

# Hierarchical Process Reward Models are Symbolic Vision Learners

## Supplementary Material

### A. Synthetic Data Construction

To train the symbolic vision encoder for diagram parsing, we develop a synthetic data generation pipeline that produces paired (diagram, logic form) examples. This pipeline uses computational geometry to generate diverse geometric configurations with accompanying structured symbolic representations. Algorithm 1 summarizes the complete synthetic data generation pipeline, integrating theorem construction, diagram rendering, logic form enrichment, and conversation formatting.

#### A.1. Theorem Generation via Constructive Synthesis

Our data generation process begins with synthetic theorem construction, following AlphaGeometry [31]. Unlike AlphaGeometry, we generate only geometric configurations through a sequence of constructive operations, without requiring any deductive reasoning steps.

**Construction Operations.** We employ a fixed set of geometric construction primitives:

- **Point placement:** Random sampling in normalized coordinates  $[0, 1]^2$
- **Line construction:** Connect existing points or construct perpendiculars, parallels, and angle bisectors
- **Circle construction:** Define circles by center and radius, or through three non-collinear points
- **Intersection points:** Compute line-line, line-circle, and circle-circle intersections
- **Derived constructions:** Midpoints, perpendicular bisectors, circumcenters, and incenters

**Configuration Sampling.** For each synthetic example, we randomly sample a construction sequence of length  $k \sim \text{Uniform}(5, 15)$  operations. Each operation is selected with probability proportional to the current complexity budget, ensuring diverse configurations while avoiding degenerate cases (e.g., collinear points, overlapping circles).

**Symbolic Representation.** Each configuration is serialized into a structured logic form  $\mathcal{S}$  containing:

$$\mathcal{S} = \{\mathcal{P} \text{ (point names/positions)}, \mathcal{E} \text{ (line instances)}, \mathcal{G} \text{ (shape instances)}, \mathcal{M} \text{ (shape indicators)}, \mathcal{R} \text{ (geometric relations)}\}, \quad (11)$$

where  $\mathcal{P}$  encodes normalized coordinates,  $\mathcal{E}$  represents connectivity,  $\mathcal{G}$  describes shape types,  $\mathcal{M}$  stores structural properties of shapes, and  $\mathcal{R}$  captures primitive relations.

#### A.2. Diagram Rendering from Logic Forms

Given logic forms, we render a corresponding geometric diagram  $\mathbf{I} \in \mathbb{R}^{512 \times 512 \times 3}$  using matplotlib’s vector graphics engine.

**Rendering Configuration.** We enforce fixed plotting mode to ensure consistent visual appearance across the dataset:

- **Background:** White background (RGB = (255, 255, 255))
- **Point style:** Black scatter markers (size 15px)
- **Line style:** Black solid lines (width 1.2px, opacity 0.8)
- **Circle style:** Blue unfilled circles (width 1.2px, opacity 0.8)
- **Label style:** Black text with automatic positioning (font size 15pt)

**Coordinate System.** Point coordinates in  $\mathcal{P}$  are normalized to  $[0, 1]^2$ . We apply a  $y$ -axis flip transformation  $y' = 1 - y$  to align with matplotlib’s bottom-left origin convention. The viewport is dynamically scaled to encompass all geometric elements with a 40% margin.

#### A.3. Logic Form Enrichment

Raw synthetic configurations lack explicit structural properties that aid diagram understanding. We augment each logic form with **indicator attributes** that describe shape-specific characteristics. For each shape instance  $g \in \mathcal{G}$ , we use keyword matching to extract predefined spatial shape types:

- **Triangles:** type (equilateral, isosceles, right) with angle and equality measurements.
- **Quadrilaterals:** type (square, rectangle, parallelogram, trapezoid) with parallel/perpendicular relationships.

These indicators are stored in the  $\mathcal{M}$  component of the logic form (Eq. 11), providing rich semantic annotations for training.

---

**Algorithm 1** Synthetic Data Generation

---

**Require:** Number of examples  $N$ , construction operation set  $\Omega$

**Ensure:** Dataset  $\mathcal{D} = \{(\mathbf{I}_i, \mathcal{S}_i, \mathcal{T}_i)\}_{i=1}^N$  where  $\mathbf{I}$  is diagram,  $\mathcal{S}$  is logic form,  $\mathcal{T}$  is conversation

```
1: for  $i = 1$  to  $N$  do
2:   // Stage 1: Theorem Generation
3:   Sample construction sequence length  $k \sim \text{Uniform}(5, 15)$ 
4:   Initialize empty configuration  $\mathcal{Q} = \emptyset$ 
5:   for  $j = 1$  to  $k$  do
6:     Sample operation  $\omega \sim \Omega$  with complexity-weighted probability
7:     Apply  $\omega$  to  $\mathcal{Q}$  (add points, lines, circles, intersections)
8:     if configuration is degenerate (collinear points, overlapping circles) then
9:       Reject  $\omega$  and resample
10:    end if
11:  end for
12:  Serialize  $\mathcal{Q}$  to logic form  $\mathcal{S}_i = \{\mathcal{P}, \mathcal{E}, \mathcal{G}, \mathcal{M}_{\text{empty}}, \mathcal{R}\}$ 
13:
14:  // Stage 2: Diagram Rendering
15:  Normalize point coordinates:  $\mathcal{P} \leftarrow \{(n, (x, 1 - y)) \mid (n, (x, y)) \in \mathcal{P}\}$ 
16:  Compute viewport bounds  $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$  with 40% margin
17:  Initialize canvas  $\mathbf{I}_i \in \mathbb{R}^{512 \times 512 \times 3}$  with white background
18:  Render lines from  $\mathcal{E}$  (black, width 1.2px)
19:  Render circles from  $\mathcal{C}$  (blue, unfilled, width 1.2px)
20:  Render points and labels from  $\mathcal{P}$  (black markers, font size 15pt)
21:  Save  $\mathbf{I}_i$  as PNG file
22:
23:  // Stage 3: Logic Form Enrichment
24:  for each shape  $g \in \mathcal{G}$  do
25:    Extract shape type and constituent points
26:    Classify  $g$  into special shape categories via keyword matching
27:    Create indicator  $\iota_g \leftarrow \{\text{shape, properties}\}$ 
28:    Add  $\iota_g$  to indicator set  $\mathcal{M}$ 
29:  end for
30:  Update logic form:  $\mathcal{S}_i \leftarrow \{\mathcal{P}, \mathcal{E}, \mathcal{G}, \mathcal{M}, \mathcal{R}\}$ 
31:
32:  // Stage 4: Conversation Construction
33:  Create conversation  $\mathcal{T}_i \leftarrow \{\text{human} : \text{"<image> Generate logic forms \dots"}, \text{gpt} : \mathcal{S}_i\}$ 
34:  Add  $(\mathbf{I}_i, \mathcal{S}_i, \mathcal{T}_i)$  to dataset  $\mathcal{D}$ 
35: end for
36: Save diagrams  $\{\mathbf{I}_i\}$  as PNG files, metadata  $\mathcal{T}_i$  as JSON
37: return  $\mathcal{D}$ 
```

---

#### A.4. Conversation Dataset Construction

To train the model in an instruction-following format, we convert logic forms into multi-turn conversational data.

