DiGraphHal-Bench: Evaluating Multimodal Large Language Models on Complex Directed Graphs

Supplementary Material

A. Overview

This supplementary material provides extended technical details and analyses for **DiGraphHal-Bench**. The content is organized into three major sections to facilitate systematic exposition and reproducibility.

- First, we comprehensively elaborate on the **design** of all twelve fine-grained tasks over four high-level capabilities: *Structural*, *Visual*, *Semantic*, and *Comprehensive*. Each task is detailed in terms of its motivation, data generation protocols, operational definitions, and evaluation methodologies (Appendices B to E).
- Second, We report the exact **VQA sample counts** for all task categories and modalities to ensure transparency and reproducibility. We further perform an in-depth empirical analysis of our results, providing **detailed metric breakdowns** to dissect performance across individual tasks through SFT impact evaluations and rigorous ablation studies. Notably, we also summarize a series of **counterintuitive and thought-provoking findings**, providing a critical foundation for understanding the limitations of current MLLM reasoning (Appendix F).
- Finally, we present **qualitative case studies** that intuitively compare representative models and highlight common failure modes across tasks (Appendix G).

We will publicly release the benchmark and codebase to enable and support future research.

B. Structural Category Design Details

B.1. Structural Comprehension

Motivation of Subgraph Condition Design. The subgraph condition evaluates model robustness under partial observations and its ability to resist hallucinating elements not fully present in the view. It further captures a practical global-to-local reasoning paradigm, wherein coarse global understanding is progressively refined through localized analysis of specific subgraphs.

Subgraph Data Generation Protocol. The generation of subgraph images follows a principled and reproducible methodology to ensure that the resulting crops are both coherent and challenging:

1. **Subgraph Sampling:** Selecting a coherent subgraph of interconnected nodes and edges from the full graph via a random walk.
2. **Bounding Box Definition:** Computing an **inner bbox** as the minimal rectangle that encloses all elements of the sampled subgraph.
3. **Context Expansion:** Defining an **outer bbox** that encompasses the inner bbox and extends to include half of its nearest neighboring elements, thereby introducing contextual surroundings for cropping.
4. **Middle Bbox Sampling:** Randomly selecting a **middle bbox** from the region between inner and outer bboxes, ensuring the complete subgraph is centered while introducing partial, incomplete graphical elements at the periphery as **visual distractors**.
5. **Cropped Image Extraction:** Directly cropping the subgraph image from the full graph according to the middle bbox. Notably, the resulting crop may also contain fully visible nodes or edges **beyond** the initially sampled subgraph, as the inner, middle, and outer bboxes can encompass adjacent visible elements surrounding the target region.

Operational Definitions of Full Visibility. To ensure a fair and reproducible evaluation, we employ precise operational definitions for fully visible graphical elements. This methodological rigor eliminates ambiguity, a hallmark of high-quality benchmark design.

- **Fully Visible Node:** A node is "fully visible" if its entire shape and full textual label are visible within the image boundaries, without any truncation.
- **Fully Visible Edge:** An edge is "fully visible" if its full path, arrowhead, connected start and end nodes, and its entire textual label (if present) are fully visible.

B.2. Masked Subpath Query

Motivation of Masked Subpath Query Design. While holistic structural comprehension represents a crucial capability, it is not always a necessary or even practical requirement for all visual reasoning tasks. For instance, addressing a localized query about a specific region within a large, high-resolution graph does not inherently demand a full Graph2Code conversion. In such scenarios, exhaustive parsing can be computationally inefficient and may introduce cascading errors that compromise downstream reasoning.

A more pragmatic and efficient strategy is **targeted reasoning**, where the model selectively localizes and interprets only the subgraph relevant to the query. This paradigm closely parallels Visual Grounding (VG) and serves as a foundation for advanced reasoning agents capable of employing Chain-of-Thought (CoT)-based decomposition, systematically partitioning complex global problems into coherent, localized analytical steps.

To this end, we formulate the Masked Subpath Query task to rigorously examine a model's targeted reasoning competence, achieved by transforming practical VQA instances into a standardized structured representation.

Data Generation Protocol. The question–answer pairs for all Masked Subpath Query tasks are generated through a rigorous, programmatic pipeline:

1. **Path Sampling:** For each source graph, five diverse paths are sampled using a random walk algorithm.
2. **Masking Application:** A masking strategy (prefix, middle, or suffix) is randomly applied to each path to form a structured query.
3. **Subpath Enumeration:** A Depth-First Search (DFS) is executed on the source graph to exhaustively identify all subpaths satisfying the query's structural and complexity constraints, ensuring correctness and completeness of the ground truth.
4. **Query Curation:** Only instances with three or fewer valid ground-truth answers are retained to ensure well-posedness and minimize ambiguity.

Illustrative examples of the three masking strategies are provided below:

1. **Middle Masking:** $n_1 \to n_2 \to \{\} \to [e_1] \to n_6$
2. **Prefix Masking:** $\{k\} \to n_4 \to n_5 \to [e_1] \to n_6$
3. **Suffix Masking:** $n_1 \to n_2 \to [e_1] \to n_3 \to \{k\}$

The statistics of VQA samples for the Masked Subpath Query task are summarized in Tab. 1 and Tab. 2. The distributions of complete path length, masked subpath length and subpath counts (test set) are illustrated in Fig. 1, Fig. 2 and Fig. 3, respectively, covering a diverse range of single- and multi-question cases.

Table 1. Statistics of VQA samples for SCQ sub-task. $\to$ indicates a random sampling process from the complete dataset, applied likewise throughout.

Structurally-Conditioned Masked Subpath Query		Simple	Cyclic
Test	Fully-Merged	4576 $\to$ 1000	485
	Structurally-Matched	4609 $\to$ 1000	263
Train	Fully-Merged	8157 $\to$ 2000	1008
	Structurally-Matched	8137 $\to$ 2000	506

Table 2. Statistics of VQA samples for SSQ sub-task.

Shortest Subpath Query	Total	Simple	Cyclic
Test	1334	1257 (94.23%)	77 (5.77%)
Train	2476	2320 (93.70%)	156 (6.30%)

Prompting Protocol. Masked Subpath Query comprises two sub-tasks. Structurally-Conditioned Query assesses a model's ability to infer masked relational structures under different masking formalisms and subpath complexity constraints. It comprises four conditions formed by combining *Fully-Merged vs. Structurally-Matched* and *Cyclic vs. Simple* subpath settings. Prompt 1 and Prompt 2 illustrate the two evaluation axes introduced above, jointly covering all four task conditions. The Shortest Subpath Query further incorporates an explicit optimization constraint into the evaluation protocol. Prompt 3 exemplifies this sub-task.

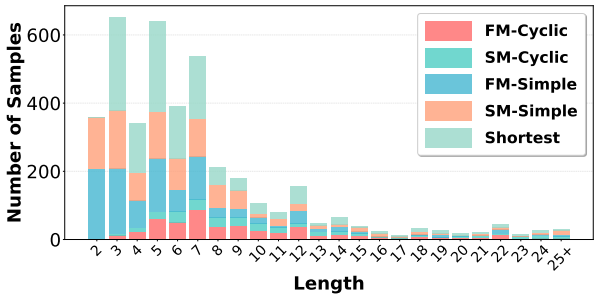


Figure 1. Distributions of path lengths in the test set (stacked bar chart; the same test-set configuration are adopted in all subsequent figures). Path length is defined as the number of nodes along the path (consistent throughout). FM and SM denote Fully-Merged and Structurally-Matched conditions, respectively (hereafter).

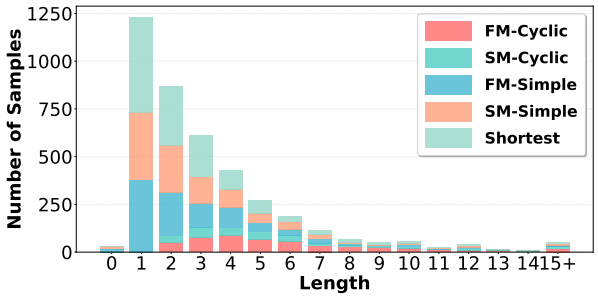


Figure 2. Distributions of masked subpath lengths (stacked bar chart). A value of 0 indicates that the masked segment contains only a single edge label.

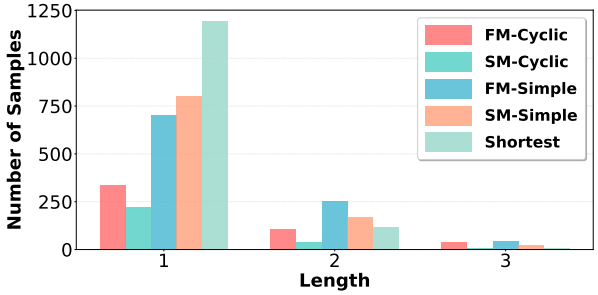


Figure 3. Distributions of subpath counts (grouped bar chart).

Evaluation Protocol. To quantitatively assess model performance on the Masked Subpath Query task, we designed a hierarchical evaluation protocol that computes four key metrics: Precision, Recall, F1-score, and a stricter form of Accuracy that accounts for contextual correctness. To ensure methodological consistency, this same evaluation protocol is also applied to assess the model performance on **Semantic Capability** and **Comprehensive Capability** categories.

1. **Model Output Parsing:** The raw string output from

Prompt 1. Prompt for Masked Subpath Query: Fully-Merged Cyclic Subpath.

Prompt for Fully-Merged Cyclic Subpath Query

Task: According to the flowchart in the picture and the structured path query given below (formatted as **<important>partially masked path</important>**), find all possible subpaths of the masked part (i.e., the steps to be queried), where {} represents a masked `node_label`, and {[]} represents a masked `edge_label`; all {} and {[]}together constitute a masked subpath.

Note: All paths consisting of the masked subpath you need to find, the first node before the masked subpath (if any), and the first node after the masked subpath (if any) are not simple paths, that is, the path must have a **<important>directed cycle</important>**, but each directed cycle can only appear once at most; in another way, there cannot be repeated directed cycles in the path.

REQUIREMENTS:

1. You need to output the entire path you completed, including all node labels and edge labels, and the edge labels need to be wrapped in []. The output path format is consistent with the format of a complete path given above.
2. If there are multiple possible subpaths, output multiple possible paths. Each path is separated by "\n".
3. You can choose whether to think before answering, but your answer must be marked with "**** output **\n**".

[Example1] a partially masked path:

start gaming → {} → [No] → end game

**** output ****

start gaming → continue gaming? → [Yes] → start gaming → [No] → end game

[Example2] a partially masked path:

start gaming → continue gaming? → {2}

**** output ****

start gaming → continue gaming? → [Yes] → start gaming → [No] → end game

Note: You **MUST NOT** omit the **EDGE_LABEL** in the path in your answer, especially check the edge_label of the **BRANCH EDGE** after the **DECISION NODE** (mostly diamond-shaped and with node label ends with a question mark "?"), and **WRAP** the EDGE_LABEL in [].

The partially masked path given is as follows:

{`Input: masked path query`}

The flowchart image given is as follows:

`<image>`

the model is parsed into a structured format to isolate individual predicted paths. Each path is subsequently split into its constituent node and edge labels, resulting in a list of predicted paths, where each path is represented as an ordered list of strings.

2. **Precision, Recall, and F1 Calculation:** Precision and Recall are computed based on sub-sequence matching. A predicted path is considered a successful match if it contains any of the ground-truth paths as a complete, ordered sub-sequence.
   (a) **Precision (Prec.):** The fraction of predicted paths that successfully contain at least one ground-truth path.
   (b) **Recall (Rec.):** The fraction of all ground-truth paths that are successfully contained within at least one of the predicted paths.

Prompt 2. Prompt for Masked Subpath Query: Structurally-Matched Simple Subpath Query.

Prompt for Structurally-Matched Simple Subpath Query

Task: The same as the Task in Prompt 1.

Note: All paths consisting of the masked subpath you need to find, the first node before the masked subpath (if any), and the first node after the masked subpath (if any) are all **simple paths**, that is, the path must **not** have a **<important>DIRECTED CYCLE </important>**; in another way, there cannot be any directed cycle in the path.

REQUIREMENTS:

1. If there are multiple possible subpaths, output multiple possible paths. Each path is separated by "\n".
2. You can choose whether to think before answering, but your answer must be marked with "** output **\n".

[Example1] A partially masked path:

start gaming → {} → {[]} → {} → end game

**** output ****

start gaming → continue gaming? → [No] → start learning → end game

[Example2] A partially masked path:

start gaming → continue gaming? → [No] → {} → {}

**** output ****

start gaming → continue gaming? → [No] → start learning → end game

Note: You **MUST NOT** omit the **EDGE_LABEL** in the path in your answer, especially check the edge_label of the **BRANCH EDGE** after the **DECISION NODE** (mostly diamond-shaped and with node label ending with a question mark "?"), and **WRAP** the EDGE_LABEL in `[]`.

The partially masked path given is as follows:

{`Input: masked path query`}

The flowchart image given is as follows:

`<image>`

   (c) **F1-score (F1):** The harmonic mean of Precision and Recall, providing a single, balanced measure of performance.
3. **Contextual Accuracy Calculation:** A more stringent **Accuracy (Acc.)** metric is computed to ensure that the model's predictions are not only correct but also **contextually faithful** to the original query. It is computed only for predictions that have successfully matched a ground-truth path, following three steps:
   (a) **Mask Position Parsing:** The original masked subpath query is parsed to locate the mask placeholder ({} or $\{n\}$) as *prefix*, *middle*, or *suffix*.
   (b) **Context Isolation:** From the successfully matched predicted path, the identified ground-truth subpath is removed algorithmically, isolating the remaining contextual elements.
   (c) **Context Validation:** These contextual elements are validated against the non-masked portions of the original query. For instance, if the mask was a prefix, the elements following the matched subpath in the prediction are checked against the corresponding elements in the query. A prediction is deemed accurate only if this contextual

Prompt 3. Prompt for Masked Subpath Query: Shortest Subpath Query.

Prompt for Shortest Subpath Query

Task: According to the flowchart in the picture and the structured path query given below (formatted as **<important>masked path</important>**), find all the **shortest subpaths** corresponding to the masked parts {{}} (i.e., the steps to be queried).

Note: The paths consisting of the masked subpath you need to find, the first node before the masked subpath (denoted as start_node), and the first node after the masked subpath (denoted as end_node) are **not necessarily simple paths**. That is, the path may have a **DIRECTED CYCLE**, but each directed cycle can appear **at most once**; in other words, there cannot be **repeated directed cycles** in the found subpaths. However, the masked subpath you need to complete must be the **shortest subpath** between start_node and end_node, regardless of whether this shortest subpath has a directed cycle.

REQUIREMENTS:

1. If there are multiple possible subpaths, output multiple possible paths. Each path is separated by "\n".
2. You can choose whether to think before answering, but your answer must be marked with "`** output **`\n".

[Example 1] A partially masked path:

start gaming → {} → end game

**** output ****

start gaming → continue gaming? → [No] → end game

[Example 2] A partially masked path:

start gaming → {} → continue gaming? → [No] → end game

**** output ****

start gaming → continue gaming? → [Yes] → start gaming → continue gaming? → [No] → end game

The partially masked path given is as follows:

{`Input: masked path query`}

The flowchart image given is as follows:

<image>

validation succeeds.

4. **Metric Aggregation:** The four metrics are aggregated over the dataset to yield the reported results.

C. Visual Category Design Details

C.1. Edge Layout Perception

Motivation of Edge Layout Perception Design. The Edge Layout Perception task evaluates a model's ability to perceive and reason over the spatial arrangements of edges, which are slender, directed structures fundamental to graph topology. This task is motivated by an empirical analysis of contemporary graph parsing models, which reveals that certain edge configurations, such as elongated edges, edges oriented against the dominant flow, and intersecting edges, pose disproportionately high difficulty. By explicitly focusing on these visually and topologically complex scenarios, the task provides a precise measure of a model's robustness in handling challenging structural layouts that commonly induce errors in downstream graph reasoning.

Operational Definitions of Edge Layout Types. The precise, quantitative criteria used to define each edge layout type are detailed below:

- **Long Edges:** An edge is classified as "long" if its bounding box height or width is at least one-third of the corresponding SVG image dimension ($h_{edge} \geq \frac{1}{3}h_{SVG}$ or $w_{edge} \geq \frac{1}{3}w_{SVG}$).
- **Reverse Edges:** An edge is "reverse" if its primary vector opposes the graph's dominant directional flow (*e.g.*, Top-to-Bottom, Left-to-Right), which is determined by parsing directives from the source code.
- **Crossing Edge Pairs:** An edge pair is "crossing" if their vector representations intersect, as determined by a standard line-segment intersection algorithm.

Tab. 3 presents the statistics for these three types.

Table 3. Statistics of VQA samples for ELP task.

Edge layout		Long	Reverse	Crossing
Test	Graphviz	347	369	39
	Mermaid	409	371	121
Train	Graphviz	628	677	63
	Mermaid	762	679	206

C.2. Local Structure Comparison

Motivation of Local Structure Comparison Design. The Local Structure Comparison task examines a model's sensitivity to spatial modifications in graph layouts. Mastery of this capability is crucial for practical applications, including automated diagram version control, visual regression testing, and summarizing structural changes between different process graph versions.

Data Generation Protocol. We automatically generate image pairs to evaluate structural comparison under two complementary sub-tasks.

- **Direct Structure Difference.** For the Direct sub-task, modifications, such as random deletion of edges and any resulting isolated nodes, are applied to the parsed front-end representation (*e.g.*, SVG) of an already rendered graph. This **post-rendering** procedure strictly preserves the spatial layout of all shared elements, thereby evaluating the model's ability to detect explicit, pixel-level visual changes.
- **Indirect Structure Difference.** For the Indirect sub-task, the same modifications are applied to the declarative source code (*e.g.*, Mermaid) prior to rendering. This **pre-rendering** procedure allows the layout engine to recompute element positions, potentially introducing spatial shifts. The design tests layout-invariant reasoning, requiring models to focus on topological changes rather than low-level visual discrepancies.

C.3. Elements Localization

Motivation of Absolute and Relative Localization Design. The Absolute Localization task assesses a model's

core object detection capabilities within graphical structures, establishing a baseline for identifying discrete elements. The Relative Localization task evaluates a model's reasoning over spatial relationships, a prerequisite for interpreting connections, logical flow, and hierarchy in visual graphs. Mastery of both tasks is crucial for accurate analysis of real-world graph structures.

C.3.1. Absolute Localization Sampling Protocol. A detailed description of the element sampling and filtering protocol is provided below.

- **Edge Selection Criteria.** To ensure a fair evaluation on clearly visible elements, we apply a filtering criterion to **ensure a minimum level of visibility**. An edge is sampled only if its smallest bounding box dimension is at least 5.0 pixels ($\min(h, w) \geq 5.0px$).
- **Instance Construction.** For each evaluation instance, 5 nodes and 5 eligible edges are randomly sampled from each graph to form the query set.

C.3.2. Relative Localization. The formal mathematical definitions for all eight spatial relationships based on bounding box coordinates are provided below.

Operational Definitions of Spatial Relationships. Let the bounding box of obj_1 be $(x_{1,1}, y_{1,1}, x_{2,1}, y_{2,1})$ and obj_2 be $(x_{1,2}, y_{1,2}, x_{2,2}, y_{2,2})$.

- **Cardinal Directions:** A relationship is cardinal if there is containment on one axis (*e.g.*, the x-range of obj_1 contains that of obj_2, or vice-versa) and no overlap on the other axis.
  - **DIRECTLY_ABOVE:** $x_{1,1} \leq x_{1,2}$ and $x_{2,1} \geq x_{2,2}$ and $y_{2,1} < y_{1,2}$.
  - **DIRECTLY_BELOW:** $x_{1,1} \geq x_{1,2}$ and $x_{2,1} \leq x_{2,2}$ and $y_{1,1} > y_{2,2}$.
  - **DIRECTLY_LEFT:** $y_{1,1} \leq y_{1,2}$ and $y_{2,1} \geq y_{2,2}$ and $x_{2,1} < x_{1,2}$.
  - **DIRECTLY_RIGHT:** $y_{1,1} \geq y_{1,2}$ and $y_{2,1} \leq y_{2,2}$ and $x_{1,1} > x_{2,2}$.
- **Diagonal Directions:** A relationship is diagonal if there is no overlap on either the x-axis or the y-axis.
  - **UPPER_LEFT:** $x_{2,1} < x_{1,2}$ and $y_{2,1} < y_{1,2}$.
  - **UPPER_RIGHT:** $x_{1,1} > x_{2,2}$ and $y_{2,1} < y_{1,2}$.
  - **LOWER_LEFT:** $x_{2,1} < x_{1,2}$ and $y_{1,1} > y_{2,2}$.
  - **LOWER_RIGHT:** $x_{1,1} > x_{2,2}$ and $y_{1,1} > y_{2,2}$.

