KNOSSOS AND ARIADNE: BENCHMARKING AND LEARNING COMPLETE DIAGRAM TOPOLOGY EXTRACTION WITH VISION–LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Structural diagrams are widely used to represent complex systems and relational information across scientific, engineering, procedural, and spatial domains. Recent vision–language models (VLMs) have become increasingly capable of recognizing diagram elements and reasoning about their content, while complete diagram topology extraction remains comparatively underexplored. In this paper, we study **diagram-to-graph topology extraction**: extracting all diagram entities and the complete relations among them. To enable large-scale supervised training and systematic evaluation of this task, we introduce **Knossos**, a benchmark of 19,200 diagrams across six diverse domains, with 245,179 nodes and 439,740 edges. Its symbolic generation process provides exact alignment between rendered diagrams and annotations of complete topology, relation types, and connector geometry. To address the modeling challenge of complete topology extraction, we also present **Ariadne**, a structured framework that decomposes the task into node inventory extraction and source-conditioned edge prediction. Extensive experiments show that training on Knossos substantially improves complete topology extraction in smaller open-source VLMs. Ariadne further improves over one-step extraction under matched supervision, demonstrating the additional benefit of structured decomposition. It achieves the highest average Edge F1 among the evaluated methods on Knossos, while both backbone variants also improve over their unadapted counterparts on the real-world external benchmark. The benchmark and code are available on `https://anonymous.4open.science/r/knossos-ariadne-2C4C/`

1 INTRODUCTION

Structural diagrams are essential for representing complex structured and relational information across science, engineering, education, and technical domains. Recent advances in large vision–language models (VLMs) have broadened the scope of diagram understanding, spanning scientific and chart question answering, visual-graph reasoning, and specialized engineering-diagram analysis (Masry et al., 2022; Zhu et al., 2025; Zhao et al., 2025; Li et al., 2025a). These advances have improved the ability of VLMs to recognize diagram elements and reason about their content. However, structural diagrams often contain many interacting entities and densely routed relations, making complete structural understanding substantially more challenging than recognizing individual elements (Hou et al., 2024; Hu et al., 2026).

Despite this progress, existing diagram-understanding tasks typically evaluate only selected entities and relations, for example through question answering or localized reasoning (Lu et al., 2022; Li et al., 2024b). Such evaluations do not require extracting the complete topology: a model may answer individual questions correctly while still omitting, reversing, or hallucinating other relations (Zhu et al., 2025; Hu et al., 2026). We therefore study **diagram-to-graph topology extraction**, which aims to extract the complete graph topology encoded in a diagram, where diagram entities correspond to nodes and their relations to edges. Once the complete topology is available, many structure-centric question answering and reasoning tasks become simpler by directly analyzing the extracted textual graph representation.

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.

Complete diagram-to-graph topology extraction, however, faces two key challenges: the scarcity of large-scale benchmarks that support both training and evaluation, and the difficulty of modeling complete diagram topology. On the data side, constructing complete topology supervision requires exhaustively labeling all diagram entities and relations, making large-scale dataset construction far more demanding than question-answer supervision (Guo et al., 2026; Hu et al., 2026). On the modeling side, VLMs must identify all relevant entities and extract their complete relations from potentially dense diagrams. As diagram complexity grows, the resulting graph serialization becomes longer and harder to generate consistently, increasing the likelihood of omissions, hallucinations, and structural inconsistencies (Jiang et al., 2026; Zheng et al., 2025).

To address these challenges, we introduce two complementary contributions: **Knossos**, a large-scale benchmark for supervised topology learning and evaluation, and **Ariadne**, a structured framework for complete topology extraction[1]. Knossos is generated from symbolic topology specifications, ensuring exact alignment between rendered diagrams and complete topology annotations by construction. This pipeline scales to 19,200 diagrams across six domains while varying topology structures, layouts, connector routes, and visual assets. Ariadne addresses the modeling challenge by first extracting a complete node inventory and then decomposing topology extraction into source-group edge predictions, replacing a single long graph serialization with shorter structured generations that are deterministically assembled into the complete topology. We train Ariadne on the Knossos training split using its complete topology annotations as supervision. Extensive experiments show that Knossos provides effective large-scale supervision for complete topology extraction, while Ariadne further improves over matched one-step models through structured source-conditioned prediction. Evaluation on an external real-world benchmark further demonstrates cross-domain generalization beyond the synthetic training distribution. Our contributions are summarized as follows:

- We establish a large-scale supervised setting for diagram-to-graph topology extraction, moving beyond isolated entities or relations toward complete diagram topology extraction with vision–language models.
- We introduce Knossos, a large-scale benchmark containing 19,200 diagrams across six diverse domains, with complete annotations of nodes, directed and typed relations, and connector geometry for both training and evaluation.
- We present Ariadne, a structured framework that decomposes complete topology extraction into node inventory construction and short source-conditioned edge predictions.
- We conduct extensive experiments across Knossos and an external real-world benchmark, demonstrating the effectiveness of large-scale topology supervision, the benefit of structured decomposition, and cross-domain generalization.

2 Task Formulation and the Knossos Benchmark

2.1 Diagram-to-Graph Topology Extraction

Given a diagram image I, the task is to extract its topology $G = (V, E)$, where V is the set of diagram nodes and E is a multiset of directed edges. Each node $v_i \in V$ has a diagram-local identifier and a visible text label, and each edge $e = (s, t) \in E$ connects source node s to target node t. When relation semantics are defined, an edge is additionally assigned a type r, yielding a typed edge (s, t, r). Multiple typed edges may share the same endpoint pair, allowing distinct relations between the same two nodes to be preserved.

2.2 Knossos: Multi-Domain Synthetic Diagram Benchmark

We introduce Knossos, a large-scale synthetic diagram benchmark for supervised topology extraction. Knossos contains 19,200 diagrams spanning six complementary domains: Food Web, Network, Workflow, Natural Process, Circuit, and Map Route, with representative examples shown in Figure 1(a). Together, these domains cover scientific, engineering, procedural, and spatial structures

[1]The names draw from the Greek myth of the Labyrinth at Knossos, where Ariadne gave Theseus a thread to navigate its passages. Here, Knossos evokes the benchmark's intricate structures, while Ariadne provides the thread for extracting their underlying topology.

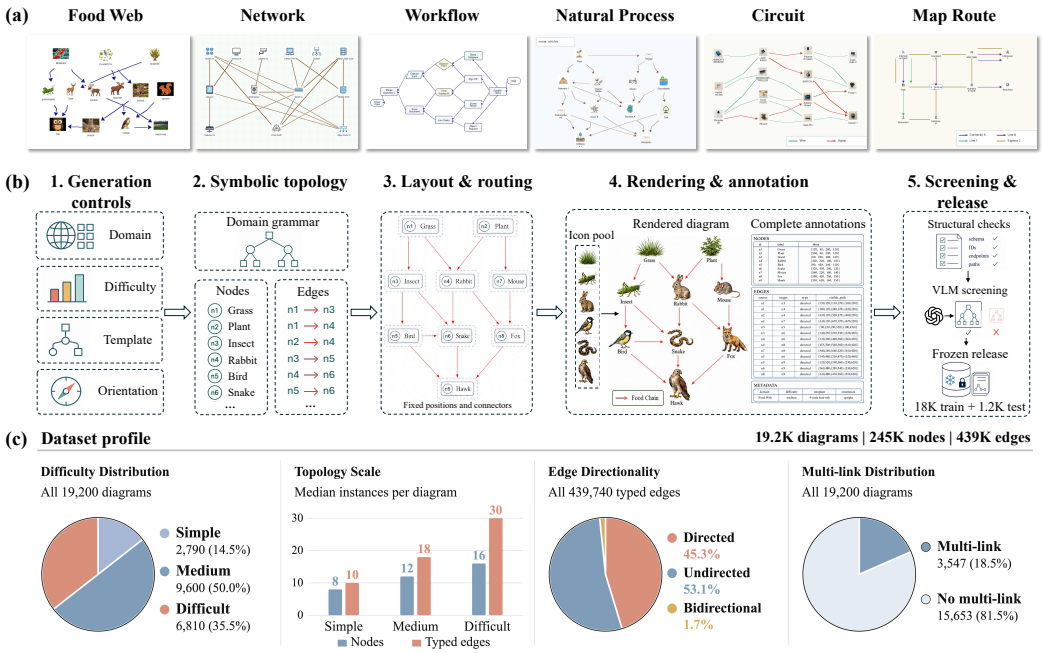

Figure 1: **Overview of the Knossos benchmark.** (a) Representative diagrams from the six domains. (b) The generation pipeline samples generation controls, instantiates a symbolic topology, places nodes and routes edges, and renders the diagram together with aligned node, typed-edge, and geometry annotations. Structural validation and VLM-based screening produce the frozen training and test release. (c) Dataset statistics summarize difficulty, topology scale, edge directionality, and multi-link prevalence across 19,200 diagrams, 245,179 nodes, and 439,740 typed edges.

while introducing diverse extraction challenges, including dense directed connectivity, unarrowed connections, branching and merging structures, legend-defined relation types, and multiple typed edges between common endpoint pairs. The dataset contains 245,179 node instances and 439,740 edge instances, with the nodes drawn from 310 domain-specific canonical concepts. The frozen release is divided into 18,000 training images and 1,200 held-out test images. Each diagram is paired with its complete graph topology, relation types when defined by the diagram, and geometric annotations for both nodes and edges, including node bounding boxes and connector paths. These annotations support both topology extraction and geometric grounding.

2.3 Symbolic Generation Pipeline

Figure 1(b) summarizes the Knossos generation pipeline. The generator samples the domain, difficulty, structural template, and orientation, instantiates nodes from the corresponding domain vocabulary, and constructs edges using the domain-specific grammar and selected template. Once the topology is fixed, the layout stage assigns node positions and bounding boxes, after which connectors are routed between the corresponding node boundaries. Depending on the domain and template, connectors may follow direct, bent, or orthogonal paths, and the routed connector paths are retained as geometric traces associated with the corresponding edges. Finally, the renderer combines the fixed topology, layout, connector geometry, and visual assets to produce the diagram image. Complete annotations are serialized from the same finalized node and edge objects, including node labels and bounding boxes, graph topology, relation types when defined, and connector traces. This shared representation keeps the rendered diagram and its annotations aligned by construction. Detailed domain grammars and rendering configurations are provided in Appendix B.

2.4 Structural and Visual Diversity

Structural diversity is introduced through parameterized, domain-specific template families rather than unconstrained graph sampling. These templates cover hierarchical, clustered, branching, lay-

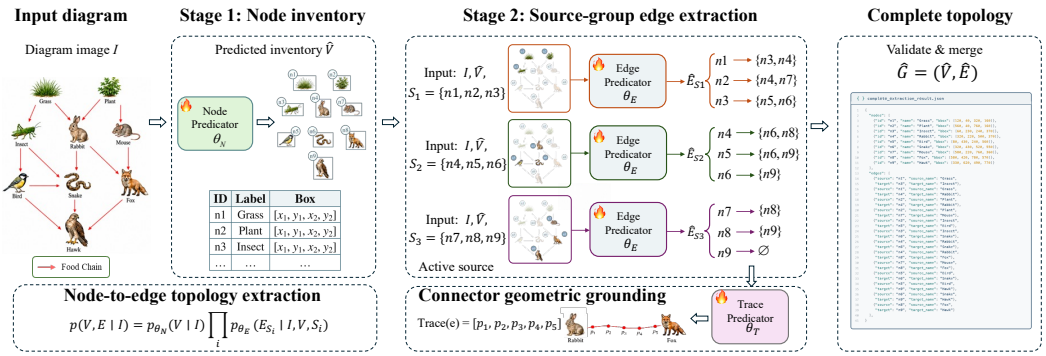

Figure 2: **Overview of Ariadne.** Ariadne decomposes topology extraction into node inventory prediction and source-group edge extraction. Edge prediction is restricted to active sources within each group while retaining all predicted nodes as candidate targets. The group-wise edge multisets are validated and merged into the final graph, with node bounding boxes and separately predicted connector traces providing auxiliary geometric grounding.

ered, and spatial structures. Seeded rules vary graph size, density, depth, branching, cross-links, relation types, and parallel edges within each grammar. Difficulty controls both topology and presentation: harder samples generally contain more nodes and edges, more layers and cross-links, and longer, more intersecting connector paths. Training and test instances are generated with disjoint random seeds for template instantiation, topology sampling, and layout variation.

Visual diversity is introduced through offline-prepared, label-matched asset pools, with each eligible canonical concept associated with multiple visual styles and variants; nodes without image assets are rendered programmatically using role-specific shapes. At rendering time, assets are sampled only from the pool corresponding to the selected node concept. Seeded presentation choices further vary, where applicable, orientation, label casing, background, connector appearance, arrowhead size, and legend placement. Visible labels are drawn by the renderer rather than by the asset-generation model, so visual variation does not alter the underlying node identity or graph topology. Implementation details are provided in Appendix B.3.