**Conversation Template.** Each training example  $(\mathbf{I}, \mathcal{S})$  is formatted as:

```
{
  "id": "sample_*",
  "image": "/path/to/diagram.png",
  "conversations": [
    {
      "from": "human",
      "value": "<image>\nGenerate the logic forms
```

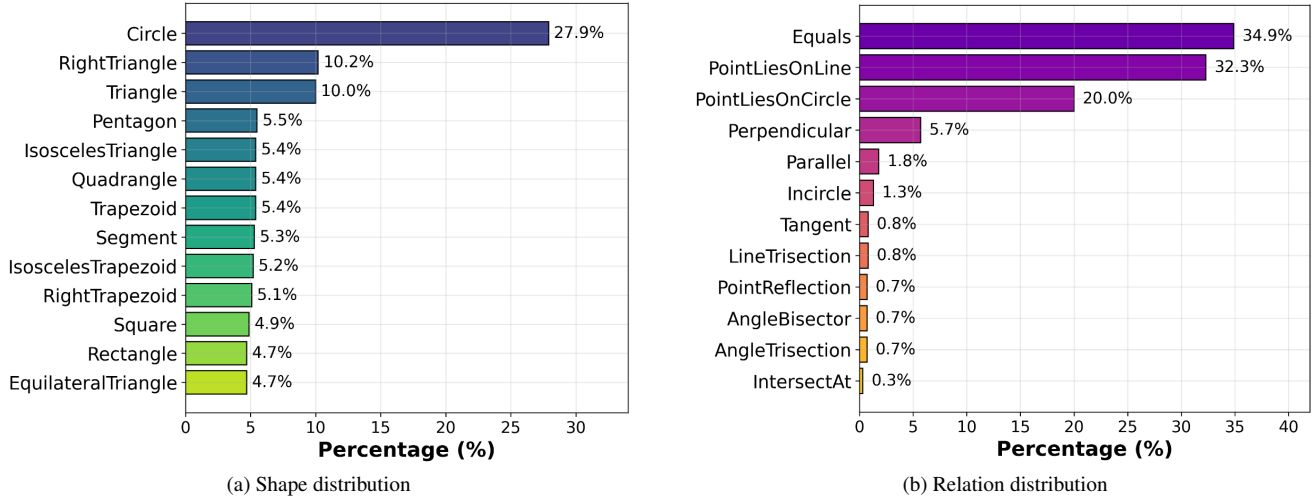


Figure 5. Distributions of geometric shapes and relations in our synthetic dataset. The dataset exhibits diverse shape categories and relation types, providing comprehensive supervision for learning topological diagram structures and hierarchical geometric relationships.

```

        for this geometric diagram."
    },
    {
      "from": "gpt",
      "value": "logic form"
    }
  ]
}

```

## B. Data Statistics

During training, we generate a 105K synthetic dataset, consisting of a 100K subset used to train SymParser and a 5K subset used for SymHPR. On average, each diagram contains 6 points, 7 lines, 2 polygonal shapes, and 1 circle, reflecting a moderate level of geometric complexity. The shape distribution is well-balanced across 13 predefined categories, with `Circle` (27.9%), `RightTriangle` (10.2%), and `Triangle` (10.0%) appearing most frequently, while the remaining categories each contribute between 4.7% and 5.5%. For shape properties, `Parallel` (55.5%), `Perpendicular` (22.4%), and `Equals` (22.1%) relations collectively capture the dominant geometric constraints in typical constructions. At the relational level, `Equals` constraints constitute the largest proportion (34.9%), followed by `PointLiesOnLine` (32.3%) and `PointLiesOnCircle` (20.0%). Less frequent but crucial relations—including `Perpendicular` (5.7%), `Parallel` (1.8%), `Incircle` (1.3%), and `Tangent` (0.8%)—ensure coverage of more intricate geometric configurations. This diverse composition provides a rich supervisory signal, enabling the model to acquire robust and hierarchical representations of geometric primitives, constraints, and structural dependencies. The full distributions of shape types and geometric relations are illustrated in Figs. 5a and 5b.

## C. Logic Forms

During hierarchical reward modeling, we use a 9K dataset comprising 5K synthetic samples and 4K diagrams from an extended version of PGDP [40], where we augment the original diagrams with additional geometric relationships. Moreover, we corrected several annotation errors in the PGDP dataset. In particular, many diagrams lacked explicit vertex point annotations required for our logic-form representation. We manually added the missing vertex points to ensure complete geometric specifications—for example, points A–F in Fig. 9 (left) and points A–G in Fig. 9 (right).

Figs. 6–9 visualize the symbolic logic forms alongside their rendered diagrams. Each example illustrates the full hierarchical structure: normalized point coordinates, line segments connecting vertices, shape primitives with geometric indicators (e.g., parallel sides, perpendicular edges), and inter-primitive relations (e.g., `PointLiesOnLine`, `Incircle`).

## D. Symbolic Decoder

Our reconstruction decoder converts structured logic-form representations into geometric diagrams to compute visual similarity rewards in Eq. 7. The decoder follows a multi-stage pipeline: it first parses the symbolic representations to extract geometric primitives (points, lines, and circles), computes derived attributes such as circle centers and radii, and finally renders the diagrams using computational geometry libraries (*e.g.*, `matplotlib`). We summarize the entire process in Algorithm 2.

### D.1. Parsing Process

**Point Positions.** We extract point/vertex coordinates using regular expression matching:

$$\mathcal{P} = \{(n_i, \mathbf{p}_i) \mid i = 1, \dots, N_p\}, \quad (12)$$

where  $n_i$  denotes the point name and  $\mathbf{p}_i = (x_i, y_i) \in [0, 1]^2$  represents normalized coordinates. We apply a coordinate transformation  $y'_i = 1 - y_i$  to align with the bottom-left origin convention.

**Line Instances.** Line segments are represented as ordered pairs of point names:

$$\mathcal{E} = \{(n_j^{(1)}, n_j^{(2)}) \mid j = 1, \dots, N_\ell\} \quad (13)$$

Each line instance string (*e.g.*, “ab”) is decomposed into constituent point identifiers that index into  $\mathcal{P}$ .

**Circle Instances.** Circles are identified through two mechanisms: (1) explicit declarations in shape representations (*e.g.*, `Circle(o)`), extracting center point names, and (2) implicit definitions via geometric relations. We maintain a set of circle center identifiers:

$$\mathcal{C} = \{c_k \mid k = 1, \dots, N_c\} \quad (14)$$

For the implicit geometric relations, we parse three types of circle-defining relations that enable radius computation, as follows:

(a) From `PointLiesOnCircle(p, Circle(c, radius))` relations, we compute the radius as the Euclidean distance:

$$r_c = \|\mathbf{p}_p - \mathbf{p}_c\|_2 \quad (15)$$

(b) For `ConcyclicPoints(p1, p2, p3, p4)`, we fit a circumcircle using least-squares optimization. The circle equation  $(x - a)^2 + (y - b)^2 = r^2$  is linearized to:

$$x^2 + y^2 = 2ax + 2by + c, \quad \text{where } c = r^2 - a^2 - b^2 \quad (16)$$

Given  $n \geq 3$  points, we solve the overdetermined linear system:

$$\begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} 2a \\ 2b \\ c \end{bmatrix} = \begin{bmatrix} x_1^2 + y_1^2 \\ \vdots \\ x_n^2 + y_n^2 \end{bmatrix} \quad (17)$$

via least-squares to obtain center  $\mathbf{c} = (a, b)$  and radius  $r = \sqrt{c + a^2 + b^2}$ . We reject solutions where the maximum fitting error exceeds 10% of the radius.

