseen entities and relations. Moreover, many existing approaches rely on link predictors as their basic reasoning primitive, decomposing a complex query into multiple link prediction problems. Such decomposition scores individual atoms rather than directly modeling joint dependencies among multiple query variables, so it requires a search over a combinatorial assignment space to reason jointly over mutually dependent variables.

In this paper, we propose VASIL (autoregressi**V**e query **A**n**S**wering via query-cond**I**tioned compositiona**L** reasoning), the first foundation model trained directly for CLQA and capable of answering

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.

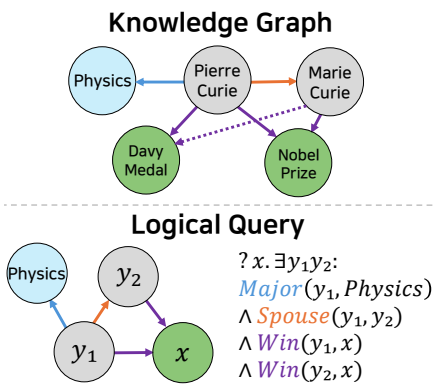

Figure 1: An example of CLQA over a KG G. y_1 and y_2 are existential variables and x is the free variable. The dashed arrow indicates a missing link in G. The target answer x can be *Davy Medal* or *Nobel Prize*.

Table 1: Comparison between VASIL and existing CLQA methods. "Mult." refers to queries with multiple free variables. "Ent." and "Rel." refer to entities and relations unseen during training, respectively.

	Query Structure			Inductive	
	Tree	Non-Tree	Mult.	Ent.	Rel.
BetaE (2020)	✓				
CQD-Beam (2021)	✓				
QTO (2023)	✓				
NPQE (2022)	✓			✓	
SAQE (2025)	✓			✓	
UltraQuery (2024b)	✓			✓	✓
LMPNN (2023)	✓	✓			
FIT (2024)	✓	✓			
NS3 (2026)	✓	✓	✓		
VASIL	✓	✓	✓	✓	✓

EFO queries with arbitrary structures over KGs containing entities and relations unseen during training. To account for variable dependencies in arbitrarily structured queries, VASIL resolves variables autoregressively while jointly predicting mutually dependent variables. At each step, our query-conditioned compositional reasoner labels entities and relations in a given KG using the components of relevant atoms. It represents entities and relations as compositions of triplets in the labeled KG, and variables as compositions of atoms in the query graph. These representations are used to predict query-conditioned distributions of the variables being resolved. Rather than training on a fixed set of predefined query structures, we introduce a dynamic query synthesizer that generates queries with diverse structures by merging multiple random walks. Theoretically, the proposed synthesizer can generate any positive conjunctive EFO query with a non-empty answer set on an input KG. Since VASIL learns to answer complex logical queries solely from the query and labeled KG structures, it can be seamlessly applied to KGs with unseen entities and relations. Our main contributions are:

- We propose VASIL, the first CLQA foundation model that can answer EFO queries with arbitrary structures over KGs containing entities and relations unseen during training.[1]
- By modeling variable distributions conditioned solely on the structures of a query and a labeled KG, VASIL can answer queries on unseen KGs without any KG-specific training.
- We introduce a dynamic query synthesizer that provably generates any possible positive existential conjunctive query, allowing VASIL to be trained on diverse query structures.
- Experiments on 37 datasets show that VASIL achieves the best performance on queries requiring reasoning beyond simple link prediction, without any dataset-specific training.

2 Related Work

Query Embedding for CLQA Query embedding methods compute query representations from learned entity embeddings through relation-specific transformations and logical operations, representing queries as geometric objects (Hamilton et al., 2018; Ren et al., 2020; Zhang et al., 2021; Choudhary et al., 2021) or probability distributions (Ren & Leskovec, 2020; Yang et al., 2022). Because these methods learn parameters tied to specific entities and relations observed during training, they fail to generalize to KGs with unseen entities or relations. Moreover, they target tree-shaped queries with a single free variable and do not directly support more general query structures involving multi-edges, cycles, or multiple free variables.

Query Answering via Iterative Link Prediction Some methods answer complex queries by decomposing a query into multiple one-hop link prediction tasks and iteratively applying a link

[1]Following one of the three options specified in the FAQ section of the author guidelines, we will share our code with the AC and reviewers during the rebuttal period, keeping it visible only to them.

predictor (Bai et al., 2023; Zhu et al., 2022; Galkin et al., 2022; Cao et al., 2025). For example, CQD (Arakelyan et al., 2021) uses a link predictor to score individual atoms and aggregates these scores using fuzzy operators. To answer queries over KGs with unseen entities and relations, UltraQuery (Galkin et al., 2024b) iteratively applies Ultra (Galkin et al., 2024a), a foundation model for link prediction. However, a link predictor can only make predictions for individual links and cannot jointly model atoms that share variables. This limits the ability of these methods to directly model dependencies among query variables when answering complex queries with non-tree structures.

CLQA using Query Graphs To handle non-tree structures or multiple free variables, some methods represent queries as graphs and use these graphs to infer answers (Wang et al., 2023; Zhang et al., 2024). For example, FIT (Yin et al., 2024) performs fuzzy inference using query graphs to support structures such as multi-edges and cycles, and NS3 (Fei et al., 2026) handles multiple free variables by selecting top-ranked entity candidates for each variable and then ranking their combinations. While these methods can handle complex query structures, they are developed for the transductive setting with fixed sets of entities and relations across training and inference.

Comparison with AnyCQ While AnyCQ (Olejniczak et al., 2025) aims to answer any conjunctive query over any KG, it differs fundamentally from VASIL in its inference mechanism and task formulation. First, AnyCQ learns a policy to search for satisfying variable assignments, but evaluating whether those assignments satisfy the query requires atom-level predictions from an external link predictor. In contrast, VASIL directly learns to predict variable distributions conditioned on the query and KG, requiring neither an external link predictor nor explicit assignment search. Second, AnyCQ targets Boolean query evaluation, whereas VASIL targets CLQA, a ranking task over candidate entities. Despite these differences, we adapt VASIL to the query answer classification task proposed by AnyCQ to provide a comparison and show that VASIL outperforms AnyCQ (§ 5.3).

3 Problem Definition

A KG $G = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ consists of a set of entities $\mathcal{V}$, a set of relations $\mathcal{R}$, and a set of triplets $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$. A model is trained on a training KG $G_{\text{tr}} = (\mathcal{V}_{\text{tr}}, \mathcal{R}_{\text{tr}}, \mathcal{E}_{\text{tr}})$ and answers queries at test time on an inference KG $G_{\text{inf}} = (\mathcal{V}_{\text{inf}}, \mathcal{R}_{\text{inf}}, \mathcal{E}_{\text{inf}})$. The setting is *transductive* if $G_{\text{inf}} = G_{\text{tr}}$, *semi-inductive* if $\mathcal{V}_{\text{inf}} \setminus \mathcal{V}_{\text{tr}} \neq \emptyset$ and $\mathcal{R}_{\text{tr}} = \mathcal{R}_{\text{inf}}$, and *fully-inductive* if $\mathcal{V}_{\text{tr}} \cap \mathcal{V}_{\text{inf}} = \emptyset$ and $\mathcal{R}_{\text{tr}} \cap \mathcal{R}_{\text{inf}} = \emptyset$.

A *term* is either a constant $c \in \mathcal{V}$ or a variable, and an *assignment* ν maps each term to an entity in $\mathcal{V}$, with constants mapped identically, i.e., $\nu(c) = c$. An *atom* $r(o, o')$ with $r \in \mathcal{R}$ is satisfied under ν if $(\nu(o), r, \nu(o')) \in \mathcal{E}$. A *literal* is an atom or its negation. An Existential First-Order (EFO) query is a first-order formula constructed from literals using conjunction, disjunction, and existential quantification. An EFO query q with free variables $x_1, \ldots, x_k$ and existential variables $y_1, \ldots, y_m$ can be expressed in Disjunctive Normal Form (DNF) as:

$$q(x_1, \ldots, x_k) = \exists y_1 \cdots \exists y_m . \bigvee_i \bigwedge_j \lambda_{ij}, \tag{1}$$

where each λ_{ij} is a literal. Its answer set $A(q; G)$ is the set of tuples $(e_1, \ldots, e_k) \in \mathcal{V}^k$ for which $q(e_1, \ldots, e_k)$ holds on G under some assignment of the existential variables.

4 Methodology

Figure 2 illustrates the overall framework of VASIL. During training, a dynamic query synthesizer merges random walks sampled from a training KG, generating queries with diverse structures without predefined query templates. During inference, queries over an inference KG are provided (§ 4.1). Given an EFO query and a KG, VASIL autoregressively resolves the query variables using representations conditioned on both the query and the input KG (§ 4.2). At each step, VASIL predicts a set of variables using a query-conditioned compositional reasoner, which initializes entities and relations from the query-derived labels and updates their representations through message passing over the labeled KG. These conditional representations are used to compute variable representations and predict a probability distribution over entities for each variable (§ 4.3). By learning to model the distributions of the variables conditioned on the query and KG structure, VASIL can be applied to KGs containing unseen entities and relations.

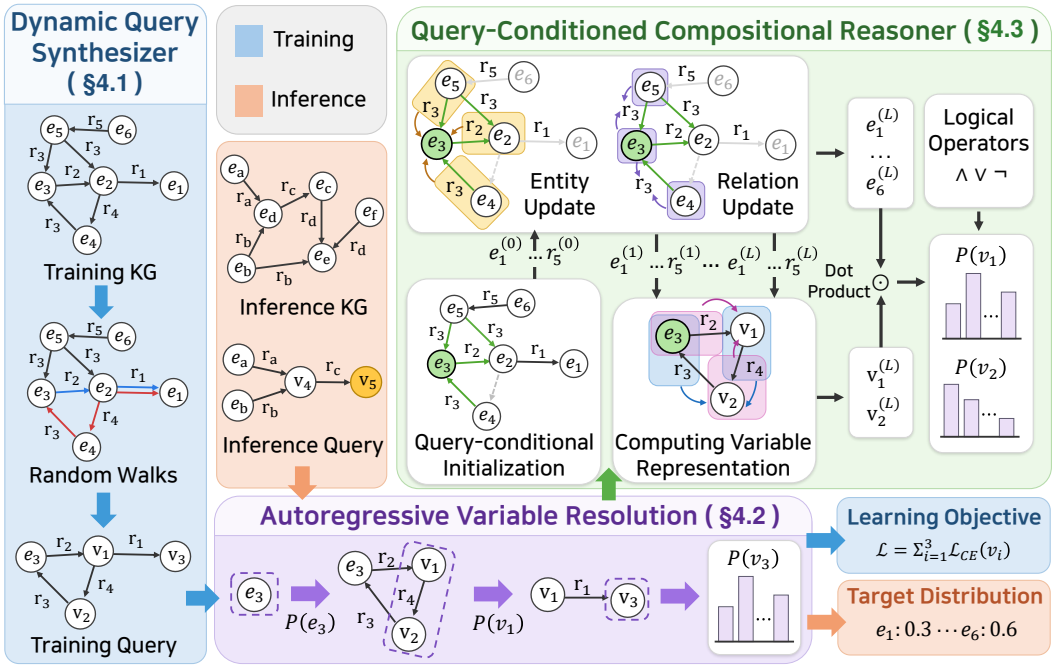

Figure 2: Overview of VASIL. During training, the Dynamic Query Synthesizer generates diverse queries, while a query is provided as input during inference. Variables are resolved autoregressively based on their dependencies. At each step, the Query-Conditioned Compositional Reasoner predicts probability distributions of the variables conditioned on the query and the KG structure.

4.1 Dynamic Query Synthesizer

To train VASIL on diverse query structures, we propose a dynamic query synthesizer that generates queries directly from a KG G by merging sampled random walks, without relying on predefined query templates. Each query is constructed from multiple random walks that start from a common seed entity. At each step of a walk, we select an incident triplet and move to the entity at its other end, regardless of the direction of the triplet. The direction and relation of each traversed triplet are preserved when constructing the query. After each step, the walk terminates with probability η, where η is a hyperparameter that controls the expected walk length.

For each occurrence of an entity e in the sampled walks, we independently map it to a query term by (1) retaining e as a constant, or (2) mapping it to a variable, where we either reuse an existing variable assigned to e (if available) or introduce a new variable for e. Thus, different occurrences of the same entity may be represented as a constant, a variable, or as distinct variables within the same query, while reusing an existing variable allows multiple atoms to share the same query variable. The mapped walks are then merged to form a query. For each variable, its associated entity provides an answer. Notably, every positive conjunctive EFO query with a non-empty answer set can be generated by this procedure, as formally stated in Proposition 4.1.

Proposition 4.1. *Given a KG G, the set of queries obtainable by the Dynamic Query Synthesizer is the class of positive conjunctive EFO queries that have a non-empty answer set on G.*

Intuitively, if a query q has an answer on G, there exists an assignment of its variables under which every atom corresponds to a triplet in G. Since there exist walks that collectively cover these triplets, q can be generated by the dynamic query synthesizer. In practice, we bound both the number and the length of the sampled walks. Appendix A provides the proof and discusses the coverage under these bounds, while Appendix B shows examples of queries generated by our dynamic query synthesizer.

Training and Inference During training, a training KG G_{tr} is given, and our dynamic query synthesizer generates diverse queries from G_{tr}. For each synthesized query, every variable is in turn treated as a free variable, and all valid assignments to that variable, including the associated entity,

are used as supervision targets. When predicting a free variable, previously predicted variables are labeled with their predicted distributions. Consequently, a single synthesized query provides supervision not only for its full structure but also for the substructures corresponding to its individual variables. During inference, VASIL answers a given query q on a KG G_{inf}, without using the dynamic query synthesizer.

4.2 Autoregressive Variable Resolution

To account for variable dependencies in queries with complex structures, VASIL decomposes an EFO query q over a KG $G = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ into a sequence of subproblems, where the goal of each subproblem is to jointly resolve a group of mutually dependent variables, and solves these subproblems autoregressively. Let $\mathcal{X}_q$ denote the set of free variables in q. We represent q as a graph, with terms as nodes and atoms as edges, and define the distance of a variable v to the free variables as $d_q(v) = \min_{u \in \mathcal{X}_q} \text{dist}_q(v, u)$, where $\text{dist}_q(v, u)$ is the shortest path distance between v and u in the query graph. The variables are resolved in decreasing order of d_q, such that variables farther from the free variables are resolved first, and their predicted distributions are used when resolving variables closer to the free variables. Consequently, the free variables are resolved in the final step.

Variables at different distances belong to different subproblems. For variables with the same distance, we regard those connected through atoms as mutually dependent and group each connected component into a subproblem. Different subproblems with the same distance are solved in parallel. For each subproblem, we consider only atoms whose variable terms are either previously resolved or belong to the current subproblem; atoms involving variables assigned to later subproblems are deferred to the subproblems in which those variables are resolved.

4.3 Query-Conditioned Compositional Reasoner

For each subproblem, the Query-Conditioned Compositional Reasoner (QCCR) predicts the variables in the subproblem by jointly considering all relevant atoms. QCCR labels G using atoms that connect current variables to constants or previously resolved variables, constructing a labeled KG G_a for each atom a. QCCR represents entities and relations as compositions of their incident triplets by aggregating messages in G_a. Using these atom-conditioned representations, it applies the same message-passing updates over the query graph, representing variables as compositions of their incident atoms. Since mutually dependent variables are connected through their incident atoms, the resulting variable representations reflect their dependencies. Finally, QCCR compares the variable and entity representations for each conditioning atom to obtain atom-conditioned predictions, which are combined across atoms into a query-conditioned distribution for each variable.