2.5 Quality Control and Dataset Release

As summarized in Figure 1(b), each generated sample undergoes structural validation followed by VLM-based visual screening. Structural checks focus on basic annotation and graph consistency, including format validity, completeness, and duplicate detection. A separate VLM-based screening step filters out samples with clearly unreadable text or visually unclear diagram elements. The final release is constructed by stratified selection across domains and difficulty levels, replacing failed samples with valid candidates from the same stratum. The frozen benchmark contains 18,000 training images and 1,200 held-out test images, with 3,000 training and 200 test examples per domain. Figure 1(c) summarizes the resulting benchmark statistics. Additional screening and release details are provided in Appendix B.5.

3 Ariadne

3.1 Node-to-Edge Topology Extraction

Figure 2 illustrates the overall framework. Following the topology formulation in Section 2.1, Ariadne decomposes topology extraction into node inventory prediction followed by source-conditioned edge prediction. The predicted node inventory provides a fixed reference vocabulary and is partitioned into K source groups $\mathcal{S}(V) = \{S_1, \ldots, S_K\}$ with target group size g. For each S_i, the edge predictor outputs only the typed edges originating from nodes in S_i. The group-wise edge multisets are then merged deterministically to form the complete graph.

We model this decomposition as

$$p_\theta(V, E \mid I) = p_{\theta_N}(V \mid I) \prod_{i=1}^{K} p_{\theta_E}(E_{S_i} \mid I, V, S_i), \tag{1}$$

where E_{S_i} is the multiset of directed edges originating from sources in S_i. At inference, the predicted group-wise edge multisets are merged as

$$\hat{E} = \biguplus_{i=1}^{K} \hat{E}_{S_i}, \qquad \hat{G} = (\hat{V}, \hat{E}). \tag{2}$$

This decomposition replaces a single long graph serialization with shorter, source-conditioned generations and, under our serialization setting, removes cross-group ordering ambiguity. Under a position-sensitive decoding assumption, the shorter generation horizon may also reduce error accumulation at later output positions. We formalize these properties and their assumptions in Appendix C.

3.2 Two-Stage Topology Prediction

Stage 1: Node inventory. The node predictor first identifies the diagram nodes and their visible text labels, producing the predicted node inventory $\hat{V}$. Each node is localized with a bounding box $b_v = [x_1, y_1, x_2, y_2]$ for geometric grounding, while a diagram-local identifier is included only to provide a stable reference in later stages. The output contains only node records and no edge or connector predictions. At test time, the predicted node inventory, rather than the ground-truth inventory, is passed to the next stage.

Stage 2: Source-group edge extraction. Given $\hat{V}$, Ariadne processes the source groups $\{S_i\}_{i=1}^{K}$ independently. Each query contains the diagram, the complete predicted node inventory, and the active sources in S_i. The edge predictor may emit relations only from these active sources, while any node in $\hat{V}$ remains available as a target. For each active source, it emits one `CHECK_NODE` block:

Example Stage 2 output for one active source

```
<CHECK_NODE id="n4" name="Sensor">
  <EDGE source="n4" source_name="Sensor"
        target="n6" target_name="Controller"
        type="signal"/>
  <EDGE source="n4" source_name="Sensor"
        target="n8" target_name="Ground"
        type="ground"/>
</CHECK_NODE>
```

Relation types are predicted only when defined by the diagram legend; otherwise the type is left empty. Directionality is encoded by the ordered source–target pair, with undirected or bidirectional connectors represented by reciprocal edges. Multiple typed relations between the same endpoints remain distinct multiset elements, and an empty `CHECK_NODE` block denotes a source with no outgoing edge. Finally, all source-group predictions are validated and merged into $\hat{E}$.

3.3 Geometric Grounding

Beyond symbolic topology, Ariadne associates each graph primitive with explicit geometric evidence. Nodes are localized by the bounding boxes predicted in Stage 1, while each predicted edge is grounded by an ordered connector trace generated after the Stage 2 edge set is fixed. For each edge, the trace predictor outputs five points sampled along its visible connector from source to target. Training targets are obtained by uniformly sampling the final rendered connector path by arc length, so the endpoints localize the connector boundaries and the intermediate points capture its routed geometry. Reciprocal edges use the same connector path in reverse order. The trace serves

only as auxiliary geometric grounding and does not modify the predicted graph topology. Detailed results are provided in Appendix D.2.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Implementation and evaluation protocol. We instantiate Ariadne with GLM-4.6V-Flash (Hong et al., 2025) and Qwen3-VL-8B (Bai et al., 2025) as two vision–language backbones, each using separate adapters for the node, edge, and trace predictors. All main Knossos results are evaluated on the frozen 1,200-image held-out test set, with 200 images from each of the six domains. At inference, the predicted node inventory is partitioned into source groups of target size 3 for edge extraction, while the trace predictor is applied only after the symbolic edge set is fixed. Complete training and decoding details are provided in Appendix E.1, with prompt templates in Appendix E.2. We use macro-averaged Node F1 and Edge F1 scores as the primary topology metrics. Node F1 measures node identification after matching predictions to reference nodes, while Edge F1 evaluates directed source–target connectivity between matched nodes. Scores are averaged across images within each domain, and the overall Knossos score is the mean across the six domains. Additional relation-type and connector-grounding metrics are reported in ablation analyses and appendix. We additionally evaluate generalization on a 162-image subset of the external TopoBench-180 test set (Guo et al., 2026), with preprocessing details provided in Appendix B.6. We report only Node F1 and Edge F1 on this benchmark.

Baselines. We organize the baselines into three groups. Open-source VLMs include Qwen3-VL-32B and Qwen3-VL-8B (Bai et al., 2025), GLM-4.6V-Flash (Hong et al., 2025), MiMo-VL-7B-RL (Yue et al., 2025), Keye-VL-1.5-8B (Yang et al., 2025), InternVL3.5-8B (Wang et al., 2025), and Ovis2.5-9B (Lu et al., 2025). Closed-source VLMs include GPT-4o (Hurst et al., 2024), GPT-5.6 (Singh et al., 2025), and Claude Opus 5 (Anthropic, 2026). For all open- and closed-source VLM baselines, we use direct full-graph generation, requiring the model to produce all diagram nodes and edges in a single response. This setting provides a direct comparison with Ariadne's decomposed node-to-edge prediction strategy. Structured visual-reasoning methods include Pixel Reasoner (Su et al., 2026a), Chain of Region (Li et al., 2025b), Speculative Verdict (Liu et al., 2026), and TopoAgent (Guo et al., 2026). We instantiate Ariadne with two backbone configurations, GLM-4.6V-Flash (9B) and Qwen3-VL-8B, allowing us to evaluate the proposed decomposition across two different open-source VLM families.

4.2 NODE EXTRACTION

Figure 3 compares node extraction results on the internal and external test sets. Node extraction is highly reliable on the Knossos internal test set: Ariadne achieves a macro-average Node F1 of 97.8% with both the GLM- and Qwen3-based variants, while strong open-source VLMs also generally exceed 95%. Recent closed-source VLMs are already close to this ceiling, with Opus 5 reaching 97.4% and GPT-5.6 reaching 96.8%. Together, these results suggest that current multimodal models already provide strong OCR and visual-semantic recognition for identifying diagram entities. Performance decreases on TopoBench-180, suggesting that additional node-specific training may introduce some distribution specialization. Given the already strong node recognition capabilities of modern VLMs, a separately trained node predictor may therefore not always be necessary. This further motivates focusing model capacity and evaluation on relation and topology extraction, where the performance gap remains substantially larger. Detailed domain-level results and additional analysis are provided in Appendix D.3.

4.3 EDGE EXTRACTION

Table 1 compares Ariadne with a broad set of baselines on Edge F1. On Knossos, recent closed-source VLMs already show strong direct topology-extraction capability: GPT-5.6 reaches 81.1% average Edge F1, substantially outperforming all direct open-source VLMs. Ariadne improves further, reaching 91.6% with the Qwen3-based variant and 88.8% with the GLM-based variant. Ariadne-Qwen uses an 8B backbone yet substantially exceeds the 32B Qwen3-VL model under

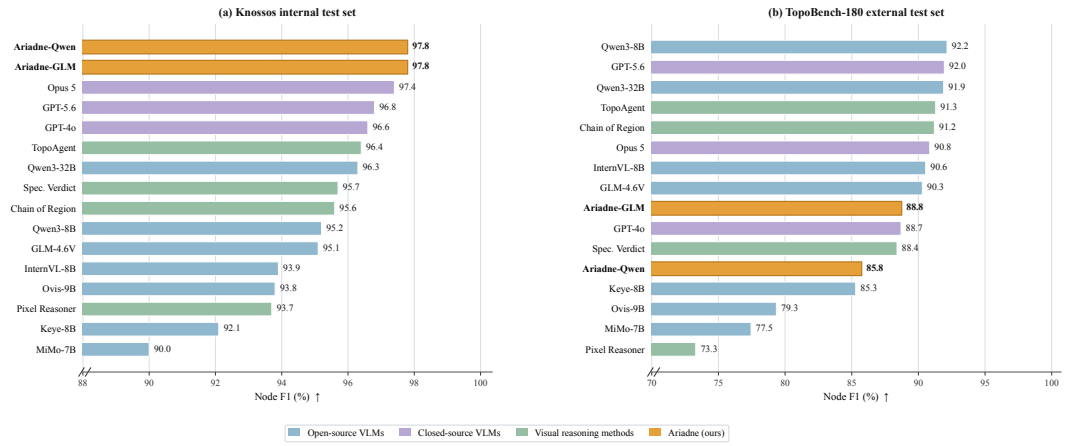

Figure 3: Node extraction results on the internal and external test sets. Panel (a) reports the unweighted mean Node F1 across the six Knossos domains. Panel (b) reports the unweighted mean across the Web-style and Network-style subsets of TopoBench-180. Colors distinguish open-source VLMs, closed-source VLMs, visual reasoning methods, and our Ariadne models. Full model names are listed in Table 1.

direct extraction (91.6% vs. 54.7%). Relative to the corresponding direct backbones, Ariadne improves Qwen3-VL-8B from 46.1% to 91.6% and GLM-4.6V-Flash from 38.9% to 88.8%, reflecting the combined effects of task adaptation and the proposed structured decomposition.

On TopoBench-180, closed-source VLMs show the strongest generalization, with Claude Opus 5 reaching 81.2% on Web-style diagrams and GPT-5.6 reaching 68.7% on Network-style diagrams. This suggests that broad multimodal pretraining provides strong out-of-distribution topology understanding without task-specific adaptation. Ariadne nevertheless substantially improves over its corresponding backbones: the GLM-based variant increases from 55.2% to 73.4% on Web-style diagrams and from 45.1% to 55.6% on Network-style diagrams, while the Qwen3-based variant improves from 44.1% to 68.6% and from 49.8% to 53.2%, respectively. The GLM-based variant also generalizes better than the Qwen3-based variant despite its lower in-domain score, highlighting the distinction between in-distribution performance and external generalization.

Edge F1 evaluates whether the two endpoints are correctly recovered, but does not require the predicted relation type to be correct. Typed-Edge F1 is stricter: a prediction is counted as correct only when both endpoints match and the edge type indicated by the diagram legend is also extracted correctly. Figure 4 summarizes relation-type prediction by averaging Typed-Edge F1 over Natural Process, Circuit, and Map Route, the three domains with explicit relation semantics. Ariadne-Qwen ranks first at 90.4%, followed by Ariadne-GLM at 88.9%. Both outperform the strongest baseline, GPT-5.6 at 84.0%, by 6.4 and 4.9 points, respectively. The gap is substantially larger relative to direct open-source VLMs: the strongest of these, Qwen3-VL-32B, reaches 49.8%. This aggregate comparison indicates that Ariadne's gains extend beyond identifying the correct endpoints to extracting the semantics of their relations. Complete domain-level results are provided in Appendix Table 5.

4.4 Analysis and Ablations

When does staged topology extraction help? Table 2a compares one-step and staged topology extraction across models with different capacities and training settings. Staging does not improve GPT-5.6 and instead reduces its Edge F1, suggesting that frontier models may already have sufficient capacity for joint entity and relation extraction. In contrast, decomposition provides a modest gain for the base Qwen3-VL-8B model and a much larger gain after task-specific adaptation: Ariadne-Qwen improves from 80.0% to 91.6% Edge F1. Node F1 changes only marginally, indicating that the improvement primarily comes from more reliable edge extraction. Overall, our results suggest that staged decomposition is particularly beneficial for smaller task-adapted models, while larger frontier models benefit less from explicitly separating the two steps.

Table 1: Edge extraction results across six Knossos domains and two external TopoBench-180 domains, measured by Edge F1 scores (%).