(c) From `Incircle(Circle(c, radius), Triangle(v1, v2, v3))`, we compute the incircle radius using Heron’s formula. For triangle side lengths  $a, b, c$  and semiperimeter  $s = (a + b + c)/2$ :

$$r = \frac{A}{s}, \quad \text{where } A = \sqrt{s(s-a)(s-b)(s-c)} \quad (18)$$

### D.2. Rendering Pipeline

The rendering process transforms the parsed geometric primitives into a rasterized image  $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ . For visual similarity computation, the reconstructed diagrams are resized to match the ground-truth sizes, ensuring identical resolution for alignment.

---

**Algorithm 2** Diagram Reconstruction

---

**Require:** Logic forms  $\hat{\mathcal{S}}$

**Ensure:** Rendered image  $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$

- 1: Parse  $\hat{\mathcal{S}}$  to extract:
  - 2: - Point set  $\mathcal{P} = \{(n_i, \mathbf{p}_i)\}$  via regex matching
  - 3: - Line set  $\mathcal{E} = \{(n_j^{(1)}, n_j^{(2)})\}$  from line instances
  - 4: - Circle set  $\mathcal{C} = \{c_k\}$ , derived from both explicit shape representations and implicit geometric relations.
  - 5: Compute circle radii  $\{r_k\}$  from geometric relations:
  - 6: - PointLiesOnCircle:  $r_c \leftarrow \|\mathbf{p}_p - \mathbf{p}_c\|_2$  (Eq. 15)
  - 7: - ConcyclicPoints:  $(\mathbf{c}, r) \leftarrow \text{FITCIRCLELEASTSQUARES}(\{\mathbf{p}_1, \dots, \mathbf{p}_4\})$  (Eq. 17)
  - 8: - Incircle:  $r \leftarrow \text{COMPUTEINCIRCLERADIUS}(\text{vertices})$  (Eq. 18)
  - 9: Establish coordinate system:
  - 10: - Compute bounding box  $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$  via Eq. 19-20
  - 11: - Extend by margin factor  $\alpha = 0.4$  via Eq. 21
  - 12: - Set equal aspect ratio
  - 13: Render primitives in order:
  - 14: - Draw points and names:  $\forall (n_i, \mathbf{p}_i) \in \mathcal{P}$
  - 15: - Draw lines:  $\forall (n_j^{(1)}, n_j^{(2)}) \in \mathcal{E}$
  - 16: - Draw circles:  $\forall c_k \in \mathcal{C}$  with radius  $r_k$
  - 17: Save canvas as PNG  $\rightarrow \mathbf{I}$
  - 18: **return**  $\mathbf{I}$
- 

**Coordinate System Setup.** We compute the axis-aligned bounding box encompassing all points and circles:

$$x_{\min} = \min_i \{x_i\} - \min_k \{r_k\}, \quad x_{\max} = \max_i \{x_i\} + \max_k \{r_k\} \quad (19)$$

$$y_{\min} = \min_i \{y_i\} - \min_k \{r_k\}, \quad y_{\max} = \max_i \{y_i\} + \max_k \{r_k\} \quad (20)$$

A margin factor  $\alpha = 0.4$  is applied to extend the viewport:

$$[x_{\min}, x_{\max}] \leftarrow [x_{\min} - \alpha \Delta_x, x_{\max} + \alpha \Delta_x] \quad (21)$$

where  $\Delta_x = x_{\max} - x_{\min}$ , and similarly for the  $y$ -axis. Equal aspect ratio is enforced to prevent distortion.

**Primitive Rendering.** Geometric elements are rendered as follows: (a) *Lines*: Each line  $(n_j^{(1)}, n_j^{(2)}) \in \mathcal{E}$  is rasterized as a black line segment connecting  $\mathbf{p}_j^{(1)}$  and  $\mathbf{p}_j^{(2)}$  with linewidth 1.2 pixels and opacity  $\alpha = 0.8$ ; (b) *Circles*: Each circle  $c_k \in \mathcal{C}$  is rendered as an unfilled circle with center  $\mathbf{p}_{c_k}$  and radius  $r_k$ , using blue stroke color (RGB: 0, 0, 255), linewidth 1.2 pixels, and opacity  $\alpha = 0.8$ ; (c) *Points and Names*: Each point  $\mathbf{p}_i$  is drawn as a black scatter marker (size 15 pixels, z-order 10). Point names are positioned using the naming algorithm [31], which computes smart offsets to avoid overlaps with lines and circles. The rendered canvas is saved as a PNG image with white background at 100 DPI.

**Failure Handling.** In cases where parsing fails or produces degenerate geometry (e.g., empty point sets, singular circle fits), the decoder generates a black image to signal reconstruction failure, resulting in minimal visual similarity reward.

## E. Training Details

**Cold Start Training (SymParser):** We initialize the symbolic vision encoder using supervised fine-tuning on a synthetic dataset of 100K diagram-logic form pairs, training for one epoch to establish foundational symbolic parsing capability. During cold start training, we use a global batch size of 256, with the learning rate to  $1 \times 10^{-5}$  (a vision-specific learning rate of  $2 \times 10^{-6}$ ). We employ the Adam optimizer without weight decay and apply a cosine learning rate schedule. To improve memory efficiency, we utilize Fully Sharded Data Parallel (FSDP), gradient checkpointing, and enable BF16 precision, avoiding CPU/GPU offloading to maximize throughput. All 3B/7B model training is performed on  $8 \times \text{A100 GPUs}$  (80GB each), with alignment training taking approximately 9-12 hours.

**Hierarchical Reward Modeling (SymHPR):** Hierarchical reward modeling uses reinforcement learning (RL) to capture the inherent structural hierarchy of geometric diagrams. We train the encoder on 9,000 samples from our synthetic dataset and

the extended PGDP data, enabling the model to learn compositional dependencies across levels: points  $\Rightarrow$  lines  $\Rightarrow$  shapes  $\Rightarrow$  relations. We adopt the Verifiers framework [2] and optimize using Group Relative Policy Optimization (GRPO). Our cluster configuration consists of 3 nodes, with 1 inference node and 2 training nodes, each equipped with  $2 \times$  A100 GPUs (40GB) and 12 CPU cores, with a SLURM time limit of 2 days. We use a per-device batch size of 4 and generate 8 rollouts per prompt (GRPO group size). Combined with gradient accumulation over 2 steps, this yields an effective batch size of 32 unique prompts per update ( $4 \text{ prompts} \times 4 \text{ GPUs} \times 2 \text{ accumulation steps}$ ). The learning rate is set to  $5 \times 10^{-7}$  with a KL penalty coefficient  $\beta = 0.03$ . The maximum generation length is 1,024 tokens.

**Self-Supervised Learning (SymVAE):** We train SymVAE on 16,000 diagrams, consisting of 5,000 samples from Geo170K [6], 7,000 synthetic diagrams, and 4,000 diagrams from PGDP [40]. The model learns self-supervised reconstruction through perceptual rewards, enabling structure-aware diagram encoding. To stabilize RL optimization, we apply power normalization and noise annealing as described in the main paper (Sec. 3.2.2).