Atom-conditioned Labeling Consider an atom a that connects a current variable v to a term o through relation ρ, where o is a constant or a previously resolved variable. The term o defines a distribution π_a over entities, where $\pi_a(e) = \mathbb{1}[e = o]$ if o is a constant and $\pi_a(e)$ is the predicted probability of e for o if o is a previously resolved variable. Each entity $e \in \mathcal{V}$ receives the label $w_a(e) = \pi_a(e)/\max_{e' \in \mathcal{V}} \pi_a(e')$. Similarly, each relation $r \in \mathcal{R}$ receives the label $w_a(r) = \mathbb{1}[r = \rho]$. In G_a, $\mathbf{e}_a^{(l)}, \mathbf{r}_a^{(l)} \in \mathbb{R}^D$ denote the D-dimensional representations of an entity e and a relation r, respectively, at layer $l \in \{0, \ldots, L\}$. These representations are initialized as:

$$\mathbf{e}_a^{(0)} = \boldsymbol{\theta}_0 + w_a(e)\,(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_0)\,, \quad \mathbf{r}_a^{(0)} = \boldsymbol{\theta}'_0 + w_a(r)\,\left(\boldsymbol{\theta}'_1 - \boldsymbol{\theta}'_0\right) \tag{2}$$

where $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1 \in \mathbb{R}^D$ are learnable vectors for unlabeled and labeled entities, respectively, and $\boldsymbol{\theta}'_0, \boldsymbol{\theta}'_1 \in \mathbb{R}^D$ are learnable vectors for unlabeled and labeled relations, respectively. Since our initialization strategy relies only on query-derived labels and the KG-agnostic parameters, $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}'_0$, and $\boldsymbol{\theta}'_1$, it can be applied to any KG, including those with unseen entities and relations.

Compositional Representation Update Given the labeled KG G_a and initial representations $\mathbf{e}_a^{(0)}$ and $\mathbf{r}_a^{(0)}$, QCCR iteratively updates entity and relation representations by computing and aggregating messages from incident triplets. For each triplet $T = (h, r, t) \in \mathcal{E}$, h receives a message $\mathbf{m}_{\text{head}}$ computed from r and t, t receives a message $\mathbf{m}_{\text{tail}}$ computed from h and r, while r receives messages $\mathbf{m}_{h \to r}$ and $\mathbf{m}_{t \to r}$ from h and t, respectively:

$$\mathbf{m}_{\text{head}}^{(l)} = W_{\text{head}}[\mathbf{t}_a^{(l)}; \mathbf{r}_a^{(l)}], \mathbf{m}_{\text{tail}}^{(l)} = W_{\text{tail}}[\mathbf{h}_a^{(l)}; \mathbf{r}_a^{(l)}], \mathbf{m}_{h \to r}^{(l)} = W_{h \to r}\mathbf{h}_a^{(l)}, \mathbf{m}_{t \to r}^{(l)} = W_{t \to r}\mathbf{t}_a^{(l)}, \tag{3}$$

where $[;]$ denotes concatenation. The learnable projection matrices $W_{\text{head}}, W_{\text{tail}} \in \mathbb{R}^{D\times 2D}$ compute messages from a triplet to its head and tail entities, respectively, while $W_{h\to r}, W_{t\to r} \in \mathbb{R}^{D\times D}$ compute messages from head and tail entities to the relation, respectively. The entity representations are updated by aggregating the messages from their incident triplets using multi-head attention (Vaswani et al., 2017), followed by a feed-forward network. Relations are updated analogously by aggregating messages from the head and tail entities, using separate parameters. Since representations are computed using only the initialized representations, the KG structure, and parameters not tied to individual entities or relations, VASIL can be applied to any KG.

Variable Representation and Prediction Given the atom-conditioned entity and relation representations obtained from G_a, QCCR computes variable representations on the query graph using the same message-passing operations and parameters used to update entity representations. Each variable v in the current subproblem is initialized as $\mathbf{v}_a^{(0)} = \boldsymbol{\theta}_0$. Consider an atom b that connects a current variable to a term o, where o is a constant or a previously resolved variable. We compute the layer-l representation of o conditioned on a as $\mathbf{o}_a^{(l)} = \sum_{e\in\mathcal{V}} \pi_b(e)\mathbf{e}_a^{(l)}$, where π_b is the distribution over entities defined by o. The relation in each atom uses its corresponding layer-wise representation from the labeled KG. Using these representations, QCCR compositionally updates the variables in the current subproblem with messages from the atoms. After L layers, we obtain $\mathbf{v}_a^{(L)}$, the final representation of v conditioned on a. QCCR compares it with each entity representation $\mathbf{e}_a^{(L)}$ to obtain the atom-conditioned probability distribution $p_{v|a} = \mathrm{softmax}_{e\in\mathcal{V}}\Big(\Big\langle \mathbf{v}_a^{(L)}, \mathbf{e}_a^{(L)} \Big\rangle / \sqrt{D}\Big)$.

Logical Query Answering For each atom a used for conditioning, QCCR computes a probability distribution conditioned on a and normalizes it. If q contains the negative literal $\neg a$, the normalized prediction is adjusted with temperature τ and complemented. Let $\mathcal{A}_v^+$ and $\mathcal{A}_v^-$ denote the sets of conditioning atoms that appear without or with negation in q, respectively. These predictions are combined multiplicatively to obtain the final distribution for each variable v:

$$p_v(e) \;\propto\; \prod_{a\in\mathcal{A}_v^+} \bar{p}_{v|a}(e) \prod_{a'\in\mathcal{A}_v^-} \Big(1 - \bar{p}_{v|a'}(e)^{1/\tau}\Big)^{\beta}, \qquad \bar{p}_{v|a}(e) = \frac{p_{v|a}(e)}{\max_{e'\in\mathcal{V}} p_{v|a}(e')}, \tag{4}$$

where τ and β are hyperparameters that control the sharpness and contribution of the negative literals, respectively. Union is handled by converting the query into DNF, resolving each conjunctive clause independently, and taking the entity-wise maximum over their normalized predictions. For queries with multiple free variables, we obtain a factorized joint distribution by multiplying the predicted distributions of the free variables.

5 Experiments

We pre-train VASIL on FB15k-237 (Toutanova & Chen, 2015), CoDEx-M (Safavi & Koutra, 2020), NELL995 (Xiong et al., 2017), and WN18RR (Dettmers et al., 2018), following the pre-training setup of UltraQuery (Galkin et al., 2024b). During pre-training, our dynamic query synthesizer (§ 4.1) generates training queries from these KGs at each epoch. We evaluate a single pre-trained model on 37 benchmarks: 25 evaluate CLQA on tree-shaped queries, 8 evaluate CLQA on queries with diverse structures, and 4 evaluate query answer classification. Rather than following their benchmark-specific inference settings, such as transductive or inductive settings, we evaluate the same pre-trained VASIL model across all benchmarks without any benchmark-specific training.

We group CLQA methods into two categories: *per-KG trained*, which are trained separately for each benchmark, and *pre-trained*, which are evaluated in a zero-shot setting after pre-training. The latter category remains relatively underexplored, with UltraQuery as the only existing method. VASIL belongs to this category. Following the standard evaluation protocol (Ren & Leskovec, 2020), we use filtered metrics and evaluate only on non-trivial answers that cannot be obtained by simply traversing a KG. The best and second-best performances are **boldfaced** and underlined, respectively. "-" indicates that a method is not applicable or its results are not publicly available. Unless otherwise stated, baseline results are taken from either the methods' original papers or the corresponding benchmark papers. Dataset statistics and implementation details are in Appendix C and Appendix D, respectively. Appendix E provides a comparison of training and inference costs with UltraQuery, showing that VASIL requires substantially less training time with comparable inference time per query.

Table 2: Comparison of VASIL with baselines on benchmarks for tree-shaped queries across four inference settings. MRR (%) is reported for all settings except Semi-Inductive, where Hits@10 (%) is reported. avg_p and avg_n denote performance for queries without and with negation, respectively.

	Transductive		Transductive+H		Semi-Inductive		Fully-Inductive	
	avg_p	avg_n	avg_p	avg_n	avg_p	avg_n	avg_p	avg_n
Per-KG trained								
BetaE (2020)	22.8	5.7	–	–	–	–	–	–
CQD-Beam (2021)	25.5	–	16.1	2.4	–	–	–	–
CQD-Hybrid (2025)	–	–	15.8	2.4	–	–	–	–
ConE (2021)	25.3	6.2	11.2	3.4	–	–	–	–
NPQE (2022)	–	–	–	–	10.0	–	–	–
GNN-QE (2022)	27.9	10.0	14.4	4.5	46.9	**29.7**	–	–
QTO (2023)	33.2	14.2	16.1	4.5	–	–	–	–
LMPNN (2023)	27.4	7.9	–	–	–	–	–	–
CLMPT (2024)	28.5	7.4	16.3	2.8	–	–	–	–
FIT (2024)	30.4	–	–	–	–	–	–	–
SAQE (2025)	**34.3**	**15.1**	–	–	–	–	–	–
Pre-trained								
UltraQuery (2024b)	23.5	7.7	10.4	3.5	46.8	26.2	31.3	20.8
VASIL	31.8	12.9	**17.4**	**5.4**	**52.5**	29.1	**41.4**	**31.2**

5.1 Evaluation on Tree-Shaped Queries

We evaluate VASIL against 18 baselines on 25 benchmarks with tree-shaped queries: two Transductive (Ren & Leskovec, 2020), three Transductive+H (Gregucci et al., 2025), nine Semi-Inductive (Galkin et al., 2022), and 11 Fully-Inductive (Galkin et al., 2024b) benchmarks. These categories reflect the original inference settings followed by per-KG-trained methods. Across all categories, we evaluate the same pre-trained VASIL without benchmark-specific training. Because most queries in Transductive benchmarks can be reduced to simple link prediction, Transductive+H was proposed to reduce this bias and place greater emphasis on queries that require reasoning over multiple missing links (Gregucci et al., 2025). Following standard practice (Galkin et al., 2024b), we report MRR (%) for Transductive, Transductive+H, and Fully-Inductive benchmarks, and Hits@10 (%) for Semi-Inductive benchmarks. Table 2 presents representative results for selected strong baselines. Detailed results by query type for all baselines are provided in Tables 23–30 of Appendix F.

Table 2 shows that a single pre-trained VASIL outperforms all baselines, including per-KG-trained methods, on Transductive+H, Semi-Inductive, and Fully-Inductive benchmarks, while remaining competitive on Transductive benchmarks. Notably, VASIL achieves these results with a single pre-trained model. It also consistently outperforms UltraQuery, the only pre-trained baseline, across all four benchmark groups. The leading performance of VASIL on Transductive+H further supports its effectiveness on queries requiring reasoning beyond simple link prediction.

5.2 Evaluation on Diverse Query Structures

We evaluate VASIL against 15 baselines on eight benchmarks originally designed for transductive CLQA with non-tree query structures or multiple free variables. VASIL is evaluated in a zero-shot manner on all eight benchmarks, without benchmark-specific training. The four DAG benchmarks (He et al., 2025) evaluate queries with shared intermediate variables and comprise two easier benchmarks (DAG-Easy) and two harder benchmarks (DAG-Hard). Two EFO_k-CQA (Yin et al., 2025) benchmarks cover queries with one or two free variables, including multi-edge and cyclic structures, while two EFO_3-Small (Fei et al., 2026) benchmarks extend evaluation to queries with up to three free variables. Following standard practice (He et al., 2025; Yin et al., 2025; Fei et al., 2026), we report Hits@10 (%) for queries with two or more free variables and MRR (%) otherwise. We run and evaluate UltraQuery, the only pre-trained baseline, as its results are unavailable. Since

Table 3: Comparison of VASIL with baselines on benchmarks for diverse query structures in the transductive setting. k denotes the number of free variables. We report Hits@10 (%) for $k = 2$ and $k = 3$, and MRR (%) otherwise. ‡ We use path decomposition to adapt UltraQuery to these datasets.

	DAG-Easy		DAG-Hard		EFO_3-Small		EFO_k-CQA	
	avg_p	avg_n	avg_p	avg_n	k=2	k=3	k=1	k=2
Per-KG trained								
BetaE (2020)	12.0	15.9	8.3	16.3	9.9	6.8	27.2	11.7
CQD (2021)	13.4	–	7.1	–	9.8	9.1	30.0	15.8
ConE (2021)	16.5	16.3	11.4	17.1	11.6	7.3	29.9	12.1
Query2Box (DAGE) (2025)	<u>25.5</u>	–	14.6	–	–	–	–	–
BetaE (DAGE) (2025)	24.5	<u>20.7</u>	15.5	<u>19.9</u>	–	–	–	–
ConE (DAGE) (2025)	24.4	20.0	<u>17.9</u>	19.7	–	–	–	–
FIT (2024)	–	–	–	–	12.2	10.8	–	–
NS3(M) (2026)	–	–	–	–	12.8	13.2	–	–
NS3(J) (2026)	–	–	–	–	**26.8**	<u>18.8</u>	–	–
Pre-trained								
UltraQuery‡ (2024b)	23.2	18.5	17.7	16.5	18.1	13.0	<u>31.4</u>	<u>18.5</u>
VASIL	**30.4**	**26.9**	**22.3**	**27.5**	<u>26.1</u>	**20.9**	**39.0**	**28.3**

UltraQuery is restricted to tree-shaped queries, we decompose each query into all possible simple paths so that every atom is included in at least one path. Table 3 compares VASIL and selected competitive baselines. The comparison with all 15 baselines, more detailed results including breakdowns by query type, and details on running UltraQuery are available in Tables 31–35 of Appendix G.

Table 3 shows that VASIL outperforms all baselines, including per-KG-trained methods and the adapted UltraQuery, across all benchmarks. The only exception is EFO_3-Small with two free variables, where it remains competitive with NS3(J), the best-performing baseline. These results show that a single pre-trained VASIL can effectively answer queries with diverse structures, including those with shared intermediate variables, cyclic structures, and multiple free variables.

5.3 Evaluation on Query Answer Classification

Although VASIL is designed for ranking-based CLQA, we additionally evaluate it on Query Answer Classification (QAC). Given a query q and a candidate assignment ν to its free variables, the goal of QAC is to determine whether q is satisfied under ν. We evaluate VASIL against six baselines on four benchmarks (Olejniczak et al., 2025): two Small benchmarks containing tree-shaped queries and two Large benchmarks containing k-hub queries. To obtain binary predictions from the entity distributions produced by VASIL, we select a threshold using the validation set of each dataset. A candidate entity is classified as TRUE if its probability exceeds this threshold and FALSE otherwise. Since results for UltraQuery are unavailable for the Large benchmarks, we run and evaluate it using path decomposition and a threshold selected on each dataset's validation set. We follow the evaluation protocol (Olejniczak et al., 2025) and report F1. Table 4 shows that VASIL achieves the best performance on QAC. Detailed results are provided in Appendix H.

5.4 Ablation Studies and Qualitative Analysis

Ablation Studies Table 5 reports ablation results of VASIL on NELL-EFO_3, which is an EFO_3-Small benchmark with diverse query structures and multiple free variables, and Award, which is a Fully-Inductive benchmark with tree-shaped queries. (i) replaces the diverse training queries generated by our dynamic query synthesizer (§ 4.1) with simple 1-hop queries. This substantially reduces performance, highlighting the importance of complex query supervision for handling diverse structures and multiple free variables. (ii) replaces autoregressive variable resolution (§ 4.2) with simultaneous resolution of all variables, removing the sequential conditioning on earlier predictions. The substantial performance degradation shows the benefit of using previously resolved variables to

Table 4: Average F1-scores on the QAC task. ‡ We use path decomposition to adapt UltraQuery to Large benchmark.