	Knossos							TopoBench-180	
Method	Food Web	Network	Workflow	Natural Process	Circuit	Map Route	Avg.	Web-style	Network-style
Open-source VLMs									
Qwen3-VL-32B	28.8	48.7	76.4	65.8	36.7	71.9	54.7	54.7	52.3
Qwen3-VL-8B	27.5	47.0	59.9	49.5	31.6	61.1	46.1	44.1	49.8
GLM-4.6V-Flash	30.0	25.0	46.3	48.4	27.3	56.3	38.9	55.2	45.1
MiMo-VL-7B-RL	24.7	37.1	54.3	40.2	26.1	61.7	40.7	42.2	46.5
Keye-VL-1.5-8B	30.7	36.7	51.1	44.3	22.0	49.2	39.0	47.9	45.5
InternVL3.5-8B	22.8	46.0	52.5	44.8	28.0	50.3	40.7	41.9	46.2
Ovis2.5-9B	25.8	37.5	56.7	39.0	25.2	45.8	38.3	42.1	48.8
Closed-source VLMs									
GPT-4o	16.8	43.3	46.8	43.1	30.5	67.0	41.2	32.8	54.1
GPT-5.6	61.9	91.0	80.5	89.6	77.0	86.6	81.1	80.3	**68.7**
Opus 5	54.7	51.1	85.6	96.5	56.9	77.5	70.4	**81.2**	51.8
Visual reasoning methods									
Pixel Reasoner	23.0	31.4	35.4	31.2	24.3	45.6	31.8	32.6	35.3
Chain of Region	18.2	40.6	45.6	40.2	28.3	57.1	38.3	35.0	51.8
Speculative Verdict	23.4	36.8	64.2	50.9	28.9	59.4	43.9	42.8	47.4
TopoAgent	28.4	51.6	54.5	41.4	37.6	65.5	46.5	45.8	56.8
Ours									
Ariadne (GLM-based)	80.4	88.5	94.3	96.7	84.3	88.3	88.8	73.4	55.6
Ariadne (Qwen3-based)	**87.2**	**93.5**	**95.5**	**97.8**	**86.4**	**89.3**	**91.6**	68.6	53.2

What limits Ariadne's external generalization? Table 2b examines why the large gains observed on the internal test set generalize only partially to the external test set. We keep the trained Qwen3 edge predictor fixed and vary only the upstream node inventory. Replacing Ariadne's task-adapted node predictions with those from the base Qwen3-VL-8B raises Node F1 from 85.8% to 92.2% and Edge F1 from 60.9% to 67.3%. This indicates that the more limited external improvement is driven in substantial part by weaker node generalization under distribution shift, which constrains the downstream edge predictor. Meanwhile, general-purpose VLMs already show strong node-recognition generalization on the external data, suggesting that additional node-specific adaptation may be less necessary in this setting. With oracle nodes, the frozen edge predictor reaches 78.2% macro-F1, 10.9 points above the strongest predicted-node source, confirming substantial error propagation from node extraction while also revealing remaining limitations in edge reasoning even when node uncertainty is removed.

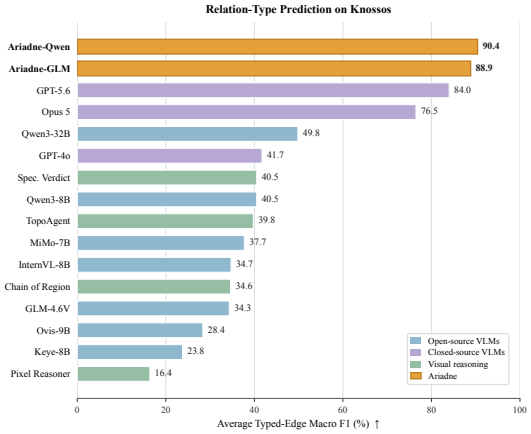

Figure 4: Average Typed-Edge F1 scores across Natural Process, Circuit, and Map Route. Each bar is the arithmetic mean of the three domain-level Macro F1 scores reported in Appendix Table 5.

What diagram structures remain challenging? Table 3 examines how diagram structure affects topology prediction. Performance decreases consistently as diagram difficulty increases, showing that overall diagram complexity remains a primary source of error. Harder diagrams typically contain denser connectivity, more cross-links, and more intersecting connector paths, increasing the difficulty of assigning relations to the correct endpoints. Multi-link diagrams are also more challenging, likely because multiple relations must be visually disentangled and preserved separately rather than collapsed into a single connection. In contrast, performance remains relatively stable across directed, bidirectional, and mixed settings, suggesting that relation direction itself is no longer a major bottleneck for current models. Overall, topology extraction is affected more by diagram complexity and visually entangled connectivity than by directionality.

(a) One-step vs. staged topology extraction on the Knossos test set. Node and Edge F1 are macro-averaged across the six domains (%).

Model	Protocol	Node F1 ↑	Edge F1 ↑
GPT-5.6	One-step	96.8	81.1
	Staged extraction	97.0	77.1
Qwen3-VL-8B (base)	One-step	95.2	46.1
	Staged extraction	95.1	49.3
Ariadne-Qwen	One-step	97.1	80.0
	Staged extraction	97.8	91.6

(b) Node-source ablation on the external TopoBench test set with a fixed trained Qwen3 edge predictor.

Node source	Average Node F1 ↑	Average Edge F1 ↑	Web-style Edge F1 ↑	Network-style Edge F1 ↑
Ariadne node predictor	85.8	60.9	68.6	53.2
Qwen3-VL-8B (base)	92.2	67.3	71.7	62.9
GPT-5.6	92.0	67.1	75.0	59.2
Oracle nodes	100.0	78.2	78.3	78.1

Table 2: Ablations on staged topology extraction and external generalization. (a) compares one-step and staged extraction across model and training settings. (b) varies the upstream node inventory while keeping the trained Qwen3 edge predictor fixed on the public TopoBench subset.

5 Discussion and Conclusion

We studied complete diagram-to-graph topology extraction, where a model must extract both the visible entities and the directed relations connecting them in the diagram. To support this task, we introduced Knossos, a benchmark of 19,200 diagrams across six domains with aligned node, edge, relation-type, and connector annotations, and Ariadne, which separates node inventory prediction from source-conditioned edge extraction. On the 1,200-image Knossos test set, Ariadne-Qwen reaches 91.6% Edge F1, compared with 81.1% for GPT-5.6, with the margin increasing from 2.0 points on simple diagrams to 12.2 points on difficult diagrams. On the external TopoBench-180 test set, Ariadne substantially improves over its direct open-source backbones, although the strongest closed-source VLMs remain more competitive.

Table 3: Edge F1 (%) across structural subsets. N denotes the number of images. To better isolate each structural factor while reducing variation from other domain-specific properties, difficulty is averaged across all six Knossos domains, directionality is evaluated within Map Route, and multi-link effects are evaluated across Circuit, Map Route, and Natural Process.

Factor	Subset	N	GPT-5.6	Ariadne-Qwen
Difficulty	Simple	164	94.4	96.5
	Medium	571	84.9	94.7
	Difficult	465	76.8	89.0
Direction	Directed	132	86.7	89.0
	Bidirectional	37	86.7	88.6
	Mixed	31	86.2	91.0
Multi-link	Absent	384	86.0	92.0
	Present	216	82.1	89.6

Limitations and outlook. First, Knossos remains synthetic, and its assets, layouts, connector styles, and rendering patterns cannot capture the full diversity of real-world diagrams. Results on the external TopoBench-180 test set show generalization beyond the synthetic distribution, but a synthetic-to-real gap remains. Second, Knossos relies on finite domain-specific vocabularies and predefined graph grammars, limiting the semantic and structural diversity seen during training. Generalization to unseen entity types, relation semantics, and topology patterns therefore remains open. Third, Ariadne is sequential: node errors can propagate to edge extraction, while source grouping requires multiple model calls per image. Finally, structured decomposition benefits smaller task-adapted VLMs most, whereas recent closed-source models already show strong direct topology-extraction capability. Whether these gains persist with stronger and larger backbones remains to be studied. Broader real-world evaluation, more efficient decoding, and stronger open-source backbones are important directions for future work.

Conclusion. Our results show that complete diagram topology extraction can be substantially improved through large-scale supervision and structured prediction. Knossos provides a large-scale benchmark for training and controlled evaluation, while Ariadne demonstrates that source-conditioned decomposition can effectively improve topology extraction, particularly for smaller task-adapted VLMs. Together, they provide a practical framework for studying diagram-to-graph topology extraction. We hope Knossos and Ariadne support further progress toward reliable topology extraction across increasingly diverse real-world diagrams.

AI USE STATEMENT

Generative AI and agentic AI tools were used to assist with code implementation, debugging, code execution, and experiment workflows, as well as synthetic visual asset generation, the creation and editing of figures and other visual elements, and language editing of the manuscript. AI agents were also used to execute and inspect computational workflows under author supervision. All AI-assisted code, experimental outputs, data-processing results, figures, and text were reviewed and verified by the authors. The authors take full responsibility for the final code, data, experiments, analyses, claims, figures, and manuscript.

REPRODUCIBILITY STATEMENT

The benchmark construction rules, asset-generation and sampling policies, annotation schema, validation procedure, training stages, and evaluation metrics are documented in the main paper and appendix. We have released an anonymous repository containing the benchmark code, validation pipeline, training recipes, evaluation scripts, and representative data. The complete training set will be released upon publication.

REFERENCES

Qihang Ai, Ruizhou Li, Menghui Wang, and Haiyun Jiang. Graph-to-vision: Multi-graph understanding and reasoning using vision-language models. *arXiv preprint arXiv:2503.21435*, 2025.

Anthropic. Claude opus 5. `https://platform.claude.com/docs/en/models/opus-5/whats-new-opus-5`, 2026. Claude Platform Documentation, accessed September 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Bangwei Guo, Xujiang Zhao, Yanchi Liu, Wei Cheng, Shengyu Chen, Dongyue Li, Masaharu Morimoto, Takayuki Kuroda, Dimitris Metaxas, and Haifeng Chen. Topoagent: A structure-aware perception-to-reasoning framework for diagram-to-graph topology extraction with large vision-language models. *arXiv preprint arXiv:2608.28701*, 2026.

Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, et al. Glm-4.5 v and glm-4.1 v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025.

Yifan Hou, Buse Giledereli, Yilei Tu, and Mrinmaya Sachan. Do vision-language models really understand visual language? *arXiv preprint arXiv:2410.00193*, 2024.

Haoxiang Hu, Yaokun Li, Zeyuan Huang, Cangjun Gao, Qiang He, Qingkun Li, Xiaoming Deng, Cuixia Ma, Yu-Kun Lai, Yong-Jin Liu, et al. Diagram2structure: Unlocking llms' diagram comprehension through diagramdiff, a framework for structuring offline diagrams. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24395–24404, 2026.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Siyu Jiang, Feiyang Chen, Xiaojin Zhang, and Kun He. Kvsmooth: Mitigating hallucination in multi-modal large language models through key-value smoothing. *arXiv preprint arXiv:2602.04268*, 2026.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

Abhishek Kumar, Isha Motiyani, Tilak Kasturi, Ethan Seefried, Prahitha Movva, and Tirthankar Ghosal. Enginuity: A dataset and benchmark for vision-language understanding of engineering diagrams. *arXiv preprint arXiv:2606.03410*, 2026.

Ming Li, Jike Zhong, Tianle Chen, Yuxiang Lai, and Konstantinos Psounis. Eee-bench: A comprehensive multimodal electrical and electronics engineering benchmark. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13337–13349. IEEE, 2025a.

Xue Li, Yiyou Sun, Wei Cheng, Yinglun Zhu, and Haifeng Chen. Chain-of-region: Visual language models need details for diagram analysis. In *International Conference on Learning Representations*, volume 2025, pp. 28547–28563, 2025b.

Yunxin Li, Baotian Hu, Haoyuan Shi, Wei Wang, Longyue Wang, and Min Zhang. Visiongraph: Leveraging large multimodal models for graph theory problems in visual context. *arXiv preprint arXiv:2405.04950*, 2024a.

Zhuowan Li, Bhavan Jasani, Peng Tang, and Shabnam Ghadar. Synthesize step-by-step: Tools, templates and llms as data generators for reasoning-based chart vqa. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13613–13623. IEEE, 2024b.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024.

Yuhan Liu, Lianhui Qin, and Shenji Wan. Small drafts, big verdict: Information-intensive visual reasoning via speculation. In *International Conference on Learning Representations*, volume 2026, pp. 5385–5412, 2026.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521, 2022.

Shiyin Lu, Yang Li, Yu Xia, Yuwei Hu, Shanshan Zhao, Yanqing Ma, Zhichao Wei, Yinglun Li, Lunhao Duan, Jianshan Zhao, et al. Ovis2. 5 technical report. *arXiv preprint arXiv:2508.11737*, 2025.

Ahmed Masry, Jia Qing Tan, Shafiq Joty, Enamul Hoque, et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Shuyin Ouyang, Jie M Zhang, Jingzhi Gong, Gunel Jahangirova, Mohammad Reza Mousavi, Jack Johns, Beum Seuk Lee, Adam Ziolkowski, Botond Virginas, and Joost Noppen. Benchmarking and evaluating vlms for software architecture diagram understanding. *arXiv preprint arXiv:2604.04009*, 2026.