**Downstream Task Fine-Tuning (SymVAE+\*):** For downstream applications, we apply LoRA [11] adaptation (rank=64) and train for one epoch on task-specific datasets. (1) **Perception (MathGlance [30]):** We fine-tune on 10K planar geometry diagrams (1/10 of the full 100K dataset) with direct-answer supervision to allow the model to adapt to the conversation instructions and enhance primitive recognition capabilities; (2) **Chart Reconstruction (VisCodex [12]):** We train on 100K chart-code pairs (1/5 of the full 598K dataset) to evaluate cross-domain symbolic representation transfer via chart-to-code generation; (3) **Reasoning:** We train on Geo170K [6] and MathV360K [28] using question-answering with chain-of-thought supervision to enable mathematical problem reasoning tasks.

**Reconstructions Beyond Geometry:** To assess the generality of our reconstruction framework beyond planar geometry, we conduct experiments on two out-of-domain diagram types: electrical circuits and molecular structures. Although circuit and molecular diagrams differ from geometric figures, a portion of their structure is still formed by basic geometric primitives such as points, lines, circles, and polygonal contours. This structural overlap enables our representation to transfer across domains. For circuits, we collect 2,903 images from the `electronics_diagrams_single` subset of MMMU-Pro [1] and EEE-Bench [2]. For molecular diagrams, we use 923 images from the MMChemOCR subset introduced in ChemVLM [3]. To ensure that the extended domains remain compatible with our symbolic rules, we adopt a preprocessing step that filters out samples with complex topologies, *e.g.*, sinusoidal AC waveforms in circuits and wedge-dash notation for three-dimensional stereochemistry, based on the visual reconstruction reward (Eq. 7). Specifically, we run SymVAE on all circuit and molecular images and retain only those examples with a visual reward greater than 0.6 ( $r_{\text{vis}} > 0.6$ ). After filtering, we obtain 536 circuit diagrams and 447 molecular diagrams for training. We use the same training configuration as in SymVAE for this experiment.

## F. SymVAE Reconstruction Visualizations

We provide reconstruction visualizations produced by SymVAE across electrical circuit diagrams and molecular structure diagrams in Figs. 10–11. These results demonstrate the model’s ability to recover both low-level primitives and global structural organization. In each demonstration, we present triplets consisting of the ground-truth input, the latent logic-form primitives, and the final reconstruction. For clarity, we omit vertex indicators in the visualizations. Since those circuit and molecular diagrams are primarily composed of basic geometric primitives, *e.g.*, points, lines, and polygonal shapes, our model can faithfully reconstruct their topologies.

We further compare our geometric reconstructions with those generated by Qwen2.5-VL-7B (Fig. 12 (a)) and GPT-4o (Fig. 12 (b)). General-purpose multimodal models struggle to preserve fine-grained geometric structure, particularly in complex diagrams, whereas our approach yields substantially more faithful geometric and relational reconstructions (Fig. 12 (c)). Notably, for these comparators, we prompt the models to generate Python plotting code and render the diagrams via Matplotlib, while our method reconstructs diagrams directly from symbolic logic forms through our dedicated decoder.

As discussed in Limitation J, the rule-based nature of our symbolic design limits flexibility when extending the system to new geometric concepts. Consequently, when input diagrams contain structures outside our geometric primitives—particularly those with higher visual complexity or domain-specific topology—our model fails to fully reconstruct them. In Fig. 13, the model is unable to recover domain-specific graphical symbols whose semantics lie beyond basic geometry. Examples include the wavy-line representation of resistors and the lettered ammeter symbol (circled “A”). Moreover, for diagrams with dense geometric content, such as the clustered shapes shown in the top rows, our model struggles to recover all structural details accurately. We discuss potential extensions to address these limitations in further work (Sec. J).

## G. Cross-Lingual and Cross-Modal Translation

**Cross-Lingual Translation.** After fine-tuning on downstream perception tasks using the GeoPeP conversational-format samples, we observe an emergent cross-lingual transfer phenomenon during inference: formal logic forms are automatically translated into fluent natural-language chain-of-thoughts. This indicates that the model internalizes the symbolic structure encoded in our logic-form representation and can project it into coherent linguistic reasoning. Remarkably, this ability emerges without any explicit CoT supervision, suggesting that our logic forms are intrinsically aligned with natural-language semantics—an alignment that other program-like syntaxes (*e.g.*, Python) fail to achieve.

As shown in Fig. 14a, the model’s reasoning faithfully follows the hierarchical structure encoded in the logic forms: beginning with vertices, then edges, and finally shape attributes before identifying the correct shape category. We hypothesize that this emergent behavior is enabled by the strong NLP reasoning priors of the base model (Qwen2.5-VL-7B). To determine whether this phenomenon reflects genuine cross-lingual transfer or merely mirrors the base model’s priors, we compare our model’s reasoning traces with those generated by the original Qwen2.5-VL-7B. Indeed, the base model exhibits a markedly different reasoning pattern—often starting from global shape descriptions and only later mentioning sides or angles—deviating from the hierarchical primitive structure defined in our logic rules. Similar patterns appear in Figs. 14b-15: our model consistently follows the correct reasoning trajectory and produces the correct answer, whereas Qwen2.5-VL-7B generates disorganized or inconsistent reasoning steps that lead to incorrect predictions.

**Cross-Modal Translation.** Another interesting finding is an emergent form of cross-modal translation. Although our training data contain only planar geometric diagrams, where the fundamental primitives are points corresponding to 2D vertices, we observe that the model can automatically adapt these primitives when inferring over solid 3D scenes. In particular, the model enriches the point instances with 3D attributes such as object shape, color, and material. As shown in Fig. 16, the inferred point primitives become `cylinder_red`, `cube_cyan`, and `cylinder_blue`, demonstrating a self-adaptation from planar vertices to 3D object descriptors. Furthermore, as illustrated in Figs. 17a–17b, when multiple objects share the same color, the model automatically introduces disambiguating markers (*e.g.*, `cyan_1`, `cyan_2`) to distinguish them. By explicitly enumerating foreground objects with these extended attributes, the model can more reliably support downstream tasks such as object counting and spatial relationship identification.

## H. Case Studies on Reasoning

Figs. 18 and 19 present qualitative comparisons on the MathVerse benchmark. Our model consistently produces correct reasoning chains grounded in accurate visual perception, while GPT-4o and Qwen2.5-VL-7B often misidentify geometric configurations, leading to erroneous conclusions. The visual tokens produced by our symbolic vision encoder are more closely aligned with natural-language feature spaces, enabling the model to learn visually grounded reasoning for multimodal problem solving and significantly reducing visual hallucinations.