	Small	Large
Per-KG trained		
QTO (2023)	69.3	–
FIT (2024)	69.6	–
GNN-QE (2022)	74.9	–
AnyCQ (2025)	<u>76.3</u>	53.7
Pre-trained		
UltraQuery (2024b)	73.2	<u>59.4</u>‡
VASIL	**77.4**	**69.4**

Table 5: Ablation studies of VASIL on NELL-EFO$_3$ and Award datasets. We report Hits@10 (%) for NELL-EFO$_3$ and MRR (%) for Award.

		NELL-EFO$_3$		Award	
		k=2	k=3	avg$_p$	avg$_n$
(i)	1-hop queries only	14.8	11.7	28.1	22.1
(ii)	w/o autoregression	22.3	16.6	25.0	9.4
(iii)	w/o labeling	17.4	11.7	23.8	16.6
(iv)	hard top-1 labels	31.4	26.1	30.7	21.3
(v)	mean aggregation	28.2	21.7	29.7	20.4
VASIL		**31.9**	**27.2**	**32.6**	**24.5**

Table 6: Top-three predictions of VASIL for a conjunctive query with two atoms and its two relaxed variants. Ground-truth answers for each query are **boldfaced**.

Query	Top-3 predictions		
$?x$: *received*$(x,$ Golden Bear$) \wedge$ *winner*(L. Delluc Prize$, x)$	1. **P. Chéreau**	2. **J. Godard**	3. **C.Gavras**
$?x.\exists y_1$: *received*$(x,$ Golden Bear$) \wedge$ *winner*(y_1, x)	1. **H. Miyazaki**	2. **M. Forman**	3. **I. Bergman**
$?x$: *received*$(x,$ Golden Bear$)$	1. **Grbavica**	2. **P. Chéreau**	3. **D. Hare**

guide subsequent predictions. (iii) removes atom-conditioned labels by setting $w_a(e) = w_a(r) = 0$ for all $e \in \mathcal{V}, r \in \mathcal{R}$ (Eq. 2). This makes the entity and relation representations computed from a given KG independent of the conditioning atom. The resulting performance reduction highlights the importance of incorporating query-specific information into KG representations. (iv) replaces the soft labels of previously predicted variables with hard top-1 labels (Eq. 2), retaining only the most likely entity in each variable's labeling. This consistently lowers performance, suggesting that propagating predicted probability distributions is helpful for subsequent predictions. Finally, (v) replaces multi-head attention aggregation with mean pooling (§ 4.3), assigning equal weight to all incoming messages. The resulting performance decline supports that adaptive message weighting is beneficial for computing atom-conditioned representations. Overall, these results show that each component of VASIL significantly contributes to its CLQA performance.

Qualitative Analysis Table 6 compares the top-three predictions of VASIL for a conjunctive query with two atoms and its two relaxed variants. Both variants retain the first atom: one replaces the constant in the second atom with an existential variable, and the other removes the second atom entirely. While the top-three predictions vary across the queries, all predictions are correct. These results suggest that VASIL combines information across atoms to rank candidate answers according to the query as a whole, even for queries that share an atom.

6 Conclusion and Future Work

We present VASIL, a foundation model for zero-shot CLQA over EFO queries with arbitrary structures on unseen KGs. We propose autoregressive variable resolution and a query-conditioned compositional reasoner to model variable dependencies and reason compositionally across atoms, alongside a dynamic query synthesizer for self-supervised CLQA training without predefined query templates. Experiments on 37 benchmarks show that a single pre-trained VASIL achieves state-of-the-art performance on answering queries requiring reasoning beyond simple link prediction, including non-tree-shaped queries and queries with multiple free variables. These findings highlight the benefits of designing and training models directly for CLQA, rather than relying on link predictors. We plan to integrate VASIL into diverse AI systems that require multi-hop relational reasoning and complex logical inference over KGs, such as knowledge-grounded QA and task-oriented dialogue systems. We also plan to develop a principled approach to modeling whether a query has any valid answers, enabling explicit predictions for queries with empty answer sets.

AI USE STATEMENT

In this work, we used generative AI tools for minor assistance with *implementing methods* and for *grammar corrections to improve writing*. We have not used generative AI tools for *assist with translation*, *interpret results*, *assist in the writing of proofs*, *help develop theoretical models or conceptual frameworks*, *formulate mathematical claims*, *provide critical ingredients for proving mathematical claims*, *propose or refine hypotheses*, *design or provide feedback on research methodology or experiments*, *support qualitative* and *thematic data analysis*. *Generating synthetic data sets*, *cleaning and reformatting dataset* are not applicable to this work. We have reviewed all AI-assisted work. We checked every AI-generated content ourselves. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies CLQA using established KG benchmarks. Potential ethical concerns include biases in the underlying KGs that may affect model predictions. In applications involving personal or sensitive information, inferring unobserved relationships may also raise privacy concerns. Such applications should include appropriate validation of predicted answers and safeguards for sensitive information.

REPRODUCIBILITY STATEMENT

We describe the proposed model and training procedure in Section 4. The proof of Proposition 4.1 is provided in Appendix A. The experimental setup and implementation details are provided in Section 5 and Appendix D, respectively. For code sharing, we have chosen one of the options specified in the FAQ section of the author guidelines and will share our code with the AC and reviewers during the rebuttal period.

REFERENCES

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Yushi Bai, Xin Lv, Juanzi Li, and Lei Hou. Answering complex logical queries on knowledge graphs via query computation tree optimization. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 1472–1491, 2023.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Yuan He, Xiaochun Cao, and Qingming Huang. SAQE: Complex logical query answering via semantic-aware representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 37(11):6651–6665, 2025.

Xuelu Chen, Ziniu Hu, and Yizhou Sun. Fuzzy logic based logical query answering on knowledge graphs. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pp. 3939–3948, 2022.

Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. Self-supervised hyperboloid representations from logical queries over knowledge graphs. In *Proceedings of the Web Conference 2021*, pp. 1373–1384, 2021.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D knowledge graph embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 1811–1818, 2018.

Weizhi Fei, Hang Yin, Zihao Wang, Shukai Zhao, Wei Zhang, and Yangqiu Song. Neural scalable symbolic search framework for complex logical queries with multiple free variables. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 1096–1107, 2026.

Michael Galkin, Zhaocheng Zhu, Hongyu Ren, and Jian Tang. Inductive logical query answering in knowledge graphs. In *Proceedings of the 36th Conference on Neural Information Processing Systems*, pp. 15230–15243, 2022.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *Proceedings of the 12th International Conference on Learning Representations*, 2024a.

Mikhail Galkin, Jincheng Zhou, Bruno Ribeiro, Jian Tang, and Zhaocheng Zhu. A foundation model for zero-shot logical query reasoning. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pp. 54137–54160, 2024b.

Cosimo Gregucci, Bo Xiong, Daniel Hernández, Lorenzo Loconte, Pasquale Minervini, Steffen Staab, and Antonio Vergari. Is complex query answering really complex? In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 20399–20428, 2025.

Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, pp. 2026–2037, 2018.

Yunjie He, Bo Xiong, Daniel Hernández, Yuqicheng Zhu, Evgeny Kharlamov, and Steffen Staab. DAGE: DAG query answering via relational combinator with logical constraints. In *Proceedings of the ACM on Web Conference 2025*, pp. 2514–2529, 2025.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs), 2016. URL `https://arxiv.org/abs/1606.08415`.

Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. Answering complex queries in knowledge graphs with bidirectional sequence encoders. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 4968–4977, 2021.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Francois Luus, Prithviraj Sen, Pavan Kapanipathi, Ryan Riegel, Ndivhuwo Makondo, Thabang Lebese, and Alexander Gray. Logic embeddings for complex query answering. *arXiv preprint arXiv:2103.00418*, 2021.

Krzysztof Olejniczak, Xingyue Huang, Mikhail Galkin, and Ismail Ilkan Ceylan. One model, any conjunctive query: Graph neural networks for answering queries over incomplete knowledge graphs. In *Proceedings of the 4th Learning on Graphs Conference*, 2025.

Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, pp. 19716–19726, 2020.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Tara Safavi and Danai Koutra. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 8328–8350, 2020.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 5998–6008, 2017.

Zihao Wang, Yangqiu Song, Ginny Wong, and Simon See. Logical message passing networks with one-hop inference on atomic formulas. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10524–10533, 2020.

Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 564–573, 2017.

Dong Yang, Peijun Qing, Yang Li, Haonan Lu, and Xiaodong Lin. GammaE: Gamma embeddings for logical queries on knowledge graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 745–760, 2022.

Hang Yin, Zihao Wang, and Yangqiu Song. Rethinking complex queries on knowledge graphs with neural link predictors. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Hang Yin, Zihao Wang, Weizhi Fei, and Yangqiu Song. EFOk-CQA: Towards knowledge graph complex query answering beyond set operation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 5876–5887, 2025.

Chongzhi Zhang, Zhiping Peng, Junhao Zheng, and Qianli Ma. Conditional logical message passing transformer for complex query answering. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4119–4130, 2024.

Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. ConE: Cone embeddings for multi-hop reasoning over knowledge graphs. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, pp. 19172–19183, 2021.

Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. Neural-symbolic models for logical queries on knowledge graphs. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 27454–27478, 2022.

A EXPRESSIVENESS OF THE DYNAMIC QUERY SYNTHESIZER

We prove the expressiveness result in Proposition 4.1 by allowing the number and length of random walks to be arbitrarily large but finite.

Notation and Background. Let $G = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ be a finite KG with $\mathcal{V} \neq \emptyset$. We consider positive conjunctive EFO queries of the form

$$q(\mathbf{x}) = \exists \mathbf{y}.\, C, \qquad C = \bigwedge_j r_j(o_j, o'_j),$$

where $\mathbf{x}$ is a tuple of distinct free variables. All variables range over $\mathcal{V}$, and terms are either variables or constants $c \in \mathcal{V}$. For every assignment ν, constants are interpreted by $\nu(c) = c$.

A conjunction C is *satisfiable over* G if there exists an assignment ν of its variables to entities in $\mathcal{V}$ such that $(\nu(o), r, \nu(o')) \in \mathcal{E}$ for every atom $r(o, o')$ in C.

We assume that two conjunctions are equivalent if they differ only by variable renaming or the presence of duplicate atoms. Since each connected component of a disconnected conjunction can be treated as a separate query, we focus on conjunctions with connected query graphs, where the query graph has one node for each term and one directed, relation-labeled edge for each atom, with edge directions ignored when determining connectivity.

Random-walk-based Synthesis. Fix integers $n, L \geq 1$ and a stopping probability $\eta \in (0, 1)$. During synthesis, we maintain an assignment ν for the generated variables and, for each $e \in \mathcal{V}$,

$$M(e) = \{o : o \text{ is a generated variable and } \nu(o) = e\}.$$

1. Choose a seed entity $e \in \mathcal{V}$ in G and represent it either by the constant e or by a fresh variable o with $\nu(o) = e$. If a fresh variable is introduced, add it to $M(e)$.
2. Run n walks from the seed entity, each of length at most L. After each step, terminate the walk with probability η, upon reaching the length bound, or if the current entity has no incident KG triplet. Otherwise, select an incident triplet and traverse it in either direction to an entity e'. Represent this occurrence of e' by the constant e', by a fresh variable o' with $\nu(o') = e'$, or by an existing variable $o'' \in M(e')$ if $M(e') \neq \emptyset$. Any fresh variable is added to $M(e')$. Record the corresponding atom using the original relation label and direction of the traversed KG triplet, and continue the walk from e'.
3. Return the conjunction of all recorded atoms.

Thus, repeated occurrences of an entity may be represented by its constant, by distinct variables, or by a shared variable. We now characterize the queries obtainable by this procedure as n and L range over all positive integers.

Proposition 4.1 Given a KG G, the set of queries obtainable by the Dynamic Query Synthesizer is the class of positive conjunctive EFO queries that have a non-empty answer set on G.

Proof. First, every synthesized query is a positive conjunctive EFO query. Its query graph is connected, since all walks start from the same seed entity and each recorded atom joins the current term to its destination. Moreover, every recorded atom corresponds to a KG triplet between the entities assigned to its terms. Hence the assignment ν, with $\nu(c) = c$ for constants, satisfies the generated conjunction, so its answer set on G is non-empty.

Conversely, let

$$q(\mathbf{x}) = \exists \mathbf{y}.\, C$$

be a positive conjunctive EFO query with a non-empty answer set on G, and let $\nu^\star$ be an assignment satisfying C. By assumption, the query graph of C is finite and connected. Choose any term z as a root. Then finitely many walks starting from z can collectively traverse every atom of C; let n be their number and L their maximum length.

Choose the seed entity $\nu^\star(z)$ and replay these walks through the entities assigned by $\nu^\star$. Represent each constant by itself. For each variable o of C, choose a fresh generated variable on its first occurrence, assign it to $\nu^\star(o)$, and reuse the same generated variable on subsequent occurrences. Thus, distinct query variables remain distinct even when $\nu^\star$ maps them to the same entity, while repeated occurrences of the same variable are represented consistently.

Since $\nu^\star$ satisfies every atom of C, each prescribed traversal corresponds to an available KG triplet. Recording its original relation and direction therefore reproduces the corresponding atom of C. Hence the generated conjunction is C up to repeated atoms and consistent variable renaming.

Finally, designate the generated variables corresponding to $\mathbf{x}$ as free and existentially quantify the remaining variables. Only finitely many prescribed traversal, representation, and stopping choices are required. Thus q is obtainable by the synthesizer for some finite n and L. □

Union and Negation. Union queries can be constructed by disjoining independently synthesized positive conjunctive EFO queries with the same free variables, while keeping their existential variables distinct. Under this construction, each constituent conjunctive query is required to have a non-empty answer set on G. Negated queries can be constructed by negating recorded positive atoms. However, the assignment used during generation need not satisfy the resulting query, which may therefore have an empty answer set.

Training Configuration. Proposition 4.1 allows any finite walk length and number of walks. In practice, however, we limit the walk length and the number of walks to keep query generation and training computationally tractable. VASIL uses at most four random walks, each of length at most four, with $\eta = 0.5$ to generate a single connected conjunction containing exactly one constant. To ensure that each generated query contains exactly one constant, once the first constant has been introduced, all subsequent terms are represented by variables. This single-constant constraint focuses training on minimally grounded queries, which constitute the most challenging setting for query answering. Every variable is in turn treated as a free variable, and all valid assignments to that variable, including the associated entity, are used as supervision targets.

B Qualitative Examples of the Synthesized Queries

Table 7 presents six example queries generated by the dynamic query synthesizer (§ 4.1) from CoDEx-M (Safavi & Koutra, 2020), one of the pre-training KGs used by VASIL.

The first three queries are tree-shaped. In the 3-way intersection, constraints from three constants converge directly at x. In the intersection-then-1-hop query, the constraints from two constants meet at an existential variable y_1, which connects to the answer variable x via a single edge. The 3-hop chain connects x to a single constant through three edges and two existential variables, y_1 and y_2.

The remaining three queries contain cycles. The multi-edge query links x to a single constant through two distinct relations, forming the smallest cyclic structure considered here. The 3-node cycle forms a triangle among x, an existential variable y_1, and a constant. The 4-node cycle connects both x and a constant to two shared existential variables, y_1 and y_2, forming a four-node cycle.

The six queries in Table 7 illustrate some of the query structures the synthesizer can produce, rather than exhaustively covering its output space. The synthesizer constructs each query graph on the fly by merging multiple random walks, without relying on predefined query templates. This process allows both tree-shaped and cyclic structures to emerge within the same generation framework.