Huitong Pan, Qi Zhang, Cornelia Caragea, Eduard Dragut, and Longin Jan Latecki. Flowlearn: Evaluating large vision-language models on flowchart understanding. In *ECAI 2024: 27th European Conference on Artificial Intelligence, 19–24 October 2024, Santiago de Compostela, Spain–Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, pp. 73–80. SAGE Publications Pvt. Ltd 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2024.

Lukas Schölch, Jonas Steinhäuser, Maximilian Beichter, Constantin Seibold, Kailun Yang, Merlin Knaeble, Thorsten Schwarz, Alexander Maedche, and Rainer Stiefelhagen. Towards automatic parsing of structured visual content through the use of synthetic data. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pp. 1607–1613. IEEE, 2022.

Kaiwen Shi, Sichen Liu, Ziyue Lin, Hangrui Guo, and Gong Cheng. Flowgen: Synthesizing diverse flowcharts to enhance and benchmark mllm reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 19761–19791, 2026.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. *Advances in Neural Information Processing Systems*, 38:8222–8251, 2026a.

Xiaoyan Su, Peijie Dong, Zhenheng Tang, Song Tang, Yuyao Zhai, Kaitao Lin, Liang Chen, Gai Yuhang, Yuyu Luo, Qiang Wang, et al. Vcg-bench: Towards a unified visual-centric benchmark for structured generation and editing. *arXiv preprint arXiv:2605.15677*, 2026b.

Liyan Tang, Grace Kim, Xinyu Zhao, Thom Lake, Wenxuan Ding, Fangcong Yin, Prasann Singhal, Manya Wadhwa, Zeyu Liu, Zayne Sprague, et al. Chartmuseum: Testing visual reasoning capabilities of large vision-language models. *Advances in Neural Information Processing Systems*, 38, 2026.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Biao Yang, Bin Wen, Boyang Ding, Changyi Liu, Chenglong Chu, Chengru Song, Chongling Rao, Chuan Yi, Da Li, Dunju Zang, et al. Kwai keye-vl 1.5 technical report. *arXiv preprint arXiv:2509.01563*, 2025.

Junyi Ye, Ankan Dash, Wenpeng Yin, and Guiling Wang. Beyond end-to-end vlms: Leveraging intermediate text representations for superior flowchart understanding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3534–3548, 2025.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Zihao Yue, Zhenru Lin, Yifan Song, Weikun Wang, Shuhuai Ren, Shuhao Gu, Shicheng Li, Peidian Li, Liang Zhao, Lei Li, et al. Mimo-vl technical report. *arXiv preprint arXiv:2506.03569*, 2025.

Xingchen Zeng, Zhewei Su, Hengming Zhang, Juyong Jiang, Jiazhi Xia, and Wei Zeng. Davinci: Reinforcing visual-structural syntax in mllms for generalized scientific diagram parsing. In *International Conference on Learning Representations*, volume 2026, pp. 88664–88699, 2026.

Yilun Zhao, Chengye Wang, Chuhan Li, and Arman Cohan. Can multimodal foundation models understand schematic diagrams? an empirical study on information-seeking qa over scientific papers. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18598–18631, 2025.

Ge Zheng, Jiaye Qian, Jiajin Tang, and Sibei Yang. Why lvlms are more prone to hallucinations in longer responses: The role of context. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4101–4113. IEEE, 2025.

Yingjie Zhu, Xuefeng Bai, Kehai Chen, Yang Xiang, Jun Yu, and Min Zhang. Benchmarking and improving large vision-language models for fundamental visual graph understanding and reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30678–30701, 2025.

A Related Work

Benchmarks for structural diagram understanding. Existing multimodal and diagram-understanding benchmarks largely formulate structured visual understanding as question answering or reasoning over selected visual content. Representative examples include AI2D (Kembhavi et al., 2016), ChartQA (Masry et al., 2022), MMBench (Liu et al., 2024), MMMU (Yue et al., 2024), and ChartMuseum (Tang et al., 2026). More recent work moves toward graph-centric visual reasoning: VisionGraph (Li et al., 2024a), VGCURE (Zhu et al., 2025), and Graph-to-Vision (Ai et al., 2025) evaluate structural and relational reasoning over visually rendered graphs, while

domain-specific benchmarks study structured understanding and extraction in software architecture, engineering, and other specialized diagrams (Ouyang et al., 2026; Kumar et al., 2026; Su et al., 2026b). Synthetic data has also been used to scale structured diagram supervision. SSVC (Schölch et al., 2022) provides synthetic structured-visual-content images with bounding-box and graph annotations, while FlowLearn (Pan et al., 2024) and FlowGen (Shi et al., 2026) use simulated or controllably generated flowcharts for structured understanding and evaluation. Most closely related, TopoBench-180 (Guo et al., 2026) provides a human-verified real-world evaluation set with canonical graph annotations for diagram-to-graph topology extraction. In contrast, Knossos provides large-scale fully supervised training and controlled evaluation across six domains, with complete topology annotations and systematic variation in topology, layout, connector routing, and visual appearance.

Structure-aware diagram understanding and topology extraction. Recent work on diagram understanding has increasingly moved from localized visual evidence toward explicit structural representations. Chain-of-Region decomposes scientific diagrams into semantically meaningful regions for fine-grained reasoning (Li et al., 2025b), while related visual-reasoning methods such as Pixel Reasoner (Su et al., 2026a) and Speculative Verdict (Liu et al., 2026) improve detailed perception through pixel-space reasoning and speculative visual inference. Moving beyond localized evidence, TextFlow (Ye et al., 2025) converts flowcharts into textual graph representations before downstream reasoning, Diagram2Structure (Hu et al., 2026) reconstructs raster diagrams into standardized structured representations, and DaVinci (Zeng et al., 2026) learns diagram structure from visual primitives and their relations. More directly related, TopoAgent (Guo et al., 2026) targets diagram-to-graph topology extraction through grounded structural prediction. Ariadne builds on this direction through supervised complete-topology training and source-group decomposition, with local predictions deterministically assembled into the final graph.

B Detailed Benchmark Construction

This appendix describes the benchmark construction pipeline in sufficient detail for reproduction. We first introduce the domain-specific node vocabularies and graph grammars used to generate valid symbolic topologies. We then describe visual-asset generation and sampling, layout and connector routing, annotation derivation, quality control, and the final balancing and release procedure. We additionally specify the boundary between the released synthetic benchmark and the separately curated real-diagram evaluation set.

B.1 Canonical vocabularies and node instances

Each domain is defined by a version-controlled JSON vocabulary of canonical node entries. An entry separates a stable key and base display name from the attributes used by generation, including a broader semantic category and, when applicable, a structural role, preferred shape, render mode, or symbol. For example, one Circuit entry is {`label: ground, display: Ground, category: reference, role: ground, symbol: ground`}. The graph grammar uses its category and role to place the node in a valid topology, while the renderer uses its display name and symbol to construct its appearance. Keeping these vocabularies with the released generator fixes the available node semantics for a dataset version and makes later vocabulary changes explicit.

Instantiating a vocabulary entry creates a diagram-specific node. To avoid repeating a small closed set of labels verbatim, the renderer may append a domain-specific suffix sampled from letters, numbers, or location terms, as in `Server 03` or `Server West`. If the same base label occurs more than once in a diagram, distinct suffixes are assigned to disambiguate the instances, as in `Ground X` and `Ground 2`. Capitalization may also vary independently. Every instance stores the exact label drawn in the image, a unique diagram-local ID, and a bounding box; these surface variations do not change its underlying vocabulary entry or structural role.

The frozen release contains 310 domain-local canonical labels across the six domains. Table 4 summarizes their coverage and the resulting node counts.

Food Web (31 names; three trophic tiers). *Producer (8):* Grass, Berries, Algae, Leaves, Acorn, Flowering Plant, Plankton, Seaweed. *Intermediate consumer (12):* Rabbit, Mouse, Squirrel, Deer,

Table 4: Canonical vocabulary and node-occurrence statistics in the frozen release. Domain identifies the six diagram domains; Canonical labels is the number of domain-local vocabulary entries represented in that domain; Node instances is the total number of node occurrences across its 3,200 diagrams; Avg./image is the mean number of nodes per diagram; and Min–max is the observed per-diagram node-count range. The total row aggregates all 19,200 diagrams, counting a shared label separately in each domain.

Domain	Canonical labels	Node instances	Avg./image	Min–max
Food Web	31	38,585	12.06	6–17
Network	40	44,315	13.85	8–17
Workflow	60	47,746	14.92	8–20
Natural Process	59	40,527	12.66	7–18
Circuit	60	45,142	14.11	7–23
Map Route	60	28,864	9.02	6–15
Total	310	245,179	12.77	6–23

Caribou, Moose, Caterpillar, Grasshopper, Dragonfly, Butterfly, Frog, Fish. *Higher consumer (11):* Bobcat, Fox, Hawk, Owl, Gray Wolf, Black Bear, Snake, Eagle, Shark, Seal, Killer Whale. The FoodWeb appearance pool also contains assets for Mushroom, Coral, Shrimp, Clam, Snail, Spruce Tree, Evergreen Tree, Oak Tree, and Pine Tree. These nine candidate labels are not sampled by the frozen graph generator and therefore are not counted in the 31-name benchmark vocabulary.

Network (40 names; 15 categories). *Compute (1):* Server. *Endpoint (7):* Client, Workstation, Laptop, Desktop, Phone, Tablet, Terminal. *Infrastructure (1):* Data Center. *IoT (3):* Sensor, Controller, IoT Device. *Network device (4):* Repeater, Bridge, Modem, Relay. *Network group (2):* Subnet, Cluster. *Network node (2):* Edge Node, Core Node. *Peripheral (2):* Printer, Camera. *Routing (2):* Router, Gateway. *Security (4):* Firewall, VPN, IDS, IPS. *Service (2):* DNS Server, Cloud. *Storage (3):* Database, Storage, NAS. *Switching (2):* Switch, Hub. *Traffic (2):* Load Balancer, Proxy. *Wireless (3):* Access Point, Base Station, Satellite.

Workflow (60 names; 10 categories). *Terminal (2):* Start, End. *Intake (3):* Receive Request, Submit Form, Capture Input. *Validation (7):* Validate Input, Check Completeness, Verify Identity, Check Eligibility, Check Inventory, Quality Check, Run Test. *Data pipeline (8):* Extract Data, Transform Data, Load Data, Query Database, Update Database, Sync Records, Generate Report, Export File. *Automation (8):* Run Model, Score Result, Classify Item, Route Task, Enqueue Job, Process Job, Retry Step, Handle Error. *Approval (7):* Manual Review, Review Result, Approve Request, Reject Request, Escalate Case, Assign Owner, Sign Off. *Communication (6):* Send Notification, Send Email, Create Ticket, Update Status, Notify Customer, Schedule Meeting. *Operations (6):* Pack Item, Ship Order, Receive Shipment, Inspect Item, Repair Item, Close Case. *Storage (4):* Archive Record, Store Document, Retrieve Record, Backup Data. *Control flow (9):* Merge Branches, Split Path, Wait for Event, Timer Expires, Parallel Tasks, Join Tasks, Decision Point, Condition Met?, Fallback Path.

Natural Process (59 realized names; nine categories). *Energy (2):* Sunlight, Heat. *Climate (2):* Wind, Cloud. *Water (8):* Rain, Snow, River, Stream, Lake, Ocean, Groundwater, Glacier. *Earth (6):* Soil, Sand, Clay, Rock, Mineral, Sediment. *Organic (4):* Leaf Litter, Organic Matter, Dead Wood, Compost. *Organism (14):* Plant, Grass, Tree, Algae, Moss, Fungi, Bacteria, Microbe, Pollinator, Insect, Worm, Fish, Bird, Small Animal. *Role (1):* Decomposer. *Chemical (8):* Carbon Dioxide, Oxygen, Nitrogen, Phosphate, Nitrate, Ammonium, Nutrient, Toxin. *Process (14):* Photosynthesis, Respiration, Decomposition, Evaporation, Condensation, Runoff, Infiltration, Sedimentation, Erosion, Absorption, Fixation, Mineralization, Uptake, Release. The candidate configuration also contains the human–nature label Pollution, but it is not selected by the frozen layer pools and does not occur in the 19,200 released annotations.

Circuit (60 names; 11 categories). *Power source (5):* Battery, Power Supply, USB Power, Solar Cell, AC Adapter. *Power conditioning (3):* Voltage Regulator, Buck Converter, Boost Converter. *Protection (1):* Fuse. *Passive (7):* Resistor, Variable Resistor, Capacitor, Polarized Capacitor, In-

ductor, Transformer, Crystal Oscillator. *Passive sensor (2):* Thermistor, Photoresistor. *Active (10):* Diode, LED, Zener Diode, Transistor, MOSFET, Op Amp, Comparator, Logic Gate, Integrated Circuit, Driver Chip. *Control (6):* Switch, Push Button, Toggle Switch, Relay, Microcontroller, Arduino Board. *Connector (6):* DC Jack, Jumper, GPIO Pin, Terminal Block, Header Connector, Test Point. *Sensor (10):* Temperature Sensor, Light Sensor, Motion Sensor, Pressure Sensor, Humidity Sensor, Current Sensor, Voltage Sensor, Hall Sensor, Microphone, Photodiode. *Load (9):* Motor, Servo Motor, Speaker, Buzzer, Lamp, Heater, Display, Fan, Pump. *Reference (1):* Ground.