Specifically, symbolic vision tokens establish a more reliable perceptual foundation for reasoning. Across many examples, the base model fails because it starts with an incorrect perceptual premise (*e.g.*, misidentifying key vertices or an angle), causing the entire reasoning chain to collapse. In contrast, our model corrects these initial perceptual errors, enabling the reasoning process to begin from an accurate visual foundation. For instance, in Fig. 18a, the base model incorrectly recognizes the  $\angle CBD$ , whereas our model correctly identifies it and naturally applies the Angle Sum Theorem to reach the correct answer. Similarly, in Fig. 18b, our model accurately perceives the perpendicular relationship, infers the right angle, and proceeds with the appropriate reasoning steps. Beyond accuracy, improved perception enables the model to generate more direct reasoning trajectories rather than overthinking or searching for unnecessarily complex alternatives. By removing perceptual ambiguity, the model can commit earlier to the correct strategy. In the more challenging examples shown in Fig. 19, our model demonstrates significantly more stable multi-step reasoning because the intermediate perceptual relations—especially in vision-only settings on MathVerse test samples—are now correctly grounded.

## I. More Results

### I.1. Neuro-Symbolic Reasoning with Adaptive Visual-Textual Rewards

Instead of integrating visual tokens directly into MLLMs—the conventional design of multimodal large language models—we adopt a neuro-symbolic architecture. Specifically, our SymParser generates executable logic forms from diagrams, which are subsequently processed by a mathematical reasoning model (Qwen2.5-Math-7B) to solve multimodal geometric problems. We evaluate this neuro-symbolic pipeline on the GeoQA benchmark.

Concretely, we first apply SymParser to 5K diagrams from Geo170K to generate paired logic forms, which are prepended

to the corresponding questions with reasoning chains-of-thought (CoTs). The resulting data serve as supervised fine-tuning (SFT) samples for Qwen2.5-Math-7B. We then apply reinforcement learning (GRPO) using conventional textual answer rewards. Since our logic forms are executable, we further introduce a **visual reconstruction reward**  $r_{\text{vis}}$  (Eq. 7), where reconstructed diagrams are generated from logic forms using our decoder. The total reward thus combines textual accuracy (final answer and format) with visual fidelity, jointly guiding the policy model toward grounded multimodal reasoning.

During training, we observe that rollouts with low visual rewards often produce irrelevant or malformed outputs, including nonsensical or “garbage” tokens unrelated to the question. In contrast, samples with higher visual rewards demonstrate coherent reasoning that effectively references visual cues. To mitigate the impact of low-quality visual grounding, we introduce an **adaptive reward weighting mechanism** that dynamically adjusts the contribution of visual rewards based on mathematical correctness. The key insight is that visual understanding should primarily support correct reasoning. When the model generates incorrect answers ( $r_{\text{math}} \leq \tau$ ), we exponentially decay the visual reward to prevent spurious visual–textual correlations. Conversely, when the answers are correct ( $r_{\text{math}} > \tau$ ), the full visual reward is preserved to reinforce accurate visual grounding.

Formally, the adaptive weight  $\gamma(r_{\text{math}})$  is defined as:

$$\gamma(r_{\text{math}}) = \begin{cases} 1.0, & \text{if } r_{\text{math}} > \tau, \\ \exp(r_{\text{math}} - \tau), & \text{if } r_{\text{math}} \leq \tau. \end{cases} \quad (22)$$

The modulated visual reward and overall reward are then defined as:

$$r_{\text{vis}} = \alpha \cdot \gamma(r_{\text{math}}) \cdot r_{\text{vis}}, \quad (23)$$

$$r_{\text{total}} = w_1 \cdot r_{\text{vis}} + w_2 \cdot r_{\text{math}}, \quad (24)$$

where  $r_{\text{vis}}$  denotes the reconstruction-based visual reward and  $r_{\text{math}}$  the mathematical correctness reward. We use  $\alpha = 1.0$ ,  $w_1 = 0.3$ , and  $w_2 = 0.7$  by default. This design ensures that visual grounding contributes positively only when it leads to correct mathematical outcomes. The exponential decay for incorrect outputs ( $\gamma < 1$ ) provides smooth gradient feedback and prevents noisy rollouts from dominating policy updates.

Compared to directly integrating visual tokens into MLLMs, our neuro-symbolic system provides explicit interpretability and modular supervision but exhibits limited flexibility—particularly in aligning visual symbolic representations with pre-trained reasoning states. We hypothesize that reinforcement learning primarily refines existing reasoning policies rather than inducing new multimodal knowledge, a finding consistent with prior observations [38]. We advocate future research toward neuro-symbolic architectures and adaptive, reasoning-aligned visual supervision as a promising direction for developing grounded multimodal intelligence.

## I.2. Close-up Results on MathVerse

Tab. 7 provides detailed breakdowns of our model’s performance on the MathVerse `test-mini` benchmark, expanding upon the aggregate results presented in the main paper (Sec. 4.2).

## I.3. Qwen2.5-VL-7B Fine-tuning Results

To isolate the contribution of our symbolic encoder, we fine-tune Qwen2.5-VL-7B with the same data and training settings as SymVAE-7B (denoted Qwen2.5+ in Table 8). SymVAE+ outperforms Qwen2.5+ by 15.2% on average, with consistent gains across all categories, confirming that improvements stem from symbolic architecture rather than training data.

## I.4. Close-up Analysis on MSE Evaluation

To verify that clean rendering does not inflate MSE differences, we apply Gaussian blur, average blur, and down-up resizing to 100 diagrams and measure MSE against the clean inputs. The degradation is minimal (1.1–2.7), far below the gap between SymVAE (6.13) and VQ-GAN (11.7): 

GauBlur: 1.1	AvgBlur: 2.7	Down-Up: 2.4	VQ-GAN: 11.7	SymVAE: 6.13
--------------	--------------	--------------	--------------	--------------

. The perceptual (LPIPS) and structural (SSIM) metrics in Tab. 2 further confirm this: SymVAE-3B achieves consistent gains across all three metrics, with a 47.6% MSE reduction, 95.2% LPIPS reduction, and 22.5% SSIM improvement.

Models with stronger structural understanding (SymHPR vs. SymParser) achieve better performance, especially on unseen domains, across both diagram reconstruction and understanding tasks. VQ-GAN reconstructions fail on complex topology and fine-grained details (Fig. 4), while our symbolic decoder achieves a 100% parsing success rate on basic primitives and relationships. Failures occur only when prerequisite conditions for subsequent parsing steps are missing (§D).

## I.5. Rendering Cost

Average runtime (ms) over 100 images at 512×512 resolution on a single A100 GPU for the neural decoder, and on CPU only for rendering engine: [VQ-GAN Decoder: 12.54][Rendering Engine: 8.09]. Rendering engine is 1.55× faster than VQ-GAN decoding.

## J. Limitations

Although our symbolic–visual framework provides clear interpretability and verifiable reasoning, it also introduces several limitations. First, our logic rules are manually designed around planar geometric primitives (points, lines, circles, and polygons). This rule-based design limits flexibility: extending the system to new geometric concepts requires manual modification of the formal grammar and the rendering engine, creating a knowledge-acquisition bottleneck characteristic of classical symbolic AI. In contrast to end-to-end neural approaches that can implicitly learn novel visual patterns through gradient-based adaptation, our symbolic representations cannot autonomously generalize beyond their predefined ontology.

Second, our training data consists exclusively of synthetic planar geometry diagrams with clean and well-structured primitives. Although we observe encouraging cross-modal transfer to 3D object scenes, such generalization remains limited. The system struggles with: (1) hand-drawn diagrams with irregular strokes or occlusions, and (2) real-world images containing cluttered backgrounds and complex topologies. Moreover, the symbolic representation requires precise and complete detection of all primitives. Any missing vertex, line, or relation produces an incomplete logic form that can propagate errors throughout the entire chain, leading to compounding failures.