C Dataset Details

Our evaluation spans 37 benchmarks across two tasks: 33 for CLQA, including 25 with tree-shaped queries and eight with diverse query structures, and four for QAC (Olejniczak et al., 2025). The tree-shaped benchmarks comprise two Transductive (Ren & Leskovec, 2020), three Transductive+H (Gregucci et al., 2025), nine Semi-Inductive (Galkin et al., 2022), and 11 Fully-Inductive benchmarks (Galkin et al., 2024b). The eight benchmarks with diverse query structures comprise four DAG benchmarks (He et al., 2025), two EFO_k-CQA benchmarks (Yin et al., 2025), and two

Table 7: Example queries generated by Dynamic Query Synthesizer

Query Shape	FOL Query	Natural-Language Question
3-way intersection	$?x : castMember(\text{Blade Runner}, x)$ $\wedge castMember(\text{Stripes}, x)$ $\wedge occupation(x, \text{film actor})$	Which film actor appeared in both Blade Runner and Stripes?
Intersection-then-1-hop	$?x.\exists y_1 : castMember(y_1, x)$ $\wedge castMember(y_1, \text{Gary Oldman})$ $\wedge genre(y_1, \text{action film})$	Who appeared in an action film alongside Gary Oldman?
3-hop chain	$?x.\exists y_1 y_2 : spouse(y_1, x)$ $\wedge castMember(y_2, y_1)$ $\wedge genre(y_2, \text{mystery film})$	Who is married to someone who appeared in a mystery film?
Multi-edge	$?x : director(x, \text{Sylvester Stallone})$ $\wedge castMember(x, \text{Sylvester Stallone})$	Which film did Sylvester Stallone both direct and appear in?
3-node cycle	$?x.\exists y_1 : occupation(x, \text{screenwriter})$ $\wedge spouse(y_1, x)$ $\wedge occupation(y_1, \text{screenwriter})$	Which screenwriter is married to another screenwriter?
4-node cycle	$?x.\exists y_1 y_2 : influencedBy(y_1, x)$ $\wedge influencedBy(y_2, x)$ $\wedge influencedBy(y_1, \text{E. Hemingway})$ $\wedge influencedBy(y_2, \text{E. Hemingway})$	Who influenced two people who were also influenced by Ernest Hemingway?

Table 8: Statistics of the pre-training KGs. Train, Valid, and Test denote the triplet counts in the corresponding splits.

KG	Entities	Relations	Train	Valid	Test
FB15k-237	14,505	237	272,115	17,526	20,438
NELL995	63,361	200	114,213	14,324	14,267
CoDEx-M	17,050	51	185,584	10,310	10,311
WN18RR	40,943	11	86,835	3,034	3,134

EFO_3-Small benchmarks (Fei et al., 2026). We exclude FB15k-based benchmarks because of known inverse-relation leakage (Toutanova & Chen, 2015). For reference, we also report statistics for the training queries associated with these benchmarks, although none of these queries are used to train VASIL. Each query structure is named using abbreviations: "p" for multi-hop path (projection), "i" for intersection, "n" for negation, and "u" for union.

C.1 Statistics of Training and Inference KGs

Table 8 summarizes statistics for the pre-training KGs, while Table 9 summarizes the statistics of the KGs used in the Transductive benchmark and the additional benchmarks listed in its caption. Train, Valid, and Test denote the triplet counts in the corresponding splits. Tables 10 and 11 summarize the statistics of the KGs used in the Semi-Inductive and Fully-Inductive benchmarks, respectively. Validation KG and Test KG denote the inference KGs used during validation and testing.

C.2 Statistics of Tree-Shaped Query Benchmarks

Tables 12–14 summarize query statistics for the Transductive (Ren & Leskovec, 2020) and Transductive+H (Gregucci et al., 2025) benchmarks. Tables 15 and 16 provide the corresponding statistics for the Semi-Inductive benchmarks (Galkin et al., 2022), while Table 17 reports test query counts for the Fully-Inductive benchmarks (Galkin et al., 2024b). In the Fully-Inductive table, values that are identical across query types are reported only once in the caption. Unavailable entries are marked as N/A or "–".

Table 9: Statistics of the KGs used in the Transductive, Transductive+H, DAG, EFO_k-CQA, EFO_3-Small, and QAC benchmarks. Train, Valid, and Test denote the triplet counts in the corresponding splits. ICEWS18 is used only for the Transductive+H benchmark.

KG	Entities	Relations	Train	Valid	Test
FB15k-237	14,505	237	272,115	17,526	20,438
NELL995	63,361	200	114,213	14,324	14,267
ICEWS18	20,840	250	213,304	25,048	24,689

Table 10: Statistics of the KGs used for the Semi-Inductive benchmark (Galkin et al., 2022). *Valid* and *Test* denote the number of missing triplets held out from the corresponding inference graph.

	Relations	Training KG		Validation KG			Test KG		
		Entities	Triplets	Entities	Triplets	Valid	Entities	Triplets	Test
106%	233	13,091	246,713	13,801	275,668	5,110	13,802	269,448	4,012
113%	234	11,601	200,839	13,022	245,759	7,925	13,021	243,034	7,447
121%	233	10,184	149,440	12,314	206,777	10,116	12,314	215,446	11,645
133%	233	8,634	114,365	11,468	186,631	12,739	11,471	183,905	12,265
150%	231	7,232	81,342	10,783	155,731	13,118	10,782	165,676	14,878
175%	218	5,560	51,261	9,801	132,706	14,346	9,781	133,247	14,446
217%	223	4,134	26,228	9,062	113,642	15,405	9,058	106,193	14,089
300%	206	2,650	12,220	8,252	89,340	13,568	8,266	93,578	14,329
550%	156	1,084	2,633	7,247	68,279	11,491	7,275	66,762	11,252

C.3 Statistics of Diverse Query Structure Benchmarks

Tables 18, 19, and 20 summarize query statistics for the DAG (He et al., 2025), EFO_3-Small (Fei et al., 2026), and EFO_k-CQA (Yin et al., 2025) benchmarks, respectively. For EFO_k-CQA, queries are further categorized by the number of free variables, existential variables (e), and constants (c), as well as by graph structure: simple directed acyclic graphs (SDAG), multi-edge structure (Multi), and cyclic graphs (Cyclic).

C.4 Statistics of QAC Benchmarks

Table 21 summarizes the query statistics for the QAC benchmark (Olejniczak et al., 2025). 3-hub, 4-hub, and 5-hub denote hub queries constructed from the corresponding number of hub entities. %easy, %hard, and %neg denote the proportions of easy answers, hard answers, and incorrect proposals in each split, respectively.

Table 11: Statistics of the KGs used for the Fully-Inductive benchmark (Galkin et al., 2024b). *Valid* and *Test* denote the number of missing triplets held out from the corresponding inference graph.

	Training KG			Validation KG				Test KG			
	Entities	Rels	Triplets	Entities	Rels	Triplets	Valid	Entities	Rels	Triplets	Test
Art	10,000	65	27,262	10,000	65	27,262	3,026	10,000	65	28,023	3,113
Award	10,000	17	23,821	10,000	13	23,821	2,646	10,000	17	25,056	2,783
Education	10,000	19	14,355	10,000	19	14,355	1,594	10,000	19	14,193	1,575
Health	10,000	31	15,539	10,000	31	15,539	1,725	10,000	31	15,337	1,703
Infrastructure	10,000	37	21,990	10,000	37	21,990	2,443	10,000	37	21,646	2,405
Location	10,000	62	85,063	10,000	62	85,063	9,451	10,000	62	80,269	8,917
Organization	10,000	34	33,325	10,000	34	33,325	3,702	10,000	34	31,314	3,357
People	10,000	40	55,698	10,000	40	55,698	6,188	10,000	40	58,530	6,503
Science	10,000	66	12,576	10,000	66	12,576	1,397	10,000	66	12,516	1,388
Sport	10,000	34	47,251	10,000	34	47,251	5,250	10,000	34	46,717	5,190
Taxonomy	10,000	59	18,921	10,000	59	18,921	2,102	10,000	59	19,416	2,157

Table 12: Statistics of the queries used for the Transductive benchmark (Ren & Leskovec, 2020). Slash-separated query types show counts for each listed query type, not their combined total.

Split	**Query Type**	**FB15k-237-BetaE**	**NELL995-BetaE**
Train	1p/2p/3p/2i/3i	149,689	107,982
	2in/3in/inp/pin/pni	14,968	10,798
Valid	1p	20,094	16,910
	Others	5,000	4,000
Test	1p	22,804	17,021
	Others	5,000	4,000

Table 13: Statistics of the positive queries used for the Transductive+H benchmark (Gregucci et al., 2025).

		1p	**2p**	**3p**	**4p**	**2i**	**3i**	**4i**	**pi**	**ip**	**2u**	**up**
FB15k-237+H	training	149,689	149,689	149,689	–	149,689	149,689	–	–	–	–	–
	validation	20,094	1,416	2,262	3,755	7,565	11,078	23,150	2,113	776	7,610	507
	test	22,804	936	1,461	2,670	6,692	9,615	20,189	1,586	604	6,695	473
NELL995+H	training	107,982	107,982	107,982	–	107,982	107,982	–	–	–	–	–
	validation	16,910	944	1,953	2,070	8,286	14,108	24,479	2,114	1,264	8,169	995
	test	17,021	1,054	1,583	2,322	8,278	13,118	22,326	1,957	1,260	8,168	1,073
ICEWS18+H	training	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	validation	30,803	2,237	3,077	4,187	11,531	26,614	38,881	13,480	4,452	7,400	3,242
	test	30,605	2,125	3,582	4,338	11,918	27,252	38,767	14,746	5,140	7,516	3,855

Table 14: Statistics of the negative queries used for the Transductive+H benchmark (Gregucci et al., 2025).

		2in	**3in**	**inp**	**pin**	**pni**
FB15k-237+H	training	14,968	14,968	14,968	14,968	14,968
	validation	5,000	3,820	2,745	2,406	5,000
	test	5,000	3,057	2,663	2,093	5,000
NELL995+H	training	10,798	10,798	10,798	10,798	10,798
	validation	4,000	5,100	2,459	2,214	4,000
	test	4,000	5,236	2,708	1,995	4,000
ICEWS18+H	training	N/A	N/A	N/A	N/A	N/A
	validation	1,347	7,487	2,642	2,716	1,440
	test	1,548	7,811	2,875	2,919	1,508

Table 15: Statistics of the positive queries used for the Semi-Inductive benchmark (Galkin et al., 2022).

		1p	2p	3p	2i	3i	pi	ip	2u	up
106%	training	135,613	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	validation	6,582	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
	test	5,446	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
113%	training	115,523	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	validation	10,256	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
	test	9,782	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
121%	training	91,228	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	validation	12,696	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
	test	14,458	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
133%	training	75,326	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	validation	15,541	50,000	50,000	50,000	50,000	50,000	50,000	20,000	20,000
	test	15,270	50,000	50,000	50,000	50,000	50,000	50,000	20,000	20,000
150%	training	56,114	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	validation	16,229	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	test	17,683	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
175%	training	38,851	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	validation	17,235	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	test	17,476	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
217%	training	22,422	30,000	30,000	50,000	50,000	50,000	50,000	50,000	50,000
	validation	18,168	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	test	16,902	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
300%	training	11,699	15,000	15,000	40,000	40,000	50,000	50,000	50,000	50,000
	validation	16,189	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	test	17,105	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
550%	training	3,284	15,000	15,000	40,000	40,000	50,000	50,000	50,000	50,000
	validation	13,616	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000
	test	13,670	50,000	50,000	50,000	50,000	50,000	50,000	50,000	50,000

Table 16: Statistics of the negative queries used for the Semi-Inductive benchmark (Galkin et al., 2022).

		2in	**3in**	**inp**	**pin**	**pni**
106%	training	50,000	40,000	50,000	50,000	50,000
	validation	1,000	1,000	1,000	1,000	1,000
	test	1,000	1,000	1,000	1,000	1,000
113%	training	50,000	40,000	50,000	50,000	50,000
	validation	1,000	1,000	1,000	1,000	1,000
	test	1,000	1,000	1,000	1,000	1,000
121%	training	50,000	40,000	50,000	50,000	50,000
	validation	5,000	5,000	5,000	5,000	5,000
	test	5,000	5,000	5,000	5,000	5,000
133%	training	50,000	40,000	50,000	50,000	50,000
	validation	5,000	5,000	5,000	5,000	5,000
	test	5,000	5,000	5,000	5,000	5,000
150%	training	50,000	40,000	50,000	50,000	50,000
	validation	5,000	5,000	5,000	5,000	5,000
	test	5,000	5,000	5,000	5,000	5,000
175%	training	50,000	40,000	50,000	50,000	50,000
	validation	10,000	10,000	10,000	10,000	10,000
	test	10,000	10,000	10,000	10,000	10,000
217%	training	30,000	30,000	50,000	50,000	50,000
	validation	10,000	10,000	10,000	10,000	10,000
	test	10,000	10,000	10,000	10,000	10,000
300%	training	15,000	15,000	50,000	40,000	50,000
	validation	10,000	10,000	10,000	10,000	10,000
	test	10,000	10,000	10,000	10,000	10,000
550%	training	10,000	10,000	30,000	30,000	30,000
	validation	10,000	10,000	10,000	10,000	10,000
	test	10,000	10,000	10,000	10,000	10,000

Table 17: Statistics of the test queries used for the Fully-Inductive benchmark (Galkin et al., 2024b). "Other positive" indicates the count for each of 2i, 3i, pi, ip, 2u, and up, while "Negative" indicates the count for each of 2in, 3in, inp, pin, and pni. Grouped counts are reported per query type rather than as totals.

Dataset	1p	2p	3p	Other positive	Negative
Art	3,113	10,000	10,000	10,000	1,000
Award	2,783	10,000	10,000	10,000	1,000
Education	1,575	10,000	10,000	10,000	1,000
Health	1,703	10,000	10,000	10,000	1,000
Infrastructure	2,405	10,000	10,000	10,000	1,000
Location	8,917	8,000	4,000	8,000	1,000
Organization	3,357	8,000	4,000	8,000	1,000
People	6,503	10,000	10,000	10,000	1,000
Science	1,388	10,000	10,000	10,000	1,000
Sport	5,190	8,000	4,000	8,000	1,000
Taxonomy	2,157	8,000	8,000	8,000	1,000

Table 18: Statistics of queries for DAG benchmarks (He et al., 2025). The label "s" indicates a multi-edge in the query structure.

Dataset	Split	**p2s**	**p3s**	**p2sp**	**i2s**	**in2s**	**u2s**
FB15k-237-DAG-Easy	valid	1,000	1,000	1,000	1,000	1,000	1,000
	test	5,000	5,000	5,000	5,000	5,000	5,000
FB15k-237-DAG-Hard	valid	1,000	1,000	1,000	1,000	1,000	1,000
	test	4,700	4,700	4,700	4,700	4,700	4,700
NELL-DAG-Easy	valid	1,000	1,000	1,000	1,000	1,000	1,000
	test	1,000	1,000	1,000	1,000	1,000	1,000
NELL-DAG-Hard	valid	1,000	1,000	1,000	1,000	1,000	1,000
	test	1,500	1,500	1,500	1,500	1,500	1,500

Table 19: Statistics of test queries for EFO_3-Small benchmarks (Fei et al., 2026). k denotes the number of free variables. "p" denotes multi-hop projection, "n" denotes negation, "f" denotes free variables, "d" denotes a disconnected dependency pattern among free variables, "m" denotes a multi-edge structure, and "c" denotes a cyclic structure.

	fp	**fpn**	**fpm**	**fd**	**fdm**	**fc**	**fcn**
$k = 2$							
FB15k-237-EFO_3	1,000	1,000	1,000	1,000	1,000	1,000	1,000
NELL-EFO_3	420	100	400	400	400	400	100
$k = 3$							
FB15k-237-EFO_3	1,000	1,000	1,000	1,000	1,000	1,000	1,000
NELL-EFO_3	1,000	1,000	1,000	100	1,000	1,000	200

Table 20: Statistics of test queries for EFO_k-CQA benchmarks (Yin et al., 2025). Columns group the number of existential variables e and structural category; c is the number of constants. k denotes the number of free variables.