Sampling Protocol. To ensure an unambiguous evaluation, we apply two filtering criteria during sampling.

1. **Overlap Filtering:** Element pairs with partial spatial overlap that do not satisfy the strict containment definitions for cardinal directions are excluded.
2. **Semantic Uniqueness Filtering:** Nodes or edges sharing duplicate labels with other elements in the same graph are discarded and resampled.

C.4. Spatial Position Awareness

Motivation of Absolute and Relative Positioning Design. Absolute Positioning evaluates a model's ability to search for and localize elements within a global context, analogous to finding a specific detail on a map or slide. Relative Positioning, in contrast, assesses the model's capacity to describe spatial locations in a coarse, human-interpretable manner, such as "top-right" or "center-left". Together, these tasks support the generation of accurate high-level descriptions of visual scenes.

C.4.1. Absolute Positioning. The model input is a query image of a subgraph and a target image. The model is then required to output the bounding box coordinates of the subgraph within the target image, or "not exists" if it is not found. Furthermore, all query subgraphs are generated using the **inner bbox** crop, ensuring they contain only complete elements to prevent any interference with bounding box prediction.

Operational Definitions of Target Graph. This subtask is administered under two conditions:

- **Full Target Graph:** The target image is a full-sized graph. With a 50% probability, this is the original graph from which the subgraph was extracted; otherwise, it is a different, distractor graph.
- **Thumbnail Target Graph:** The target image is a 5x downscaled version (i.e., 1/5th of the original height and width) of the full graph.

C.4.2. Relative Positioning. The Relative Positioning sub-task, in particular, assesses a model's capacity for coarse, human-interpretable spatial articulation (*e.g.*, describe an object as being in the "top-right corner" of an image). This capability is important for generating natural language descriptions and for high-level scene understanding.

Operational Definitions of Target Graph. A virtual grid is imposed over the target image. Given a query subgraph, the model must output the indices of all grid cells that its bounding box overlaps. The cells are indexed sequentially (left-to-right, top-to-bottom) starting from 1. This sub-task is also presented in two scale-dependent conditions:

- **Full Graph.** A grid with 250×250 pixel cells is imposed on the target image.
- **Thumbnail Graph.** A grid with 50×50 pixel cells is imposed on the target image.

For both positioning tasks, including full and thumbnail graph conditions provides significant diagnostic power to evaluate a model's **scale invariance**, a key characteristic of a well-designed diagnostic benchmark.

C.5. Visual Attribute Perception

Data Generation Protocol. We first define candidate sets of each visual attribute type for nodes and edges. For each graph, we randomly sample visual attributes for every node and edge, and render them into the corresponding **visual Mermaid and Graphviz code** representations. When constructing VQA question–answer pairs, for each graph we randomly select one candidate attribute from the candidate sets across every node and edge attribute type, and parse the rendered Mermaid/Graphviz code to obtain the ground-truth answer.

D. Semantic Category Design Details

This section provides the detailed implementation of our **two-stage data construction and verification pipeline** for the Semantic Capability tasks.

D.1. Data Construction Stage

The construction stage involves two primary prompting strategies for GPT-4o.

Prompt for Semantic Query Generation. To convert structured path queries into natural language questions, we provide GPT-4o with the structured path and the corresponding prompt in Prompt 4.

Prompt 4. Prompt for Semantic Query Generation

Prompt for Semantic Query Generation

Task: For the flowchart in the picture and the given path below, please design a question to ask about the masked part - i.e. {} or {n} (n represents the number of nodes contained in the masked part) - of the path according to the following requirements.

REQUIREMENTS for Question Generation:

1. The question must contain a problem scenario designed by you, and your questioning style must simulate a question that an actual user may ask, rather than asking directly.
2. The question must completely and accurately contain all the node labels and edge labels that are not masked in the given path. In the output question, these corresponding node labels and edge labels are marked with [].
3. If the masked part contains the number of nodes, that is, {n}, then the question you designed should clearly state that the path to be completed contains n steps.
4. In your answer, you only need to output the question you designed, and your answer does not need to include the given path.

Given structured path:
`{Input: masked path query}`
Your designed question:

Prompt for Semantic-Rewrite Query Generation. For this sub-task, the generated semantic questions are revised by GPT-4o using the following prompt in Prompt 5 to paraphrase the bracketed entity labels.

D.2. Data Verification Stage

The integrity of Semantic Query generation is crucial, as it directly impacts the quality of the subsequent

Prompt 5. Prompt for Semantic-Rewrite Query Generation

Prompt for Semantic-Rewrite Query Generation

Task: Please rewrite each bracketed content in the given semantic question while preserving their original meaning. The rewritten content should also be wrapped in square brackets []. Output your rewritten semantic question in the required format.

REQUIREMENTS for Question Rewriting:

1. Keep the same meaning but use different wording.
2. Maintain similar length and complexity.
3. Use natural and fluent language.
4. Keep technical terms if present.
5. Do not omit any string content wrapped in [], and each rewritten content should also be wrapped in square brackets [].
6. Output your rewritten semantic question directly, do not add any other text.

[Example]:
Given semantic question:
After [Baked Cake], what are the important steps that I need to complete in order to reach [Cut off the hard edges]?
Rewritten semantic question:
I have finished [Cake is baked], what are the important actions that I need to take to reach [Trim the tough edges]?

Given semantic question:
`{Input: semantic question}`
Rewritten semantic question:

Semantic-Rewrite Query task. We enforce a strict, automated verification procedure with the following rules:

1. **Path Parsing:** The original structured query path is split by the $\rightarrow$ delimiter into an ordered list of string elements (node labels and edge labels). The corresponding generated semantic question has all [] markers removed for content analysis.
2. **Element Presence Check:** The script iterates through the list of path elements and checks whether each path element appears as a substring in the processed semantic question. A flexible matching algorithm (*e.g.*, case-insensitive and whitespace-tolerant) is used instead of exact string matching. If any structured-path element is missing from the generated question, the QA pair is discarded.
3. **Numeric Constraint Verification:** If the structured query contains a numeric constraint, such as 5, the script verifies that the corresponding number (*e.g.*, "5") or its word equivalent (*e.g.*, "five", "Five") appears in the generated question. Conversely, if the structured query does not include a numeric constraint (i.e., only contains placeholders such as {}), but the generated question introduces a restrictive number or its word equivalent, the script replaces such numeric terms with a unified placeholder *"n"*, indicating the absence of explicit length constraints.
4. **Answer Ambiguity Filtering:** All QA pairs with more than three valid ground-truth answers are discarded to reduce ambiguity in the evaluation.

A QA pair is only accepted into the final benchmark if it successfully passes all applicable verification rules. Statistics for the filtering process are provided in Tab. 4.

Table 4. Filtering Statistics for Semantic Verification

Data Verification of Semantic Query Generation				Original #QA Pairs	Retained #QA Pairs	Retention Rate(%)
Test	Fully-Merged Masked SubPath Query	Simple Path Query	Graphviz	1000	553	55.30
			Mermaid	1000	546	54.60
		Complicate Path Query	Graphviz	480	167	34.79
			Mermaid	480	170	35.42
	Shortest Subpath Query		Graphviz	500	228	45.60
			Mermaid	500	227	44.60
Train	Fully-Merged Masked SubPath Query	Simple Path Query	Graphviz	2000	1055	52.75
			Mermaid	2000	1041	52.05
		Complicate Path Query	Graphviz	1008	402	39.88
			Mermaid	1008	387	38.39
	Shortest Subpath Query		Graphviz	2446	1121	45.83
			Mermaid	2446	1133	46.32

E. Comprehensive Category Design Details

This section provides the detailed implementation of the **two-stage data generation and verification stages** for the Comprehensive capability task.

E.1. Data Construction Stage

E.1.1. Non-Semantic Query Sub-Task. The generation process is based on Masked Subpath Query and its corresponding structured path. We design two main classes of semi-structured templates.

Element-Centric Class 1: What/Which Templates. These templates are designed to ask for specific elements (nodes and edges) that meet certain structural, visual and semantic criteria within the subpath.

- **Template Definitions:**
  1. *According to the flowchart in the picture, find* {} *[] () in the subpath.*
  2. *According to the flowchart in the picture, which ()s are [] () in order in the subpath.*
- **Parametrization Process:**
  1. **() - element type**: The primary element is first chosen to be either "node"/"nodes" or "edge"/"edges".
  2. **[] - style type**: A visual style is then specified as a **single** attribute or a **composition** of multiple attributes defined in the Visual Attribute Perception task (*e.g.*, a *red-filled, diamond-shaped* node).
  3. **{} - quantity option**: A quantity specifier is selected with a 50% probability for each option:
     - **"all the"**: Identify all elements that satisfy the specified requirement.
     - **"the {n}th"**, where n is an integer: To test for correctness and handling of **out-of-bounds queries**, n is chosen from $[1, L+1]$ with 80% probability and from $[L+1, L+2]$ with 20% probability, where L is the number of ground-truth elements. Models are expected to output "**not exist**s" when tested by out-of-bounds queries.

Path-Centric Class 2: How Templates. These templates focus on counting and identifying subpaths that contain elements with specific properties.

- **Template Definition:**
  - *According to the flowchart in the picture, find* {} *subpaths that include* {} *[] ().*
- **Parametrization Process:**
  1. **() - element type**: The element type ("nodes" or "edges") is selected.
  2. **The second {} - style quantity option**: Specifies a quantifier for the elements, such as "the most", or "the least".
  3. **[] - style type**: A visual style is selected (*e.g.*, a *yellow-bordered with dashed-bordered* edge).
  4. **The first {} - path quantity option**: Determines the number of paths to find. Selected with a 50% probability for each option:
     - **"all"**: Identify all subpaths that satisfy the specified requirement.
     - **"at least {n}"**, where n is an integer: To test for correctness and handling of **out-of-bounds queries**, n is chosen from $[1, M+1]$ with 80% probability and from $[M+1, M+2]$ with 20% probability, where M is the maximum possible count. Models are expected to **only output all the constraint-satisfying subpaths** when tested by out-of-bounds queries.

E.1.2. Semantic Query. For each non-semantic query and its corresponding structured path, we use GPT-4o to generate a natural-language reformulation. The prompt designs in Prompts 6 and 7 explicitly require the model to integrate all structural, visual, and semantic constraints in a fluent and coherent manner, accurately surface every unmasked node and edge label, and introduce controlled linguistic diversity. Furthermore, the prompts **bind visual attributes to explicit semantic roles** (*e.g.*, *red-filled, dashed-bordered nodes* instantiated as *urgent states*), thereby enriching graph interpretation and aligning queries with realistic usage scenarios.

E.1.3. Semantic-Rewrite Query. We take the validated

Prompt 6. Prompt for Comprehensive Semantic Query Element-Centric Class1 - What/Which Generation

Prompt for Comprehensive Semantic Query Element-Centric Class1 - What/Which Generation

Task: For the flowchart in the picture, the given path and the original question below, please design a question to ask about the masked part - i.e. {} or {n} (n represents the number of nodes included in the masked part) - of the path according to the following requirements.

REQUIREMENTS for Question Rewriting:

1. The question must contain a problem scenario designed by you, and your questioning style must simulate a question that an actual user may ask, rather than asking directly like the original question.
2. The question must completely and accurately contain all the node labels and edge labels that are not masked in the given path. In the output question, these corresponding node labels and edge labels are marked with [].
3. Your question must completely contain all the details of the original question, and no details can be omitted.
4. If the masked part contains the number of nodes, that is, {n}, then the scope of the question to be asked must be clearly stated in the question you designed, for example, {n} → node1 means that the question range is n steps before node1, and node1 → {n} means that the question range is n steps after node1.
5. You can choose whether to think before answering, but your answer must be marked with "[your designed question]\n". In [your designed question], just output the question you designed, the given path and the original question should not be included.
6. The parts of your designed question that need to be emphasized can be wrapped with **, as shown in the following example.

The output format example is as follows:

Example1:

[Given structured path]

Baked Cake → {} → Cut off the hard edges

[Original question]

According to the flowchart in the picture, which node is red-filled with solid-bordered node in order in the subpath.

[Your designed question]

After [Baked Cake], what are the **important steps** that I need to complete in order to reach [Cut off the hard edges]? All the nodes in the picture that are filled with red and have solid borders represent **important steps**. Please **output these important steps in order**.

Example2:

[Given structured path]

{3} → Succeed baking a cake

[Original question]

According to the flowchart in the picture, find the 1th solid-bordered edge in the subpath.

[your designed question]

In order to achieve [Succeed baking a cake], what is the first **important process** that I need to do **in the three steps before this**? All the solid lines in the picture represent **important processes**.

[Given structured path]

{Input: masked path query}

[original question]

{Input: original question}

Your designed question:

questions from the Semantic sub-task and use GPT-4o

Prompt 7. Prompt for Comprehensive Semantic Query Path-Centric Class2 - How Generation

Prompt for Comprehensive Semantic Query Path-Centric Class2 - How Generation

Task: The same as the Task in Prompt 6.

REQUIREMENTS for Question Rewriting:

The same as the REQUIREMENTS in Prompt 6.

The output format example is as follows:

Example1:

[Given structured path]

Baked Cake → {} → Cut off the hard edges

[Original question]

According to the flowchart in the picture, find at least 2 paths that include least green-bordered nodes.

[Your designed question]

After [Baked Cake], what steps do I need to complete in order to reach [Cut off the hard edges]? Please tell me **at least two** step paths that include **least difficult steps**. Note: All green-bordered nodes in the picture represent **difficult steps**.

Example2:

[Given structured path]

{3} → Succeed baking a cake

[Original question]

According to the flowchart in the picture, find at least 2 paths that include least blue-bordered edges.

[Your designed question]

In order to achieve [Succeed baking a cake], I need to **complete three consecutive steps before this**. Please help me find **at least two** such three-step paths that include **least critical processes**. Note: All blue-bordered edges in the picture represent **critical processes**.

[Given structured path]

{Input: masked path query}

[Original question]

{Input: original question}

Your designed question:

for a second round of paraphrasing. The prompt in Prompt 8 encourages more linguistic variety and phrasing where the semantic content is highly related but does not require a direct string match with the graph labels.

Table 5. Filtering Statistics for Element-Centric Query.

Class 1 - Which, What	Renderer	Original #QA Pairs	Retained #QA Pairs	Retention Rate
Test	Graphviz	5958	3517 → 1000	59.03%
	Mermaid	5924	3444 → 1000	58.14%
Train	Graphviz	10611	6121 → 2000	57.69%
	Mermaid	10539	6313 → 2000	59.90%

E.2. Data Verification Stage

To ensure the quality of semantic queries, which in turn affects semantic-rewrite queries, we apply the following rule-based filters to GPT-4o outputs.

Prompt 8. Prompt for Comprehensive Semantic-Rewrite Query Generation

Prompt for Comprehensive Semantic-Rewrite Query Generation

Task: The same as the Task in Prompt 5.

REQUIREMENTS for Question Rewriting:
The same as the REQUIREMENTS in Prompt 5.

The output format example is as follows:

Example:
[Given semantic question]
After [Baked Cake], what are the **important steps** ...

[Your rewritten semantic question]
I have completed [Cake is baked], what are the **critical steps** ...

[Given semantic question]
`{Input: semantic question}`
Your rewritten semantic question:

Table 6. Filtering Statistics for Path-Centric Query.

Class 2 - How	Renderer	Original #QA Pairs	Retained #QA Pairs	Retention Rate
Test	Graphviz	1762	1027 → 1000	58.29%
	Mermaid	1805	1036 → 1000	57.40%
Train	Graphviz	3204	1804	56.30%
	Mermaid	3237	1893	58.48%

Verification Rules for Element-Centric Queries:

1. **Path Parsing:** The structured path is parsed into a list of N string elements (node/edge labels).
2. **Element Presence Check:** For each element in the list from the previous step, we perform a flexible substring check to ensure it is present in the generated semantic question. This check is robust to minor variations (*e.g.*, stemming, pluralization) rather than requiring an exact string match.
3. **Retention Criterion:** A question-answer pair is retained only if **all** elements from the unmasked subpath are successfully found within the semantic question; otherwise, it is discarded.

Statistics summarizing the filtering process for Element-Centric Query are presented in Tab. 5.

Verification Rules for Path-Centric Queries: In addition to the rules above, Class 2 questions undergo a numerical check:

1. **Numerical Constraint Handling:** If the non-semantic template contains a numerical constraint (*e.g.*, "at least 5"), the generated semantic question must include the corresponding digit (5) or its textual representation (*e.g.*, "five", "Five"), case-insensitively.
2. **Retention and Focus Criterion:** Class-2 question–answer pairs are discarded if they do not satisfy the above rules. Additionally, any remaining pairs where the number of ground-truth answers **exceeds 3** are removed to maintain task focus.

Tab. 6 reports the statistics of the Path-Centric Query filtering process. As illustrated in Figs. 4 to 9, the distributions of complete-path lengths, masked-subpath lengths, and valid subpath counts for element-centric class1 and path-centric class2 queries. (test set, Graphviz rendered; similar results for Mermaid) remain highly consistent between non-semantic and semantic variants, demonstrating the effectiveness and **distributional fidelity** of our two-stage data curation pipeline.

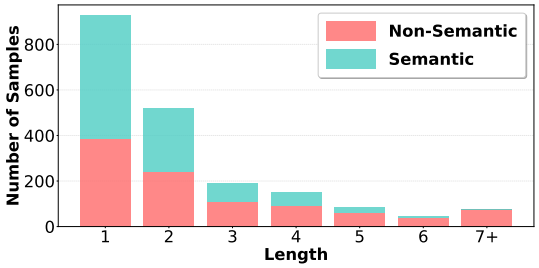


Figure 4. Distributions of element-centric class1 unmasked subpath lengths (stacked bar chart).

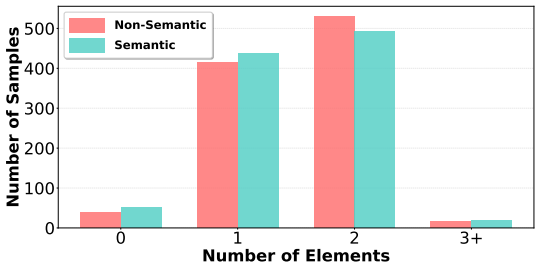


Figure 5. Distributions of required masked-subpath element counts (nodes/edges) for element-centric class1 (stacked bar chart); 0 indicates counterfactual or answerless cases.

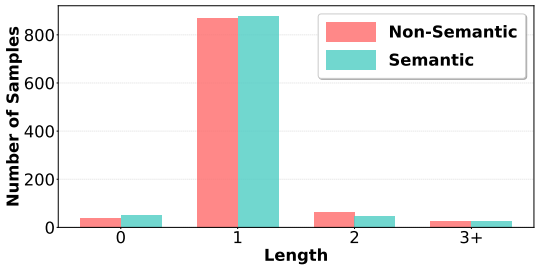


Figure 6. Distributions of element-centric class1 valid subpath counts (grouped bar chart).

F. Details of Experiment Results

The aggregated results summarizing twelve fine-grained tasks spanning four high-level capability categories are presented in Tab. 7. Across modalities, the test set of the benchmark comprises **34,483 Graphviz and 34,631**

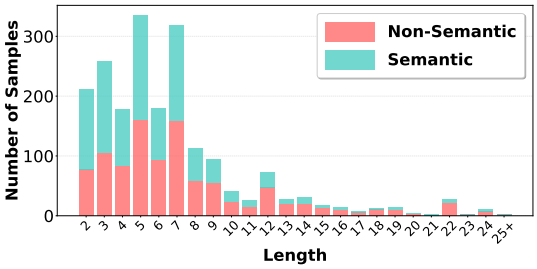


Figure 7. Distributions of path-centric class2 path lengths (stacked bar chart).