Map Route (60 names; six categories). *Transit (10):* Station, Platform, Terminal, Transfer Hub, Depot, Bus Stop, Train Stop, Tram Stop, Ferry Pier, Taxi Stand. *Indoor (10):* Lobby, Office, Laboratory, Storage Room, Elevator, Stairwell, Hallway, Conference Room, Reception, Server Room. *Campus (10):* Main Gate, Library, Dormitory, Cafeteria, Parking Lot, Auditorium, Gymnasium, Courtyard, Admin Building, Lecture Hall. *Warehouse (10):* Loading Dock, Packing Area, Shelf Zone, Conveyor, Sorting Area, Cold Storage, Forklift Bay, Inventory Desk, Dispatch Gate, Return Station. *Landmark (10):* Bridge, Park, Tower, Plaza, Checkpoint, Fountain, Museum, Monument, Tunnel Entrance, Overlook. *Utility (10):* Junction, Entrance, Exit, Security Post, Control Room, Service Gate, Emergency Exit, Information Desk, Rest Area, Access Point.

B.2 Domain graph grammars

Each domain grammar begins from semantic node roles and a collection of valid structural motifs rather than sampling unconstrained random graphs. Difficulty then changes graph scale, density, and geometric ambiguity in ways appropriate to that domain. The statistics below are measured from the frozen release.

Across domains, generation follows the same high-level procedure: sample difficulty-conditioned grammar parameters and role slots, instantiate a symbolic graph satisfying the resulting structural constraints, and only then assign layout, connector routing, and visual assets. Figure 5 illustrates this procedure with the layered Natural Process grammar. Let L_k denote the nodes sampled for process stage k. For every adjacent pair L_k, L_{k+1}, the generator gives each node in L_k at least one outgoing edge into L_{k+1} and each node in L_{k+1} at least one incoming edge from L_k, before adding difficulty-dependent adjacent-stage and intra-stage relations. The example demonstrates how difficulty modifies the grammar—the number and cardinality of stages, relation density, and routing complexity— rather than merely enlarging an otherwise fixed graph.

Food Web. The grammar samples producer, intermediate-consumer, and higher-consumer tiers and arranges them in one of four orientations. In the frozen release, simple images contain 6–8 nodes and 5–9 edges, medium images contain 9–13 nodes and 7–15 edges, and difficult images contain 16–17 nodes and 14–22 edges. Most relations follow plausible trophic direction between adjacent tiers, while denser examples add long-range links and directed, bidirectional, or unarrowed connector variants. Medium and difficult cases may also contain close visual alternatives and connector crossings that do not denote additional relations. Node, label, and arrow sizes are reduced together in dense layouts so that difficulty is not produced by simply adding edges to an unchanged canvas.

Network. Network diagrams combine hub–spoke, layered infrastructure, tiered, and cluster-bridge structures, with device roles constraining plausible connections. Simple examples contain 8–10 nodes and 8–12 edges, medium examples 12–15 nodes and 13–21 edges, and difficult examples 14–17 nodes and 23–31 edges. Increasing difficulty introduces multiple hubs or clusters, shorter distances between candidate endpoints, and more links sharing the same devices or bridge regions. Connectors primarily express unarrowed structural connectivity, with routing adjusted in denser cases to keep connectivity visually traceable.

Workflow. Workflow templates encode three control-flow families used in the frozen release: staged grids, ladder-feedback workflows, and swimlanes. Each template defines role slots and a backbone that preserves the functions of terminals, processes, decisions, splits, and merges. Simple, medium, and difficult examples contain 8–9, 13–14, and 20 nodes, respectively, with 10–11, 18, and 29–32 directed edges. Harder cases increase the number of stages or parallel lanes and may intro-

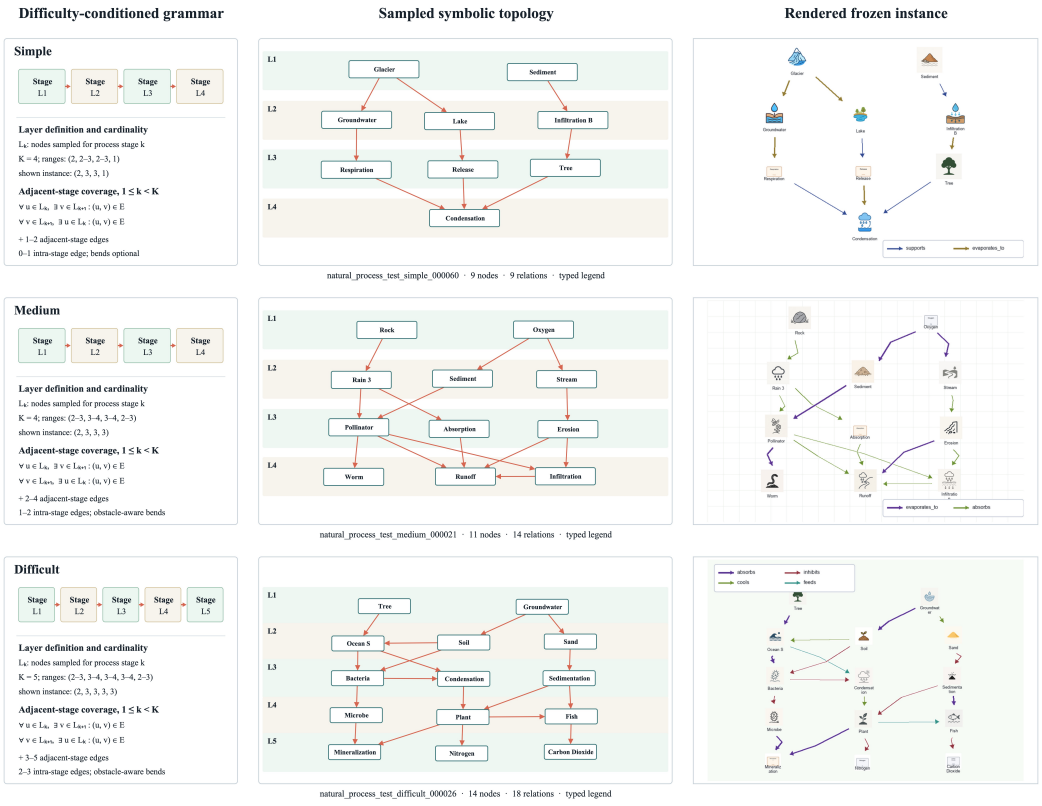

Figure 5: Difficulty-conditioned Natural Process grammar and three frozen instantiations. Left: the executable constraints used at each difficulty. Here L_k is the node set at process stage k, K is the number of stages, and the two coverage constraints require every node in adjacent stages to participate in at least one forward relation. The listed ranges are grammar-level cardinality ranges, while "shown instance" gives the realized cardinalities in that row. Middle: the exact symbolic nodes and edge multiset of the selected example, with geometry regularized only for legibility. Right: the corresponding unmodified rendered image and its relation-type legend. From top to bottom, the examples contain 9/9, 11/14, and 14/18 nodes/relations. All three are drawn directly from the frozen test set; no topology, annotation, or rendered connector was edited for the figure.

duce skip or feedback relations. Role-specific shapes are rendered locally, while typed connector styles are assigned after topology creation and communicated through a matching legend.

Natural Process. Natural Process diagrams organize sources, carriers, reservoirs, flows, organisms, and materials into layered transformation or transport structures. Simple and medium cases use four layers with 7–9 and 10–14 nodes, while difficult cases use five layers with 13–18 nodes. Their corresponding edge ranges are 6–13, 10–23, and 15–33. The grammar first establishes progression between adjacent layers, ensures that nodes participate in the process, and then adds difficulty-dependent cross-layer or reversible within-layer links. An image may use one or several semantic relation types and may contain multiple typed relations between the same endpoint pair. Parallel connectors are visually separated and distinguished through the legend.

Circuit. Circuit diagrams use layered functional structures that progress from power, sensing, or passive components through control and active components to loads and grounds. Simple and medium diagrams use four layers, whereas difficult diagrams use five; the frozen ranges are 7–10, 10–14, and 16–23 nodes and 6–12, 9–22, and 18–35 edges. Endpoint roles determine whether a relation is rendered as wire, power, ground, or signal: links to ground use the ground relation, power-source links favor power, and control, sensor, or active components favor signal or wire. Harder

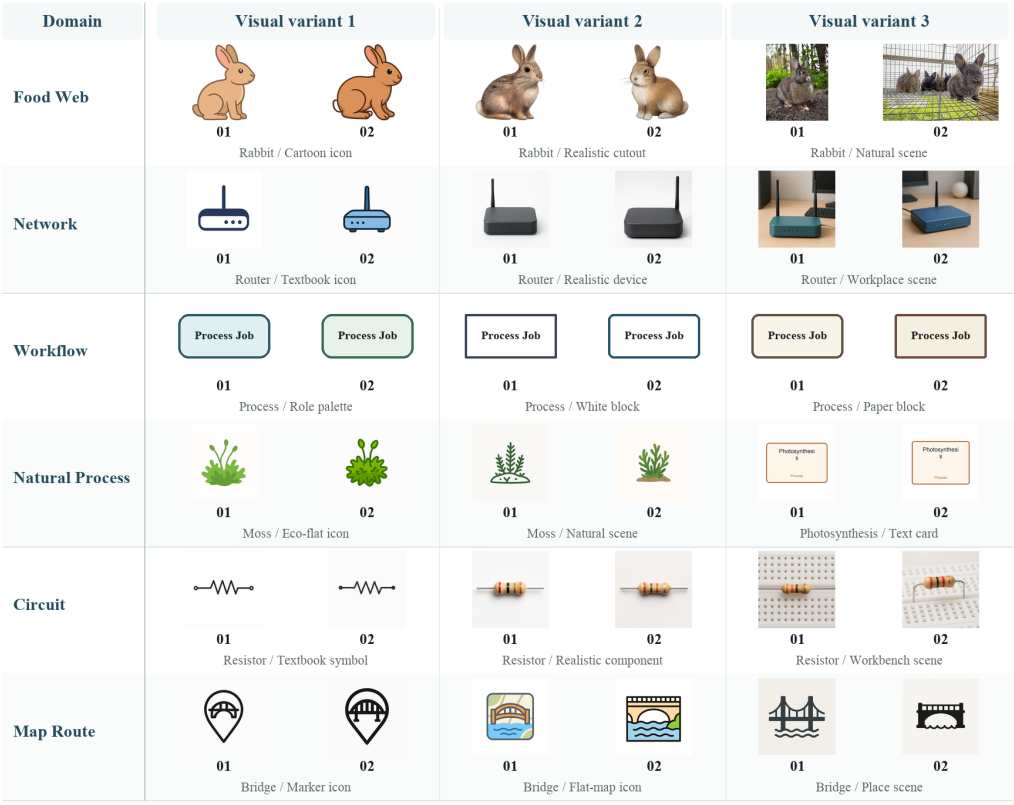

Figure 6: Representative visual-appearance pools in Knossos. Each row is a domain, each column is an appearance family, and each cell shows the two candidate variants available for one representative vocabulary entry. Workflow appearances are rendered programmatically rather than generated. The figure shows the candidate pool; the frozen renderer uses the domain-specific subset and sampling policy described below.

diagrams add denser inter-layer and parallel typed connections. Shared endpoints and parallel links are visually separated and identified in a legend.

Map Route. Map Route diagrams cover city and campus grids, indoor corridors, transit networks, subway-like lines, floorplans, warehouse routes, cluster bridges, and free spatial maps. Simple examples contain 6–8 nodes and 4–7 edges, medium examples contain 8–12 nodes and 6–16 edges, and difficult examples contain 8–15 nodes and 8–25 edges. Difficulty additionally increases intersections, shared corridors, and alternative paths between nearby destinations. Routes may be direct, softly bent, or orthogonal and may be directed or unarrowed. Overlapping routes and parallel links are visually separated, while route types and direction conventions are indicated by the legend.

B.3 Visual asset pipeline

Visual appearance is sampled only after the symbolic graph has been fixed. The graph generator determines node identities and roles, typed relations, and layout constraints; the asset pipeline supplies alternative appearances for the resulting nodes. Consequently, changing an asset cannot change the graph or any evaluation target.

Asset construction. For each image-eligible vocabulary entry, the pipeline constructs label–style–variant generation jobs from its canonical label, semantic role, and requested visual style. Prompts request a single centered subject in a square composition and explicitly exclude text, arrows, diagram lines, brands, borders, and watermarks. Two independent candidates are generated for each label–

style pair and screened for recognizability, unwanted text, and visual quality. Assets are generated with the OpenAI Images API using `gpt-image-1-mini` at 1024×1024 resolution, medium quality, and PNG output. Visible node labels are rendered separately from the symbolic annotations using Pillow rather than by the image model, ensuring exact agreement between displayed text and stored ground truth. The 1024×1024 resolution above refers only to source assets before they are fitted into diagram nodes, not to the resolution of the rendered diagrams.