(a) $k = 1$

		e=0	e=1		e=2		
Dataset	c	SDAG	SDAG	Multi	SDAG	Multi	Cyclic
FB15k-237-EFO_k	1	1,000	2,000	3,000	4,000	12,000	3,000
	2	1,500	4,500	4,500	15,000	30,000	6,000
	3	1,500	6,000	6,000	27,000	54,000	9,000
NELL-EFO_k	1	400	800	1,000	1,600	4,000	1,000
	2	500	1,500	1,500	5,000	10,000	2,000
	3	500	2,000	2,000	9,000	18,000	3,000

(b) $k = 2$

		e=0		e=1			e=2		
Dataset	c	SDAG	Multi	SDAG	Multi	Cyclic	SDAG	Multi	Cyclic
FB15k-237-EFO_k	1	2,000	3,000	12,000	27,000	6,000	12,000	48,000	39,000
	2	6,000	6,000	30,000	54,000	12,000	57,000	162,000	96,000
	3	6,000	6,000	48,000	90,000	18,000	0	0	0
NELL-EFO_k	1	800	1,000	4,400	9,000	2,000	4,800	16,000	13,000
	2	2,000	2,000	10,000	18,000	4,000	19,000	54,000	32,000
	3	2,000	2,000	16,000	30,000	6,000	0	0	0

Table 21: Statistics of test queries for QAC benchmarks (Olejniczak et al., 2025)

	2p	**3p**	**pi**	**ip**	**inp**	**pin**	**3-hub**	**4-hub**	**5-hub**
FB15k-237-QAC									
queries	500	500	500	500	500	500	300	300	300
answers	9,818	9,828	9,632	9,358	9,808	9,898	2,036	1,988	2,028
%easy	26.5%	24.0%	27.5%	28.7%	35.8%	32.9%	18.7%	16.6%	17.0%
%hard	23.5%	26.0%	22.5%	21.3%	14.2%	17.1%	31.3%	33.4%	33.0%
%neg	50.0%	50.0%	50.0%	50.0%	50.0%	50.0%	50.0%	50.0%	50.0%
NELL-QAC									
queries	500	500	500	500	500	500	300	300	300
answers	9,708	9,702	9,478	9,694	9,698	9,888	2,174	2,186	1,922
%easy	23.6%	22.6%	25.2%	23.6%	35.8%	32.8%	15.9%	14.4%	13.7%
%hard	26.4%	27.4%	24.8%	26.4%	14.2%	17.2%	34.1%	35.6%	36.3%
%neg	50.0%	50.0%	50.0%	50.0%	50.0%	50.0%	50.0%	50.0%	50.0%

Table 22: Computational cost comparison between VASIL and UltraQuery (Galkin et al., 2024b) on a single NVIDIA A6000 GPU. For training, we report the time per 1,000 steps (Time, s) and the total GPU hours required for each training phase (GPU-h). For inference, we report the per-query inference time (ms) averaged across all query types and datasets. Memory denotes peak GPU memory usage. UltraQuery requires pre-training the Ultra backbone (Galkin et al., 2024a) followed by fine-tuning for complex query answering, so we report the total training cost across both phases. In contrast, VASIL is trained end-to-end in a single phase.

		Training			Inference	
		Time (s)	GPU-h	Memory (GB)	Time (ms)	Memory (GB)
UltraQuery	Ultra pretrain	483.9	53.8	45.1	69.0	5.1
	UltraQuery fine-tune	610.8	6.8	13.6		
	Total	–	60.6	–		
VASIL	End-to-end (single phase)	353.3	14.7	5.0	77.1	4.9

D Implementation and Training Details of VASIL

VASIL is implemented using a pre-LN architecture (Xiong et al., 2020) with 6 layers, a hidden dimension of 64, and a dropout rate of 0.1. Both entity and relation multi-head attention (Vaswani et al., 2017) use 8 heads. The feed-forward networks use GELU (Hendrycks & Gimpel, 2016) activations, and all linear projections include bias terms. For all benchmarks, we set the negation hyperparameters to $\tau = 0.5$ and $\beta = 4$.

We select the pre-training hyperparameters through grid search, with model selection based on the validation queries of FB15k-237-BetaE (Ren & Leskovec, 2020). We train for 300 epochs, with 500 steps per epoch, a batch size of 1, and gradient accumulation over 16 steps. We use AdamW (Loshchilov & Hutter, 2019) with a weight decay of 0.01, excluding bias terms from weight decay. The learning rate is fixed at 2×10^{-4} for the first 200 epochs and then decayed to 1×10^{-5} over the final 100 epochs using a cosine schedule (Loshchilov & Hutter, 2017).

To expose the model to diverse KG environments, we first select a KG G from the pre-training KGs and operate the dynamic query synthesizer (§ 4.1) on randomly sampled subgraphs of G. For each query, we sample a triplet-retention probability uniformly from $[0.3, 0.7]$ and construct a subgraph by independently retaining each triplet in G with that probability. To further simulate incomplete graph observations, we independently remove each edge appearing in the generated query from G with probability 0.5 before labeling G. This reduces the model's reliance on queried edges being explicitly present in G and encourages it to infer missing knowledge.

All experiments are conducted with Python 3.11.13, PyTorch 2.6.0, and CUDA 11.8 on a single NVIDIA RTX A6000 GPU.

E Scalability Analysis

Table 22 compares the training and inference costs of VASIL and UltraQuery (Galkin et al., 2024b). All measurements are obtained on a single NVIDIA A6000 GPU, with UltraQuery evaluated under its default configuration. Since UltraQuery fine-tunes a pretrained Ultra model (Galkin et al., 2024a), its overall training cost includes both Ultra pretraining and UltraQuery fine-tuning, whereas VASIL is trained end-to-end in a single phase. As shown in Table 22, VASIL substantially reduces both training time and memory requirements while maintaining comparable inference efficiency. This efficiency is achieved while consistently outperforming UltraQuery across the evaluated settings as shown in Section 5.1 and Section 5.2.

We further compare the time complexity of UltraQuery and VASIL. Consider a query with $|\mathcal{A}|$ atoms and a KG with $|\mathcal{V}|$ entities, $|\mathcal{R}|$ relations, and $|\mathcal{E}|$ triplets. Both models use L GNN layers on the whole KG with hidden dimension D, while UltraQuery additionally employs L' GNN layers on a relation graph with $|\mathcal{E}_R|$ edges following the methodology of Ultra (Galkin et al., 2024a). The time complexity of UltraQuery is $O\big(|\mathcal{A}| \cdot \big(L' \cdot (|\mathcal{R}|D^2 + |\mathcal{E}_R|D) + L \cdot (|\mathcal{V}|D^2 + |\mathcal{E}|D)\big)\big)$, and

the time complexity of VASIL is $O\big(|\mathcal{A}| \cdot L \cdot (|\mathcal{V}|D^2 + |\mathcal{R}|D^2 + |\mathcal{E}|D)\big)$. Since the relation graph size is bounded by the KG size, the two models have the same asymptotic time complexity for a fixed number of layers. UltraQuery fine-tune is slightly faster in practice, partly due to optimized implementations. Our implementation leaves room for further low-level optimization, suggesting that implementation differences contribute to the observed runtime gap.

F Detailed Results on Tree-Shaped Queries

We report detailed results for all baselines on tree-shaped queries, broken down by query type (§ 5.1). The baselines include GQE (Hamilton et al., 2018), Query2Box (Ren et al., 2020), BetaE (Ren & Leskovec, 2020), CQD-CO (Arakelyan et al., 2021), CQD-Beam (Arakelyan et al., 2021), FuzzQE (Chen et al., 2022), ConE (Zhang et al., 2021), GNN-QE (Zhu et al., 2022), QTO (Bai et al., 2023), LMPNN (Wang et al., 2023), CLMPT (Zhang et al., 2024), FIT (Yin et al., 2024), UltraQuery (Galkin et al., 2024b), SAQE (Cao et al., 2025), CQD-Hybrid (Gregucci et al., 2025), Edge-type Heuristic (Galkin et al., 2022), NPQE (Galkin et al., 2022), and NPQE w/ GNN (Galkin et al., 2022).

Following the standard practice (Galkin et al., 2024b), we report MRR (%) for the Transductive, Transductive+H, and Fully-Inductive benchmarks, and Hits@10 (%) for the Semi-Inductive benchmarks. VASIL is trained exclusively on four pre-training KGs: FB15k-237 (Toutanova & Chen, 2015), NELL995 (Xiong et al., 2017), CoDEx-M (Safavi & Koutra, 2020), and WN18RR (Dettmers et al., 2018). Thus, VASIL uses only the inference graph of each evaluation benchmark, without access to each training graph. avg_p and avg_n denote the macro-averages over the corresponding positive and negative query types, respectively. The best results are shown in **bold**, and the second-best results are underlined.

F.1 Results on Transductive Benchmarks

Tables 23 and 24 report results on two Transductive benchmarks (Ren & Leskovec, 2020): FB15k-237-BetaE and NELL995-BetaE. Despite using a single pre-trained model, VASIL remains competitive with methods trained separately on each benchmark.

F.2 Results on Transductive+H Benchmarks

Tables 25 and 26 report results on three Transductive+H benchmarks (Gregucci et al., 2025): FB15k-237+H, NELL995+H, and ICEWS18+H. These benchmarks exclude queries that can be answered by solving simpler subqueries alone, thus mitigating the bias present in the Transductive benchmarks. They also include longer-path (4p) and larger-intersection (4i) query types to increase both difficulty and structural diversity. Since reasoning over simpler subqueries is insufficient on these benchmarks, the strong performance of VASIL demonstrates its ability to leverage the constraints imposed by the full query structure.

F.3 Results on Semi-Inductive Benchmarks

Tables 27 and 28 report results on nine Semi-Inductive benchmarks (Galkin et al., 2022): 106%, 113%, 121%, 133%, 150%, 175%, 217%, 300%, and 550%. The ratio measures the number of entities in the inference KG as a percentage of that in the training KG. Since VASIL is applied zero-shot, its performance remains relatively stable across different ratios.

F.4 Results on Fully-Inductive Benchmarks

Tables 29 and 30 report results on all 11 fully-inductive benchmarks (Galkin et al., 2024b): Art, Award, Education, Health, Infrastructure, Location, Organization, People, Science, Sport, and Taxonomy. VASIL's consistent gains over UltraQuery (Galkin et al., 2024b) across these benchmarks demonstrate effective zero-shot transfer to KGs with entirely different entities and relations.

Table 23: MRR(%) results on positive queries across two Transductive benchmarks (Ren & Leskovec, 2020).

		avg_p	1p	2p	3p	2i	3i	pi	ip	2u	up
FB15k-237-BetaE	GQE	16.3	35.0	7.2	5.3	23.3	34.6	16.5	10.7	8.2	5.7
	Query2Box	20.1	40.6	9.4	6.8	29.5	42.3	21.2	12.6	11.3	7.6
	BetaE	20.9	39.0	10.9	10.0	28.8	42.5	22.4	12.6	12.4	9.7
	CQD-CO	21.8	46.7	9.5	6.3	31.2	40.6	23.6	16.0	14.5	8.2
	CQD-Beam	22.3	46.7	11.6	8.0	31.2	40.6	21.2	18.7	14.6	8.4
	FuzzQE	24.2	42.2	13.3	10.2	33.0	47.3	26.2	18.9	14.6	10.8
	ConE	23.4	41.8	12.8	11.0	32.6	47.3	25.5	14.0	14.5	10.8
	GNN-QE	26.8	42.8	14.7	11.8	38.3	54.1	31.1	18.9	16.2	13.4
	QTO	33.5	49.0	21.4	21.2	**43.1**	**56.8**	38.1	28.0	22.7	21.4
	LMPNN	24.1	45.9	13.1	10.3	34.8	48.9	22.7	17.6	13.5	10.3
	CLMPT	25.9	45.7	13.7	11.3	37.4	52.0	28.2	19.0	14.3	11.1
	FIT	27.4	46.7	14.6	12.8	37.5	51.6	30.1	21.9	18.0	13.1
	UltraQuery	24.4	40.6	11.5	8.9	35.7	51.0	29.6	18.6	13.2	10.2
	SAQE	**34.6**	**50.1**	**22.8**	**23.7**	42.9	56.4	**39.6**	**29.6**	**24.1**	**22.3**
	VASIL	29.6	46.4	17.4	15.7	40.1	54.0	33.9	24.8	18.6	15.9
NELL995-BetaE	GQE	18.6	32.8	11.9	9.6	27.5	35.2	18.4	14.4	8.5	8.8
	Query2Box	22.9	42.2	14.0	11.2	33.3	44.5	22.4	16.8	11.3	10.3
	BetaE	24.6	53.0	13.0	11.4	37.6	47.5	24.1	14.3	12.2	8.5
	CQD-CO	28.8	60.4	17.8	12.7	39.3	46.6	30.1	22.0	17.3	13.2
	CQD-Beam	28.6	60.4	20.6	11.6	39.3	46.6	25.4	23.9	17.5	12.2
	FuzzQE	29.3	58.1	19.3	15.7	39.8	50.3	28.1	21.8	17.3	13.7
	ConE	27.2	53.1	16.1	13.9	40.0	50.8	26.3	17.5	15.3	11.3
	GNN-QE	28.9	53.3	18.9	14.9	42.4	52.5	30.8	18.9	15.9	12.6
	QTO	32.9	60.7	24.1	21.6	42.5	50.6	31.3	26.5	20.4	17.9
	LMPNN	30.8	60.6	22.1	17.5	40.1	50.3	28.4	24.9	17.2	15.7
	CLMPT	31.2	58.9	22.1	18.4	41.8	51.9	28.8	24.4	18.6	16.2
	FIT	33.4	60.8	23.8	21.2	44.3	54.1	31.7	26.6	20.3	17.6
	UltraQuery	22.6	38.9	11.2	9.7	36.3	47.7	25.1	16.0	10.2	8.4
	SAQE	**33.9**	61.9	**25.7**	**22.5**	43.6	51.8	31.2	**27.6**	**21.9**	**18.8**
	VASIL	**33.9**	**62.9**	22.9	20.9	**45.5**	**55.5**	**32.9**	27.5	20.2	16.5

Table 24: MRR(%) results on negative queries across two Transductive benchmarks (Ren & Leskovec, 2020).

		avg_n	2in	3in	inp	pin	pni
FB15k-237-BetaE	BetaE	5.4	5.1	7.9	7.4	3.6	3.4
	FuzzQE	8.5	9.7	12.6	7.8	5.8	6.6
	ConE	5.9	5.4	8.6	7.8	4.0	3.6
	GNN-QE	10.2	10.0	16.8	9.3	7.2	7.8
	QTO	15.5	16.8	26.7	15.1	13.6	5.4
	LMPNN	7.8	8.7	12.9	7.7	4.6	5.2
	CLMPT	7.9	7.7	13.7	8.0	5.0	5.1
	FIT	-	14.0	20.0	10.2	9.5	–
	UltraQuery	8.2	8.1	15.0	7.0	5.3	5.8
	SAQE	**16.7**	**17.2**	**27.6**	**16.8**	**14.1**	7.6
	VASIL	13.2	12.5	22.0	12.9	9.7	**9.0**
NELL995-BetaE	BetaE	5.9	5.1	7.8	10.0	3.1	3.5
	FuzzQE	8.0	8.3	10.2	11.5	4.6	5.4
	ConE	6.4	5.7	8.1	10.8	3.5	3.9
	GNN-QE	9.7	9.9	14.6	11.4	6.3	6.3
	QTO	12.9	13.8	**17.9**	16.9	9.9	5.9
	LMPNN	8.0	8.5	10.8	12.2	3.9	4.8
	CLMPT	7.0	6.6	8.1	11.8	3.8	4.5
	FIT	–	12.6	16.4	15.3	8.3	–
	UltraQuery	7.2	6.4	11.6	9.7	4.4	4.1
	SAQE	**13.4**	**15.2**	17.7	**17.3**	**10.8**	6.2
	VASIL	12.6	13.0	17.7	16.8	8.4	**7.3**