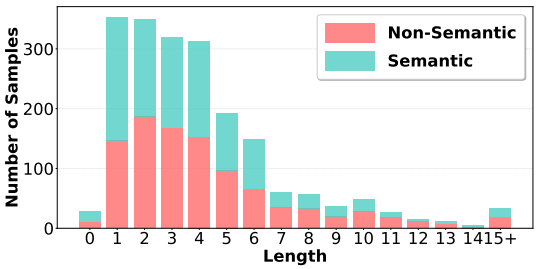


Figure 8. Distributions of path-centric class2 masked subpath lengths (stacked bar chart).

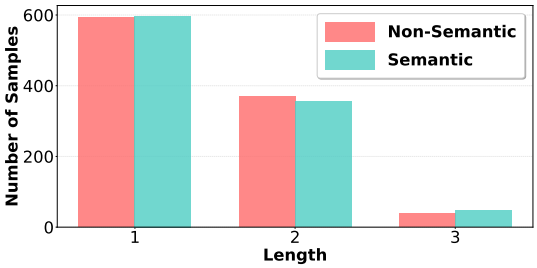


Figure 9. Distributions of path-centric valid subpath counts.

Mermaid VQA samples in total. Following a training-to-testing ratio of approximately 1.8:1, the specific distribution within the test set is as follows: the Structural tasks contain 10,082 samples for both formats; the Visual tasks include 16,399 Graphviz and 16,581 Mermaid samples; the Semantic tasks consist of 2,002 Graphviz and 1,968 Mermaid samples; and the Comprehensive tasks contribute 6,000 samples for each format.

F.1. Detailed Results on Structural Category

F.1.1. Analysis of Structural Comprehension Task. Structural Comprehension reveals a universal performance degradation from full-graphs to subgraphs, stemming from models' inability to handle incomplete peripheral elements acting as visual distractors.

Tab. 8 provides a deeper analysis of this drop in the **Graph Parsing** task. The degradation is notably asymmetric: precision drops far more significantly than recall. For example, GPT-4o's edge precision plummets from 87.58% to 71.30%, while its recall remains high (86.18% to 83.30%). This widening precision/recall gap indicates that visual distractors primarily **induce hallucinations (false positives) rather than omissions**. This vulnerability is exacerbated in the more challenging **Graph2Code** task (Tab. 9). Beyond accuracy, structural quality metrics decay sharply. All models show a clear drop in graph similarity.

Collectively, these fine-grained metrics expose a fundamental limitation: **models lack robustness to partial views**, leading to specific errors in **hallucination** (low Prec) and **directional biases** (low Dir. Acc.).

F.1.2. Analysis of Masked Subpath Query Task. The **Masked Subpath Query** task (Tab. 10) advances the evaluation from holistic perception to fine-grained structural reasoning, posing a substantially greater challenge.

In **Structurally-Conditioned Query**, performance is highly sensitive to query formalism and topological complexity. As reported in Sec. 5.2.1, performance drops sharply when moving from simple to cyclic paths. The detailed metrics further reveal divergent behavior across model scales. For large models, the explicit topological prior in **Structurally-Matched** queries serves as an effective reasoning cue, mitigating complexity-induced degradation. For instance, Gemini 2.5's accuracy drop narrows from -20.11% (FM) to -8.41% (SM). In contrast, this strict formalism confounds smaller models like InternVL3.5-8B, whose accuracy collapses from 12.84%/40.35% (FM-Cyclic/FM-Simple) to 6.87%/25.06% (SM-Cyclic/SM-Simple).

The **Shortest Subpath Query** introduces an additional optimization constraint, revealing a critical weakness in contextualized structural reasoning. Although top models like o3 and Gemini achieve strong F1 scores (81.53% and 77.04%), their corresponding contextual accuracy (Acc.) lags considerably (59.24% and 45.37%). This disparity indicates that while models can identify the correct shortest subpath (high Prec./Rec.), they **struggle to preserve contextual fidelity**, often hallucinating adjacent non-masked elements. Such inconsistency underscores a fundamental limitation in reasoning coherently under strict structural constraints.

F.1.3. Discussion on Vision Transformer (ViT) Topology Gaps. We suggest that current Transformer-based visual encoders inherently struggle with these topological properties compared to natural scene understanding due to **three primary factors**. First, the standard patch tokenization and continuous embedding process "soften" discrete connectivity; the use of fixed patch strides can inadvertently fragment edge-node intersections, obscuring the precise junctions critical for graph parsing. Second, the self-attention mechanism is designed to optimize for soft semantic similarity rather than hard topological constraints, often leading the model to rely on layout-based shortcuts rather

Table 7. Summary of model performance on **DiGraphHal-Bench**, averaged over Mermaid and Graphviz (same below). This table corresponds to Fig. 2 in the main paper. Superscripts: $*^{Spec.}$ denotes the specialist models and $*^{CL}$ denotes the curriculum model. The same convention applies to other abbreviations. Values in bold indicate the best performance for each metric, and "-" denotes failure cases after SFT; this convention applies throughout.

Models	Structural		Visual					Semantic		Comprehensive		
	Graph Parsing (GP)	Masked Subgraph Query (MSQ)	Edge Layout Perception (ELP)	Local Structure Comparison (LSC)	Elements Localization (EL)	Spatial Position Awareness (SPA)	Visual Attribute Perception (VAP)	Semantic Query (S-SQ)	Semantic-Rewrite Query (S-SRQ)	Non-Semantic Query (C-NSQ)	Semantic Query (C-SQ)	Semantic-Rewrite Query (C-SRQ)
	F1	F1	F1	F1	Acc.	F1	F1	F1	F1	F1	F1	F1
GPT-4o	86.95	39.74	17.30	66.30	61.29	39.40	48.77	56.89	52.18	37.46	37.17	31.15
GPT-5	**93.41**	83.50	53.34	**87.77**	**81.79**	**64.52**	82.38	**76.37**	**76.62**	**71.28**	60.52	51.08
o3	90.05	80.20	46.02	84.17	76.61	63.98	79.45	71.63	71.20	58.49	49.09	46.07
Gemini 2.5	89.58	**84.04**	**65.41**	83.98	81.72	53.42	**89.16**	75.94	72.63	55.26	**72.97**	**59.71**
Claude 4.5	86.47	47.23	35.32	64.74	68.42	36.31	57.09	64.13	60.71	27.24	39.85	31.40
GLM-4.5V	86.50	44.58	22.21	73.01	61.01	37.65	52.27	54.95	52.40	38.13	35.72	31.06
Qwen2.5-VL-72B	82.07	37.68	26.73	58.14	67.11	44.21	38.35	49.32	44.28	40.02	41.29	33.04
LLaVA-OV-72B	65.70	20.77	0.63	46.42	37.75	43.96	32.25	30.04	23.18	20.31	22.52	18.29
Qwen2.5-VL-7B	73.73	26.80	1.15	39.74	26.07	33.07	39.66	37.72	35.18	14.88	17.30	14.02
LLaVA-OV-7B	38.97	7.39	0.01	14.00	33.48	36.56	22.93	14.22	12.04	8.96	9.43	8.44
InternVL3.5-8B	80.45	35.27	4.48	45.36	53.71	37.77	39.87	38.01	33.04	22.19	28.66	22.94
Qwen2.5-VL-7B $^{Spec.}$	86.37	35.93	26.61	69.59	61.32	27.74	44.93	43.94	36.34	41.03	41.96	30.97
Qwen2.5-VL-7B ^{CL}	69.40	16.72	3.33	40.26	-	22.34	39.31	32.44	39.02	37.77	35.92	29.26

Table 8. Detailed model performance on **Graph Parsing Sub-Task**. (n) denotes that both Graphviz and Mermaid test sets contain n VQA samples, consistent across all subsequent tables. Superscripts: $*^{s}$ denotes the structural specialist model.

Models	Full-Graph (1000)						Subgraph (1000)					
	Edge			Node			Edge			Node		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
GPT-4o	87.58	86.18	86.73	96.34	97.50	96.71	71.30	83.30	74.88	89.45	97.77	92.44
GPT-5	94.13	94.77	94.39	97.70	97.68	97.61	**91.32**	96.63	**92.98**	95.17	97.09	**95.52**
o3	93.11	93.67	93.32	97.78	97.68	97.65	85.22	95.35	88.70	92.96	**99.25**	95.41
Gemini 2.5	**96.63**	**97.18**	**96.83**	**99.14**	**98.91**	**98.94**	90.14	**97.25**	92.53	90.73	97.49	92.81
Claude 4.5	94.43	93.99	94.09	98.66	98.19	98.33	74.80	88.97	79.14	90.64	97.37	93.02
GLM-4.5V	93.52	92.62	92.91	98.73	98.41	98.46	74.07	91.25	79.71	89.47	99.24	93.33
Qwen2.5-VL-72B	85.62	83.62	84.27	94.25	96.90	95.21	66.45	87.52	73.25	88.02	97.62	91.54
LLaVA-OV-72B	62.51	59.63	60.71	76.88	90.60	82.34	47.16	63.10	52.04	72.53	94.82	80.17
Qwen2.5-VL-7B	82.39	76.48	78.92	95.98	94.62	94.88	52.80	72.52	58.89	62.75	59.43	58.18
LLaVA-OV-7B	37.12	34.41	35.11	-	-	-	28.78	41.93	32.55	49.53	57.37	51.38
InternVL3.5-8B	84.44	82.71	83.31	90.29	95.55	92.30	57.61	85.69	66.01	84.93	97.60	89.75
Qwen2.5-VL-7B ^{s}	78.23	78.04	77.79	93.32	92.17	92.48	80.45	87.86	82.17	**96.18**	94.24	94.62
Qwen2.5-VL-7B^{CL}	74.86	68.50	71.03	90.35	86.59	87.98	59.65	56.84	54.90	89.40	74.08	77.87

Table 9. Detailed model performance on **Graph2Code Sub-Task**, where GED, Sim, and Dir. Acc. represent graph edit distance, graph similarity (calculated using $Sim = 1 - nGED$, where $nGED = \frac{GED}{\text{\#nodes + \#edges}}$), and direction accuracy (measuring the correctness of directional strings generated in the predicted code relative to the ground-truth code), respectively.

Models	Full Graph (1000)					Sub Graph (1000)				
	GED↓	Sim↑	Acc.	F1	Dir. Acc. (324)	GED↓	Sim↑	Acc.	F1	Dir. Acc. (324)
GPT-4o	4.46	93.25	86.80	91.60	60.80	3.16	81.98	71.80	80.82	65.54
GPT-5	**2.22**	**96.95**	**90.03**	92.83	98.62	**1.34**	**92.72**	**86.43**	**90.57**	93.52
o3	2.95	95.93	86.43	90.33	98.19	2.41	87.16	76.57	82.33	85.65
Gemini 2.5	2.75	96.50	88.94	91.29	**99.39**	3.48	81.94	69.35	76.47	**96.45**
Claude 4.5	2.82	95.66	89.35	**92.99**	57.19	5.06	74.28	61.04	70.60	55.71
GLM-4.5V	4.51	93.01	85.28	89.82	82.53	3.46	78.24	65.47	73.98	67.28
Qwen2.5-VL-72B	8.30	86.45	70.25	79.80	50.47	3.73	79.23	66.54	76.34	50.16
LLaVA-OV-72B	19.29	72.22	51.91	62.44	51.85	10.85	67.39	51.83	62.71	51.85
Qwen2.5-VL-7B	11.47	82.46	70.41	78.26	47.18	5.46	73.69	60.43	71.20	42.93
LLaVA-OV-7B	27.99	48.16	25.45	35.39	27.94	9.21	56.91	34.08	43.40	30.62
InternVL3.5-8B	6.05	90.81	79.77	85.95	66.79	4.65	75.73	60.90	70.15	68.68
Qwen2.5-VL-7B^{s}	8.57	88.36	76.32	84.07	62.04	2.14	89.14	81.80	87.87	62.35
Qwen2.5-VL-7B^{CL}	15.51	75.75	59.31	69.16	50.31	6.71	65.71	51.04	62.54	48.31

than underlying connectivity. Finally, the reliance on non-topology-invariant positional encodings undermines global topological consistency, as spatial coordinates in a rendered graph do not directly map to the invariant relational structures typical of symbolic graphs.

Table 10. Detailed model performance on **Structurally-Conditioned Query** and **Shortest Subpath Query Sub-Tasks**.

Models	Structurally-Conditioned Query																Shortest Subpath Query (1334)			
	Fully-Merged Cyclic Path (263)				Fully-Merged Simple Path (1000)				Structurally-Matched Cyclic Path (485)				Structurally-Matched Simple Path (1000)							
	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
GPT-4o	17.47	20.09	25.65	20.91	27.45	45.20	48.86	45.56	18.08	22.41	31.78	25.02	44.65	53.67	67.78	57.41	35.90	47.38	55.74	49.79
GPT-5	59.06	70.36	74.02	70.20	92.43	93.51	**93.93**	**93.16**	84.21	84.59	**86.50**	**84.51**	**93.42**	**93.77**	**95.25**	**93.90**	44.72	75.51	76.44	75.72
o3	56.77	65.10	65.53	63.68	88.89	90.70	91.45	90.43	74.39	74.89	75.41	73.95	90.79	91.58	92.56	91.40	**59.24**	**81.24**	**82.43**	**81.53**
Gemini 2.5	**74.32**	**77.38**	**78.63**	**76.09**	**94.43**	**94.93**	89.45	91.00	**84.52**	**84.90**	84.95	83.43	92.93	93.60	92.89	92.64	45.37	77.34	77.15	77.04
Claude 4.5	15.07	17.63	37.89	20.76	58.11	66.80	75.89	68.21	26.12	26.73	39.51	28.76	61.88	66.75	79.95	69.61	42.21	45.77	57.83	48.81
GLM-4.5V	26.64	37.28	34.33	35.03	43.32	62.34	60.64	60.74	26.26	38.40	37.36	37.50	22.47	56.79	57.51	56.52	13.14	32.87	34.25	33.11
Qwen2.5-VL-72B	12.37	17.38	21.58	18.07	29.38	42.15	49.63	43.23	15.76	19.32	31.56	21.81	52.73	58.93	70.52	61.81	25.07	40.94	50.51	43.50
LLaVA-OV-72B	6.83	9.61	11.36	9.96	32.12	37.07	43.14	37.53	4.86	10.84	15.10	11.89	24.05	24.41	24.39	26.64	5.73	15.51	24.23	17.83
Qwen2.5-VL-7B	5.62	9.48	10.40	9.46	23.71	39.65	38.35	37.94	5.05	12.66	14.69	13.19	20.03	43.80	48.63	44.61	8.00	26.99	32.85	28.78
LLaVA-OV-7B	2.73	4.03	4.60	4.12	4.17	8.83	9.20	8.49	0.65	2.34	4.26	2.75	3.59	15.47	25.32	16.71	0.70	4.26	7.30	4.86
InternVL3.5-8B	12.84	14.49	12.87	15.23	40.35	47.62	58.98	50.25	6.87	20.58	28.12	22.07	25.06	40.93	50.81	42.85	21.91	44.61	52.58	45.95
Qwen2.5-VL-7B ^{s}	19.27	22.16	20.11	20.46	57.44	57.46	57.58	56.55	9.84	11.18	11.35	11.10	33.88	36.15	35.19	35.63	55.96	57.42	55.23	55.89
Qwen2.5-VL-7B ^{CL}	7.55	9.02	8.13	8.37	45.72	47.62	42.44	44.00	2.80	6.21	5.89	5.99	21.17	24.19	21.99	22.62	1.69	2.77	2.54	2.61

Table 11. Detailed model performance on **Edge Layout Perception Task**. $(n|m)$ denotes the number of VQA samples in the Graphviz and Mermaid test sets, respectively. Superscripts: $*^{v}$ denotes the visual specialist model. Two metric sets are reported for Crossing Edge Pairs: strict (exact match) and partial (relaxed) evaluation.

Models	Long (347|409)			Reverse (369|371)			Crossing (39|121)					
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Partial Prec.	Partial Rec.	Partial F1
GPT-4o	16.56	33.61	20.14	20.19	29.01	22.54	0.83	1.66	1.10	14.21	28.28	17.35
GPT-5	**63.99**	79.98	68.15	66.56	74.70	68.45	14.20	14.60	14.13	32.87	34.34	32.72
o3	60.65	68.17	61.18	62.82	70.03	64.47	5.74	5.65	5.40	19.11	22.33	19.44
Gemini 2.5	63.20	**90.37**	**69.82**	**75.48**	**86.23**	**78.21**	**24.85**	**37.73**	**27.54**	**63.97**	**81.99**	**68.87**
Claude 4.5	23.63	32.33	25.09	37.84	36.01	36.35	0.00	0.00	0.00	11.70	9.88	10.37
GLM-4.5V	33.14	28.58	29.78	50.12	55.28	51.00	5.80	6.44	5.89	29.67	36.66	31.31
Qwen2.5-VL-72B	26.15	42.62	29.39	39.18	47.16	41.01	2.12	4.03	2.72	16.14	20.32	16.88
LLaVA-OV-72B	0.00	0.00	0.00	1.83	1.93	1.77	0.00	0.00	0.00	0.12	1.28	0.22
Qwen2.5-VL-7B	0.08	0.29	0.11	3.15	3.01	2.71	0.42	0.42	0.42	0.83	0.83	0.83
LLaVA-OV-7B	0.03	0.08	0.04	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3.5-8B	1.93	10.00	3.03	6.15	29.03	9.01	0.09	0.21	0.12	1.99	6.64	2.66
Qwen2.5-VL-7B ^{v}	36.62	34.33	34.09	41.69	37.22	37.40	2.11	2.11	2.11	15.50	14.23	14.57
Qwen2.5-VL-7B ^{CL}	0.56	1.36	0.62	9.05	7.24	7.56	0.00	0.00	0.00	4.18	3.42	3.60

Table 12. Detailed model performance on **Local Structure Comparison Task**.

Models	Direct Structure Difference (992)						Indirect Structure Difference (992)					
	Edge			Node			Edge			Node		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
GPT-4o	50.25	35.57	39.82	83.78	81.95	82.39	76.08	58.44	63.38	80.93	79.91	79.59
GPT-5	**80.79**	**67.98**	**71.81**	**98.19**	**97.76**	**97.85**	**87.76**	**83.50**	**83.92**	**97.95**	**97.40**	**97.49**
o3	75.23	61.90	65.55	96.65	96.31	96.30	85.93	77.02	78.94	96.42	95.97	95.91
Gemini 2.5	72.79	59.93	63.70	96.94	96.73	96.70	83.50	81.96	81.28	94.46	94.50	94.25
Claude 4.5	65.35	56.81	57.57	74.04	75.52	73.83	64.48	67.94	61.56	65.84	70.03	65.99
GLM-4.5V	66.97	48.27	53.20	91.34	88.60	89.22	78.25	62.78	66.26	85.58	83.47	83.34
Qwen2.5-VL-72B	47.62	36.52	39.06	75.99	74.94	74.92	59.40	58.32	54.47	64.60	66.51	64.11
LLaVA-OV-72B	24.84	18.74	19.62	79.95	75.75	76.82	30.23	33.58	27.75	64.07	63.35	61.49
Qwen2.5-VL-7B	36.79	26.24	27.94	73.36	69.81	69.74	24.17	25.66	21.91	41.05	41.93	39.35
LLaVA-OV-7B	3.01	15.20	4.86	29.06	38.82	30.24	5.02	26.31	8.15	9.81	33.03	12.76
InternVL3.5-8B	34.83	28.25	28.01	80.88	79.17	79.18	27.63	36.89	26.59	47.06	53.50	47.64
Qwen2.5-VL-7B ^{v}	55.44	47.13	49.17	89.47	86.35	87.19	55.53	58.66	55.27	88.95	86.96	86.70
Qwen2.5-VL-7B ^{CL}	38.69	26.81	29.15	66.31	62.37	62.89	33.98	28.89	28.52	42.24	42.47	40.45

F.2. Detailed Results on Visual Category

F.2.1. Analysis of Holistic Layouts and Structural Differences Perception. In **Edge Layout Perception** (Tab. 11), models struggle with global edge properties. For Long and Reverse edges, modest F1 scores mask asymmetric performance. For example, Gemini's highest Long edge F1 (69.82%) stems from high recall (90.37%) but low precision (63.20%). This indicates that **the primary failure mode is hallucination** (false positives), manifesting as predicting numerous incorrect edges rather than omitting relevant ones.