Finally, for our neuro-symbolic reasoning design, the reasoning performance is still tied to the expressivity of the symbolic representation. When the symbolic abstraction fails to encode a necessary relation, the model cannot recover it through language alone. Thus, improving perception and symbolic completeness remains essential for fully reliable multi-step reasoning. Overall, our work provides modular components for diagram parsing: symbolic vocabulary, reward functions, and rendering engines. We encourage the community to extend these with richer primitives (*e.g.*, paths, strokes, curves) for broader visual domains.

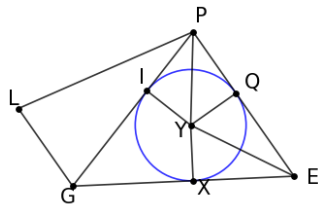
**Future Directions.** Rather than relying on hand-crafted logic rules, future work could explore meta-learning approaches that automatically discover primitive vocabularies from data, similar to neural program synthesis. Learning symbolic vocabularies directly from large-scale multimodal corpora would allow the system to expand beyond fixed rule sets while preserving interpretability. Another promising direction is to integrate neural symbolic parsing with differentiable program induction, enabling the symbolic space to grow adaptively for new domains such as chemistry, physics diagrams, or engineering drafts. In addition, incorporating uncertainty estimation into symbolic extraction could improve robustness and reduce cascading perceptual errors during multi-step reasoning. Finally, neuro-symbolic cycles, where a symbolic encoder and a neural reasoner interact in a bidirectional loop, may yield systems that combine flexible neural pattern recognition with rigorous symbolic verification.

## References

- [1] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, Yu Su, Wenhua Chen, and Graham Neubig. MMMU-Pro: A More Robust Multi-discipline Multimodal Understanding Benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. 6
- [2] Ming Li et al. EEE-Bench: A Comprehensive Multimodal Electrical And Electronics Engineering Benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 6
- [3] Junxian Li et al. ChemVLM: Exploring the Power of Multimodal Large Language Models in Chemistry Area. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 2025. 6

### Synthesis Diagram

Image:

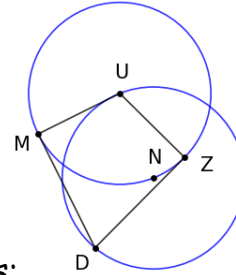


Logic Forms:

```
"Points": ["P", "E", "G", "L", "X", "Q", "I", "Y"],
"Lines": ["PE", "EG", "GL", "LP", "GP", "YX",
          "YQ", "YI", "YP", "YE"],
"Shapes": ["Circle(Y)", "Trapezoid(P, E, G, L)"],
"Properties": [
  "Trapezoid(P, E, G, L) with Parallel(Line(P, E),
  Line(L, G))"
],
"Relations": [
  "PointLiesOnCircle(I, Circle(Y, radius_0_0))",
  "PointLiesOnCircle(Q, Circle(Y, radius_0_0))",
  "PointLiesOnCircle(X, Circle(Y, radius_0_0))",
  "Incircle(Circle(Y, radius_0_0), Triangle(P, G, E))"
],
"Positions": {
  "P": (0.574, 0.344), "E": (0.777, 0.635),
  "G": (0.330, 0.654), "L": (0.221, 0.498),
  "X": (0.574, 0.643), "Q": (0.660, 0.467),
  "I": (0.480, 0.463), "Y": (0.568, 0.531)
}
```

### Synthesis Diagram

Image:



Logic Forms:

```
"Points": ["U", "N", "D", "M", "Z"],
"Lines": ["DM", "DZ", "UM", "UZ"],
"Shapes": ["Circle(N)", "Circle(U)"],
"Relations": [
  "PointLiesOnCircle(U, Circle(N, radius_0_0))",
  "PointLiesOnCircle(D, Circle(N, radius_0_0))",
  "PointLiesOnCircle(N, Circle(U, radius_1_0))",
  "PointLiesOnCircle(Z, Circle(U, radius_1_0))",
  "PointLiesOnCircle(M, Circle(U, radius_1_0))",
  "Tangent(Line(D, M), Circle(U, radius_1_0))",
  "Tangent(Line(D, Z), Circle(U, radius_1_0))"
],
"Positions": {
  "U": (0.465, 0.410),
  "N": (0.533, 0.588),
  "D": (0.412, 0.732),
  "M": (0.295, 0.496),
  "Z": (0.600, 0.545)
}
```

Figure 6. Visualizations of synthetic diagrams paired with their corresponding logic forms.

**Synthesis Diagram**

Image:

Logic Forms:

```

"Points": ["W", "C", "D", "Q", "E", "J", "K", "Y"],
"Lines": ["WC", "CD", "DQ", "QW", "QC", "YE", "YJ",
          "YK", "YW", "YQ"],
"Shapes": ["Circle(Y)", "Square(W, C, D, Q)"],
"Properties": [
  "Square(W, C, D, Q) with Parallel(Line(W, C), Line(Q, D)),
  Parallel(Line(C, D), Line(W, Q)),
  Perpendicular(Line(W, C), Line(C, D)),
  Perpendicular(Line(C, D), Line(D, Q)),
  Equals(LengthOf(Line(W, C)), LengthOf(Line(C, D))),
  Equals(LengthOf(Line(C, D)), LengthOf(Line(D, Q))),
  Equals(LengthOf(Line(D, Q)), LengthOf(Line(Q, W)))"];
],
"Relations": [
  "PointLiesOnCircle(E, Circle(Y, radius_0_0))",
  "PointLiesOnCircle(J, Circle(Y, radius_0_0))",
  "PointLiesOnCircle(K, Circle(Y, radius_0_0))",
  "Incircle(Circle(Y, radius_0_0), Triangle(W, C, Q))"
],
"Positions": {
  "W": (0.738, 0.777), "C": (0.221, 0.738),
  "D": (0.260, 0.221), "Q": (0.777, 0.260),
  "E": (0.498, 0.500), "J": (0.750, 0.625),
  "K": (0.586, 0.766), "Y": (0.598, 0.613)
}

```

**Synthesis Diagram**

Image:

Logic Forms:

```

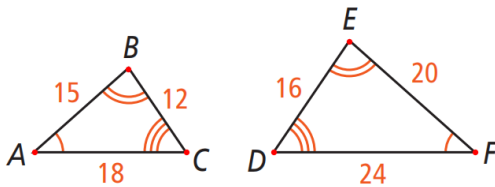
"Points": ["B", "G", "T", "M", "J", "Z", "N"],
"Lines": ["BG", "GT", "TM", "MB", "GM", "GJ", "BT", "BZ",
          "BN"],
"Shapes": ["IsoscelesTrapezoid(B, G, T, M)"],
"Properties": [
  "IsoscelesTrapezoid(B, G, T, M) with Parallel(Line(B, G),
  Line(M, T)), Equals(LengthOf(Line(G, T)), LengthOf(Line(B,
  M)))"
],
"Relations": [
  "AngleBisector(Line(G, J), Angle(T, G, M))",
  "AngleTrisection(Angle(T, B, M), Angle(Z, B, T), Angle(Z, B,
  N), Angle(N, B, M))"
],
"Positions": {
  "B": (0.777, 0.566),
  "G": (0.736, 0.285),
  "T": (0.221, 0.273),
  "M": (0.287, 0.725),
  "J": (0.459, 0.395),
  "Z": (0.246, 0.449),
  "N": (0.268, 0.590)
}

```

Figure 7. Visualizations of synthetic diagrams paired with their corresponding logic forms.