Table 25: MRR(%) results on positive queries across three Transductive+H benchmarks (Gregucci et al., 2025).

		avg_p	1p	2p	3p	2i	3i	pi	ip	2u	up	4p	4i
FB15k-237+H	GNN-QE	13.6	42.8	5.2	4.0	6.0	8.8	5.6	9.9	32.5	10.0	4.3	20.0
	UltraQuery	12.2	40.6	4.5	3.5	5.2	7.2	5.3	10.1	29.4	8.3	3.8	16.4
	CQD-Beam	15.5	**46.7**	4.4	2.4	**11.3**	12.8	6.0	11.5	40.1	10.6	1.1	23.8
	CQD-Hybrid	14.7	**46.7**	4.8	3.1	6.0	8.6	5.5	12.9	**42.2**	12.0	2.4	17.4
	ConE	12.2	41.8	4.6	3.9	9.1	10.3	3.8	7.9	22.8	6.0	3.5	20.3
	QTO	14.5	**46.7**	4.9	3.7	8.7	10.1	6.1	13.5	30.6	11.2	3.9	20.2
	CLMPT	**15.9**	45.3	5.3	**4.7**	10.2	12.2	5.6	**14.9**	33.6	**14.2**	**4.5**	24.0
	VASIL	15.0	46.4	**6.0**	4.0	10.7	**13.1**	**7.0**	13.3	25.1	10.4	3.8	**25.4**
NELL995+H	GNN-QE	18.4	53.6	8.0	6.0	10.7	13.3	16.0	13.5	47.5	9.8	4.7	19.4
	UltraQuery	13.6	38.9	6.1	4.1	7.9	10.2	15.8	9.3	28.1	9.5	4.2	15.6
	CQD-Beam	22.3	60.4	9.6	4.2	18.5	19.6	18.9	22.6	46.3	18.5	2.0	24.8
	CQD-Hybrid	20.9	60.4	9.0	6.1	12.1	14.4	17.4	21.2	46.4	19.3	3.5	20.4
	ConE	20.1	53.1	7.9	6.7	21.8	23.6	14.9	11.8	39.9	8.8	5.2	27.6
	QTO	22.1	60.3	9.8	8.0	14.6	15.8	17.6	21.1	49.1	18.9	7.0	20.9
	CLMPT	24.9	58.1	10.1	7.8	22.7	25.0	17.2	**24.4**	**50.0**	**22.0**	**7.2**	29.1
	VASIL	**25.1**	**62.9**	**10.9**	**8.3**	**24.7**	**25.9**	**20.8**	22.8	44.9	18.4	**7.2**	**29.8**
ICEWS18+H	GNN-QE	11.1	12.2	0.9	0.5	16.1	**26.5**	19.1	3.5	**7.6**	1.1	0.4	**34.0**
	UltraQuery	5.3	6.3	1.2	1.2	7.0	11.7	8.8	1.3	3.3	1.2	0.8	15.9
	CQD-Beam	10.4	**16.6**	2.5	**1.5**	13.0	19.5	17.1	6.7	6.8	**5.9**	1.1	24.0
	CQD-Hybrid	11.9	**16.6**	2.6	**1.5**	15.0	25.5	17.5	5.8	6.8	5.6	0.9	33.2
	ConE	1.2	3.5	0.9	0.9	1.2	0.5	1.2	1.6	1.1	0.9	0.6	0.3
	QTO	11.8	**16.6**	2.6	1.4	15.7	25.0	18.4	6.2	6.7	4.9	1.1	31.5
	CLMPT	8.0	4.7	0.8	0.1	12.0	23.0	9.7	2.1	2.7	2.2	0.1	31.0
	VASIL	**12.2**	16.1	**2.9**	1.3	**16.4**	25.2	**20.2**	**6.9**	6.3	5.6	**1.2**	31.7

Table 26: MRR(%) results on negative queries across three Transductive+H benchmarks (Gregucci et al., 2025).

		avg_n	2in	3in	inp	pin	pni
FB15k-237+H	GNN-QE	5.1	6.8	**6.5**	3.3	**3.7**	5.0
	UltraQuery	3.9	5.3	5.5	2.2	2.6	3.7
	CQD-Beam	2.5	3.3	2.6	1.2	0.6	4.9
	CQD-Hybrid	2.4	4.7	1.6	1.3	1.0	3.2
	ConE	4.0	5.1	4.9	**3.6**	2.9	3.3
	QTO	4.5	10.6	3.1	1.5	2.0	5.3
	CLMPT	3.6	6.8	2.3	2.5	1.6	4.8
	VASIL	**6.2**	**12.5**	5.4	2.1	2.1	**9.0**
NELL995+H	GNN-QE	5.1	5.5	**6.4**	4.4	**5.8**	3.3
	UltraQuery	4.2	4.5	5.9	3.6	4.3	2.7
	CQD-Beam	2.9	4.2	1.5	2.6	1.5	4.9
	CQD-Hybrid	2.9	5.1	1.2	2.4	1.4	4.3
	ConE	4.7	4.6	6.0	**6.4**	3.7	2.7
	QTO	5.3	10.2	2.3	2.4	3.1	**8.4**
	CLMPT	4.0	6.5	2.4	4.5	4.1	2.3
	VASIL	**6.4**	**13.0**	4.6	3.8	3.3	7.3
ICEWS18+H	GNN-QE	3.3	4.5	6.9	0.8	0.9	**3.5**
	UltraQuery	2.4	2.3	4.8	**1.6**	1.2	2.2
	CQD-Beam	1.6	1.5	2.9	0.9	0.2	2.7
	CQD-Hybrid	1.9	1.7	4.0	1.1	0.3	2.2
	ConE	1.6	1.7	2.9	1.3	1.1	0.9
	QTO	**3.7**	**4.9**	**8.7**	0.9	1.2	3.0
	CLMPT	0.9	1.2	2.1	0.2	0.1	1.0
	VASIL	3.5	4.1	8.1	1.3	**1.3**	2.9

Table 27: Hits@10(%) results on positive queries across 9 semi-inductive datasets (Galkin et al., 2022).

		avg_p	1p	2p	3p	2i	3i	pi	ip	2u	up
	Edge-type Heuristic	11.7	15.8	10.4	11.5	13.4	16.4	12.2	9.7	6.3	9.2
	NPQE	9.6	16.3	8.4	8.8	10.8	11.5	8.8	7.7	6.8	7.0
	NPQE w/ GNN	11.1	20.1	8.3	6.2	14.0	15.5	10.5	8.7	10.3	6.6
550%	GNN-QE	33.1	37.7	23.4	17.0	50.7	74.9	43.3	20.4	15.1	15.4
	UltraQuery	53.6	70.3	40.7	28.9	79.7	90.7	63.4	48.2	33.6	26.5
	SAQE	37.9	42.7	25.2	17.8	63.7	82.0	49.2	26.5	18.3	16.4
	VASIL	**61.4**	**77.3**	**50.7**	**42.1**	**83.8**	**92.5**	**70.6**	**56.2**	**39.5**	**39.7**
	Edge-type Heuristic	12.2	20.9	10.9	11.6	12.1	14.8	11.6	10.8	6.4	10.3
	NPQE	10.7	19.6	9.5	9.5	11.0	12.0	9.7	9.5	7.5	8.3
	NPQE w/ GNN	17.9	31.5	13.7	8.2	24.3	27.0	18.0	13.7	13.8	10.6
300%	GNN-QE	42.6	50.9	32.1	26.4	65.4	78.1	49.5	35.7	21.7	23.4
	UltraQuery	52.6	72.6	40.0	29.3	76.8	85.3	46.5	**59.6**	33.0	29.9
	SAQE	44.7	60.3	32.0	25.9	68.6	79.8	51.1	37.4	24.0	22.8
	VASIL	**58.6**	**79.1**	**47.4**	**39.9**	**81.0**	**87.8**	**64.7**	52.2	**38.2**	**37.2**
	Edge-type Heuristic	11.4	18.8	9.9	10.2	12.1	16.1	11.9	9.8	4.8	9.4
	NPQE	11.7	22.3	9.3	8.8	12.9	15.5	11.5	10.0	7.1	7.8
	NPQE w/ GNN	23.5	41.0	16.0	10.4	32.5	40.8	23.7	17.7	16.9	12.6
217%	GNN-QE	49.2	61.3	36.1	29.8	72.6	86.8	58.4	42.5	29.3	25.8
	UltraQuery	53.7	72.7	40.4	29.0	78.2	89.2	62.4	47.9	33.5	29.9
	SAQE	49.4	63.2	36.7	29.4	73.2	86.7	58.4	42.8	28.3	25.9
	VASIL	**59.5**	**78.7**	**47.1**	**39.6**	**82.1**	**91.2**	**68.0**	**52.7**	**38.9**	**37.3**
	Edge-type Heuristic	10.1	17.7	8.2	9.9	10.7	13.0	9.8	8.2	5.3	8.5
	NPQE	11.2	25.5	8.2	8.4	12.4	13.9	9.9	8.7	7.0	6.8
	NPQE w/ GNN	28.6	45.9	19.2	11.5	39.9	48.8	29.4	22.6	25.3	14.6
175%	GNN-QE	50.7	65.4	36.3	31.6	73.8	84.3	56.5	41.5	39.3	28.0
	UltraQuery	50.0	72.3	35.6	25.7	74.9	84.5	57.2	42.3	32.5	25.3
	SAQE	51.8	65.9	37.9	32.4	73.7	83.9	58.7	43.9	**40.3**	29.9
	VASIL	**55.8**	**77.9**	**42.3**	**35.4**	**78.8**	**87.7**	**63.5**	**46.9**	35.9	**33.5**
	Edge-type Heuristic	9.6	17.4	7.9	8.7	10.4	12.7	9.0	7.7	4.5	8.0
	NPQE	10.8	22.8	8.9	8.5	11.7	12.9	9.1	9.1	6.3	7.7
	NPQE w/ GNN	25.9	48.9	17.2	9.1	36.6	43.1	25.1	19.7	21.1	12.2
150%	GNN-QE	49.3	69.1	35.7	29.7	73.1	81.3	52.9	38.7	**34.3**	28.7
	UltraQuery	47.4	72.6	32.9	23.8	73.0	81.6	52.5	37.0	28.0	25.0
	SAQE	49.1	67.2	36.8	31.0	72.1	80.6	52.1	40.0	34.0	27.9
	VASIL	**52.9**	**78.3**	**39.0**	**32.4**	**77.0**	**84.1**	**57.8**	**42.3**	33.2	**31.6**
	Edge-type Heuristic	9.0	17.7	7.3	7.1	10.1	11.8	8.0	7.5	4.3	7.0
	NPQE	10.2	24.8	9.3	7.7	10.1	9.9	7.4	9.3	5.6	7.5
	NPQE w/ GNN	29.7	52.1	20.6	9.4	42.3	49.2	29.3	24.3	25.5	14.2
133%	GNN-QE	**51.4**	69.2	**38.7**	**31.1**	73.4	**79.9**	53.8	**43.7**	**42.2**	30.4
	UltraQuery	45.1	70.5	31.7	20.8	70.3	76.4	48.9	36.0	27.6	23.4
	SAQE	–	–	–	–	–	–	–	–	–	–
	VASIL	50.9	**76.4**	38.5	30.4	**74.9**	79.8	**54.5**	41.3	31.7	**30.5**
	Edge-type Heuristic	8.6	23.3	6.5	6.9	7.6	9.5	6.7	6.4	3.7	6.9
	NPQE	9.6	28.0	6.5	6.1	9.2	10.7	7.8	7.1	4.9	6.0
	NPQE w/ GNN	28.0	53.6	17.1	8.4	38.0	45.9	26.7	21.9	26.1	13.9
121%	GNN-QE	46.2	74.2	31.1	27.2	63.4	70.5	48.1	**37.3**	**38.3**	25.4
	UltraQuery	43.0	73.7	26.6	20.8	65.2	74.2	46.1	32.6	25.8	22.1
	SAQE	–	–	–	–	–	–	–	–	–	–
	VASIL	**48.3**	**78.4**	**32.7**	**28.5**	**69.9**	**78.7**	**52.3**	**37.3**	28.4	**28.8**
	Edge-type Heuristic	7.0	15.2	4.3	5.5	8.2	10.4	6.4	4.4	3.4	5.1
	NPQE	8.1	21.5	6.4	6.4	7.8	8.6	5.9	5.9	3.8	6.3
	NPQE w/ GNN	27.2	53.6	16.0	7.3	39.2	46.7	26.2	21.1	24.0	11.0
113%	GNN-QE	**49.4**	71.7	**34.9**	**31.3**	**67.5**	**74.7**	**50.7**	**38.9**	**44.3**	**30.9**
	UltraQuery	38.7	68.8	22.8	17.3	60.7	68.3	39.8	27.9	23.5	19.2
	SAQE	–	–	–	–	–	–	–	–	–	–
	VASIL	44.0	**75.3**	29.7	25.4	64.6	70.6	45.3	33.3	25.4	26.7
	Edge-type Heuristic	6.4	14.5	4.3	4.4	8.1	9.7	4.8	3.7	3.1	4.7
	NPQE	7.7	19.2	5.9	7.6	8.1	9.2	4.3	4.5	4.1	6.1
	NPQE w/ GNN	31.6	54.3	21.6	7.8	44.2	51.6	28.4	27.7	35.2	14.0
106%	GNN-QE	**50.4**	71.9	**37.0**	**32.4**	**67.9**	**73.7**	**48.2**	**41.8**	**48.0**	32.7
	UltraQuery	37.0	67.9	23.2	16.6	59.3	64.3	34.2	24.5	24.3	19.1
	SAQE	49.9	71.6	**37.0**	32.1	67.0	73.4	47.5	41.1	46.4	**33.2**
	VASIL	41.0	**73.4**	26.4	22.7	63.7	68.8	40.9	26.9	23.0	23.1

Table 28: Hits@10(%) results on negative queries across 9 semi-inductive datasets (Galkin et al., 2022).