The performance collapse on Crossing Edge Pairs is clarified by strict vs. partial metrics. Models fail to identify the intersecting pair (*e.g.*, Gemini's 27.54% strict F1), yet the higher partial F1 (68.87%) shows they can detect constituent edges. This points to a failure in relational spatial reasoning, not just perception.

In **Local Structure Comparison** (Tab. 12), metrics confirm the performance trade-off from the main paper (Sec. 5.2.2) and reveal a persistent "edge blindness". Models are universally more proficient at detect-

Table 13. Detailed model performance across two tasks: **Elements Localization** and **Spatial Position Awareness**.

Models	Elements Localization						Spatial Position Awareness											
	Absolute Localization (922|931)				Relative Localization (922|931)		Absolute Positioning (497)						Relative Positioning (497)					
	node		edge		node	edge	Full Target			Thumbnail Target			Full Target			Thumbnail Target		
	Acc @0.5	Acc @0.25	Acc @0.5	Acc @0.25	Acc.	Acc.	Exist. Acc	BBox Acc @0.5	BBox Acc @0.25	Exist. Acc	BBox Acc @0.5	BBox Acc @0.25	Prec.	Rec.	F1	Prec.	Rec.	F1
GPT-4o	0.99	3.25	0.01	0.32	72.25	50.34	97.50	11.81	35.74	93.39	0.00	0.00	51.44	32.11	37.76	50.70	36.96	41.05
GPT-5	12.83	32.11	2.33	12.69	**90.42**	73.17	96.04	**42.45**	**68.41**	72.95	0.00	0.00	**71.59**	**58.95**	**67.58**	63.86	75.61	66.45
o3	13.41	32.03	1.36	9.80	89.90	63.33	96.15	37.97	64.28	78.07	0.00	0.00	70.21	55.71	60.04	**65.00**	**78.04**	**67.91**
Gemini 2.5	15.89	34.01	**2.47**	**13.73**	87.08	**76.37**	96.89	10.43	31.39	82.72	0.00	0.00	62.36	46.00	50.86	64.41	52.97	55.99
Claude 4.5	5.49	16.05	0.17	2.24	76.58	60.27	98.37	4.13	21.74	92.18	0.00	0.00	59.53	39.30	45.35	29.53	29.11	27.28
GLM-4.5V	1.32	5.61	0.04	0.36	68.34	53.68	**99.08**	5.21	18.36	65.43	0.00	0.00	50.36	40.64	42.89	37.07	32.17	32.42
Qwen2.5-VL-72B	**17.71**	**37.24**	0.81	4.43	73.85	60.36	76.26	30.53	57.47	70.64	0.00	0.00	55.64	47.92	49.20	41.77	42.32	39.23
LLaVA-OV-72B	0.79	4.22	0.05	0.86	27.96	47.54	98.10	3.04	14.70	**93.84**	0.00	0.00	55.62	41.76	45.32	50.41	42.60	42.60
Qwen2.5-VL-7B	0.34	3.57	0.12	0.92	27.87	24.27	98.70	0.91	6.35	90.33	0.00	0.00	48.25	33.39	34.95	37.78	30.80	31.20
LLaVA-OV-7B	0.03	0.80	0.02	0.12	34.80	32.17	77.76	0.86	7.56	74.65	0.00	0.00	31.25	30.89	26.79	28.51	60.71	46.34
InternVL3.5-8B	0.85	3.97	0.01	0.29	67.57	39.85	98.15	0.82	6.00	90.90	0.00	0.00	44.55	36.37	37.27	42.65	40.37	38.28
Qwen2.5-VL-7B ^{v}	8.58	24.78	1.83	3.56	60.90	61.73	98.80	17.62	44.50	88.60	0.00	0.00	38.41	26.80	28.46	35.71	26.02	27.02
Qwen2.5-VL-7B ^{CL}	0.33	2.04	0.15	0.38	–	–	67.10	0.71	6.74	59.40	0.00	0.00	32.06	26.10	27.13	31.98	14.37	17.54

ing node changes than edge changes. For instance, GPT-5's strong Indirect F1 (90.71%) averages near-perfect node detection (97.49% F1) with far weaker edge detection (83.92% F1). This **perceptual bias** against filamentary structures remains a critical barrier.

F.2.2. Analysis of Localization and Spatial Awareness. In **Elements Localization** (Tab. 13), a clear performance gap emerges. While models are competent at Relative Localization, the previously observed "**edge blindness**" persists; top models are significantly more accurate at identifying the relative positions of nodes than edges (*e.g.*, GPT-5 scores 90.42% vs. 73.17%, respectively). This proficiency with coarse relational understanding contrasts sharply with performance on Absolute Localization, which is exceedingly poor. At an IoU threshold of 0.5, accuracy for nodes is below 18% for all models and drops to near-zero for edges. This demonstrates that models can process coarse spatial relationships but **lack the fine-grained visual grounding** for precise coordinate prediction, a weakness especially pronounced for filamentary structures.

This weakness is further amplified in the **Spatial Position Awareness** task (Tab. 13). The Absolute Positioning sub-task reveals a profound lack of scale invariance. While models can determine subgraph existence, their ability to predict a bounding box is already limited on full-sized graphs (*e.g.*, GPT-5 at 42.45% Acc@0.5) and collapses entirely to 0% on thumbnails.

In striking contrast, the Relative Positioning sub-task shows a complex interaction with scale. Moving to thumbnails causes a drop in precision for most models (*e.g.*, GPT-5: 71.59% $\rightarrow$ 63.86%) but a significant increase in recall (58.95% $\rightarrow$ 75.61%). This trade-off results in a net gain in F1 score for many top models (*e.g.*,o3: 60.04% $\rightarrow$ 67.91%). This suggests that for coarse spatial judgments, scale invariance not only exists, but downscaled (thumbnail) views may even be more effective, despite this effectiveness coming at the cost of introducing more false positive errors.

F.2.3. Analysis of Visual Attributes Perception. Within Node Attribute Perception, models generally exhibit high proficiency (Tab. 14). Top-tier models like GPT-5 and Gemini consistently achieve high F1 scores across all four conditions: Shape, Border Color, Fill Color, and Border Style, with Gemini's F1 scores ranging from 84.77% to 95.21%. This indicates a robust capability to identify the visual properties of salient, well-defined objects. However, even for nodes, performance of all models **dips on boundary-related attributes** like Border Color compared to Shape or Fill Color, hinting at an underlying weakness with linear features.

This weakness becomes more pronounced in Edge Attribute Perception, where **a clear performance gap** emerges between top-tier and smaller models. For most MLLMs, the transition leads to a severe performance collapse, confirming a widespread "edge blindness" phenomenon. However, leading models such as Gemini, GPT-5, and o3 demonstrate better resilience. While their performance on edges is lower than on node shape or fill attributes, it remains relatively strong and is notably comparable to their performance on node border attributes. This nuanced result reveals a critical **perceptual bias in processing the attributes of filamentary structures compared to enclosed shapes**, a hurdle that top-tier models are better equipped to overcome.

F.3. Detailed Results on Semantic Category

The evaluation of Semantic Capability assesses a model's ability to ground natural language queries to a graph's content. As noted in the main paper (Sec. 5.2.3), our findings (Tab. 15) reveal a critical vulnerability in

Table 14. Detailed model performance on **Visual Attributes Perception Task**.

Models	Node (1000)												Edge (1000)					
	Shape			Border Color			Fill Color			Border Style			Color			Style		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
GPT-4o	81.04	72.09	74.71	50.52	48.86	46.87	70.27	64.28	64.55	72.54	51.28	55.68	40.82	40.89	39.01	43.24	36.53	35.17
GPT-5	**95.52**	91.18	91.97	75.94	79.79	75.73	89.31	90.77	**88.79**	90.37	88.07	87.93	80.07	80.67	79.35	79.13	78.16	77.94
o3	94.84	95.53	94.64	74.99	74.31	72.22	**89.44**	89.92	88.39	88.28	86.58	85.91	77.65	75.63	75.50	74.63	73.17	71.72
Gemini 2.5	95.46	**95.70**	**95.21**	**84.19**	**88.35**	**84.77**	87.60	**91.87**	88.13	**92.18**	**92.49**	**91.40**	**86.59**	**89.16**	**87.40**	**88.73**	**90.82**	**89.45**
Claude 4.5	92.63	77.03	81.09	59.62	55.53	53.92	87.13	80.94	81.22	84.70	50.95	59.25	51.52	55.87	51.10	60.44	33.62	39.52
GLM-4.5V	92.17	88.60	88.87	31.71	27.50	26.73	86.11	75.93	78.07	74.73	64.07	60.36	40.67	41.01	37.40	51.10	50.79	44.64
Qwen2.5-VL-72B	63.98	59.48	60.01	35.38	33.15	30.82	60.37	59.31	57.12	71.97	60.63	62.01	20.55	20.49	18.55	33.60	31.59	29.87
LLaVA-OV-72B	72.28	64.50	65.57	28.86	33.27	28.46	55.37	58.36	54.10	52.44	44.03	43.17	11.37	18.38	12.76	20.46	23.71	20.55
Qwen2.5-VL-7B	76.23	76.28	73.66	25.89	40.82	28.43	51.20	64.37	53.29	55.76	56.99	51.72	14.01	40.97	18.81	32.29	47.09	36.27
LLaVA-OV-7B	37.07	43.62	37.64	17.73	30.40	20.34	35.23	49.63	38.82	41.70	44.22	39.77	5.83	14.13	7.35	14.35	20.90	16.08
internvl 8b	77.07	75.28	74.95	29.22	40.78	31.12	54.50	64.56	56.16	53.99	54.91	49.87	17.79	31.30	20.72	32.89	37.11	32.68
Qwen2.5-VL-7B ^{v}	77.65	78.19	76.73	29.88	36.56	29.64	60.70	62.54	58.54	51.66	63.57	55.01	30.32	31.77	29.11	37.21	48.08	40.63
Qwen2.5-VL-7B ^{CL}	75.61	70.90	70.84	25.70	37.30	26.71	53.77	57.85	51.34	51.56	51.31	46.62	17.74	39.17	21.62	34.86	47.31	37.84

Table 15. Detailed model performance on **Semantic Category**. Superscripts: $*^{m}$ denotes the semantic specialist model.

Models	Semantic Query (1001|984)				Semantic-Rewrite Query (1001|984)			
	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
GPT-4o	47.10	58.81	57.00	56.89	31.56	53.45	53.02	52.18
o3	64.62	73.72	71.44	71.63	56.95	73.42	71.21	71.20
Gemini 2.5	**71.37**	**77.36**	**76.47**	**75.94**	**66.92**	**74.42**	**73.08**	**72.63**
Claude 4.5	50.69	65.95	65.09	64.13	44.54	62.22	62.34	60.71
GLM-4.5V	48.53	57.41	54.37	54.95	42.78	54.56	51.98	52.40
Qwen2.5-VL-72B	39.94	51.38	50.60	49.32	29.88	46.01	46.77	44.28
LLaVA-OV-72B	28.51	32.67	29.10	30.04	15.21	25.22	22.44	23.18
Qwen2.5-VL-7B	36.23	40.91	36.44	37.72	17.21	38.40	33.92	35.18
LLaVA-OV-7B	11.59	15.29	13.98	14.22	6.88	13.12	11.66	12.04
InternVL3.5-8B	35.92	40.30	37.54	38.01	26.13	35.32	32.43	33.04
Qwen2.5-VL-7B ^{m}	43.89	47.44	42.48	43.94	30.87	39.41	35.01	36.34
Qwen2.5-VL-7B ^{CL}	33.60	35.77	31.04	32.44	31.93	42.50	37.44	39.02

MLLMs: a heavy reliance on superficial lexical matching over genuine semantic understanding.

In **Semantic Query**, which requires direct lexical mapping, Tab. 15 details the performance hierarchy. Top-tier models like Gemini 2.5 achieve high, balanced scores (75.94% F1; 77.36% Prec./76.47% Rec./71.37% Acc.). The failure of open-source models is not a specific bias but a comprehensive one; for instance, Qwen2.5-VL-72B's low F1 (49.32%) stems from equally weak precision (51.38%) and recall (50.60%), indicating a fundamental inability both to identify correct elements (low Rec.) and to suppress false positives (low Prec.). Moreover, its contextual accuracy is even lower (39.94%), underscoring the difficulty of maintaining **contextual fidelity**, consistent with the observations reported in Sec. F.1.2.

Semantic-Rewrite Query reveals how grounding fails under linguistic variation. While the main paper notes the divergence between F1 and accuracy drops, Tab. 15 offers a sharper diagnostic. For top models like o3, paraphrasing barely impacts subpath grounding ability (F1 stable: 71.63% $\rightarrow$ 71.20%; Prec./Rec. similar). However, its contextual reasoning collapses, reflected in a steep accuracy decline (64.62% $\rightarrow$ 56.95%). **models retain grounding but lose coherent reasoning once linguistic cues shift.** For other models like GPT-4o, both grounding (F1: 56.89% $\rightarrow$ 52.18%) and reasoning (Acc.: 47.10% $\rightarrow$ 31.56%) degrade sharply.

F.4. Detailed Results on Comprehensive Category

The Comprehensive Capability category serves as the ultimate test of an MLLM's ability to holistically integrate structural, visual, and semantic information. As noted in the main paper (Sec. 5.2.4), this integrative reasoning remains highly challenging. The detailed results in Tab. 16 provide a more granular view of these failures, particularly the divergent performance across query types and the impact of linguistic complexity.

The **Non-Semantic Query** baseline exposes fundamental **reasoning biases** across models. A pronounced disparity emerges between element-centric class1 and path-centric class2 reasoning, suggesting distinct underlying processing mechanisms. For example, o3 demonstrates strong performance on element-centric queries (69.84% F1) but exhibits substantial degradation on path-centric (47.13% F1). In contrast, certain models such as Gemini and Claude display unexpected weaknesses in class1, potentially attributable to instruction-following ambiguity rather than perceptual limitations.

When language grounding is introduced through the **Semantic** and **Semantic-Rewrite** variants, the evaluation focus shifts from structural reasoning to joint multi-modal alignment. Under this setting, models exhibit new failure modes rather than merely amplified biases. Specifically, element-centric queries become particularly vulnerable to linguistic ambiguity, resulting in **a sharp degradation in both reasoning fidelity and visual–semantic grounding**. For instance, Gemini's class1 F1 drops drastically from 69.79% (Semantic) to 46.56% (Semantic-Rewrite), accompanied by a pronounced decline in precision (68.10% $\rightarrow$ 44.37%).

In contrast, path-centric class2 exhibits the divergent trend described in the main paper, where linguistic variation benefits reasoning for certain models. Notably, the phenomenon in which Semantic-Rewrite performance

Table 16. Detailed model performance on **Comprehensive Category**. Superscripts: $*^{c}$ denotes the comprehensive specialist model.

Models	Non-Semantic Query (1000)							Semantic Query (1000)							Semantic-Rewrite Query (1000)						
	Element-Centric			Path-Centric				Element-Centric			Path-Centric				Element-Centric			Path-Centric			
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
GPT-4o	43.74	48.93	44.98	26.87	30.43	32.84	29.94	38.33	47.05	40.69	31.36	34.44	36.51	33.65	27.12	36.63	29.61	26.16	33.02	36.06	32.68
GPT-5	**75.85**	**76.53**	**75.50**	66.44	69.13	68.90	67.05	60.72	65.27	61.87	58.99	61.32	60.84	59.16	38.77	43.76	40.09	60.25	64.15	64.41	62.07
o3	69.92	71.48	69.84	44.76	48.20	50.63	47.13	61.43	66.14	62.61	34.74	37.03	36.69	35.57	43.01	46.47	42.12	46.68	50.87	53.15	50.01
Gemini 2.5	36.45	38.89	36.92	**73.28**	**75.90**	**76.41**	**73.59**	**68.10**	**74.24**	**69.79**	**74.07**	**78.12**	**78.83**	**76.15**	**44.37**	**52.70**	**46.56**	**67.88**	**73.06**	**74.23**	**72.86**
Claude 4.5	28.15	30.97	28.69	24.13	26.26	28.83	25.78	46.10	59.31	49.63	28.35	30.56	33.91	30.07	29.27	43.52	32.95	25.94	30.21	34.33	29.84
GLM-4.5	42.04	42.63	41.63	31.36	36.37	36.16	34.62	39.54	42.89	40.24	31.07	33.16	32.00	31.20	28.61	32.77	29.62	30.57	34.36	33.71	32.49
Qwen2.5-VL-72B	49.88	50.71	49.55	29.32	32.13	31.96	30.48	48.80	55.66	50.65	31.92	33.94	32.67	31.93	32.38	38.27	33.86	28.59	34.42	33.47	32.21
LLaVA-OV-72B	24.30	30.47	26.03	13.20	15.58	15.17	14.59	23.51	46.10	26.17	15.95	18.32	22.27	18.86	16.70	26.26	18.57	10.39	17.97	20.72	18.00
Qwen2.5-VL-7B	13.88	14.53	13.77	13.27	16.81	16.74	15.99	17.12	17.08	16.98	16.30	18.97	18.13	17.61	12.09	12.05	12.00	8.35	17.53	16.12	16.03
LLaVA-OV-7B	7.68	7.32	7.38	5.37	10.02	9.45	10.53	6.49	6.47	6.45	9.24	13.55	12.75	12.40	5.45	5.72	5.45	3.14	12.33	11.58	11.42
InternVL3.5-8B	25.35	28.97	26.01	15.75	19.40	19.59	18.36	31.54	37.18	32.92	23.83	25.98	25.35	24.40	22.37	27.36	23.48	14.83	23.94	23.70	22.39
Qwen2.5-VL-7B ^{c}	55.08	53.75	53.41	28.23	33.14	27.23	28.65	57.44	54.72	55.09	31.27	33.59	27.05	28.82	37.94	35.90	36.06	26.55	30.04	24.39	25.87
Qwen2.5-VL-7B ^{CL}	53.72	51.64	51.67	25.76	28.24	21.86	23.86	49.40	47.71	47.53	27.57	28.93	22.13	24.31	37.77	36.61	36.31	24.54	26.44	20.19	22.20

Table 17. Detailed ablation results on **Comprehensive Category** (F1).

Model (Qwen2.5-VL-7B)	Non-Semantic Query							Semantic Query							Semantic-Rewrite Query						
	Element-Centric			Path-Centric				Element-Centric			Path-Centric				Element-Centric			Path-Centric			
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
Base	13.88	14.53	13.77	13.27	16.81	16.74	15.99	17.12	17.08	16.98	16.30	18.97	18.13	17.61	12.09	12.05	12.00	8.35	17.53	16.12	16.03
Structural	29.24 (+15.36)	32.13 (+17.60)	29.56 (+15.79)	23.76 (+10.49)	26.71 (+9.90)	26.98 (+10.24)	25.49 (+9.50)	27.05 (+9.93)	28.34 (+11.26)	27.03 (+10.05)	22.87 (+6.57)	24.88 (+5.91)	23.39 (+5.26)	22.93 (+5.32)	17.80 (+5.71)	18.19 (+6.14)	17.60 (+5.60)	14.26 (+5.91)	19.35 (+1.82)	17.96 (+1.84)	17.67 (+1.64)
Visual	26.99 (+13.11)	39.93 (+25.40)	27.14 (+13.37)	14.56 (+1.29)	16.57 (-0.24)	16.00 (-0.74)	15.46 (-0.53)	29.94 (+12.82)	36.10 (+19.02)	31.54 (+14.56)	22.09 (+5.79)	23.45 (+4.48)	24.58 (+6.45)	22.64 (+5.03)	21.49 (+9.40)	26.23 (+14.18)	22.46 (+10.46)	14.38 (+6.03)	18.85 (+1.32)	18.99 (+2.87)	17.89 (+1.86)
Semantic	12.75 (-1.13)	12.56 (-1.97)	12.47 (-1.30)	17.16 (+3.89)	22.02 (+5.21)	19.55 (+2.81)	19.81 (+3.82)	17.62 (+0.50)	17.31 (+0.23)	17.39 (+0.41)	26.04 (+9.74)	31.14 (+12.17)	**29.48 (+11.35)**	**28.84 (+11.23)**	11.00 (-1.09)	10.84 (-1.21)	10.90 (-1.10)	13.61 (+5.26)	26.38 (+8.85)	23.08 (+6.96)	23.59 (+7.56)
Comprehensive	**55.08 (+41.20)**	**53.75 (+39.22)**	**53.41 (+39.64)**	**28.23 (+14.96)**	**33.14 (+16.33)**	**27.23 (+10.49)**	**28.65 (+12.66)**	**57.44 (+40.32)**	**54.72 (+37.64)**	**55.09 (+38.11)**	**31.27 (+14.97)**	**33.59 (+14.62)**	27.05 (+8.92)	28.82 (+11.21)	**37.94 (+25.85)**	35.90 (+23.85)	36.06 (+24.06)	**26.55 (+18.20)**	**30.04 (+12.51)**	**24.39 (+8.27)**	**25.87 (+9.84)**
Curriculum	53.72 (+39.84)	51.64 (+37.11)	51.67 (+37.90)	25.76 (+12.49)	28.24 (+11.43)	21.86 (+5.12)	23.86 (+7.87)	49.40 (+32.28)	47.71 (+30.63)	47.53 (+30.55)	27.57 (+11.27)	28.93 (+9.96)	22.13 (+3.99)	24.31 (+6.70)	37.77 (+25.68)	**36.61 (+24.56)**	**36.31 (+24.31)**	24.54 (+16.19)	26.44 (+8.91)	20.19 (+4.07)	22.20 (+6.17)

surpasses Semantic is validated across all metrics. For o3, both F1 (35.57% → 50.01%) and, importantly, accuracy (34.74% → 46.68%) increase substantially. This suggests that for path-centric tasks, **semantic perturbation can disrupt reliance on spurious lexical cues and thereby promote a more robust and generalizable reasoning process**.