Domain-specific visual pools. Food Web uses three appearance families over 40 candidate labels, of which 31 occur in the frozen graph vocabulary. Network uses textbook-icon and realistic-device assets. Natural Process combines two generated icon families for image-eligible concepts with locally rendered text cards for abstract process nodes. Circuit and Map Route each use three visual families over their 60-category vocabularies. Workflow does not use generated image assets; its nodes are rendered programmatically using role-specific shapes. Representative examples are shown in Figure 6. The resulting candidate pools contain 240 Food Web, 240 Network, 212 Natural Process, 360 Circuit, and 360 Map Route assets.

Sampling in the frozen renderer. Asset sampling is domain-specific and seeded for reproducibility. Food Web samples independently from label-matched visual candidates for each node, whereas Network selects a consistent visual style at the image level. Natural Process, Circuit, and Map Route alternate between fixed-style and mixed-style rendering to increase visual diversity. Map Route additionally includes locally rendered text-only nodes. The selected asset identity is stored with each node annotation to support exact replay.

B.4 Geometry and annotation derivation

Let B_i denote the finalized bounding box of node v_i, and let $e = (s, t)$ denote a directed edge, optionally associated with a relation type r. The renderer obtains source and target anchors by intersecting the center-to-center ray with the boundaries of B_s and B_t. It then constructs a logical connector polyline P_e using the routing family associated with the sampled domain and template: direct, softly bent, or orthogonal. When obstacle-aware routing is used, candidate detours are selected according to their interference with non-endpoint nodes and path length.

Terminal clearance and arrow placement may slightly shorten the path actually drawn. We therefore store both the logical polyline P_e and the rendered polyline $\widetilde{P}_e$ (`visible_path`). Arrowheads are derived from the terminal direction of $\widetilde{P}_e$, and reciprocal directions use the same path in reverse. Any image-resizing transform is applied consistently to node boxes, centers, path vertices, and arrowheads, preserving image–annotation alignment.

A five-point trace is derived deterministically from the rendered path. Let $\widetilde{p}_e : [0, 1] \to \mathbb{R}^2$ denote its normalized arc-length parameterization in the final image coordinate system. The parameter $t \in [0, 1]$ specifies relative traversal distance along the connector, while $\widetilde{p}_e(t)$ remains an (x, y) location on the resized image canvas. The geometric target is

$$\tau_e = \big(\widetilde{p}_e(j/4)\big)_{j=0}^{4}. \tag{3}$$

The sampled coordinates are rounded to the nearest integer pixel. For the reciprocal direction, the five trace points are stored in reverse order. Node records, edges, relation types when present, connector geometry, and training targets are all serialized from the same finalized objects; none is recovered from rendered pixels or entered manually. Figure 7 visualizes these annotations on a released Food Web sample. Every node is overlaid with its finalized bounding box, and every connector with its five sampled trace points.

B.5 Quality control and release construction

Before large-scale generation, human researchers inspect preview batches from each domain to verify that the generated diagrams are visually coherent and semantically plausible. The review focuses on overall image quality, node and connector readability, layout and routing plausibility, consistency with the intended domain structure, and whether increasing difficulty produces meaningfully more challenging yet still interpretable diagrams. Generation rules and rendering configurations are refined when systematic issues are identified.

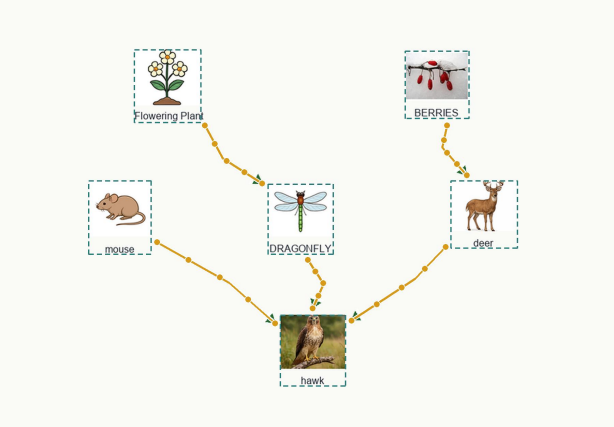

Figure 7: Geometry annotations on a released Food Web sample. Teal dashed boxes denote finalized node bounding boxes. Gold polylines denote rendered connector paths (`visible_path`), and the five circular markers on each path are trace points sampled at equal arc-length intervals in source-to-target order. All coordinates are stored in pixels on the final image canvas.

After the generation templates and rendering rules are validated through human inspection, the pipeline performs large-scale generation, producing 24,000 candidate diagrams across the six domains, with 4,000 candidates generated for each domain. Visual screening is then applied directly to the rendered images using GPT-4o with temperature zero. The binary screening criteria cover overall clarity, text readability, node and connector visibility, arrow and legend clarity, and excessive overlap. Dense layouts, crossings, and mild local clutter are allowed as long as the diagram remains visually interpretable, preventing difficult examples from being systematically filtered out. Visual screening continued until all release quotas were filled. In total, GPT-4o evaluated 19,223 candidate diagrams, of which 19,200 passed and 23 failed.

For release construction, candidates are deterministically shuffled within each domain and difficulty level and assigned to training and test pools before screening. The selected release contains 3,000 training and 200 test images per domain, yielding 18,000 training images and 1,200 held-out test images in total, with matching JSON annotations and manifest entries. Released diagrams use one of two fixed resolutions: 900×630 pixels for landscape layouts and 630×900 pixels for portrait layouts. Of the released images, 14,318 (74.6%) use the landscape resolution and 4,882 (25.4%) use the portrait resolution. The canvas dimensions are stored in every annotation, and all node boxes, connector paths, arrowheads, and trace points are defined in this final pixel coordinate system. An audit of the complete release found exact agreement between the recorded canvas dimensions and the actual PNG resolutions.

B.6 TopoBench-180 evaluation subset

We evaluate on a 162-image subset of TopoBench-180 (Guo et al., 2026), comprising 54 AI2D/Web and 108 Network diagrams. Relative to the original 180-image collection, we exclude 16 internal Network diagrams and two Network diagrams with unresolved edge references. These exclusions reflect data availability and annotation integrity, rather than model performance. We convert name-keyed annotations to stable node-instance IDs and remove annotation-only naming suffixes while preserving visible labels and distinct repeated-label instances. All evaluation images are obtained from the released files or their provided source links. The resulting annotations contain 1,786 nodes and 3,063 ordered relations. We use this subset exclusively for external evaluation and report Node F1 and Edge F1.

C Theoretical Derivation of Source-Group Decomposition

Prior work has observed that longer vision–language generations can be more susceptible to hallucination and semantic drift (Zheng et al., 2025; Jiang et al., 2026). These observations motivate

examining how Ariadne's source-group decomposition changes both the output representation and the autoregressive decoding process by replacing one long graph serialization with multiple shorter generations.

We condition throughout on a fixed node inventory and isolate the edge-decoding stage; errors introduced by upstream node prediction are outside the scope of the derivation. Our analysis proceeds from representation to decoding. We first establish that source grouping preserves the target edge multiset, quantify the cross-group ordering freedom in a one-shot representation, and examine its implications for sequence supervision. We then bound the edge-decoding horizon and, under an explicit position-sensitive error model, derive bounds on the reduction in cumulative position-dependent error.

C.1 Lossless Decomposition and Serialization Ambiguity

We begin by separating the target topology from the order in which its edges are serialized. Let

$$\mathcal{S}(V) = \{S_1, \ldots, S_K\}$$

be a partition of the node inventory, and let E_{S_i} denote the multiset of directed edges whose sources belong to S_i. Since every directed edge has a unique source and therefore belongs to exactly one source group,

$$E = \biguplus_{i=1}^{K} E_{S_i}. \tag{4}$$

Thus, source grouping is lossless with respect to graph topology: it changes the decoding representation without restricting the set of edge multisets that can be represented.

Although the target topology is unchanged, its serialization may still admit multiple equivalent orderings. Let

$$m_i = |E_{S_i}|, \qquad m = |E| = \sum_{i=1}^{K} m_i. \tag{5}$$

Fix the within-group edge orders, denoted by $\mathcal{O}$, and consider a flat one-shot representation that permits arbitrary cross-group interleavings. All other serialization choices are held fixed. An interleaving is determined by assigning m_i of the m output positions to each group; the fixed within-group orders then determine which edge occupies each assigned position. The number of equivalent interleavings is therefore

$$N_{\text{interleave}} = \binom{m}{m_1, \ldots, m_K} = \frac{m!}{\prod_{i=1}^{K} m_i!}. \tag{6}$$

For this one-shot representation, we quantify cross-group ordering freedom by its log-cardinality,

$$A_{\text{one}} = \log N_{\text{interleave}} = \log m! - \sum_{i=1}^{K} \log m_i!, \tag{7}$$

where $\log$ denotes the natural logarithm. This quantity measures the size of the equivalence class, not the entropy of a particular model distribution. Ariadne instead decodes each E_{S_i} in a separate call and combines the results by multiset union. Since edges from different groups are not interleaved within a shared output sequence, its corresponding cross-group ambiguity is

$$A_{\text{group}} = 0, \tag{8}$$

giving

$$\Delta A = A_{\text{one}} - A_{\text{group}} = \log \frac{m!}{\prod_{i=1}^{K} m_i!}. \tag{9}$$

Possible ordering choices within each group remain unchanged.

This comparison establishes a difference in representation, rather than a difference in prediction error. Its relevance to training arises when supervision specifies a single reference sequence: multiple orderings may encode the same topology, but the sequence-level objective evaluates the probability of the particular reference ordering. We next make this distinction explicit.

Implication for sequence supervision. Fix the input $X = (I, V)$ and retain the within-group orders $\mathcal{O}$ defined above. Let $\Sigma_{\mathcal{O}}(E)$ denote the set of $N_{\text{interleave}}$ equivalent interleavings. For a sequence model p_θ, define their total probability mass and the conditional distribution within this class as

$$\begin{aligned} q &= \sum_{y \in \Sigma_{\mathcal{O}}(E)} p_\theta(y \mid X), \\ \pi_\theta(y \mid E, \mathcal{O}, X) &= \frac{p_\theta(y \mid X)}{q}, \qquad y \in \Sigma_{\mathcal{O}}(E), \end{aligned} \tag{10}$$

assuming $q > 0$. Because the within-group orders are fixed, q is the probability assigned to this interleaving class, not necessarily to all serializations of the target graph.

For a reference sequence $y^\star \in \Sigma_{\mathcal{O}}(E)$ with positive probability, its negative log-likelihood decomposes exactly as

$$-\log p_\theta(y^\star \mid X) = -\log q - \log \pi_\theta(y^\star \mid E, \mathcal{O}, X). \tag{11}$$

The first term evaluates the total probability assigned to the equivalent interleavings. The second is the additional loss incurred by specifying one reference ordering rather than marginalizing over this class. In the idealized uniform case, $\pi_\theta(y^\star \mid E, \mathcal{O}, X) = 1/N_{\text{interleave}}$, so the additional term equals $\log N_{\text{interleave}} = A_{\text{one}}$. Otherwise, it depends on the probability assigned to the particular reference.

This decomposition connects the ordering freedom counted above to an ordering-dependent component of sequence supervision. Source-group decoding removes the need to choose a cross-group interleaving, while leaving possible within-group ordering choices intact. It does not, however, guarantee a lower total training loss or graph-prediction error, because the grouped model may assign different probabilities to the target outputs. The comparison also depends on the one-shot format: enforcing a unique cross-group order removes this particular ordering freedom, whereas canonically ordered training targets alone do not constrain the model's output support. The analysis so far concerns how topology is represented and supervised. The following subsections examine a separate consequence of source grouping—a shorter decoding horizon—and its implications for position-dependent error.

C.2 Bound on the Decoding Horizon

Source grouping also shortens the maximum edge-decoding horizon of each autoregressive generation. A one-shot representation predicts all $m = |E|$ edges in a single sequence,

$$L_{\text{one}} = m, \tag{12}$$

whereas source-group decoding requires at most

$$L_{\text{group}} = \max_i m_i \leq m, \tag{13}$$

where $m_i = |E_{S_i}|$. The inequality is strict whenever at least two source groups contain edges. Thus, source grouping preserves the complete edge set while replacing one global decoding horizon with multiple shorter local ones. This reduction motivates the position-sensitive error analysis below.

C.3 Position-Sensitive Decoding Model

Empirical studies of LVLM generation have reported greater hallucination risk and semantic drift in longer responses, including increased error toward later decoding positions (Zheng et al., 2025; Jiang et al., 2026). Motivated by this behavior, we consider a stylized position-sensitive decoding model.