### PGDP Extension

Image:

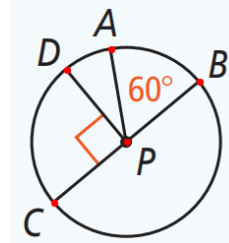


Logic Forms:

```
"Points": ["E", "A", "F", "B", "D", "C"],
"Lines": ["AB", "BC", "ED", "EF", "DF", "AC"],
"Relations": [
  "Equals(LengthOf(Line(C, A)), 18)",
  "Equals(LengthOf(Line(F, D)), 24)",
  "Equals(LengthOf(Line(A, B)), 15)",
  "Equals(LengthOf(Line(E, F)), 20)",
  "Equals(LengthOf(Line(D, E)), 16)",
  "Equals(LengthOf(Line(B, C)), 12)",
  "Equals(MeasureOf(Angle(B, A, C)),
    MeasureOf(Angle(E, F, D)))",
  "Equals(MeasureOf(Angle(D, E, F)),
    MeasureOf(Angle(A, B, C)))",
  "Equals(MeasureOf(Angle(B, C, A)),
    MeasureOf(Angle(E, D, F)))"
],
"Positions": {
  "E": (0.686, 0.225), "A": (0.077, 0.789),
  "F": (0.928, 0.789), "B": (0.259, 0.364),
  "D": (0.540, 0.789), "C": (0.371, 0.789)
}
```

### PGDP Extension

Image:



Logic Forms:

```
"Points": ["A", "C", "D", "B", "P"],
"Lines": ["DP", "CP", "CB", "PB", "AP"],
"Shapes": ["Circle(P)"],
"Relations": [
  "PointLiesOnLine(P, Line(C, B))",
  "PointLiesOnCircle(A, Circle(P, radius_0_0))",
  "PointLiesOnCircle(D, Circle(P, radius_0_0))",
  "PointLiesOnCircle(B, Circle(P, radius_0_0))",
  "PointLiesOnCircle(C, Circle(P, radius_0_0))",
  "Perpendicular(Line(D, P), Line(C, P))",
  "Equals(MeasureOf(Angle(B, P, A)), 60)"
],
"Positions": {
  "A": (0.440, 0.236),
  "C": (0.253, 0.790),
  "D": (0.295, 0.311),
  "B": (0.731, 0.354),
  "P": (0.492, 0.571)
}
```

Figure 8. PGDP extension cases: original PGDP diagram alongside its enriched logic-form representation.

**PGDP Extension**

**Image:**

**Logic Forms:**

```

"Points": ["A", "B", "C", "D", "E", "F"],
"Lines": ["AB", "AC", "BC", "FE", "FC", "EC", "AD",
          "AF", "DF", "DB", "BE", "DE"],
"Relations": [
  "PointLiesOnLine(B, Line(A, C))",
  "PointLiesOnLine(E, Line(F, C))",
  "PointLiesOnLine(D, Line(A, F))",
  "Equals(LengthOf(Line(A, D)), LengthOf(Line(D, F)))",
  "Equals(LengthOf(Line(F, E)), LengthOf(Line(E, C)))",
  "Equals(LengthOf(Line(B, C)), LengthOf(Line(A, B)))",
  "Equals(LengthOf(Line(E, D)), x)",
  "Equals(LengthOf(Line(D, B)), y)",
  "Equals(LengthOf(Line(A, C)), 3x-6)",
  "Equals(LengthOf(Line(C, F)), 2x+1)"
],
"Positions": {
  "A": (0.074, 0.231),
  "B": (0.489, 0.231),
  "C": (0.905, 0.231),
  "D": (0.074, 0.511),
  "E": (0.490, 0.511),
  "F": (0.074, 0.791)
}

```

**PGDP Extension**

**Image:**

**Logic Forms:**

```

"Points": ["A", "B", "C", "D", "E", "F", "G", "H"],
"Lines": ["DB", "DA", "BA", "FG", "FH", "GH", "BE",
          "EG", "EC"],
"Shapes": ["Circle(E)"],
"Relations": [
  "PointLiesOnLine(B, Line(D, A))",
  "PointLiesOnLine(G, Line(F, H))",
  "PointLiesOnCircle(A, Circle(E, radius_0_0))",
  "PointLiesOnCircle(C, Circle(E, radius_0_0))",
  "PointLiesOnCircle(D, Circle(E, radius_0_0))",
  "PointLiesOnCircle(F, Circle(E, radius_0_0))",
  "PointLiesOnCircle(H, Circle(E, radius_0_0))",
  "Perpendicular(Line(D, B), Line(E, B))",
  "Perpendicular(Line(F, G), Line(E, G))",
  "Equals(LengthOf(Line(G, H)), LengthOf(Line(F, G)))",
  "Equals(LengthOf(Line(D, A)), 9)",
  "Equals(LengthOf(Line(F, G)), x)",
  "Equals(LengthOf(Line(B, E)), 5)",
  "Equals(LengthOf(Line(E, C)), y)",
  "Equals(LengthOf(Line(E, G)), 5)"
],
"Positions": {
  "A": (0.681, 0.122), "B": (0.397, 0.242),
  "C": (0.838, 0.275), "D": (0.113, 0.367),
  "E": (0.493, 0.479), "F": (0.162, 0.692),
  "G": (0.465, 0.731), "H": (0.770, 0.766)
}

```

Figure 9. PGDP extension cases: original PGDP diagram alongside its enriched logic-form representation.