Despite these isolated improvements, the overall modest performance, together with the consistent disparity between accuracy and F1, indicates that achieving robust, holistic multi-modal integration remains a fundamentally unresolved challenge.

F.5. Detailed Analysis on SFT Results

Our SFT experiments demonstrate that specialist training significantly enhances fine-grained reasoning and mitigates key hallucinations, though persistent deficits and training challenges remain.

F.5.1. Mitigation of Visual Hallucination. SFT effectively reduces visual hallucinations.

- **Mitigation of "Edge Blindness":** The visual specialist markedly narrows the node–edge perceptual gap. In Tab. 14, its F1 for edge attributes (*e.g.*, edge color: 29.11%) surpasses the baseline's F1 for node attributes (*e.g.*, node border color: 28.43%).
- **Layout-Invariant Reasoning:** The visual specialist effectively reproduces the layout-invariant reasoning patterns characteristic of top-tier models. As shown in Tab. 12, its performance improves or remains comparable under layout perturbations (edge: Direct F1 49.17% → Indirect F1 55.27%; node: Direct F1 87.19% → Indirect F1 86.70%), in contrast to the baseline, which exhibits substantial degradation (edge: 27.94% → 21.91%; node: 69.74% → 39.35%).
- **Localization Precision:** As shown in Tab. 13, the visual specialist achieves substantial improvements in fine-grained localization accuracy. In **Absolute Localization**, Acc@0.5 rises from 0.34% → 8.58% for nodes, with an even larger gain at the stricter Acc@0.25 threshold (3.57% → 24.78%). Similar trends are observed in **Relative Localization** (27.87% → 60.90% for nodes; 24.27% → 61.73% for edges) and **Absolute Positioning** (Full Target), where BBox Acc@0.5 markedly improves from 0.91% to 17.62%.

F.5.2. Enhancement of Complex Fine-Grained Reasoning. Specialist training substantially strengthens both topological and semantic reasoning capabilities.

- **Topological Reasoning:** As shown in Tab. 10, the structural specialist yields notable gains on ambiguous **Fully-Merged (FM) Query** (*e.g.*, FM-SP F1: 37.94%

→ 56.55%) and **Shortest Subpath Query** (28.78% → 55.89%), evidencing improved topology reasoning.

- **Semantic-Neutral Reasoning:** In Tab. 16, the comprehensive specialist achieves consistent gains across both element-centric and path-centric **Comprehensive Non-Semantic** tasks, indicating improved robustness to semantic neutrality and structural ambiguity.
- **Semantic Generalization:** As reported in Tab. 15, the semantic specialist demonstrates enhanced generalization, with the accuracy gap between **Semantic Query** and **Semantic-Rewrite** narrowing from 19.02% to 13.02%. The comprehensive specialist further reinforces this trend, achieving substantial performance gains across all three **Comprehensive** tasks.

F.5.3. Remaining Deficits. Despite SFT, critical limitations persist, leaving substantial room for advancement.

- **Persistent Failure:** Models still fail key challenges. Although the structural specialist improves performance on accuracy of **Structurally-Matched** queries, it worsens performance on Prec./Rec./F1 (*e.g.*, SM-SP F1: 44.61% → 35.63%) (Tab. 10). Furthermore, even the visual specialist scores 0.00% Acc@0.5 on **Absolute Positioning (Thumbnail)** (Tab. 13), confirming SFT does not solve the lack of scale invariance.
- **Catastrophic Forgetting:** The curriculum model provides clear evidence of catastrophic forgetting. It consistently underperforms the baseline on foundational tasks (*e.g.*, Graph2Code Sub F1: 62.54% vs. 71.20% baseline, Tab. 9; Semantic Query F1: 32.44% vs. 37.72% baseline, Tab. 15). However, it improves on its final training step (*e.g.*, Comprehensive element-centric Semantic Query F1: 47.53% vs. 16.98% baseline, Tab. 16), confirming that sequential training leads to degradation of previously learned foundational skills due to catastrophic forgetting.

F.5.4. Roadmap for Enhanced Graph Reasoning. Beyond standard SFT, several promising avenues exist to mitigate structural hallucinations and enhance the efficacy of graph reasoning. We propose a possible technical roadmap focused on two strategic directions. First, we advocate for the adaptation of established hallucination-mitigation techniques into topology-aware variants, specifically designed to suppress structural inconsistencies at the perception source. Second, we emphasize the development of "thinking-with-image" perception-reasoning loops, which allow models to iteratively refine their reasoning process through multi-step verification. These strategic trajectories provide a viable paradigm shift toward structurally-grounded graph reasoning in MLLMs.

F.6. Detailed Results on Ablation Study

Our ablation analysis (Tab. 17) disentangles the contributions of foundational capabilities to integrative multimodal reasoning. The results highlight their essential role and exhibit distinct, capability-specific effects across tasks as presented in Sec. 5.2.6 of the main text.

Foundational capabilities underpin cross-capability generalization. Decomposed metrics demonstrate that structural, visual, and semantic competencies constitute prerequisite foundations for higher-level reasoning.

- The **structural specialist** delivers the most consistent improvements, enhancing all metrics across all six sub-tasks (*e.g.*, C-NSQ class1 F1 +15.79%; class2 Acc. +10.49%). This establishes structural reasoning as a broadly transferable capability across both element- and path-centric settings.
- The **visual specialist** shows a more localized impact, primarily benefiting class1 sub-tasks. Its gains are driven largely by substantial recall increases (*e.g.*, C-NSQ class1 Rec. +25.40% vs. Prec. +13.11%), indicating that visual training chiefly sharpens element discovery by reducing false negatives.

Semantic-only training selectively benefits path-centric reasoning. The semantic specialist exhibits diverging effects across tasks. It systematically degrades class1 performance (*e.g.*, C-NSQ Prec. -1.13%, Rec. -1.97%) while substantially improving class2 outcomes, particularly under semantic ambiguity (C-SQ Acc. +9.74%, Prec. +12.17%, Rec. +11.35%; C-SRQ Acc. +5.26%, Prec. +8.85%, Rec. +6.96%). These findings suggest that semantic supervision strengthens path-level inference yet simultaneously undermines the fine-grained visual–structural grounding needed for element-level reasoning, potentially due to an overreliance on linguistic regularities learned during training.

The comprehensive specialist surpasses the curriculum learner. The Acc., Prec., and Rec. trends clarify the "comparable performance" stated in the main paper.

- The **comprehensive specialist** attains the highest and most balanced gains (*e.g.*, C-NSQ C1 Acc./Prec./Rec. all rising by 39% or more), underscoring the effectiveness of direct integrative training.
- The **curriculum learner** matches the specialist only on class1 (*e.g.*, C-NSQ Prec. +39.84%) but performs clearly worse on class2 (*e.g.*, C-SQ F1 +6.70% vs. +11.21% for the specialist). This discrepancy indicates that naive sequential training is fundamentally inadequate for integrative reasoning, plausibly because successive stages induce **catastrophic forgetting** that disrupts previously acquired capabilities.

Table 18. Impact of Flow Orientation on Model Performance.

Flow Orientation (TB = Top-to-Bottom, LR = Left-to-Right, etc.)	Tasks				
	Graph2Code		Shortest Subpath Query		ELP
	Sim.	Dir. Acc.	F1	Acc.	F1
TB	94.72	98.68	56.90	42.50	17.46
BT	86.86	13.64	43.85	26.19	20.35
LR	94.41	100	49.42	32.89	31.32
RL	85.68	11.63	32.89	22.36	7.86

Table 19. Performance Discrepancy between Hierarchical and Cluttered Layouts.

Layout	Tasks			
	Graph2Code	Shortest Subpath Query		ELP
	Sim.	F1	Acc.	F1
Hierarchical	97.62	65.30	51.04	26.55
Cluttered	92.96	44.05	32.79	21.76

F.7. Summary on Counterintuitive and Thought-Provoking Findings

Our comprehensive analysis reveals a spectrum of findings that range from counterintuitive anomalies to thought-provoking insights into multimodal reasoning.

Specifically, prefix masking experiments uncover a pronounced **causal asymmetry**: MLLMs exhibit significantly higher proficiency in forward cause $\rightarrow$ effect extension than in reverse inference. This discrepancy is likely rooted in the intrinsic directional biases of next-token prediction objectives, where models favor sequential generation over structural backtracking (*e.g.*, the forward-biased extension in Fig. 20).

We also identify an intricate **semantic-visual trade-off** (main text line 527; Supp. line 751), where semantic perturbations can paradoxically improve graph grounding. By diminishing the model's reliance on linguistic shortcuts, these perturbations force a more rigorous alignment with visual evidence, suggesting that dominant language priors may suppress latent visual cues.

Furthermore, our experiments highlight two pivotal structural challenges: (i) **Anisotropic visual attention and layout sensitivity**: As quantified in Tab. 18 and Tab. 19, we observe profound layout-driven performance variances. Non-canonical layouts frequently induce structural failures, while cluttered visualizations further amplify topological errors by obscuring node-edge connectivity and disrupting global structural coherence. This suggests that current encoders remain overly sensitive to the geometric regularity of the rendering. (ii) **Non-monotonic scaling of robustness**: We demonstrate that increased model scale does not strictly yield enhanced topological integrity. Counterintuitively, larger models occasionally exhibit heightened vulnerability to pixel-level perturbations (main text line 456), indicating a potential trade-off between local sensitivity and global structural abstraction. This finding underscores that scaling model capacity alone is insufficient for guaranteeing robust topological reasoning in the absence of specialized structural constraints.

G. Case Study

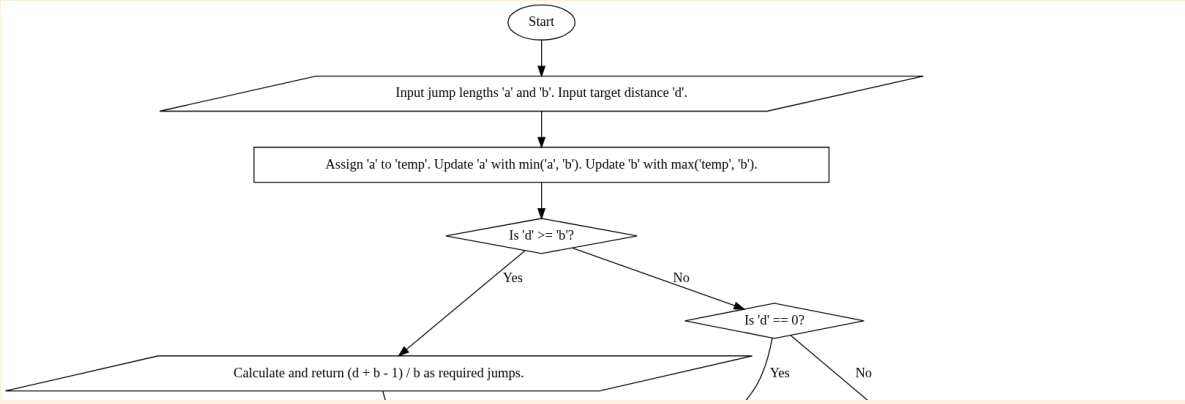
To intuitively illustrate the scope and challenges of the evaluation tasks in DiGraphHal-Bench, we present a set of qualitative case studies. As summarized in Tab. 20, these cases encompass all twelve fine-grained tasks across the four high-level capability categories, offering visual evidence of model behavior, performance differences, and representative failure modes, identifying boundary completion (Figs. 10 and 11), long-range path-continuity, edge-attribute, compositional-constraint (Figs. 23 to 26), and answerless hallucinations (Figs. 21 and 23).

Table 20. Qualitative Case Studies for DiGraphHal-Bench Tasks.

Capability	**Task**	**Case**
Structural	Structural Comprehension (SC)	Figs. 10 and 11
	Masked Subpath Query (MSQ)	Fig. 12
Visual	Edge Layout Perception (ELP)	Fig. 13
	Local Structure Comparison (LSC)	Fig. 14
	Elements Localization (EL)	Fig. 15
	Spatial Position Awareness (SPA)	Fig. 16
	Visual Attributes Perception (VAP)	Fig. 17
Semantic	Semantic Query (S-SQ)	Figs. 18 and 19
	Semantic-Rewrite Query (S-SRQ)	Fig. 20
Comprehensive	Non-Semantic Query (C-NSQ)	Figs. 21 to 23
	Semantic Query (C-SQ)	Figs. 24 and 25
	Semantic-Rewrite Query (C-SRQ)	Fig. 26

Case Study Analysis and Summary. These cases reveal several consistent challenges for current MLLMs: (i) Structural hallucination and long-range reasoning failures, where models hallucinate out-of-view elements in partial subgraphs and struggle to maintain path continuity in cyclic or topologically complex structures; (ii) Pervasive edge blindness, as models largely succeed on node-level perception but fail to reliably discriminate fine-grained edge attributes such as direction, style, or crossings, exposing limitations in existing MLLMs for filamentary structures; (iii) Semantic misalignment under compositional constraints, where models violate counting or extremal conditions (*e.g.*, step limits, "most") and rely on superficial lexical cues rather than performing joint structural–semantic reasoning to accurately capture the intended relationships.

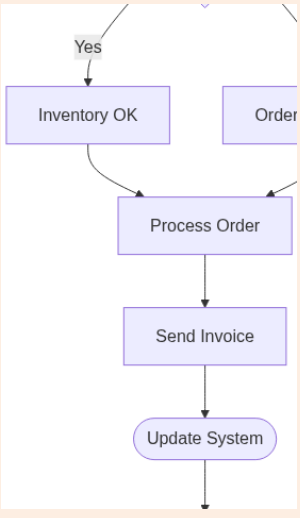
Collectively, these qualitative observations reinforce our quantitative results and highlight the value of DiGraphHal-Bench as a diagnostic benchmark for fine-grained, integrated multi-modal graph reasoning.



Q: Enumerate all fully visible edges in image.
GT: Start -> Input jump lengths ‘a’ and ‘b’. Input target distance ‘d’.\n Input jump lengths ‘a’ and ‘b’. Input target distance ‘d’. -> Assign ‘a’ to ‘temp’. Update ‘a’ with min(‘a’, ‘b’). Update ‘b’ with max(‘temp’, ‘b’). \n Assign ‘a’ to ‘temp’. Update ‘a’ with min(‘a’, ‘b’). Update ‘b’ with max(‘temp’, ‘b’). -> Is ‘d’ >= ‘b’? \n Is ‘d’ >= ‘b’? -> [Yes] -> Calculated and return (d + b - 1) / b as required jumps. \n Is ‘d’ >= ‘b’? -> [No] -> Is ‘d’ == 0?

GPT-5: Is 'd' >= 'b'? -> [Yes] -> Calculate and return (d + b - 1) / b as required jumps. ✗
Gemini: Is 'd' == 0? -> [Yes] -> Calculate and return (d + b - 1)/ b as required jumps. ✗
Qwen2.5vl-72b: Is 'd' == 0? -> Calculate and return (d + b - 1) / b as required jumps. ✗
Is 'd' == 0? -> [No] -> End ✗
Qwen2.5vl-7b-structural: CORRECT. ✓

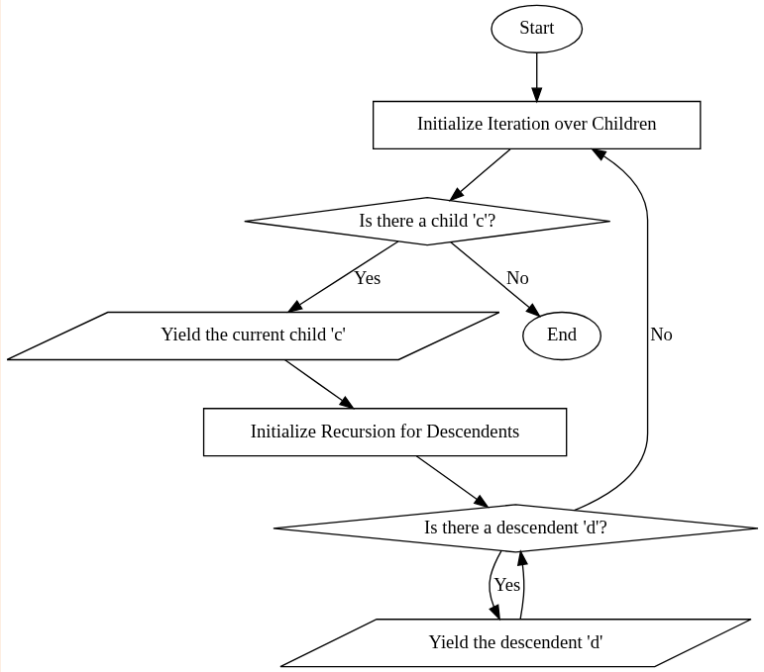
Figure 10. Structural Comprehension: Graph Parsing. Color coding for model predictions: incorrect predictions (red), missed ground truth answers (green), error analysis (blue). ✓ and × denote correct and incorrect predictions, respectively. Applied likewise throughout. Qwen2.5vl-7b-structural denotes the SFT-trained structural specialist; same notation applies.



Q: Convert the fully visible subgraph flowchart in the image into Mermaid code.
GT: flowchart TD\n D[\"Inventory OK\"]\n F[\"Process Order\"]\n G[\"Send Invoice\"]\n H([\"Update System\"])\n D --> F\n F --> G\n G --> H

GPT-5: flowchart TD\n A[\"Inventory OK\"]\n B[\"Order\"]\n C[\"Process Order\"]\n D[\"Send Invoice\"]\n E([\"Update System\"])\n A --> C\n B --> C\n C --> D\n D --> E ✗
Gemini: flowchart TD\n A{ }\n B[\"Inventory OK\"]\n C[\"Order\"]\n D[\"Process Order\"]\n E[\"Send Invoice\"]\n F([\"Update System\"])\n\n A -->|Yes| B\n A --> C\n B --> D\n C --> D\n D --> E\n E --> F ✗
Qwen2.5vl-72b: flowchart TD\n A[\"Inventory OK\"]\n B[/Order/]\n C[Process Order]\n D[Send Invoice]\n E{Update System}\n\n A -->|Yes| C\n B --> C\n C --> D\n D --> E ✗
Qwen2.5vl-7b-structural: CORRECT. ✓

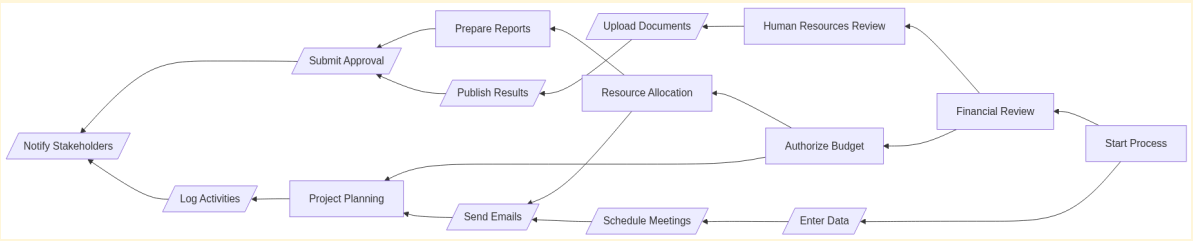
Figure 11. Structural Comprehension: Graph2Code.