Let ϵ_j denote the expected error associated with an edge-level decision made at position j within one autoregressive generation. For this stylized comparison, we assume that ϵ_j captures the position-dependent component of edge-level error and is shared between the one-shot and grouped decoding processes. We first assume only monotonicity:

$$\epsilon_{j+1} \geq \epsilon_j. \tag{14}$$

This assumption does not assert that output position itself causes error. Rather, it models a regime in which later decisions are no easier on average than earlier ones, potentially reflecting accumulated uncertainty, semantic drift, or increasing dependence on previously generated context.

For the comparison below, consider a one-shot serialization in which the source groups are written as contiguous blocks in the order $S_1, \ldots, S_K$. Define the cumulative number of edges preceding group i as

$$M_{i-1} = \sum_{h=1}^{i-1} m_h, \qquad M_0 = 0. \tag{15}$$

Under one-shot decoding, the edges of group i therefore occupy positions $M_{i-1}+1, \ldots, M_{i-1}+m_i$, giving

$$\mathcal{E}_{\text{one}} = \sum_{i=1}^{K} \sum_{j=1}^{m_i} \epsilon_{M_{i-1}+j}. \tag{16}$$

Under source-group decoding, each group starts a new autoregressive generation, so that

$$\mathcal{E}_{\text{group}} = \sum_{i=1}^{K} \sum_{j=1}^{m_i} \epsilon_j. \tag{17}$$

Since $M_{i-1} + j \geq j$, monotonicity in Eq. 14 immediately gives

$$\mathcal{E}_{\text{group}} \leq \mathcal{E}_{\text{one}}. \tag{18}$$

Hence, in the position-sensitive regime, restarting autoregressive decoding across source groups weakly reduces the cumulative error component associated with output position.

C.4 Bound on Position-Dependent Error Reduction

We now strengthen the monotonicity assumption by bounding the growth of the position-dependent error. Let

$$0 \leq \alpha \leq \epsilon_{j+1} - \epsilon_j \leq \beta \tag{19}$$

for all relevant decoding positions j. Here, α and β bound the per-position increase in expected edge-level error.

Define the reduction induced by source grouping as

$$\Delta\mathcal{E} = \mathcal{E}_{\text{one}} - \mathcal{E}_{\text{group}}. \tag{20}$$

Using Eqs. 16 and 17,

$$\Delta\mathcal{E} = \sum_{i=1}^{K} \sum_{j=1}^{m_i} \left(\epsilon_{M_{i-1}+j} - \epsilon_j\right). \tag{21}$$

For every term,

$$\epsilon_{M_{i-1}+j} - \epsilon_j = \sum_{r=j}^{M_{i-1}+j-1} \left(\epsilon_{r+1} - \epsilon_r\right). \tag{22}$$

Applying Eq. 19 yields

$$\alpha M_{i-1} \leq \epsilon_{M_{i-1}+j} - \epsilon_j \leq \beta M_{i-1}. \tag{23}$$

Summing over all m_i edges in group i gives

$$\alpha m_i M_{i-1} \leq \sum_{j=1}^{m_i} \left(\epsilon_{M_{i-1}+j} - \epsilon_j\right) \leq \beta m_i M_{i-1}. \tag{24}$$

Summing again over all source groups,

$$\alpha \sum_{i=1}^{K} m_i M_{i-1} \leq \Delta\mathcal{E} \leq \beta \sum_{i=1}^{K} m_i M_{i-1}. \tag{25}$$

Because

$$\sum_{i=1}^{K} m_i M_{i-1} = \sum_{h<i} m_h m_i = \frac{1}{2}\left(m^2 - \sum_{i=1}^{K} m_i^2\right), \tag{26}$$

we obtain

$$\boxed{\frac{\alpha}{2}\left(m^2 - \sum_{i=1}^{K} m_i^2\right) \le \Delta\mathcal{E} \le \frac{\beta}{2}\left(m^2 - \sum_{i=1}^{K} m_i^2\right)} \tag{27}$$

with

$$\Delta\mathcal{E} = \mathcal{E}_{\text{one}} - \mathcal{E}_{\text{group}}.$$

Equation 27 makes explicit how the benefit of restarting decoding depends on the edge distribution across source groups. If $K = 1$, then $\sum_i m_i^2 = m^2$ and the bound correctly gives zero reduction. For a fixed total number of edges m, the quantity

$$m^2 - \sum_i m_i^2 \tag{28}$$

is larger when edges are distributed more evenly across groups, indicating a larger potential reduction in position-dependent error.

For approximately balanced groups, $m_i \approx m/K$, giving

$$\sum_{i=1}^{K} m_i^2 \approx \frac{m^2}{K}. \tag{29}$$

The lower bound then becomes

$$\Delta\mathcal{E} \gtrsim \frac{\alpha m^2}{2}\left(1 - \frac{1}{K}\right). \tag{30}$$

Under this stylized regime, the potential benefit of source-group restarting therefore increases with graph size and with the extent to which the full edge set is distributed across multiple decoding groups.

Scope of the analysis. The analysis separates model-independent structural guarantees from model-dependent conclusions. The first two results are deterministic under the serialization setting defined above. Source grouping preserves the complete target topology while removing cross-group serialization ambiguity, and it strictly shortens the maximum decoding horizon whenever multiple nonempty source groups are present. These properties do not depend on any assumption about VLM error behavior.

The subsequent error analysis is conditional on the position-sensitive decoding model. In particular, the inequality $\mathcal{E}_{\text{group}} \le \mathcal{E}_{\text{one}}$ requires the assumption that expected edge-level error is nondecreasing with decoding position, while the quantitative error-reduction bound further assumes bounded per-position error growth. Prior work provides empirical motivation for this regime by showing increased hallucination and semantic drift in longer LVLM generations (Zheng et al., 2025; Jiang et al., 2026), but these observations do not imply that every VLM follows such an error profile. Our derivation therefore provides unconditional guarantees about representation and decoding length, together with conditional guarantees about error reduction when longer autoregressive generations become progressively harder to maintain consistently.

D Additional Experimental Results

D.1 Relation-type prediction by domain

Table 5 reports domain-level Typed-Edge F1 on the three Knossos domains with explicit relation semantics. Across all three typed domains, both Ariadne variants consistently outperform their corresponding direct backbones. Ariadne-Qwen achieves the highest Typed-Edge F1 in each domain, with Ariadne-GLM close behind. Circuit remains the most challenging domain for most methods, while Natural Process generally shows the strongest performance. The large variation of several closed-source models across domains further suggests that relation-type prediction remains sensitive to domain-specific semantics.

Table 5: Typed-Edge F1 (%) on the three Knossos domains with explicit relation semantics.

Method	Natural Process	Circuit	Map Route
Open-source VLMs			
Qwen3-VL-32B	49.5	30.7	69.3
Qwen3-VL-8B	41.8	24.0	55.8
GLM-4.6V-Flash	34.9	19.8	48.3
MiMo-VL-7B-RL	34.5	21.3	57.4
Keye-VL-1.5-8B	31.6	11.5	28.2
InternVL3.5-8B	38.9	21.2	44.1
Ovis2.5-9B	24.9	17.1	43.1
Closed-source VLMs			
GPT-4o	38.5	26.6	60.1
GPT-5.6	88.9	76.6	86.5
Opus 5	96.3	56.1	77.1
Visual reasoning methods			
Pixel Reasoner	16.3	11.7	21.2
Chain of Region	29.8	22.9	51.2
Speculative Verdict	43.1	20.3	58.2
TopoAgent	36.3	29.6	53.4
Ours			
Ariadne (GLM-based)	95.9	82.9	87.8
Ariadne (Qwen3-based)	**97.3**	**85.0**	**89.0**

Table 6: Geometric grounding on the fixed 120-image Knossos subset. Box mIoU and R@0.5 are percentages; connector L2 is measured in original-image pixels.

Method	Box mIoU ↑	R@0.5 ↑	Connector L2 ↓
GPT-5.6	81.81	93.73	11.34
Ariadne-GLM	86.42	96.96	6.14
Ariadne-Qwen3	**96.59**	**97.55**	**4.13**

D.2 Bounding-box and connector localization

We evaluate geometric grounding on a fixed 120-image subset of the Knossos test set, with 20 images per domain and difficulty proportions approximately preserved. The subset contains 1,548 reference nodes and 2,759 connector instances corresponding to typed edges. Bounding-box evaluation uses each model's predicted nodes, whereas connector localization is conditioned on reference nodes and edges to isolate geometric localization quality from topology errors.

For bounding boxes, predictions are matched to reference nodes by normalized visible names using one-to-one maximum-IoU assignment. We report mean IoU (mIoU) over matched nodes and recall at IoU ≥ 0.5 (R@0.5). Connector localization is evaluated using the mean L2 distance between the five predicted and reference points along each connector. All distances are measured in the original image coordinate system.

Table 6 shows that both Ariadne variants improve geometric grounding over GPT-5.6, with Ariadne-Qwen3 performing best overall. Its box mIoU increases from 81.81% to 96.59%, while connector localization error decreases from 11.34 to 4.13 pixels. Ariadne-GLM also substantially reduces connector localization error while improving bounding-box localization.

The domain-level results in Table 7 show that the gains are broadly consistent across diagram types. Ariadne-Qwen3 achieves the highest box mIoU in all six domains and lower connector localization error than GPT-5.6 throughout. The largest reductions in connector error occur on Food Web and Circuit, which are also among the most challenging domains for connector localization.

Figure 8 provides qualitative examples of the same comparison.

D.3 Node extraction by domain

Table 8 reports the complete domain-level Node F1 results underlying the macro-averages summarized in the main paper, together with results on the external TopoBench-180 test set. The Knossos average gives equal weight to all six domains. Across Knossos, node extraction is consistently

Table 7: Per-domain geometric grounding on the same 120-image subset. Box mIoU is reported in percent and connector L2 in pixels.

	Box mIoU ↑			Connector L2 ↓		
Domain	GPT-5.6	Ariadne-GLM	Ariadne-Qwen3	GPT-5.6	Ariadne-GLM	Ariadne-Qwen3
Food Web	88.71	86.20	96.06	17.54	6.80	5.74
Network	91.78	92.91	98.55	7.51	4.07	4.14
Workflow	95.15	84.62	97.75	6.80	4.78	2.86
Natural Process	67.74	85.78	94.94	6.68	6.09	3.37
Circuit	69.12	82.98	93.49	20.24	8.36	4.67
Map Route	69.37	85.29	99.19	8.27	8.11	4.00

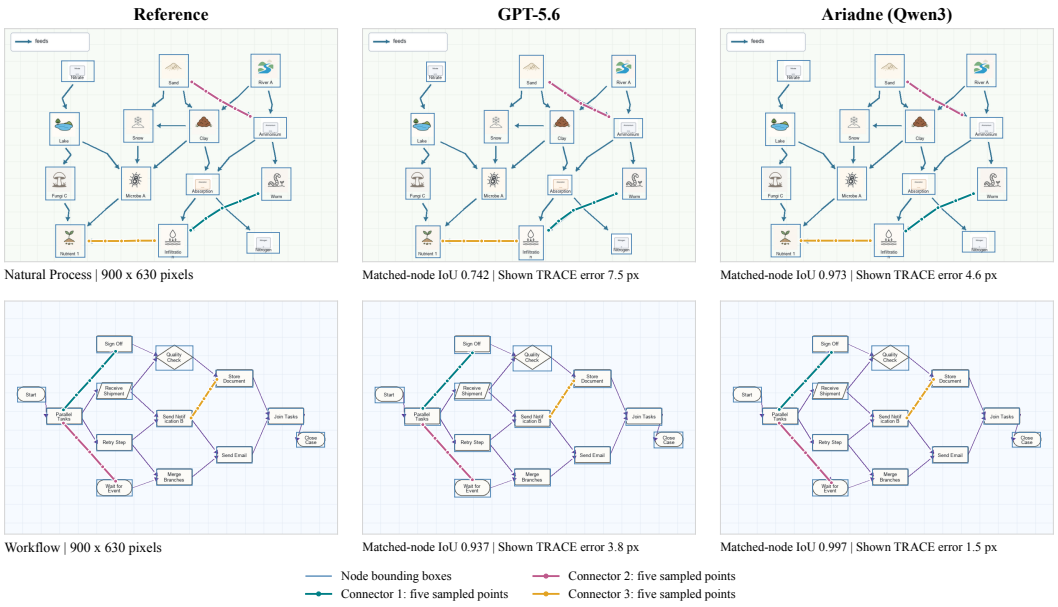

Figure 8: Geometric grounding examples on two original Knossos test images. Boxes show node localization, while colored polylines and markers show reference-conditioned connector predictions. No predicted coordinates are manually adjusted.

strong for most models, with Food Web, Network, Workflow, and Natural Process generally approaching ceiling performance. Circuit is noticeably more challenging, producing the largest drop in Node F1 for many baselines, while Map Route also shows somewhat greater variation. Both Ariadne variants remain stable across domains and achieve the highest Knossos average of 97.8%. On TopoBench-180, however, general-purpose VLMs often retain stronger node-level generalization than the task-adapted Ariadne node predictors. This domain-level breakdown further supports the observation that node-specific adaptation improves in-domain accuracy but may introduce some specialization to the synthetic training distribution.