		avg_n	2in	3in	inp	pin	pni
550%	Edge-type Heuristic	5.1	3.4	3.7	11.2	3.9	3.4
	GNN-QE	15.8	13.4	26.3	17.2	13.7	8.6
	UltraQuery	33.3	32.7	56.2	29.3	23.9	24.3
	SAQE	17.7	15.7	32.1	17.6	13.8	11.0
	VASIL	**40.6**	**38.5**	**61.4**	**41.5**	**35.9**	**25.7**
300%	Edge-type Heuristic	5.8	3.6	4.1	11.2	5.1	5.0
	GNN-QE	22.4	19.3	33.1	24.6	21.3	13.7
	UltraQuery	31.2	31.5	46.7	28.1	25.0	**24.5**
	SAQE	21.6	18.5	33.8	24.7	18.0	12.9
	VASIL	**34.8**	**35.2**	**49.0**	**37.9**	**30.3**	21.5
217%	Edge-type Heuristic	5.4	2.9	4.7	10.2	4.5	4.6
	GNN-QE	30.1	26.7	47.4	30.3	22.7	23.3
	UltraQuery	33.7	32.5	52.4	29.7	27.6	**26.1**
	SAQE	29.5	26.4	46.9	29.2	22.9	22.3
	VASIL	**38.4**	**37.8**	**58.4**	**39.1**	**32.9**	23.9
175%	Edge-type Heuristic	4.1	2.6	2.9	8.4	3.8	2.7
	GNN-QE	33.6	33.3	46.4	29.2	24.9	**34.0**
	UltraQuery	29.7	30.4	46.2	25.2	23.3	23.6
	SAQE	**34.8**	**34.7**	47.2	30.1	**28.7**	33.4
	VASIL	33.4	32.5	**51.9**	**33.6**	**28.7**	20.5
150%	Edge-type Heuristic	4.4	2.8	3.6	8.7	4.2	2.5
	GNN-QE	30.5	28.3	40.1	27.8	**27.7**	27.7
	UltraQuery	25.6	25.1	39.3	24.2	20.7	18.6
	SAQE	**31.3**	**29.4**	39.1	27.8	27.4	**28.7**
	VASIL	29.1	29.0	**40.8**	**31.9**	25.3	18.4
133%	Edge-type Heuristic	3.8	2.6	2.9	7.4	3.8	2.5
	GNN-QE	**34.1**	**35.6**	**40.3**	**27.8**	**28.6**	**38.1**
	UltraQuery	23.7	24.5	33.6	20.8	21.4	18.4
	SAQE	–	–	–	–	–	–
	VASIL	24.4	25.6	30.4	27.0	22.1	16.7
121%	Edge-type Heuristic	3.7	2.2	2.4	7.2	3.5	3.0
	GNN-QE	**29.1**	**30.5**	35.2	25.5	**21.8**	**32.6**
	UltraQuery	23.2	22.9	**36.5**	20.4	18.7	17.5
	SAQE	–	–	–	–	–	–
	VASIL	24.6	23.7	36.2	**27.6**	20.5	15.0
113%	Edge-type Heuristic	2.4	1.1	2.4	4.2	2.9	1.6
	GNN-QE	**33.9**	**36.8**	**39.5**	**23.7**	**29.6**	**39.8**
	UltraQuery	19.2	20.3	29.7	14.0	17.5	14.5
	SAQE	–	–	–	–	–	–
	VASIL	19.1	19.4	22.6	20.3	19.8	13.4
106%	Edge-type Heuristic	3.0	2.8	2.2	4.4	2.6	2.9
	GNN-QE	37.7	39.6	**43.2**	**25.3**	31.1	**49.4**
	UltraQuery	16.3	18.9	22.0	13.1	14.5	12.8
	SAQE	**37.8**	**39.8**	42.8	23.9	**33.6**	48.8
	VASIL	17.1	16.4	21.0	16.9	16.2	14.8

Table 29: MRR(%) results on positive queries across 11 fully-inductive datasets (Galkin et al., 2024b).

		avg_p	1p	2p	3p	2i	3i	pi	ip	2u	up
Art	UltraQuery	24.8	23.5	5.9	5.7	51.3	72.6	38.3	17.2	4.7	4.2
	VASIL	**35.8**	**35.3**	**17.7**	**17.9**	**60.5**	**78.5**	**48.2**	**34.3**	**14.3**	**15.9**
Award	UltraQuery	22.4	13.8	6.4	3.1	46.0	**72.1**	31.9	17.2	7.7	3.3
	VASIL	**32.6**	**30.9**	**21.0**	**16.0**	**50.4**	69.6	**36.4**	**31.1**	**22.3**	**15.3**
Education	UltraQuery	24.0	16.5	8.8	6.2	51.8	85.0	27.2	10.4	5.3	5.1
	VASIL	**30.0**	**24.4**	**17.1**	**15.8**	**56.4**	**86.2**	**29.0**	**22.6**	**11.0**	**7.8**
Health	UltraQuery	42.1	65.3	18.3	23.0	77.3	87.8	32.4	19.2	48.3	7.0
	VASIL	**51.1**	**70.9**	**32.2**	**43.4**	**82.4**	**89.6**	**37.6**	**28.1**	**53.6**	**22.2**
Infrastructure	UltraQuery	39.1	53.0	20.8	18.0	66.5	83.0	45.5	26.0	24.5	15.0
	VASIL	**59.9**	**72.2**	**50.6**	**47.1**	**80.4**	**89.6**	**60.8**	**54.2**	**44.9**	**39.6**
Location	UltraQuery	56.6	89.7	39.4	13.3	84.8	**87.2**	67.8	46.5	57.0	24.0
	VASIL	**69.7**	**92.0**	**57.2**	**45.3**	**86.1**	86.6	**73.6**	**65.2**	**78.4**	**43.1**
Organization	UltraQuery	19.4	10.3	4.9	2.2	35.5	**76.5**	27.5	11.3	5.1	1.5
	VASIL	**24.2**	**16.3**	**12.8**	**9.3**	**38.2**	69.1	**30.9**	**20.7**	**11.7**	**8.6**
People	UltraQuery	37.1	58.1	12.6	5.7	73.6	87.1	53.4	19.3	14.8	9.1
	VASIL	**54.6**	**75.8**	**35.9**	**31.9**	**85.9**	**94.1**	**68.6**	**51.0**	**23.1**	**25.4**
Science	UltraQuery	23.8	24.3	9.2	13.8	53.5	69.1	21.4	10.3	9.5	3.3
	VASIL	**31.4**	**40.4**	**17.3**	**22.1**	**60.0**	**69.7**	**25.8**	**19.8**	**18.3**	**9.2**
Sport	UltraQuery	25.4	29.4	8.4	12.9	44.5	56.5	35.4	18.9	10.0	12.3
	VASIL	**29.6**	**33.8**	**13.5**	**15.3**	**48.8**	**60.3**	**40.0**	**27.4**	**14.8**	**12.6**
Taxonomy	UltraQuery	29.0	37.7	9.6	11.4	55.8	**76.6**	36.0	15.6	11.3	7.3
	VASIL	**36.0**	**43.0**	**16.9**	**23.9**	**59.1**	76.5	**44.8**	**24.3**	**18.6**	**16.8**

Table 30: MRR(%) results on queries containing negation across 11 fully-inductive datasets (Galkin et al., 2024b).

		avg_n	2in	3in	inp	pin	pni
Art	UltraQuery	8.4	3.4	28.4	4.2	3.0	2.9
	VASIL	**19.0**	**13.2**	**42.7**	**14.2**	**12.7**	**12.2**
Award	UltraQuery	14.9	13.5	45.1	4.1	3.2	8.4
	VASIL	**24.5**	**24.2**	**48.7**	**16.5**	**17.1**	**15.8**
Education	UltraQuery	15.1	6.3	54.2	6.2	4.4	4.4
	VASIL	**20.5**	**13.1**	**60.7**	**8.9**	**9.2**	**10.5**
Health	UltraQuery	47.2	54.7	79.2	40.2	30.5	31.6
	VASIL	**55.3**	**59.5**	**84.8**	**50.2**	**44.4**	**37.8**
Infrastructure	UltraQuery	26.0	20.0	52.1	24.5	18.4	14.9
	VASIL	**47.1**	**43.5**	**67.3**	**45.7**	**41.5**	**37.5**
Location	UltraQuery	38.6	50.4	48.6	35.3	27.3	31.6
	VASIL	**54.7**	**71.5**	**60.3**	**50.3**	**43.3**	**47.9**
Organization	UltraQuery	10.7	7.7	35.0	2.8	3.5	4.3
	VASIL	**16.7**	**14.0**	**41.7**	**8.2**	**10.4**	**9.0**
People	UltraQuery	21.5	15.6	59.2	14.0	7.5	11.1
	VASIL	**37.9**	**25.7**	**82.5**	**37.4**	**24.7**	**19.2**
Science	UltraQuery	9.6	8.3	21.0	9.1	4.6	5.1
	VASIL	**16.9**	**15.5**	**28.8**	**17.5**	**13.9**	**8.8**
Sport	UltraQuery	18.5	12.6	50.1	10.1	11.6	8.2
	VASIL	**20.9**	**16.3**	**53.2**	**11.7**	**11.7**	**11.5**
Taxonomy	UltraQuery	18.4	12.2	39.4	13.4	17.7	9.5
	VASIL	**29.6**	**19.6**	**50.6**	**19.0**	**30.0**	**28.8**

G Detailed Results on Diverse Query Structures

We report detailed results on diverse query structures (§ 5.2), including additional baselines and results broken down by query type. The baselines include Query2Box (Ren et al., 2020), BetaE (Ren & Leskovec, 2020), ConE (Zhang et al., 2021), CQD-Beam (Arakelyan et al., 2021), CQD-CO (Arakelyan et al., 2021), BiQE (Kotnis et al., 2021), LogicE (Luus et al., 2021), LMPNN (Wang et al., 2023), FIT (Yin et al., 2024), NS3(M) (Fei et al., 2026), NS3(J) (Fei et al., 2026), UltraQuery (Galkin et al., 2024b), Query2Box (DAGE) (He et al., 2025), BetaE (DAGE) (He et al., 2025), and ConE (DAGE) (He et al., 2025).

Following the standard practice, we report MRR (%) for the DAG benchmarks (He et al., 2025) and EFO_k-CQA queries with one free variable (Yin et al., 2025), and Hits@10 (%) for the EFO_3-Small benchmarks (Fei et al., 2026) and EFO_k-CQA queries with two free variables. For queries with multiple free variables, we rank candidate tuples by the sum of the log probabilities assigned to their constituent values across free variables. "–" denotes an unavailable result.

Path Decomposition for UltraQuery. All UltraQuery results in these experimental settings use path decomposition (‡). For each free variable, we enumerate all simple paths from the anchor entities to that variable in the original query and evaluate each path independently using UltraQuery. Predictions from multiple paths are then combined using UltraQuery's original intersection operator.

G.1 Results on DAG Benchmarks

Table 31 reports results on four DAG benchmarks (He et al., 2025): FB15k-237-DAG-Easy, NELL-DAG-Easy, FB15k-237-DAG-Hard, and NELL-DAG-Hard. VASIL achieves the highest average performance in all four settings, demonstrating its ability to model dependencies between intermediate variables without decomposing the query into independent paths.

G.2 Results on EFO_k-CQA Benchmarks

Tables 32 and 33 report single-free-variable results, while Table 34 reports two-free-variable results on the EFO_k-CQA benchmarks (Yin et al., 2025): FB15k-237-EFO_k-CQA and NELL-EFO_k-CQA. Following prior work, the Avg score is computed as a micro average across query counts. The grouped columns further distinguish e, the number of existential variables, and three query-graph structures. SDAG denotes a simple directed acyclic graph, Multi denotes a multi-edge structure in which multiple relation edges can connect the same pair of terms, and Cyclic denotes a query graph containing a cycle.

VASIL achieves the highest overall average on both benchmarks, demonstrating its ability to handle queries with multiple edges and cycles while preserving dependencies between variables.

G.3 Results on EFO_3-Small Benchmarks

Table 35 reports results on two EFO_3-Small benchmarks (Fei et al., 2026): FB15k-237-EFO_3 and NELL-EFO_3. VASIL remains competitive with NS3(J) on queries with two free variables and achieves the highest average performance on queries with three free variables for both KGs. These results suggest that VASIL can effectively reason over queries with multiple free variables by accounting for dependencies among them.

Table 31: MRR(%) results on four DAG benchmarks (He et al., 2025). avg. averages all six query types, while avg_p averages the five positive types. ‡ indicates that the path decomposition is applied.

Model	avg	avg_p	p2s	p3s	p2sp	i2s	u2s	in2s
NELL-DAG-Easy								
Query2Box	-	21.2	20.5	34.0	0.1	23.3	28.2	-
Query2Box (DAGE)	-	42.4	37.6	49.4	41.7	40.6	42.8	-
BetaE	19.8	18.3	15.3	28.8	13.1	17.7	16.7	27.3
BetaE (DAGE)	40.2	41.3	36.9	57.1	35.0	39.9	37.8	34.7
ConE	26.7	26.5	23.6	39.4	19.5	25.3	25.0	27.7
ConE (DAGE)	39.1	40.2	33.5	57.4	38.4	37.9	33.7	33.9
CQD-Beam	-	22.3	22.6	44.8	17.5	24.9	1.6	-
BiQE	-	-	20.7	45.4	20.8	28.4	-	-
UltraQuery‡	37.6	38.8	35.3	56.7	24.2	39.1	38.6	31.8
VASIL	**51.1**	**52.1**	**46.0**	**61.3**	**55.2**	**52.4**	**45.6**	**46.2**
NELL-DAG-Hard								
Query2Box	-	6.7	7.5	5.2	0.1	8.5	12.3	-
Query2Box (DAGE)	-	24.2	25.4	20.1	21.3	24.9	29.2	-
BetaE	15.5	13.0	14.4	16.2	8.0	13.3	13.0	28.2
BetaE (DAGE)	27.5	26.3	27.7	32.3	16.4	26.1	29.2	33.6
ConE	20.4	18.6	20.3	18.9	12.0	19.6	22.3	29.5
ConE (DAGE)	31.2	30.7	30.7	**38.4**	24.8	28.4	31.1	34.1
CQD-Beam	-	11.9	12.3	21.7	10.8	12.7	2.2	-
BiQE	-	-	13.6	19.5	12.0	15.4	-	-
UltraQuery‡	27.1	27.0	27.5	34.7	15.9	27.4	29.7	27.3
VASIL	**36.9**	**34.9**	**34.5**	36.8	**32.1**	**36.2**	**35.0**	**46.7**
FB15k-237-DAG-Easy								
Query2Box	-	7.8	6.8	11.6	9.3	6.5	4.6	-
Query2Box (DAGE)	-	8.5	**7.4**	12.6	10.1	7.3	**5.0**	-
BetaE	5.5	5.6	4.8	8.2	7.5	5.0	2.7	4.5
BetaE (DAGE)	7.6	7.8	6.3	12.1	9.6	6.7	4.1	6.6
ConE	6.1	6.3	4.9	9.2	8.9	5.5	3.1	4.8
ConE (DAGE)	8.1	8.5	6.9	11.7	12.4	6.9	4.8	6.1
CQD-Beam	-	4.5	4.3	8.7	5.6	3.7	0.1	-
BiQE	-	-	2.1	2.7	4.1	5.7	-	-
UltraQuery‡	7.2	7.6	5.3	13.2	10.9	5.8	2.8	5.1
VASIL	**8.5**	**8.6**	6.1	**13.9**	**12.9**	**7.8**	2.5	**7.6**
FB15k-237-DAG-Hard								
Query2Box	-	4.5	4.3	2.6	7.2	4.6	3.6	-
Query2Box (DAGE)	-	5.0	4.8	2.8	7.9	5.3	**4.4**	-
BetaE	3.7	3.6	3.6	1.6	6.4	3.9	2.4	4.3
BetaE (DAGE)	4.9	4.6	4.9	1.7	8.3	4.8	3.5	6.1
ConE	4.2	4.1	3.5	2.3	7.4	4.2	2.9	4.7
ConE (DAGE)	5.1	5.1	4.8	2.1	9.7	4.8	4.2	5.3
CQD-Beam	-	2.3	2.7	1.6	4.6	2.4	0.1	-
BiQE	-	-	3.1	2.0	3.4	1.9	-	-
UltraQuery‡	7.9	8.3	5.5	15.9	11.3	6.0	2.8	5.7
VASIL	**9.4**	**9.6**	**6.1**	**18.1**	**13.1**	**8.2**	2.5	**8.3**

Table 32: MRR(%) results of queries with one free variable on FB15k-237-EFO$_k$-CQA (Yin et al., 2025). We denote e as the number of existential variables and c as the number of constant entities. SDAG represents the Simple Directed Acyclic Graph, Multi for multigraph, and Cyclic for the cyclic graph. Avg(c) and Avg(e) are the average scores of queries with the number of constant entities / existential variables fixed. ‡ indicates that the path decomposition is applied.