Q: Find all possible cyclic subpaths of the masked part: Yield the current child 'c' -> {} -> Is there a descendent 'd'?
GT: 1. Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [No] -> Initialize Iteration over Children -> Is there a child 'c'? -> [Yes] -> Yield the current child 'c' -> Initialize Recursion for Descendents
2. Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [Yes] -> Yield the descendent 'd'
3. Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [Yes] -> Yield the descendent 'd' -> Is there a descendent 'd'? -> [No] -> Initialize Iteration over Children -> Is there a child 'c'? -> [Yes] -> Yield the current child 'c' -> Initialize Recursion for Descendents (#nodes in path > #nodes in graph, omitted)

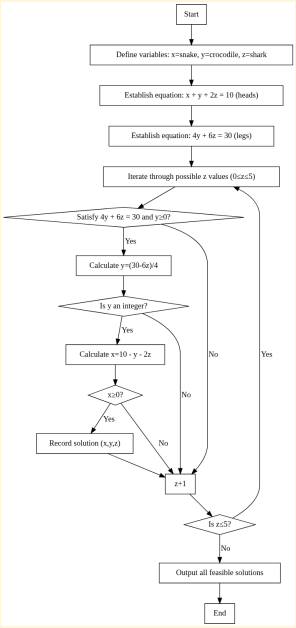
GPT-5: 1. Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [Yes] -> Yield the descendent 'd' ✓
2. Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [Yes] -> Yield the descendent 'd' -> Is there a descendent 'd'? -> [No] -> Initialize Iteration over Children -> Is there a child 'c'? -> [No] -> End ✗
3. Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [No] -> Initialize Iteration over Children -> Is there a child 'c'? -> [Yes] -> Yield the current child 'c' ✗ -> Initialize Recursion for Descendents
4. Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [Yes] -> Yield the descendent 'd' -> Is there a descendent 'd'? -> [No] -> Initialize Iteration over Children -> Is there a child 'c'? -> [Yes] -> Yield the current child 'c' ✗ -> Initialize Recursion for Descendents, and overlong
Gemini: CORRECT. ✓
Qwen2.5vl-72b: Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [Yes] -> Yield the descendent 'd' ✓
Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [No] -> Initialize Iteration over Children -> Is there a child 'c'? -> [Yes] -> Yield the current child 'c' -> Initialize Recursion for Descendents ✗
Qwen2.5vl-7b-structural: Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [Yes] -> Yield the descendent 'd' -> Initialize Recursion for Descendents -> Is there a descendent 'd'? -> [No] -> End -> Initialize Iteration over Children -> Is there a child 'c'? -> [Yes] -> Yield the current child 'c' ✗

Figure 12. Masked Subpath Query: Fully-Merged Cyclic Subpath Query.



Q: List all crossing edge pairs where two edges intersect on the 2D plane.
GT: 1. [Authorize Budget -> Project Planning, Resource Allocation -> Send Emails]
2. [Resource Allocation -> Prepare Reports, Upload Documents -> Publish Results]

GPT-5: 1. [Resource Allocation -> Send Emails, Authorize Budget -> Project Planning] ✓
2. [Upload Documents -> Resource Allocation, Resource Allocation -> Prepare Reports] ✗
[Resource Allocation -> Prepare Reports, Upload Documents -> Publish Results] ✗
Gemini: 1. [Submit Approval -> Publish Results, Human Resources Review -> Upload Documents] ✗
2. [Prepare Reports -> Resource Allocation, Human Resources Review -> Upload Documents] ✗
3. [Authorize Budget -> Project Planning, Resource Allocation -> Send Emails] ✓
[Resource Allocation -> Prepare Reports, Upload Documents -> Publish Results] ✗
Qwen2.5vl-72b: 1. [Submit Approval -> Prepare Reports, Upload Documents -> Human Resources Review] ✗
2. [Submit Approval -> Prepare Reports, Resource Allocation -> Publish Results] ✗
3. [Upload Documents -> Human Resources Review, Resource Allocation -> Publish Results] ✗
[Authorize Budget -> Project Planning, Resource Allocation -> Send Emails] ✗
[Resource Allocation -> Prepare Reports, Upload Documents -> Publish Results] ✗
Qwen2.5vl-7b-visual: [Upload Documents -> Submit Approval, Resource Allocation -> Publish Results] ✗
[Authorize Budget -> Project Planning, Resource Allocation -> Send Emails] ✗
[Resource Allocation -> Prepare Reports, Upload Documents -> Publish Results] ✗



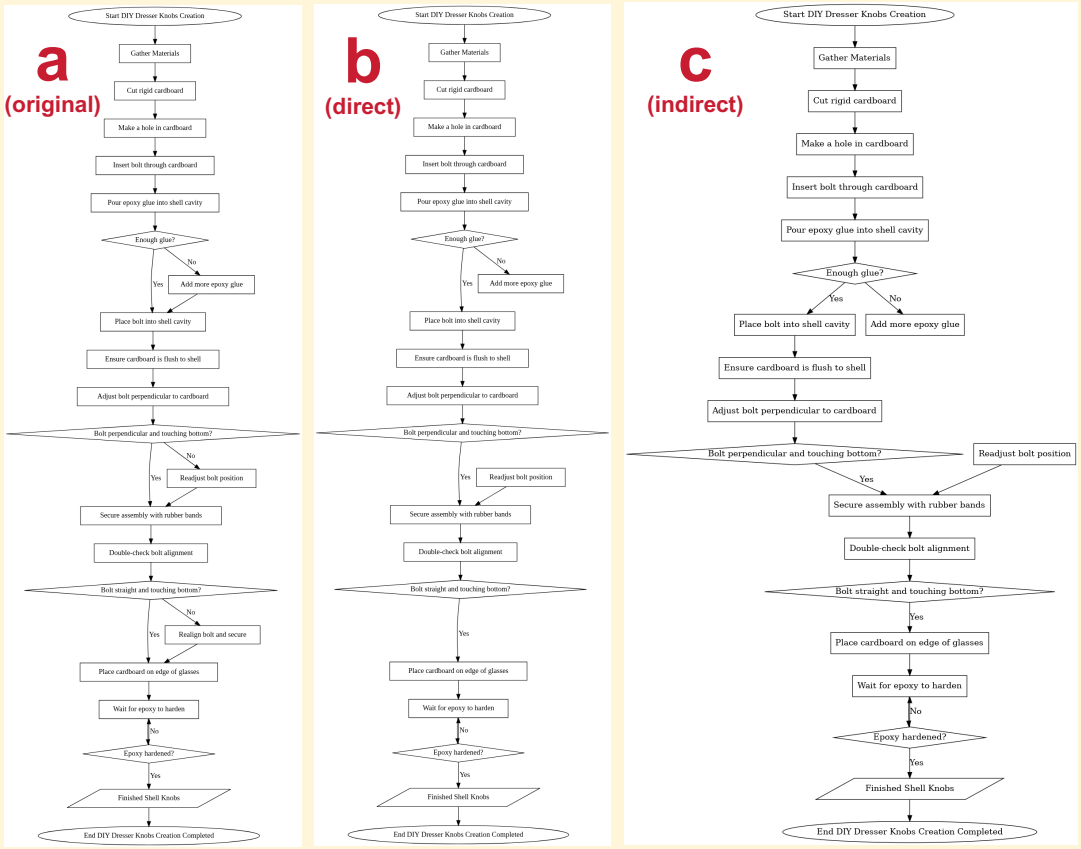
Q: List all reverse edges that opposing the dominant flow.
GT: Is z≤5? -> [Yes] -> Iterate through possible z values (0≤z≤5)

GPT-5: Is z≤5? -> [Yes] -> Satisfy 4y + 6z = 30 and y≥0? ✗ -> Iterate through possible z values (0≤z≤5)
Gemini: CORRECT. ✓
Qwen2.5vl-72b: 1. z+1 -> Satisfy 4y + 6z = 30 and y≥0? ✗
2. Record solution (x,y,z) -> z+1 ✗
3. Calculate x=10 - y - 2z -> z+1 ✗
4. Calculate y=(30-6z)/4 -> z+1 ✗
Qwen2.5vl-7b-visual: Is z≤5? -> [No] -> Iterate through possible z values (0≤z≤5) ✗ [Yes]

Q: List all long edges whose bbox height or width is at least 1/3 of the whole image dimension.
GT: 1. Satisfy 4y + 6z = 30 and y≥0? -> [No] -> z+1
2. Is z≤5? -> [Yes] -> Iterate through possible z values (0≤z≤5)

GPT-5: Is z≤5? -> [Yes] -> Satisfy 4y + 6z = 30 and y≥0? ✗ -> Iterate through possible z values (0≤z≤5)
Gemini: 1. Is z<=5? -> [Yes] -> Iterate through possible z values (0<=z<=5) ✓
2. Satisfy 4y + 6z = 30 and y>=0? -> [No] -> z+1 ✓
3. Is y an integer? -> [No] -> z+1 ✗ <1/3
Qwen2.5vl-72b: 1. Satisfy 4y + 6z = 30 and y≥0? -> No -> Iterate through possible z values (0≤z≤5) ✗ -> z+1
2. Record solution (x,y,z) -> z+1
3. Is z≤5? -> Yes -> Satisfy 4y + 6z = 30 and y≥0? ✗ -> Iterate through possible z values (0≤z≤5)
Qwen2.5vl-7b-visual: Iterate through possible z values (0≤z≤5) -> Satisfy 4y + 6z = 30 and y≥0? ✗

Figure 13. Edge Layout Perception. Qwen2.5vl-7b-visual denotes the SFT-trained visual specialist; same notation applies.



Q: Identify the elements missing from (b) compared to (a).

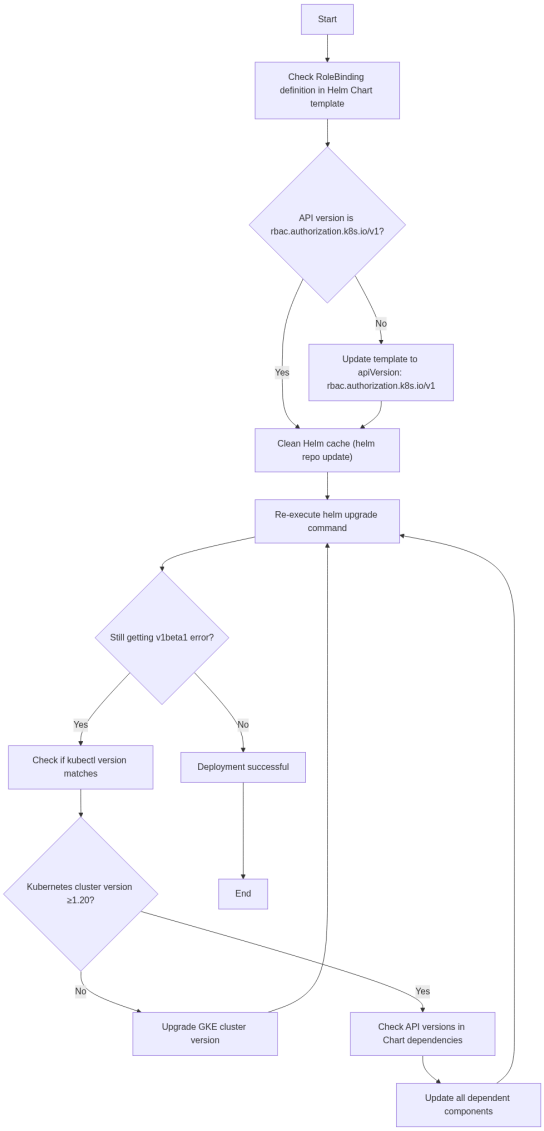
Q: Identify the elements missing from (c) compared to (a).

GT: {**nodes**: [Realign bolt and secure],
edges: [Realign bolt and secure -> Place cardboard on edge of glasses, Bolt straight and touching bottom? -> [No] -> Realign bolt and secure, Add more epoxy glue -> Place bolt into shell cavity, Bolt perpendicular and touching bottom? -> [No] -> Readjust bolt position}

GPT-5: {nodes: [Realign bolt and secure], ✓
edges: [Bolt straight and touching bottom? -> [No] -> Realign bolt and secure ✓, Realign bolt and secure -> Place cardboard on edge of glasses ✓]} Add more epoxy glue -> Place bolt into shell cavity, Bolt perpendicular and touching bottom? -> [No] -> Readjust bolt position ✗
Gemini: {nodes: [Realign bolt and secure], ✓
edges: [Bolt straight and touching bottom? -> [No] -> Realign bolt and secure ✓, Realign bolt and secure -> Bolt straight and touching bottom?]} ✗
Qwen2.5vl-72b: {nodes: [Readjust bolt position, Realign bolt and secure], ✗
edges: [Bolt perpendicular and touching bottom? -> [No] -> Readjust bolt position ✓, Bolt straight and touching bottom? -> [No] -> Realign bolt and secure ✓]} Realign bolt and secure -> Place cardboard on edge of glasses, Add more epoxy glue -> Place bolt into shell cavity ✗
Qwen2.5vl-7b-visual: {nodes: [], ✗
edges: [Bolt perpendicular and touching bottom? -> [No] -> Readjust bolt position ✓, Bolt perpendicular and touching bottom? -> [Yes] -> Secure assembly with rubber bands, Epoxy hardened? -> [No] -> Wait for epoxy to harden]} ✗

GPT-5: {nodes: [Realign bolt and secure], ✓
edges: [Add more epoxy glue -> Place bolt into shell cavity ✓, Readjust bolt position -> Secure assembly with rubber bands ✗, Bolt straight and touching bottom? -> [No] -> Realign bolt and secure ✓, Realign bolt and secure -> Place cardboard on edge of glasses ✓]} Bolt perpendicular and touching bottom? -> [No] -> Readjust bolt position ✗
Gemini: {nodes: [Realign bolt and secure] ✓,
edges: [Bolt perpendicular and touching bottom? -> [No] -> Readjust bolt position ✓, Bolt straight and touching bottom? -> [No] -> Realign bolt and secure ✓, Realign bolt and secure -> Place cardboard on edge of glasses ✓]} Add more epoxy glue -> Place bolt into shell cavity ✗
Qwen2.5vl-72b: {nodes: [Realign bolt and secure] ✓,
edges: [Bolt straight and touching bottom? -> [No] -> Realign bolt and secure ✓]} Realign bolt and secure -> Place cardboard on edge of glasses, Add more epoxy glue -> Place bolt into shell cavity, Bolt perpendicular and touching bottom? -> [No] -> Readjust bolt position ✗
Qwen2.5vl-7b-visual: CORRECT. ✓

Figure 14. Local Structure Comparison. Left: Direct Structure Difference; Right: Indirect Structure Difference.



GPT-5:
node pairs:[upper_right,upper_left,directly_above] ✓
edge pairs:[upper_left ✓,upper_left ✓,directly_left ✗]
Gemini:
node pairs:[upper_right,upper_left,directly_above] ✓
edge pairs:[upper_left,upper_left,lower_left] ✓
Qwen2.5vl-72b:
node pairs:[upper_left ✗,upper_left ✓,directly_above ✓]
edge pairs:[upper_left,upper_left,lower_left] ✓
Qwen2.5vl-7b-visual:
node pairs: [upper_left ✗, upper_left ✓, upper_left ✗]
edge pairs: [upper_right, upper_right, upper_right] ✗

Absolute Localization (node & edge)
Q: Detect and locate the given nodes and edges in the image.
GT: {nodes: {Re-execute helm upgrade command: [461, 900, 721, 978], Still getting v1beta1 error?: [173, 1028, 414, 1268]},
edges: {Still getting v1beta1 error? -> [Yes] -> Check if kubectl version matches: [147, 1211, 236, 1338], Kubernetes cluster version ≥1.20? -> [Yes] -> Check API versions in Chart dependencies: [256, 1641, 762, 1818]}}

GPT-5: {nodes:{Re-execute helm upgrade command: [300, 905, 690, 1005],Still getting v1beta1 error?: [130, 1115, 480, 1295]},
edges: {Still getting v1beta1 error? -> [Yes] -> Check if kubectl version matches: [120, 1210, 360, 1390], Kubernetes cluster version ≥1.20? -> [Yes] -> Check API versions in Chart dependencies: [450, 1630, 920, 1765]}}
Gemini: {nodes: {Re-execute helm upgrade command: [332, 451, 651, 520], Still getting v1beta1 error?: [333, 550, 650, 651]},
edges: {Still getting v1beta1 error? -> [Yes] -> Check if kubectl version matches: [158, 621, 412, 680],Kubernetes cluster version ≥1.20? -> [Yes] -> Check API versions in Chart dependencies: [234, 848, 816, 891]}}
Qwen2.5vl-72b: {nodes: {Re-execute helm upgrade command: [470, 1600, 730, 1680], Still getting v1beta1 error?: [470, 1450, 730, 1530],
edges: {Still getting v1beta1 error? -> [Yes] -> Check if kubectl version matches: [590, 1530, 610, 1720]}}
Qwen2.5vl-7b-visual: {nodes: {Re-execute helm upgrade command: [330, 530, 750, 580], Still getting v1beta1 error?: [100, 1030, 330, 1080]},
edges: {}}

Relative Localization (node & edge)
Q: Determine the position of the first node / edge in a given node / edge pair relative to the second node / edge in the image:
node pairs: [[Update template to apiVersion: rbac.authorization.k8s.io/v1, End],
[Still getting v1beta1 error?, Update all dependent components], [Start, Check RoleBinding definition in Helm Chart template]]
edge pairs: [[Start -> Check RoleBinding definition in Helm Chart template, API version is rbac.authorization.k8s.io/v1? -> [No] -> Update template to apiVersion: rbac.authorization.k8s.io/v1],
[Check RoleBinding definition in Helm Chart template -> API version is rbac.authorization.k8s.io/v1?, Check API versions in Chart dependencies -> Update all dependent components],
[Kubernetes cluster version ≥1.20? -> [No] -> Upgrade GKE cluster version, Deployment successful -> End]]
GT: node pairs: [upper_right, upper_left, directly_above]
edge pairs: [upper_left, upper_left, lower_left]

Figure 15. Elements Localization.