D.4 Protocol efficiency ablation

We compare the inference cost of one-step generation and staged decomposition on the same 30 Knossos test images, with 10 images from each difficulty level. For each backbone, both protocols use the same model configuration. One-step generation requires a single model invocation per image, whereas staged inference performs one node-prediction call followed by one call for each active source group. Table 9 reports mean per-image inference cost. GPT-5.6 latency, token usage, and API cost are obtained directly from archived request metadata. For GLM-4.6V-Flash, latency is reconstructed from consecutive completion timestamps within each continuously running GPU shard, excluding model loading. The comparison is therefore paired within each backbone; ab-

Table 8: Node F1 (%) across the frozen KNOSSOS internal test set and TopoBench-180. Avg. is the unweighted macro-average of the six KNOSSOS domain-level scores. Dashes indicate results that are not available.

	KNOSSOS							TopoBench-180	
Method	Food Web	Network	Workflow	Natural Process	Circuit	Map Route	Avg.	Web-style	Network-style
Open-source VLMs									
Qwen3-VL-32B	99.3	99.5	99.4	97.9	87.8	93.9	96.3	96.3	87.5
Qwen3-VL-8B	99.3	99.6	99.4	96.0	86.3	90.9	95.2	95.4	88.9
GLM-4.6V-Flash	98.9	98.7	99.0	98.0	85.4	90.8	95.1	94.3	86.3
MiMo-VL-7B-RL	87.0	98.6	92.3	88.7	83.3	90.4	90.0	73.7	81.2
Keye-VL-1.5-8B	97.0	98.4	96.4	92.5	80.8	87.5	92.1	86.7	83.9
InternVL3.5-8B	99.2	99.0	98.5	96.7	80.6	89.2	93.9	94.7	86.4
Ovis2.5-9B	96.7	99.1	98.6	97.3	80.2	90.9	93.8	80.1	78.6
Closed-source VLMs									
GPT-4o	99.9	99.1	99.4	98.9	89.2	93.0	96.6	93.3	84.1
GPT-5.6	99.9	99.2	99.2	98.8	88.9	94.7	96.8	**96.4**	87.5
Opus 5	99.9	99.9	99.9	99.2	90.5	95.1	97.4	93.6	**88.1**
Visual reasoning methods									
Pixel Reasoner	99.8	99.8	99.2	91.0	82.3	90.0	93.7	75.2	71.4
Chain of Region	99.8	98.6	99.5	99.2	87.5	89.2	95.6	95.6	86.8
Speculative Verdict	99.6	99.8	99.3	99.2	85.5	91.0	95.7	93.6	83.2
TopoAgent	99.9	99.6	99.0	98.4	88.8	92.4	96.4	95.6	87.0
Ours									
Ariadne (GLM-based)	99.7	99.5	99.8	99.2	93.4	95.0	**97.8**	94.6	82.9
Ariadne (Qwen3-based)	99.7	99.7	99.9	99.2	93.6	94.9	**97.8**	92.2	79.3

Table 9: Inference efficiency on the same 30-image test slice (10 simple, 10 medium, and 10 difficult). S/M/D denote mean latency (seconds per image) by difficulty; Overall is the mean across all 30 images.

Backbone	Protocol	Calls/img	API calls/img	S	M	D	Overall	Tokens/img	Cost/img ($)
GLM-4.6V-Flash	One-step	1.00	0.00	61.81	106.00	153.92	107.24	–	0.000
GLM-4.6V-Flash	Staged	5.23	0.00	83.39	126.86	204.33	138.19	–	0.000
GPT-5.6	One-step	1.00	1.00	13.89	26.98	28.90	23.26	3,843	0.052
GPT-5.6	Staged	5.23	5.23	21.99	34.55	54.81	37.12	10,747	0.147

solute latency between local GLM inference and the GPT-5.6 API should not be interpreted as a hardware-versus-service benchmark.

Staged inference uses $5.23\times$ as many model invocations on this slice. Relative to one-step generation, mean latency increases by 28.9% for GLM-4.6V-Flash and 59.6% for GPT-5.6. For GPT-5.6, token usage increases by $2.80\times$, with a comparable increase in estimated API cost. The overhead also grows with diagram difficulty because larger node inventories induce more source-group calls. These results quantify the main efficiency trade-off of staged decomposition alongside its accuracy gains reported in the main ablation.

Table 10: Stage-specific supervised-training configuration. "Initialization" denotes the adapter state from which each specialist is trained. Steps are optimizer steps after gradient accumulation.

Stage	Epochs	Steps	Learning rate	Initialization
NODE	1	1,125	5×10^{-5}	Pretrained VLM
EDGE	1	5,960	1×10^{-5}	Node predictor adapter
TRACE	1	5,801	1×10^{-5}	Node predictor adapter

E Additional Reproducibility Material

E.1 Training and inference configuration

Both Ariadne instantiations use the same data, optimization protocol, prompts, and output schemas. Table 10 summarizes the stage-specific training hyperparameters and adapter initialization. Edge and trace predictors each initialize from the completed node adapter but are trained independently.

All trainable Ariadne predictors and same-backbone direct full-graph baselines are trained with QLoRA using 4-bit NF4 quantization, double quantization, and bfloat16 computation. We use LoRA rank $r = 16$, scaling factor $\alpha = 32$, and dropout 0.05. Adapters are applied to the language attention and MLP projections, the corresponding vision projections, and the vision–language merger. Training optimizes assistant-response tokens only, while image, system, and user-prompt tokens are masked from the supervised cross-entropy loss. We use paged 8-bit AdamW with zero weight decay, gradient clipping at 1.0, cosine learning-rate decay, 3% linear warmup, and random seed 7. Gradient checkpointing is enabled. The effective batch size is 16 with a per-device batch size of one. All Ariadne predictors and direct full-graph baselines are trained on four NVIDIA A100 GPUs with four gradient-accumulation steps. We train direct full-graph baselines using GLM-4.6V-Flash (9B) and Qwen3-VL-8B with the same corresponding backbones, training data, and backbone-specific optimization settings as their Ariadne counterparts. Model evaluation is performed after training on the frozen evaluation splits rather than during the training loop.

Inference uses deterministic greedy decoding with one beam. For Ariadne, the node and edge predictors use maximum generation budgets of 2,048 tokens, while the trace predictor uses 1,024 tokens. Generation terminates early upon producing the corresponding closing tag (`</NODES>`, `</EDGE_GRAPH>`, or `</TRACES>`). The predicted node inventory is partitioned into source groups of target size three, while every group retains the complete predicted inventory as its candidate target set. The direct full-graph baselines instead generate the complete node inventory and edge structure within a single response. Evaluation loads model weights in 4-bit mode, and identical frozen test examples are used for the GLM- and Qwen3-based Ariadne variants and their corresponding direct full-graph baselines.

E.2 Prompts for topology extraction

Ariadne separates topology extraction into a node-inventory predictor followed by a source-group edge predictor. The two predictors are trained with separate objectives and independent QLoRA adapters. In particular, edge prediction does not consume or produce connector traces; the symbolic topology is complete once the source-group edge predictions are merged. During edge training, the complete reference node inventory is provided as input, whereas end-to-end inference uses the inventory predicted by Stage 1. Text enclosed in square brackets below denotes per-example content inserted by the data or evaluation pipeline and is not literal prompt text.

Node inventory prompt. The node predictor is restricted to node discovery and geometric localization. Its output contains visible node labels, diagram-local identifiers, and bounding boxes, but no relations or connector predictions.

NODE system prompt

```
You are the Node Inventory Predictor. Given a topology diagram
image, detect only the visible nodes and return their ids, names, and
bounding boxes. Do not extract edges, traces, legends, or final
graphs.
```

NODE user prompt

```
Extract only the visible node inventory from this diagram.

Return only a <NODES> block. Do not output edges or traces.

Rules:
- List every visible node exactly once.
- Use node ids n1, n2, ... in reading order.
- Preserve the visible label text as the node name.
- Each box covers the full visible node region, including its
  symbol and text label.
- Use integer coordinates in [x1,y1,x2,y2] format.
```

NODE response schema

```
<NODES>
  <NODE id="n1" name="Grass" box="[120,40,200,120]"/>
  <NODE id="n2" name="Rabbit" box="[320,220,410,350]"/>
</NODES>
```

Source-group edge prompt. The edge predictor receives the complete node inventory as target context but may emit outgoing relations only for the active source group. Because the source groups partition the predicted node inventory, the resulting edge multisets can be merged deterministically after independent decoding.

EDGE system prompt

```
You are the Edge Predictor. Extract typed topology relations
for the selected source nodes and output EDGE elements only.
```

EDGE user prompt template

```
Extract typed topology relations for the selected source nodes.

You are given:
1. the diagram image;
2. the complete fixed node list with id, name, and box; and
3. the current source node group.

Use only the provided node ids and names. Output only relations
whose source is in the current source group. For every positive
relation, output EDGE directly; do not output TRACE. Preserve the
visible target and exact relation type. If one source-target pair
has multiple types, output one EDGE per type and do not merge them.

If an active source has no outgoing relation, output an empty
CHECK_NODE. Use sparse NO_EDGE only for hard negatives such as a
near miss, a crossing without connection, or the wrong direction.
Return only <EDGE_GRAPH> and do not explain.

Complete node list:
[NODE_INVENTORY]

Current source node group:
[ACTIVE_SOURCES]

Rules:
- Put each relation under its source CHECK_NODE.
- Emit every active source CHECK_NODE exactly once.
- EDGE carries source, target, their names, and the typed label.
- Never output TRACE or repeat node boxes in the response.
```

EDGE response schema

```
<EDGE_GRAPH>
<CHECK_NODE id="n4" name="Sensor">
  <EDGE source="n4" source_name="Sensor"
        target="n6" target_name="Controller"
        type="signal"/>
  <EDGE source="n4" source_name="Sensor"
        target="n8" target_name="Ground"
        type="ground"/>
</CHECK_NODE>
</EDGE_GRAPH>
```

Direction is represented by the ordered source–target pair. When a diagram legend defines semantic relation types, `type` stores the corresponding relation label; otherwise, it is left empty. Undirected and bidirectional connectors are represented by reciprocal ordered edges, while multiple typed relations sharing the same endpoints remain distinct multiset elements. `NO_EDGE` provides sparse hard-negative supervision and is never inserted into the predicted topology.

E.3 Independent geometric grounding prompt

Connector geometry is predicted only after the symbolic edge set has been fixed. A third, independently trained QLoRA adapter receives the diagram, node inventory, and already-decided edges and predicts a five-point trace for each relation. The trace predictor cannot add, remove, redirect, or retype an edge. Each input edge is assigned an immutable `edge_id`, which links the predicted geometry back to its corresponding symbolic relation.

For the geometric-grounding evaluation reported in this paper, the trace predictor is conditioned on reference nodes and reference edges, isolating connector-localization quality from upstream topology errors.

TRACE system prompt

```
You are the Ariadne Trace Predictor. Given a diagram, fixed nodes, and
already-decided typed edges, localize each edge with exactly five
source-to-target coordinates. Never revise or emit an EDGE.
```

TRACE user prompt template

```
Localize the already-decided typed edges in the diagram.

The EDGE decisions are fixed. Do not add, remove, rename, retype,
or emit an EDGE. For each input edge, output exactly one TRACE with
the same edge_id. Each TRACE must contain exactly five points
ordered from the source boundary to the target boundary.
Intermediate points must follow the visible connector rather than
a straight-line guess. Return only <TRACES> and stop after
</TRACES>.

Complete fixed node list:
[NODE_INVENTORY]

Already-decided typed edges:
[EDGE_LIST_WITH_EDGE_IDS]
```

TRACE response schema

```
<TRACES>
  <TRACE edge_id="e1"
    points="[[180,100],[220,145],[260,190],[300,235],[340,280]]"/>
</TRACES>
```

E.4 DIRECT FULL-GRAPH BASELINE PROMPT

For the general-purpose open- and closed-source VLM baselines, we use direct full-graph generation. Each model receives the diagram and produces the complete node inventory and edge set in a single response, using the same node and edge output conventions as Ariadne.

Direct full-graph user prompt

```
Extract the complete typed topology graph from this diagram in one
response. First output a <NODES> inventory, then an <EDGE_GRAPH>.
Do not explain.

Rules:
- List every visible node exactly once using ids n1, n2, ... in
  from reading order; preserve its visible name and integer box.
- For every positive relation, output an EDGE with the correct
  source, target, and relation type when defined.
- Inspect every connector separately and determine direction only
  from visible arrowheads, never from layout or semantics.
- Preserve the exact domain-specific relation type and all
  multiple types between the same endpoint pair.
- Put each relation under its source CHECK_NODE and cover every
  visible connector.
- Encode a one-arrow connector as one ordered relation.
- Encode a two-arrow or no-arrow connector as two reciprocal
  ordered relations; do not confuse bidirectional and undirected.
- Return only the structured <NODES> and <EDGE_GRAPH> blocks.
```