Model	c	e=0	e=1		e=2			Avg(c)	Avg
		SDAG	SDAG	Multi	SDAG	Multi	Cyclic		
BetaE	1	16.2	17.9	10.9	10.6	8.5	16.5	11.1	20.7
	2	35.6	20.2	19.1	15.7	15.7	27.1	17.8	
	3	53.3	32.4	33.1	21.7	21.6	37.4	24.8	
	Avg(e)	37.4	25.7	23.5	18.8	18.1	30.5		
LogicE	1	17.4	19.0	11.5	11.0	8.5	16.8	11.5	21.3
	2	36.7	21.2	19.8	16.5	16.1	27.3	18.4	
	3	55.5	34.6	34.5	22.3	22.0	37.5	25.4	
	Avg(e)	38.9	27.3	24.5	19.4	18.5	30.6		
ConE	1	18.6	19.9	11.8	11.4	9.3	18.7	12.3	23.1
	2	39.1	22.4	20.8	18.1	17.6	30.7	20.1	
	3	58.8	36.4	37.0	24.6	23.8	41.7	27.6	
	Avg(e)	41.4	28.7	26.0	21.3	20.1	34.2		
CQD-CO	1	**22.2**	19.5	9.0	9.2	6.4	15.6	10.0	21.9
	2	35.3	20.1	19.1	16.4	16.2	27.6	18.4	
	3	40.3	32.9	34.3	24.4	24.0	40.2	26.8	
	Avg(e)	33.9	26.2	23.7	20.5	19.4	31.9		
LMPNN	1	20.5	21.4	11.2	11.6	8.7	17.0	11.9	20.5
	2	42.0	22.6	18.5	16.5	14.9	26.5	17.9	
	3	62.3	35.9	31.6	22.1	19.8	35.5	24.0	
	Avg(e)	44.2	28.8	22.7	19.4	16.9	29.4		
FIT	1	**22.2**	25.0	**17.4**	13.9	11.7	23.3	15.6	30.3
	2	45.3	29.6	28.5	23.8	24.3	35.5	26.5	
	3	64.5	44.8	45.4	33.3	33.5	44.4	36.2	
	Avg(e)	46.7	36.2	33.6	28.6	27.9	37.9		
UltraQuery‡	1	16.4	18.2	9.4	11.0	9.1	20.5	11.8	27.2
	2	42.7	24.8	23.2	19.8	20.6	36.1	23.0	
	3	66.5	40.0	43.7	29.3	30.3	48.9	33.5	
	Avg(e)	45.1	31.0	29.2	24.6	24.6	39.9		
VASIL	1	22.1	**26.5**	16.9	**16.9**	**13.6**	**25.2**	**17.3**	**33.0**
	2	**46.6**	**31.3**	**30.0**	**26.0**	**26.1**	**40.2**	**28.6**	
	3	**69.1**	**47.1**	**48.7**	**36.1**	**35.9**	**54.1**	**39.4**	
	Avg(e)	**48.9**	**38.1**	**35.4**	**31.1**	**30.1**	**44.7**		

Table 33: MRR(%) results of queries with one free variable on NELL995-EFO$_k$-CQA (Yin et al., 2025). The notation of e, c, SDAG, Multi, Cyclic, $\mathrm{Avg}(c)$ and $\mathrm{Avg}(e)$ are the same as Table 32. ‡ indicates that the path decomposition is applied.

Model	c	e=0	e=1		e=2			$\mathrm{Avg}(c)$	Avg
		SDAG	SDAG	Multi	SDAG	Multi	Cyclic		
BetaE	1	13.9	26.4	35.0	8.6	14.9	19.1	17.5	33.6
	2	58.8	31.5	43.8	22.4	30.6	34.7	30.7	
	3	78.8	48.6	58.3	29.6	39.0	47.0	39.5	
	$\mathrm{Avg}(e)$	53.1	38.5	48.3	25.2	33.3	38.2		
LogicE	1	18.3	29.2	39.6	12.1	19.0	20.4	21.1	36.9
	2	63.5	34.4	47.3	26.4	34.0	37.6	34.2	
	3	79.6	51.2	59.3	33.1	42.2	50.1	42.6	
	$\mathrm{Avg}(e)$	56.3	41.3	50.9	28.8	36.7	41.0		
ConE	1	16.7	26.9	36.6	11.1	16.9	22.3	19.6	36.6
	2	60.5	33.6	46.6	25.3	33.1	40.1	33.6	
	3	79.9	50.6	59.2	33.2	42.2	52.6	42.8	
	$\mathrm{Avg}(e)$	54.9	40.3	50.0	28.4	36.2	43.4		
CQD-CO	1	22.3	30.6	37.3	13.3	17.9	20.7	20.9	38.2
	2	59.8	34.0	45.2	28.8	35.4	38.9	35.3	
	3	62.7	48.8	59.9	36.4	44.1	52.6	44.3	
	$\mathrm{Avg}(e)$	50.1	40.2	49.9	31.6	38.1	42.7		
LMPNN	1	20.7	29.8	33.3	13.4	16.5	21.8	19.8	35.1
	2	63.5	35.4	43.3	27.0	30.2	37.6	32.3	
	3	80.8	50.7	56.0	33.6	39.2	47.6	40.7	
	$\mathrm{Avg}(e)$	57.4	41.5	46.7	29.4	33.6	40.0		
FIT	1	–	–	–	–	–	–	–	–
	2	–	–	–	–	–	–	–	
	3	–	–	–	–	–	–	–	
	$\mathrm{Avg}(e)$	–	–	–	–	–	–		
UltraQuery‡	1	12.6	21.9	31.6	9.7	17.0	20.0	17.9	35.6
	2	57.9	30.8	43.0	24.0	32.9	40.3	32.6	
	3	78.8	47.3	58.2	32.5	41.4	52.1	41.9	
	$\mathrm{Avg}(e)$	52.4	36.8	47.2	27.4	35.7	42.8		
VASIL	1	**24.3**	**36.7**	**42.9**	**19.1**	**24.8**	**27.1**	**27.1**	**44.9**
	2	**67.5**	**42.0**	**51.4**	**35.3**	**42.1**	**47.4**	**42.3**	
	3	**83.3**	**56.9**	**64.6**	**42.5**	**50.5**	**60.7**	**50.9**	
	$\mathrm{Avg}(e)$	**60.8**	**47.9**	**55.4**	**37.8**	**44.7**	**50.7**		

Table 34: Joint Hits@10(%) for EFO_k-CQA queries with two free variables (Yin et al., 2025). For NELL, the number of constants is fixed at two, while for FB15k-237, results are averaged across different numbers of constants. The notation follows Table 32. ‡ indicates that the path decomposition is applied.

Model	$e=0$		$e=1$			$e=2$			Avg
	SDAG	Multi	SDAG	Multi	Cyclic	SDAG	Multi	Cyclic	
FB15k-237-EFO$_k$-CQA									
BetaE	5.9	5.7	5.1	4.9	12.3	2.2	2.1	6.6	4.6
LogicE	6.4	6.3	5.7	5.4	13.6	2.6	2.3	7.0	5.0
ConE	6.7	7.2	6.0	5.8	14.2	2.7	2.4	7.6	5.4
CQD-CO	6.4	6.9	6.3	6.4	13.9	3.0	2.7	7.6	5.6
LMPNN	6.8	7.5	5.9	5.6	13.1	2.6	2.3	6.4	5.0
FIT	6.7	7.0	7.1	7.2	11.7	3.2	3.4	7.1	5.9
NS3(M)	6.9	7.6	7.4	7.7	14.6	3.3	3.3	9.0	6.6
NS3(J)	–	–	–	–	–	–	–	–	–
UltraQuery‡	15.7	16.9	14.9	16.2	25.0	4.9	5.2	15.6	12.2
VASIL	**22.9**	**22.0**	**22.0**	**22.7**	**31.0**	**8.1**	**8.4**	**20.9**	**17.3**
NELL-EFO$_k$-CQA (c=2)									
BetaE	19.2	31.8	21.2	26.5	21.7	13.8	17.5	18.5	18.8
LogicE	21.1	32.8	25.4	30.5	23.3	18.0	21.5	20.5	22.3
ConE	17.0	30.9	19.3	25.0	24.9	12.9	17.2	20.3	18.8
CQD-CO	23.0	38.0	29.7	34.2	26.4	21.4	25.4	24.0	26.0
LMPNN	23.4	36.4	25.5	29.4	24.0	16.6	19.7	21.5	21.5
FIT	–	–	–	–	–	–	–	–	–
NS3(M)	–	–	–	–	–	–	–	–	–
NS3(J)	–	–	–	–	–	–	–	–	–
UltraQuery‡	23.1	38.0	22.9	34.0	25.5	16.3	23.5	26.8	24.8
VASIL	**40.8**	**53.7**	**40.6**	**48.4**	**37.3**	**31.2**	**38.6**	**39.3**	**39.4**

Table 35: Hits@10(%) results on EFO_3-Small (Fei et al., 2026), grouped by dataset and the number k of free variables. Avg is the macro average across query types. ‡ indicates that the path decomposition is applied.

Model	fpn	fp	fpm	fd	fdm	fcn	fc	Avg.
FB15k-237-EFO$_3$, $k = 2$								
BetaE	4.1	9.0	6.7	3.7	2.2	17.4	3.9	6.7
LogicE	3.3	9.5	7.0	5.8	2.8	18.0	3.7	7.2
ConE	4.4	9.3	7.6	5.3	2.8	17.7	9.2	8.0
LMPNN	2.6	9.7	8.1	4.0	1.9	15.3	6.9	6.9
CQD-CO	1.8	9.5	9.2	4.1	2.2	16.1	4.0	6.7
FIT	3.3	9.7	7.7	5.7	3.2	14.9	10.1	7.8
NS3 (M)	4.5	14.4	14.1	6.0	3.9	13.7	6.0	8.9
NS3 (J)	**11.4**	**28.5**	**28.3**	**16.2**	**10.0**	**36.5**	19.2	**21.4**
UltraQuery‡	2.8	21.6	15.5	9.5	4.9	16.6	31.3	14.6
VASIL	9.9	**28.5**	20.0	14.8	8.8	20.5	**39.2**	20.3
FB15k-237-EFO$_3$, $k = 3$								
BetaE	1.9	4.0	4.4	2.4	2.3	1.6	0.3	2.4
LogicE	1.9	5.2	5.5	2.6	2.3	1.2	0.5	2.7
ConE	2.6	5.4	5.4	2.7	2.7	0.5	0.3	2.8
LMPNN	1.7	6.0	7.3	2.2	2.5	7.3	1.9	4.1
CQD-CO	0.7	8.3	8.6	2.2	2.3	9.9	0.0	4.6
FIT	2.9	8.5	6.7	3.1	4.2	5.9	3.5	5.0
NS3 (M)	5.2	14.9	17.9	3.7	5.2	7.3	3.1	8.2
NS3 (J)	**8.4**	19.4	**26.2**	8.0	8.4	**17.0**	8.9	13.8
UltraQuery‡	1.6	17.9	18.3	5.5	6.0	7.8	15.2	10.3
VASIL	5.0	**25.8**	22.8	**9.5**	**8.9**	10.4	**20.6**	**14.7**
NELL-EFO$_3$, $k = 2$								
BetaE	6.6	10.0	22.2	11.8	9.8	0.6	30.6	13.1
LogicE	6.8	10.0	21.5	16.8	11.2	0.7	31.9	14.1
ConE	6.8	9.4	21.5	12.6	11.1	12.0	32.4	15.1
LMPNN	4.4	10.7	17.6	12.1	8.9	14.6	33.3	14.5
CQD-CO	4.2	9.0	26.4	11.5	8.7	8.7	21.8	12.9
FIT	8.8	11.9	24.0	17.1	11.6	12.9	29.1	16.6
NS3 (M)	9.5	13.8	30.5	16.9	13.5	8.5	24.4	16.7
NS3 (J)	**15.8**	**32.8**	**53.9**	**26.9**	**21.9**	24.2	50.2	**32.2**
UltraQuery‡	3.6	24.0	43.4	6.8	12.5	21.7	38.6	21.5
VASIL	**15.8**	31.4	51.1	24.1	21.0	**27.4**	**52.7**	31.9
NELL-EFO$_3$, $k = 3$								
BetaE	4.4	12.0	23.9	12.5	21.3	0.4	3.0	11.1
LogicE	5.1	13.3	25.3	17.5	28.1	1.7	2.8	13.4
ConE	4.8	11.6	23.2	12.1	20.9	7.7	2.6	11.8
LMPNN	4.6	13.9	22.1	12.3	19.6	5.7	12.2	12.9
CQD-CO	0.1	16.3	27.9	14.1	24.3	0.0	12.4	13.6
FIT	5.7	16.4	27.2	18.6	31.2	7.7	9.5	16.5
NS3 (M)	8.9	20.1	34.4	19.5	30.9	4.8	8.4	18.1
NS3 (J)	**14.7**	25.3	41.0	21.4	31.6	11.7	21.0	23.8
UltraQuery‡	2.7	19.5	36.0	9.2	16.7	8.9	16.3	15.6
VASIL	8.8	**32.0**	**49.4**	**24.3**	**36.7**	**13.2**	**25.8**	**27.2**

Table 36: Average F1-scores of baseline methods on two QAC benchmarks (Olejniczak et al., 2025): FB15k-237-QAC and NELL-QAC. ‡ indicates that the path decomposition is applied.

		Small						Large		
		2p	**3p**	**pi**	**ip**	**inp**	**pin**	**3-hub**	**4-hub**	**5-hub**
FB15k-237-QAC	SQL	66.0	61.7	70.0	67.0	78.1	74.8	37.0	32.2	35.3
	QTO	67.1	64.4	70.8	67.7	78.5	75.9	–	–	–
	FIT	68.0	65.1	71.4	67.8	78.6	76.7	–	–	–
	GNN-QE	77.1	73.5	80.1	81.2	**79.0**	77.0	–	–	–
	UltraQuery	75.2	68.9	79.8	76.8	75.9	78.6	63.6‡	63.2‡	62.0‡
	AnyCQ	75.8	71.3	82.1	78.8	76.7	75.7	52.4	49.9	51.9
	VASIL	**77.3**	**74.1**	**83.8**	**82.7**	78.4	**78.7**	**73.6**	**73.9**	**76.1**
NELL-QAC	SQL	60.9	58.8	63.3	59.6	**76.7**	74.9	33.9	31.4	27.0
	QTO	63.9	64.1	68.2	61.7	74.5	75.3	–	–	–
	FIT	63.9	64.6	68.4	61.7	73.6	75.7	–	–	–
	GNN-QE	70.4	69.7	71.2	72.1	72.2	74.9	–	–	–
	UltraQuery	66.3	65.6	73.2	71.1	73.2	73.4	57.7‡	56.6‡	53.5‡
	AnyCQ	**76.2**	**72.3**	79.0	75.4	**76.7**	75.3	57.2	52.6	58.2
	VASIL	73.9	70.6	**79.7**	**78.2**	76.3	**75.8**	**63.6**	**62.8**	**66.7**

H Detailed Results on Query Answer Classification

We report detailed results on four QAC benchmarks (Olejniczak et al., 2025), FB15k-237-QAC-Small, FB15k-237-QAC-Large, NELL-QAC-Small, and NELL-QAC-Large (§ 5.3). The *Small* benchmarks comprise 2p, 3p, pi, ip, inp, and pin queries, while the *Large* benchmarks comprise 3-hub, 4-hub, and 5-hub queries. The baselines include SQL (Olejniczak et al., 2025), QTO (Bai et al., 2023), FIT (Yin et al., 2024), GNN-QE (Zhu et al., 2022), UltraQuery (Galkin et al., 2024b), and AnyCQ (Olejniczak et al., 2025).

Because VASIL assigns a score to each candidate entity, we select a decision threshold on the validation set and apply the same threshold to the test set. For the Large queries, we evaluate UltraQuery using the path-decomposition procedure described in Appendix G. We then select its decision threshold on the validation set and apply the same threshold at test time. As shown in Table 36, although VASIL is not trained specifically for the QAC task, it achieves the best F1-score on most query types. The gains are particularly pronounced on the Large queries, where VASIL consistently outperforms the baselines across the 3-hub, 4-hub, and 5-hub structures.