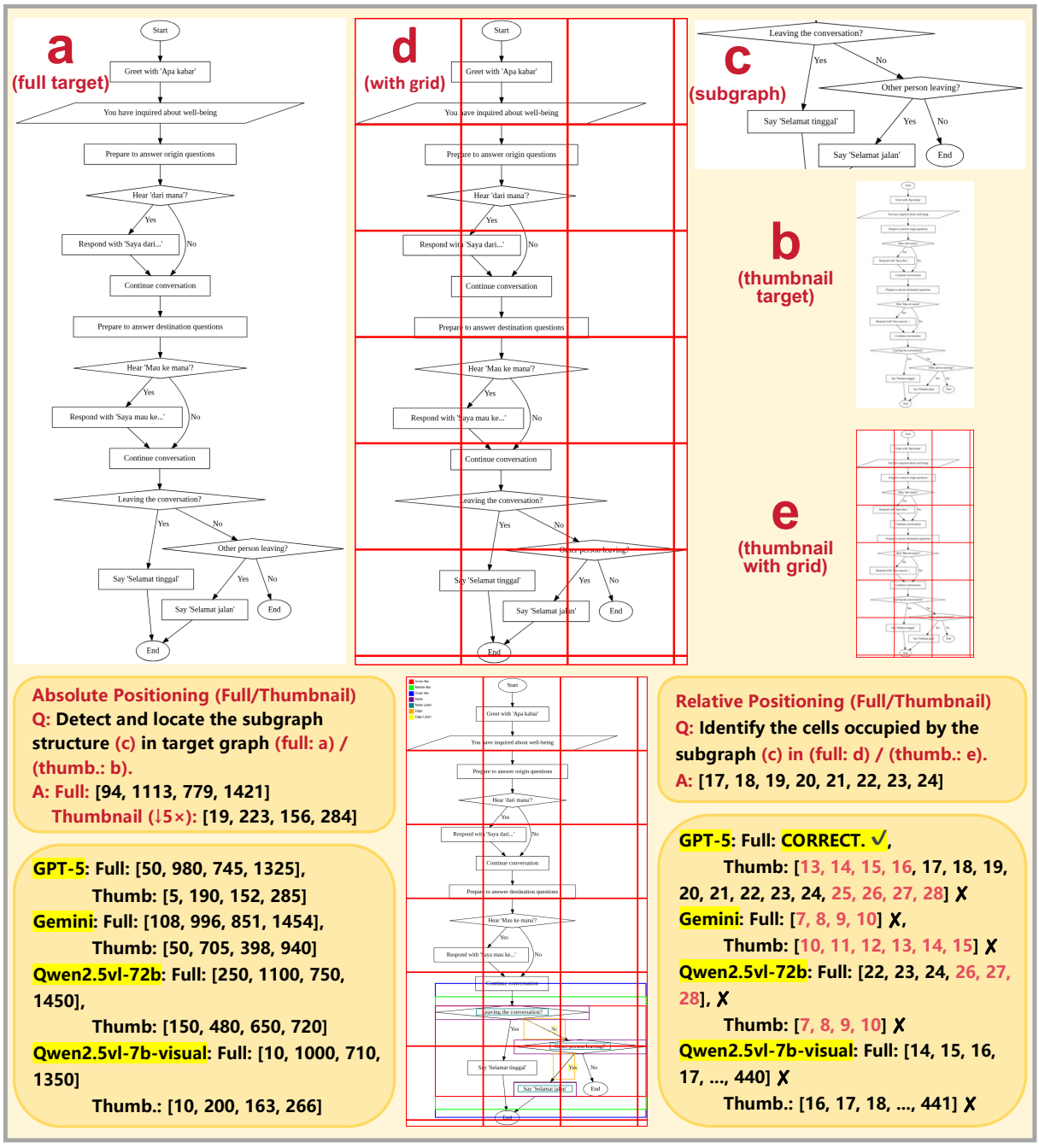
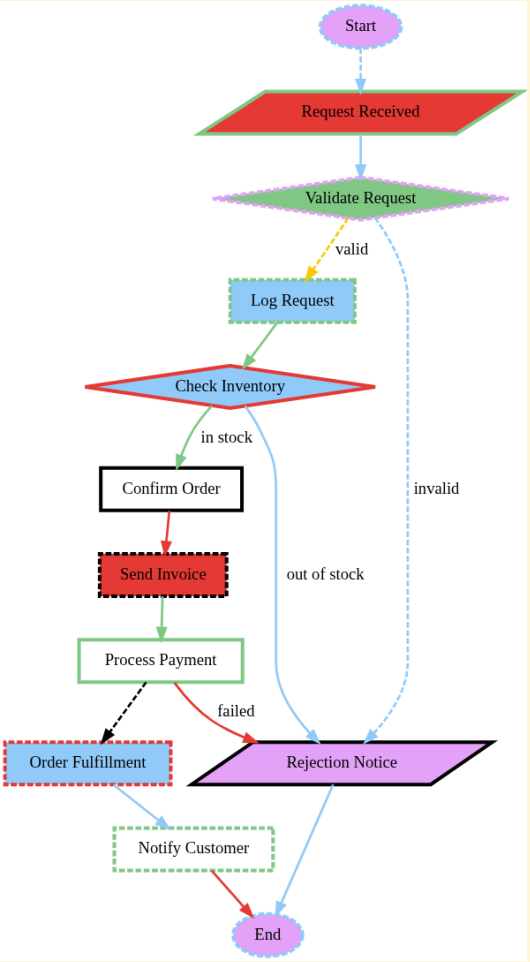


Figure 16. Spatial Position Awareness.



Q: Find all the nodes with red fill in the graph.
GT: [Request Received, Send Invoice]

GPT-5 & Gemini & Qwen2.5vl-72b & Qwen2.5vl-7b-visual: CORRECT. ✓

Q: Find all the box-shaped nodes in the graph.
GT:[Log Request, Confirm Order, Send Invoice, Process Payment, Order Fulfillment, Notify Customer]

GPT-5 & Gemini: CORRECT. ✓
Qwen2.5vl-72b: [Confirm Order, Send Invoice, Process Payment, Order Fulfillment, Notify Customer] ✗ Log Request
Qwen2.5vl-7b-visual: [Log Request, Check Inventory, Confirm Order, Send Invoice, Process Payment, Order Fulfillment, Notify Customer] ✗

Q: Find all the edges with yellow color in the graph.
GT: [Validate Request -> [valid] -> Log Request]

GPT-5: CORRECT. ✓
Gemini: CORRECT. ✓
Qwen2.5vl-72b: Validate Request -> Log Request ✗ [valid]
Qwen2.5vl-7b-visual: CORRECT. ✓

Q: Find all the edges with dashed style in the graph.
GT: [Start -> Request Received, Validate Request -> [valid] -> Log Request, Validate Request -> [invalid] -> Rejection Notice, Process Payment -> Order Fulfillment]

GPT-5: CORRECT. ✓
Gemini: CORRECT. ✓
Qwen2.5vl-72b: CORRECT. ✓
Qwen2.5vl-7b-visual: [Start -> Request Received, Validate Request -> [valid] -> Log Request, Check Inventory -> [in stock] -> Confirm Order, Check Inventory -> [out of stock] -> Rejection Notice, Rejection Notice -> Notify Customer, Notify Customer -> End] ✗ Validate Request -> [invalid] -> Rejection Notice, Process Payment -> Order Fulfillment

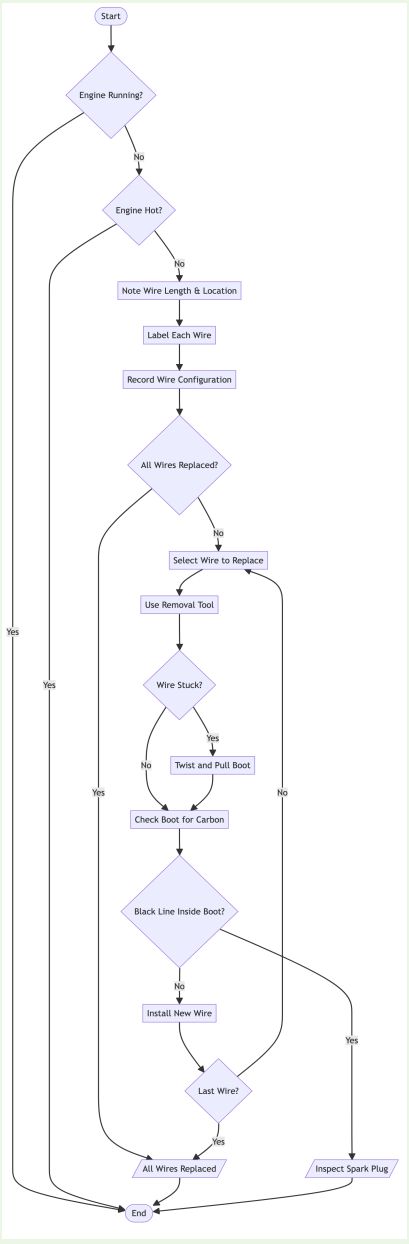
Q: Find all the nodes with black border in the graph.
GT: [Rejection Notice, Confirm Order, Send Invoice]

GPT-5: CORRECT. ✓
Gemini: CORRECT. ✓
Qwen2.5vl-72b: [Check Inventory, Confirm Order, Send Invoice, Process Payment, Order Fulfillment, Rejection Notice] ✗
Qwen2.5vl-7b-visual: [Start, Check Inventory, Confirm Order, End] ✗ Rejection Notice, Send Invoice

Q: Find all the nodes with solid border in the graph.
GT: [Request Received, Rejection Notice, Check Inventory, Confirm Order, Process Payment]

GPT-5: CORRECT. ✓
Gemini: CORRECT. ✓
Qwen2.5vl-72b: [Start, Request Received, Validate Request, Log Request, Check Inventory, Confirm Order, Send Invoice, Process Payment, Order Fulfillment, Rejection Notice, Notify Customer, End] ✗
Qwen2.5vl-7b-visual: [Start, Request Received, Validate Request, Log Request, Check Inventory, Confirm Order, Send Invoice, Process Payment, Order Fulfillment, Notify Customer, End] ✗ Rejection Notice

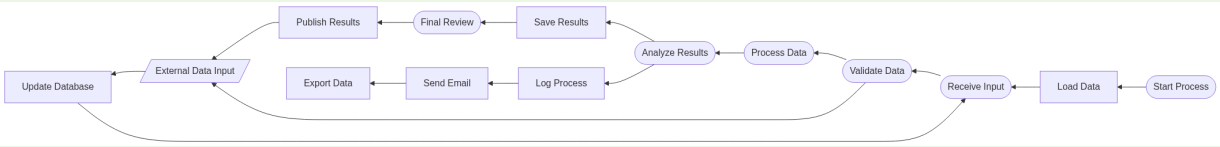
Figure 17. Visual Attributes Perception.



Q (simple subpath, i.e., no directed cycle):
Imagine you're troubleshooting the removal of a specific component, and the instructions involve a series of steps. After completing Twist and Pull Boot, you realize that there are six prior steps essential to understanding the process but are missing from your guide. Can you help identify what those six steps are?
GT-masked-subpath: {6} -> Twist and Pull Boot
GT: 1. Label Each Wire -> Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes],
2. Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes]]]

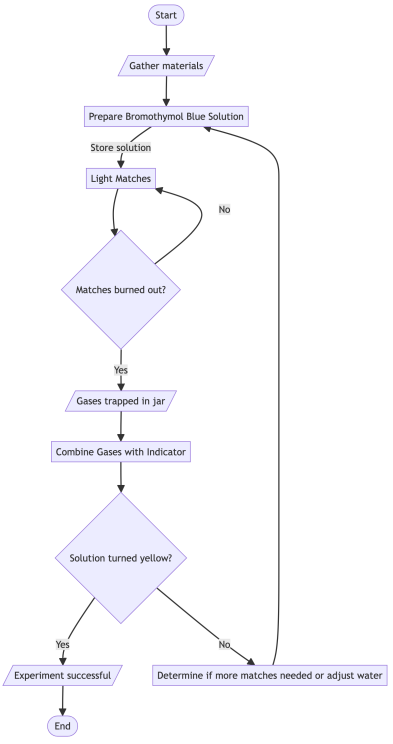
GPT-5: Label Each Wire -> Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes] -> Twist and Pull Boot ✓
Gemini: Note Wire Length & Location -> ✓ (but not required) Label Each Wire -> Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes] -> Twist and Pull Boot ✓
Qwen2.5vl-72b: Failed ✗
Qwen2.5vl-7b-semantic: Engine Running? -> [No] -> Engine Hot? -> [No] -> Note Wire Length & Location -> ✓ (but not required) Label Each Wire -> Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes] -> Twist and Pull Boot ✓ -> [No] ✗ -> Check Boot for Carbon -> Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [Yes] -> All Wires Replaced -> End
Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes]]] ✗ (This case is missed by all evaluated models.)

Figure 18. Semantic Query: Simple Subpath Query. Qwen2.5vl-7b-semantic denotes the SFT-trained semantic specialist; same notation applies.



Q (cyclic subpath, i.e., contains exactly one directed cycle):
Imagine you are designing a system where data needs to be processed before receiving user input. Initially, the system begins at Start Process, but there's a missing sequence of steps to complete the process before reaching Receive Input. The missing path contains exactly n step. Can you identify what should happen in this intermediate step to ensure the process flows correctly?
GT-masked-subpath: Start Process -> {} -> Receive Input
GT: 1. Load Data -> Receive Input -> Validate Data -> Process Data -> Analyze Results -> Save Results -> Final Review -> Publish Results -> External Data Input -> Update Database,
2. Load Data -> Receive Input -> Validate Data -> External Data Input -> Update Database

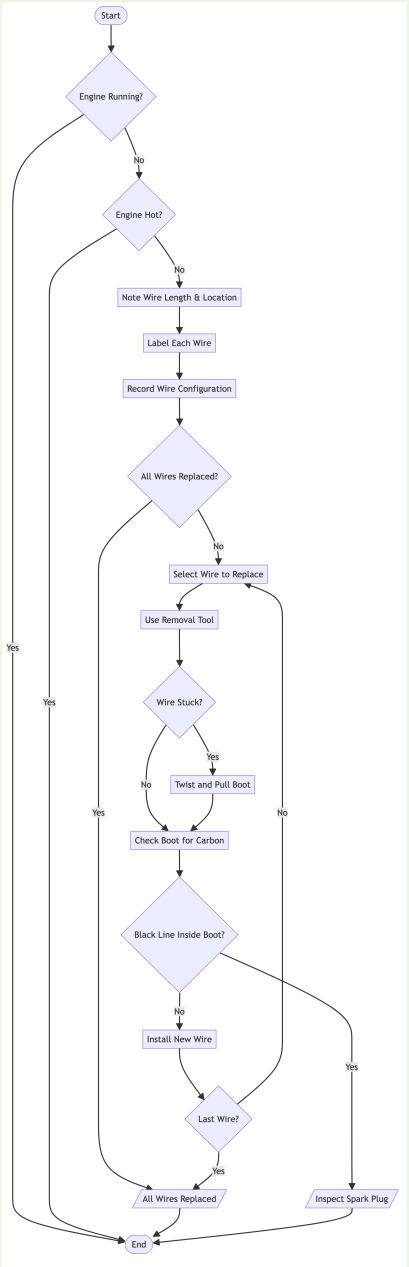
GPT-5: Failed ✗
Gemini:Start Process -> Load Data -> Receive Input -> Validate Data -> Process Data -> Analyze Results -> Log Process -> Send Email -> Export Data -> External Data Input -> Update Database -> Receive Input ✗
Qwen2.5vl-72b: Failed ✗
Qwen2.5vl-7b-semantic: Start Process -> Load Data -> Receive Input ✗ (Correct path, but non-cyclic.)
1. Load Data -> Receive Input -> Validate Data -> Process Data -> Analyze Results -> Save Results -> Final Review -> Publish Results -> External Data Input -> Update Database,
2. Load Data -> Receive Input -> Validate Data -> External Data Input -> Update Database
✗ (All cases are missed by all evaluated models.)



Q (shortest subpath): Imagine you are conducting a science experiment where you are measuring the effect of trapped gases on water levels inside a sealed jar. After observing the Gases trapped in jar, there is a process that involves multiple steps {n} before you reach the point where you decide whether more matches are needed or adjust the water - Determine if more matches needed or adjust water. Can you figure out what the intermediate step is in this experiment?
GT-masked-subpath: Gases trapped in jar -> {} -> Determine if more matches needed or adjust water
GT: Gases trapped in jar -> Combine Gases with Indicator -> Solution turned yellow? -> [No] -> Determine if more matches needed or adjust water

GPT-5: CORRECT. ✓
Gemini:CORRECT. ✓
Qwen2.5vl-72b: Gases trapped in jar -> Combine Gases with Indicator -> Solution turned yellow? -> [No] -> Determine if more matches needed or adjust water ✗
Qwen2.5vl-7b-semantic: CORRECT. ✓

Figure 19. Semantic Query: Cyclic and Shortest Subpath Query.



Q (simple subpath, i.e., no directed cycle):
Imagine you're diagnosing issues with detaching a particular part, and the directions include multiple stages. After completing Rotate and Extract Cover, you notice that six earlier steps crucial to comprehending the procedure are absent from your manual. Can you assist in pinpointing those six steps?
GT-masked-subpath: {6} -> Twist and Pull Boot
GT: 1. Label Each Wire -> Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes],
2. Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes]

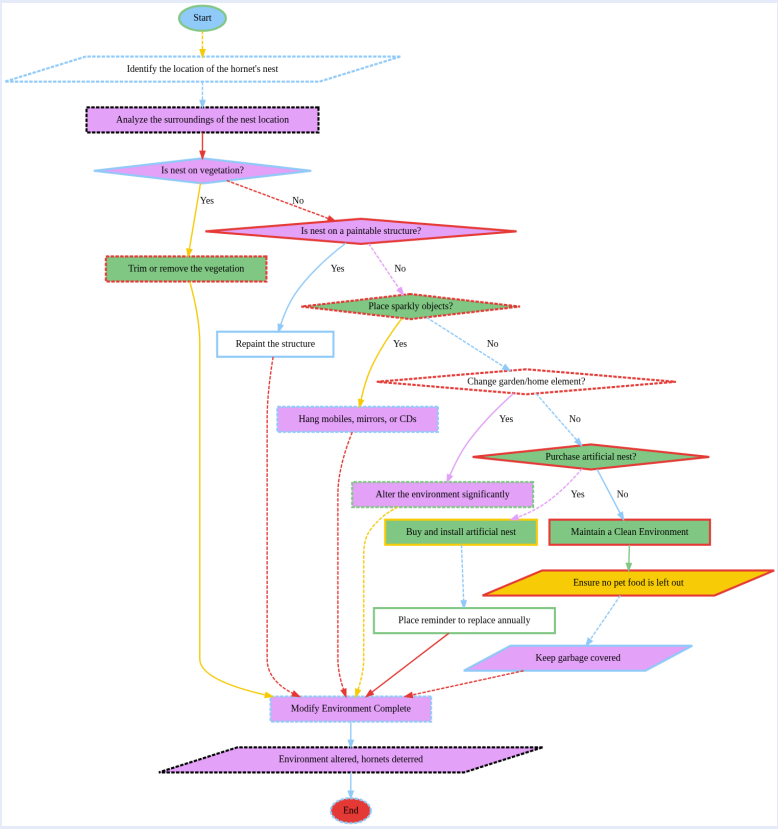
GPT-5: Start -> Engine Running? -> [No] -> Engine Hot? -> [No] -> Note Wire Length & Location -> ✓ (but not required) Label Each Wire -> Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes] -> Twist and Pull Boot ✓ -> Check Boot for Carbon (but not required) ✓

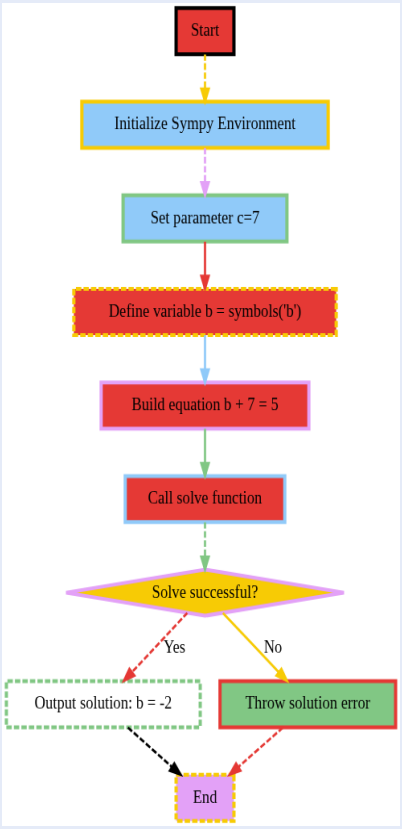
Gemini: Note Wire Length & Location -> ✓ (but not required) Label Each Wire -> Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes] -> Twist and Pull Boot ✓

Qwen2.5vl-72b: Start -> [No] -> Engine Running? -> [No] -> Engine Hot? -> [No] -> Note Wire Length & Location -> [No] -> Label Each Wire -> [No] -> Record Wire Configuration ✗

Qwen2.5vl-7b-semantic: Start -> Engine Running? -> [No] -> Engine Hot? -> [No] -> Note Wire Length & Location -> ✓ (but not required) Label Each Wire -> Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? ✓-> [No] ✗-> Twist and Pull Boot -> Check Boot for Carbon -> Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [Yes] -> All Wires Replaced -> End
Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes]]] ✗ (This case is missed by all evaluated models.)

Figure 20. Semantic-Rewrite Query: Simple Subpath Query





Non-Semantic - Element-Centric Class1: What / Which

Masked Subpath: Is nest on a paintable structure? -> [No] -> Place sparkly objects? -> {} -> Buy and install artificial nest -> Place reminder to replace annually -> Modify Environment Complete -> Environment altered, hornets deterred

Q (edge): According to the flowchart in the picture, which edges are dashed-bordered edge in order in the subpath.

GT: 1. Place sparkly objects? -> [No] -> Change garden/home element?
2. Change garden/home element? -> [No] -> Purchase artificial nest?,
3. Purchase artificial nest? -> [Yes] -> Buy and install artificial nest

Non-Semantic - Class1

Masked Subpath: Set parameter c=7 -> {} -> Solve successful?

Q (node): According to the flowchart in the picture, find the 3th red-filled with pink-bordered node in the subpath.

GT: [] (Counterfactual and answerless cases)

GPT-5: Place sparkly objects? -> [No] -> Change garden/home element? ✓ 2. Change garden/home element? -> [No] -> Purchase artificial nest?, 3. Purchase artificial nest? -> [Yes] -> Buy and install artificial nest ✗

Gemini: 1. Place sparkly objects? -> [No] -> Change garden/home element? ✓
2. Alter the environment significantly -> Buy and install artificial nest ✗
3. Buy and install artificial nest -> Place reminder to replace annually ✗

Qwen2.5vl-72b: CORRECT. ✓

Qwen2.5vl-7b-comprehensive: Place sparkly objects? -> [No] -> Change garden/home element ✓ 2. Change garden/home element? -> [No] -> Purchase artificial nest?,
3. Purchase artificial nest? -> [Yes] -> Buy and install artificial nest ✗

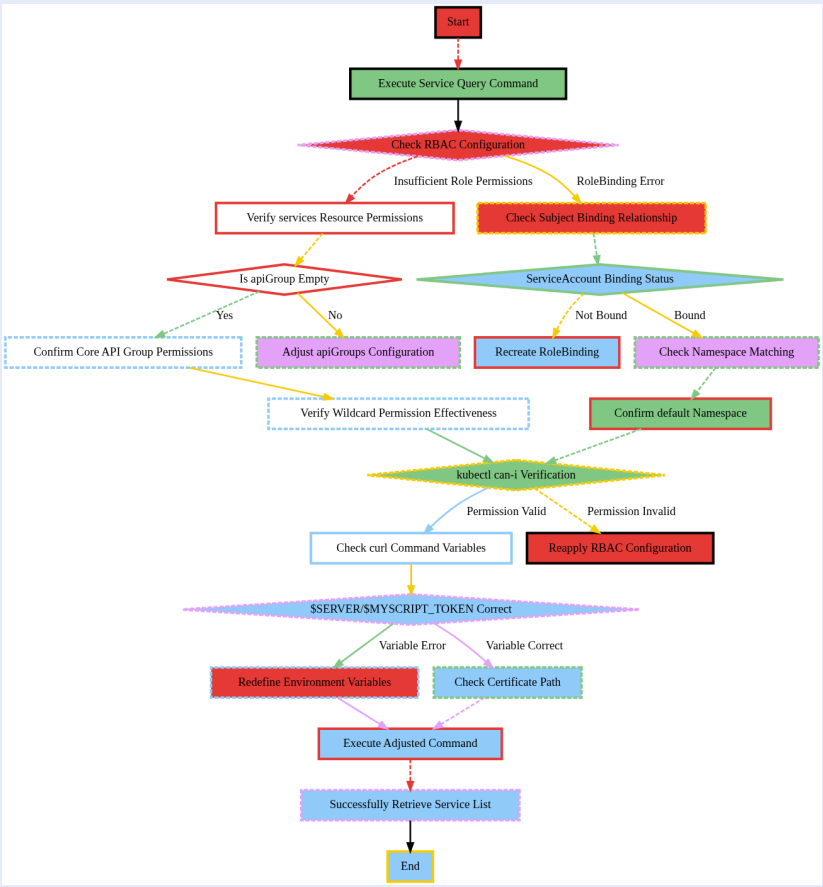
GPT-5: not exists CORRECT. ✓

Gemini: Build equation b + 7 = 5 ✗ (It satisfies the visual style requirement but violates the ordinal constraint.)

Qwen2.5vl-72b: Call solve function ✗

Qwen2.5vl-7b-comprehensive: [] CORRECT. ✓

Figure 21. Comprehensive - Non-Semantic Query: Element-Centric Class1 - What/Which. Both Simple and Cyclic subpaths are allowed, and this convention is applied consistently across subsequent cases. Qwen2.5vl-7b-comprehensive denotes the SFT-trained comprehensive specialist; same notation applies.



Non-Semantic - Path-Centric Class2: How

Masked Subpath: Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> {5}

Q (node): According to the flowchart in the picture, find at least 1 paths that include most blue-filled with red-bordered nodes.

GT: 1. Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error] -> Redefine Environment Variables -> Execute Adjusted Command -> Successfully Retrieve Service List

2. Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Correct] -> Check Certificate Path -> Execute Adjusted Command -> Successfully Retrieve Service List

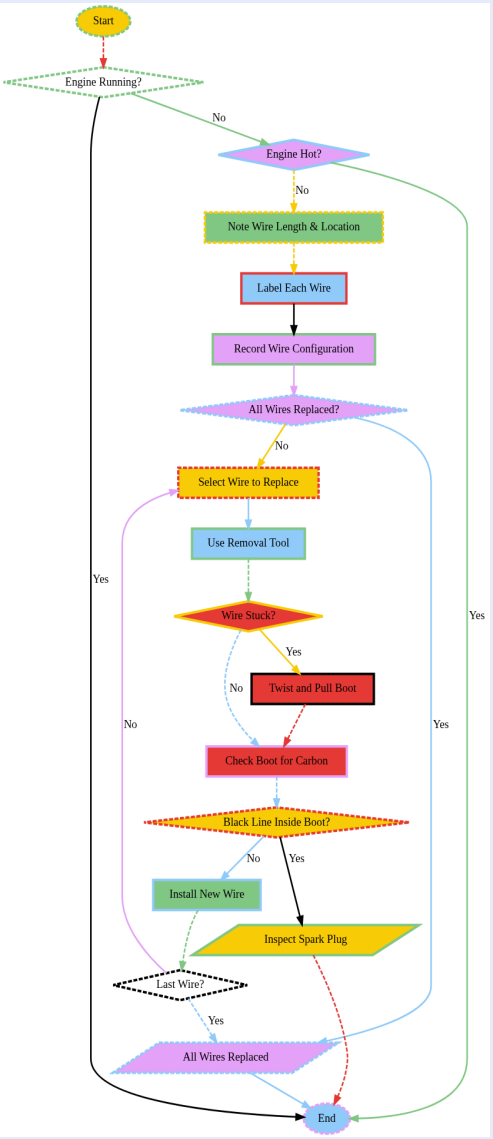
GPT-5: Check curl Command Variables -> SSSERVER/SMYSCRIPT_TOKEN Correct -> [Variable Error] -> Redefine Environment Variables -> Execute Adjusted Command -> Successfully Retrieve Service List ✓ 1 subpath missed ✗

Gemini: Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error] -> Redefine Environment Variables -> Execute Adjusted Command -> Successfully Retrieve Service List ✓ 1 subpath missed ✗

Qwen2.5vl-72b: Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace ✗

Qwen2.5vl-7b-comprehensive: Check curl Command Variables -> $SERVER=$MYSCRIPT_TOKEN Correct -> Variable Correct -> Check Certificate Path -> Execute Adjusted Command -> Successfully Retrieve Service List ✓ 1 subpath missed ✗

Figure 22. Comprehensive - Non-Semantic Query: Path-Centric Class2 - How (node).



Non-Semantic - Path-Centric Class2: How

Masked Subpath: Label Each Wire -> {} -> Use Removal Tool -> Wire Stuck? -> [Yes] -> Twist and Pull Boot

Q (edge): According to the flowchart in the picture, find at least 4 paths that include most yellow-bordered with solid-bordered edges.

GT: Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [Yes] -> Twist and Pull Boot -> Check Boot for Carbon -> Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [No] -> Select Wire to Replace

(Counterfactual and answerless cases: only one valid path exists)

GPT-5: Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace ✗ (Simple subpath correct but only one edge meets the visual attribute, violating the 'most' constraint; GT cyclic subpath has two.)

Gemini: 1. Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [No] -> Check Boot for Carbon -> Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [No] -> Select Wire to Replace ✗ (miss one yellow-bordered solid edge)

2. Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [No] -> Check Boot for Carbon -> Black Line Inside Boot? -> [Yes] -> Inspect Spark Plug -> Last Wire? -> [No] -> Select Wire to Replace ✗ (wrong subpath)

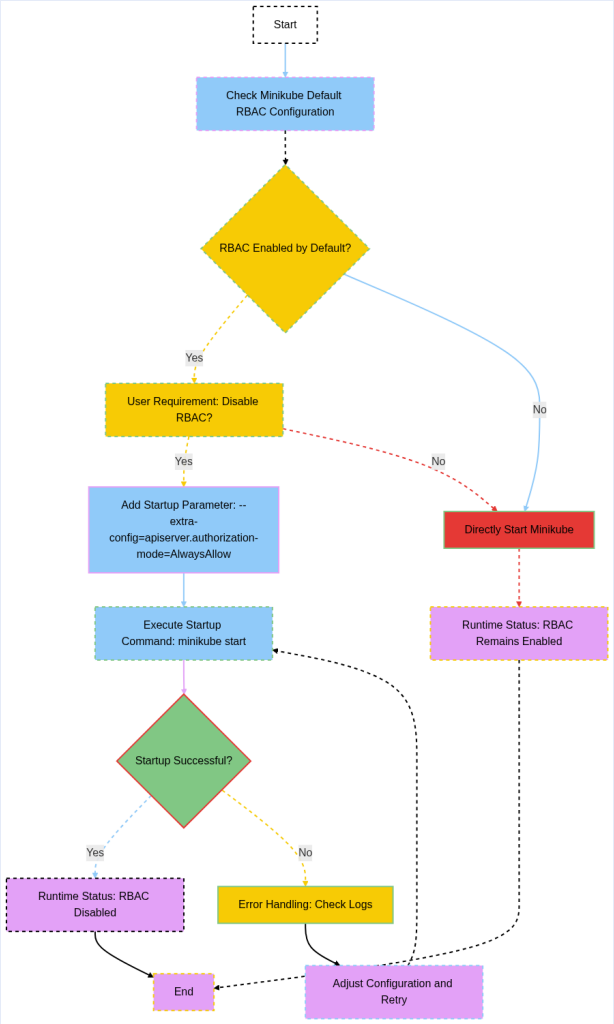
3. Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [No] -> Check Boot for Carbon -> Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [No] -> Check Boot for Carbon -> Black Line Inside Boot? -> [Yes] -> Inspect Spark Plug -> Last Wire? -> [No] -> Select Wire to Replace ✗ (wrong subpath)

4. Record Wire Configuration -> All Wires Replaced? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [No] -> Check Boot for Carbon -> Black Line Inside Boot? -> [Yes] -> Inspect Spark Plug -> Last Wire? -> [No] -> Select Wire to Replace -> Use Removal Tool -> Wire Stuck? -> [No] -> Check Boot for Carbon -> Black Line Inside Boot? -> [No] -> Install New Wire -> Last Wire? -> [No] -> Select Wire to Replace ✗ (wrong subpath)

Qwen2.5vl-72b: not exists ✗

Qwen2.5vl-7b-comprehensive: Failed ✗

Figure 23. Comprehensive - Non-Semantic Query: Path-Centric Class2 - How (edge).



Semantic - Element-Centric Class1: What/Which

Masked Subpath: Error Handling: Check Logs -> {6}

Q (edge): In the process of Error Handling: Check Logs, what are the crucial actions I need to consider in the next six steps from this point onward? All the edges in the picture that are bordered with pink represent crucial actions. Please list these actions in order.

GT: Execute Startup Command: minikube start -> Startup Successful?

GPT-5: not exists ✗

Gemnini: Adjust Configuration and Retry -> Execute Startup Command: minikube start -> Error Handling: Check Logs -> Adjust Configuration and Retry -> Execute Startup Command: minikube start -> Error Handling: Check Logs ✗

Qwen2.5vl-72b: Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace ✗

Qwen2.5vl-7b-comprehensive: Error Handling: Check Logs -> Adjust Configuration and Retry ✗ Execute Startup Command: minikube start -> Startup Successful? ✗ (GT is missed by all evaluated models.)

Semantic-Rewrite - Element-Centric Class1: What/Which

Masked Subpath: Error Handling: Check Logs -> {6}

Q (edge): In the process of Handling Errors: Review Log Files, what are the important transitions I need to take in the next six steps starting from here? All the edges in the picture that are outlined with pink signify important transitions. Please list these important transitions in sequence.

GT: Execute Startup Command: minikube start -> Startup Successful?

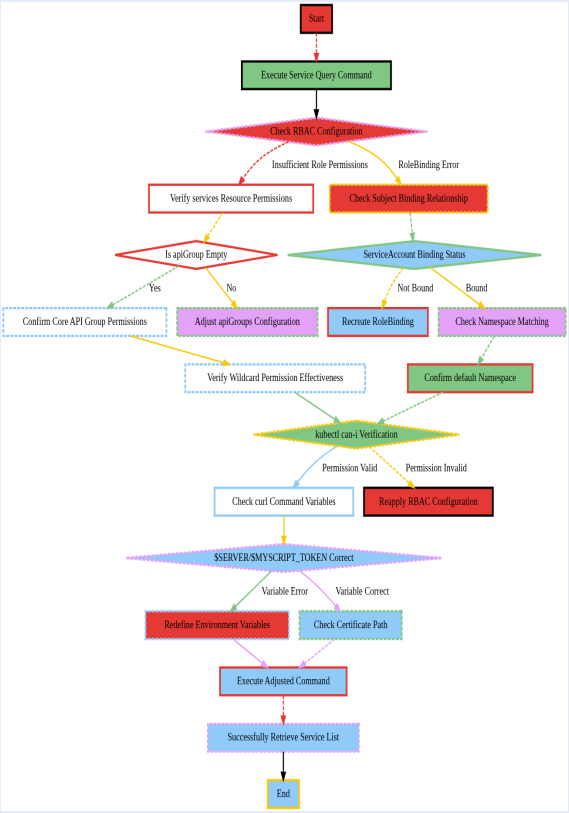
GPT-5: not exists ✗

Gemnini: Error Handling: Check Logs -> Adjust Configuration and Retry, Adjust Configuration and Retry -> Execute Startup Command: minikube start, Execute Startup Command: minikube start -> Startup Successful?, Startup Successful? -> [No] -> Error Handling: Check Logs, Error Handling: Check Logs -> Adjust Configuration and Retry,Adjust Configuration and Retry -> Execute Startup C ommand: minikube start ✗

Qwen2.5vl-72b: not exists ✗

Qwen2.5vl-7b-comprehensive: Error Handling: Check Logs -> Adjust Configuration and Retry ✗ Execute Startup Command: minikube start -> Startup Successful? ✗ (GT is missed by all evaluated models.)

Figure 24. Comprehensive - Semantic and Semantic-Rewrite Query: Element-Centric Class1 - What/Which (edge).



Semantic - Path-Centric Class2: How

Masked Subpath: {10} -> Redefine Environment Variables -> Execute Adjusted Command -> Successfully Retrieve Service List -> End

Q (node): Before reaching the Redefine Environment Variables step and continuing through Execute Adjusted Command to Successfully Retrieve Service List and ending at End, I need to complete ten consecutive steps. Could you help me find all possible paths for these ten steps that include the most user-friendly actions? Note: All green-filled nodes in the picture represent user-friendly actions.

GT: Start -> Execute Service Query Command -> Check RBAC Configuration -> [RoleBinding Error] -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error]

GPT-5: 1. Start -> Execute Service Query Command -> Check RBAC Configuration -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Not Bound] -> Recreate RoleBinding -> Confirm default Namespace -> kubectl can-i Verification -> Check curl Command Variables -> SSERVER/SMYSCRIPT_TOKEN Correct ✗

2. Start -> Execute Service Query Command -> Check RBAC Configuration -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> SSERVER/SMYSCRIPT_TOKEN Correct -> [Variable Error] ✗ (edge label missed & OCR error)

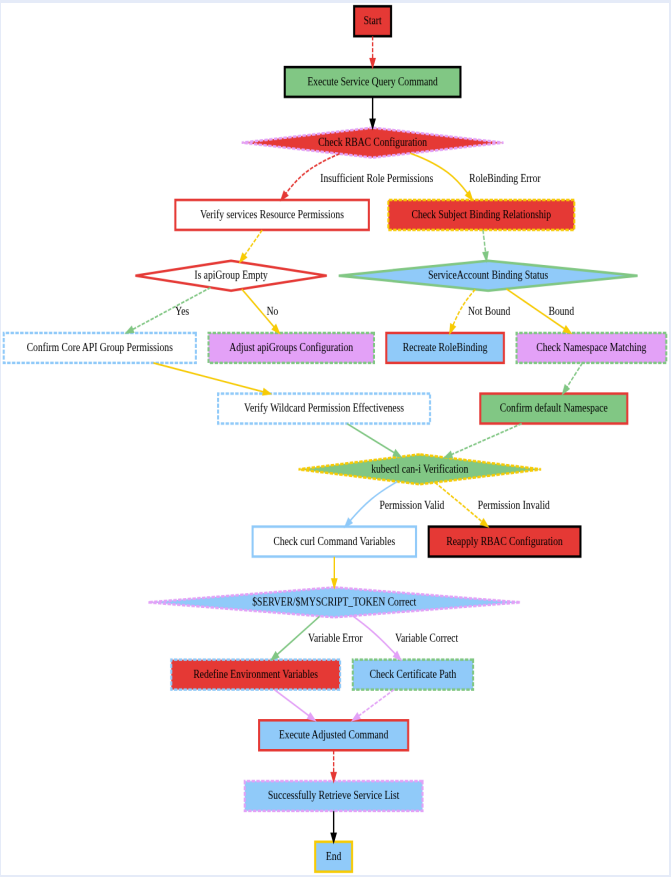
Gemini: 1. Start -> Execute Service Query Command -> Check RBAC Configuration -> [RoleBinding Error] -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error] CORRECT. ✓

2. Execute Service Query Command -> Check RBAC Configuration -> [RoleBinding Error] -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Not Bound] -> Recreate RoleBinding -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error] ✗ (hallucination)

Qwen2.5vl-72b: Start -> Execute Service Query Command -> Check RBAC Configuration -> [RoleBinding Error] -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error] ✗ (edge label missed)

Qwen2.5vl-7b-comprehensive: Start -> Execute Service Query Command -> Check RBAC Configuration -> [Insufficient Role Permissions] -> Verify services Resource Permissions -> Is apiGroup Empty -> [No] -> Adjust apiGroups Configuration (subpath correct but violates the 'most' constraint; GT subpath has two.) -> Verify Wildcard Permission Effectiveness (hallucination) -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> S$SERVER=$MYSCRIPT_TOKEN Correct -> Variable Correct ✗

Figure 25. Comprehensive - Semantic Query: Path-Centric Class2 - How (node).



Semantic-Rewrite - Path-Centric Class2: How simple & cyclic subpaths are both allowed):
Masked Subpath: {10} -> Redefine Environment Variables -> Execute Adjusted Command -> Successfully Retrieve Service List -> End
Q (node): Before reaching the Reset Environment Variables step and proceeding through Run Modified Command to Retrieve Service List Successfully and concludeing at End, I need to complete ten sequential steps. Could you assist me in identifying every possible path for these ten steps that incorporates the most user-friendly actions? Note: All nodes filled with green in the diagram signify user-friendly actions.
GT: Start -> Execute Service Query Command -> Check RBAC Configuration -> [RoleBinding Error] -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error]

GPT-5: Start -> Execute Service Query Command -> Check RBAC Configuration -> [RoleBinding Error] -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> SSERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error] ✗ (OCR error)
Gemini: Start -> Execute Service Query Command -> Check RBAC Configuration -> [RoleBinding Error] -> Check Subject Binding Relationship -> ServiceAccount Binding Status -> [Bound] -> Check Namespace Matching -> Confirm default Namespace -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> [Variable Error] -> Redefine Environment Variables ✗ (subpath correct, but mismatch in step count)
Qwen2.5vl-72b: Verify Wildcard Permission Effectiveness -> kubectl can-i Verification -> Permission Valid -> Check curl Command Variables -> $SERVER/$MYSCRIPT_TOKEN Correct -> Variable Error -> Redefine Environment Variables -> Execute Adjusted Command -> Successfully Retrieve Service List -> End ✗ (subpath correct but violates the 'most' constraint; GT subpath has two.)
Qwen2.5vl-7b-comprehensive: Start -> Execute Service Query Command -> Check RBAC Configuration -> [Insufficient Role Permissions] -> Verify services Resource Permissions -> Is apiGroup Empty -> [No] -> (subpath correct but violates the 'most' constraint; GT subpath has two.) Adjust apiGroups Configuration -> (hallucination) Verify Wildcard Permission Effectiveness -> kubectl can-i Verification -> [Permission Valid] -> Check curl Command Variables -> S$SERVER=$MYSCRIPT_TOKEN Correct -> Variable Correct ✗

Figure 26. Comprehensive - Semantic-Rewrite Query: Path-Centric Class2 - How (node).