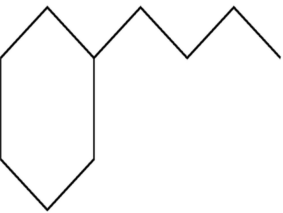
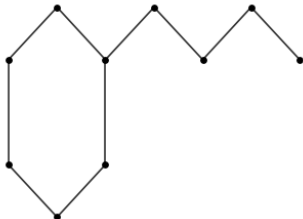
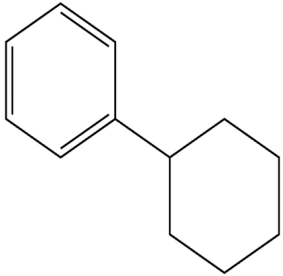
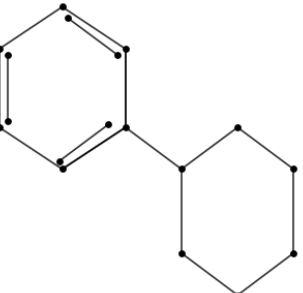
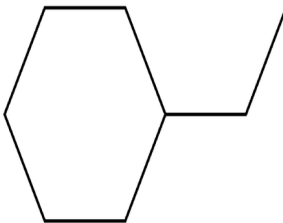
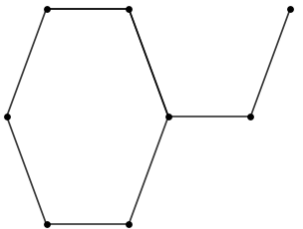
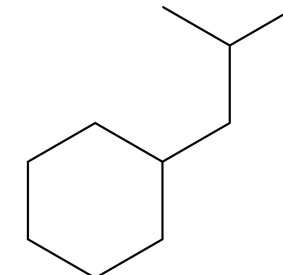
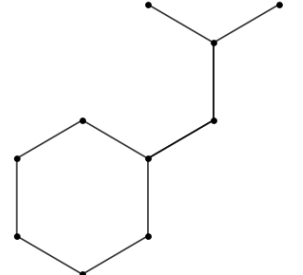
GT	Logic Form	Output
	<pre> "Points": ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J"], "Lines": ["AB", "BC", "CD", "DE", "EF", "FA", "BG", "GH", "HI", "IJ"], "Positions": { "A": (0.167, 0.000), "B": (0.333, 0.180), "C": (0.333, 0.540), "D": (0.167, 0.720), "E": (0.000, 0.540), "F": (0.000, 0.180), "G": (0.500, 0.000), "H": (0.667, 0.180), "I": (0.833, 0.000), "J": (1.000, 0.180)} </pre>	
	<pre> "Points": ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R"], "Lines": ["AB", "BC", "CD", "DE", "EF", "FG", "GB", "AH", "HI", "JK", "IL", "LM", "NO", "MP", "QR", "PA"], "Positions": { "A": (0.417, 0.417), "B": (0.611, 0.556), "C": (0.806, 0.417), "D": (1.000, 0.556), "E": (1.000, 0.833), "F": (0.806, 0.972), "G": (0.611, 0.833), "H": (0.417, 0.139), "I": (0.222, 0.000), "J": (0.367, 0.161), "K": (0.200, 0.050), "L": (0.000, 0.139), "M": (0.000, 0.417), "N": (0.022, 0.394), "O": (0.022, 0.161), "P": (0.222, 0.556), "Q": (0.050, 0.394), "R": (0.217, 0.506)} </pre>	
	<pre> "Points": ["A", "B", "C", "D", "E", "F", "G", "H"], "Lines": ["AB", "BC", "CD", "DE", "AF", "FG", "GH", "HC", "CB", "BA"], "Positions": { "A": (0.140, 0.000), "B": (0.430, 0.000), "C": (0.570, 0.381), "D": (0.860, 0.381), "E": (1.000, 0.000), "F": (0.000, 0.381), "G": (0.140, 0.762), "H": (0.430, 0.762)} </pre>	
	<pre> "Points": ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J"], "Lines": ["AB", "BC", "CD", "DE", "EF", "FG", "GH", "HI", "ID", "DC", "CB", "BJ"], "Positions": { "A": (0.971, 0.000), "B": (0.728, 0.142), "C": (0.728, 0.429), "D": (0.485, 0.571), "E": (0.485, 0.858), "F": (0.243, 1.000), "G": (0.000, 0.858), "H": (0.000, 0.571), "I": (0.243, 0.429), "J": (0.485, 0.000)} </pre>	

Figure 10. Reconstruction results of SymVAE on molecular structure diagrams. Each triplet shows the ground-truth input, the predicted logic-form primitives, and the corresponding reconstructed diagram.

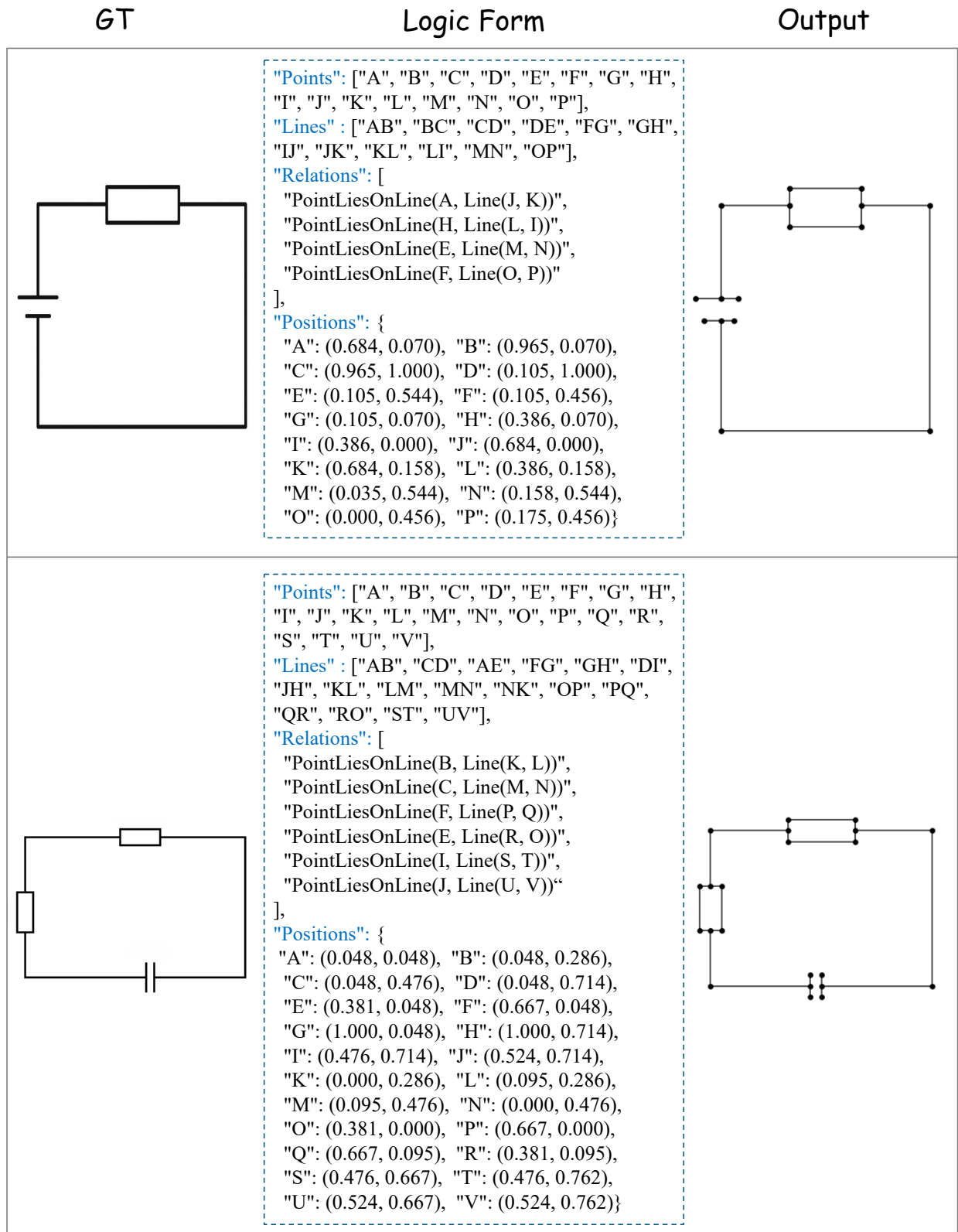


Figure 11. Reconstruction results of SymVAE on electrical circuit diagrams. Each triplet shows the ground-truth input, the predicted logic-form primitives, and the corresponding reconstructed diagram.

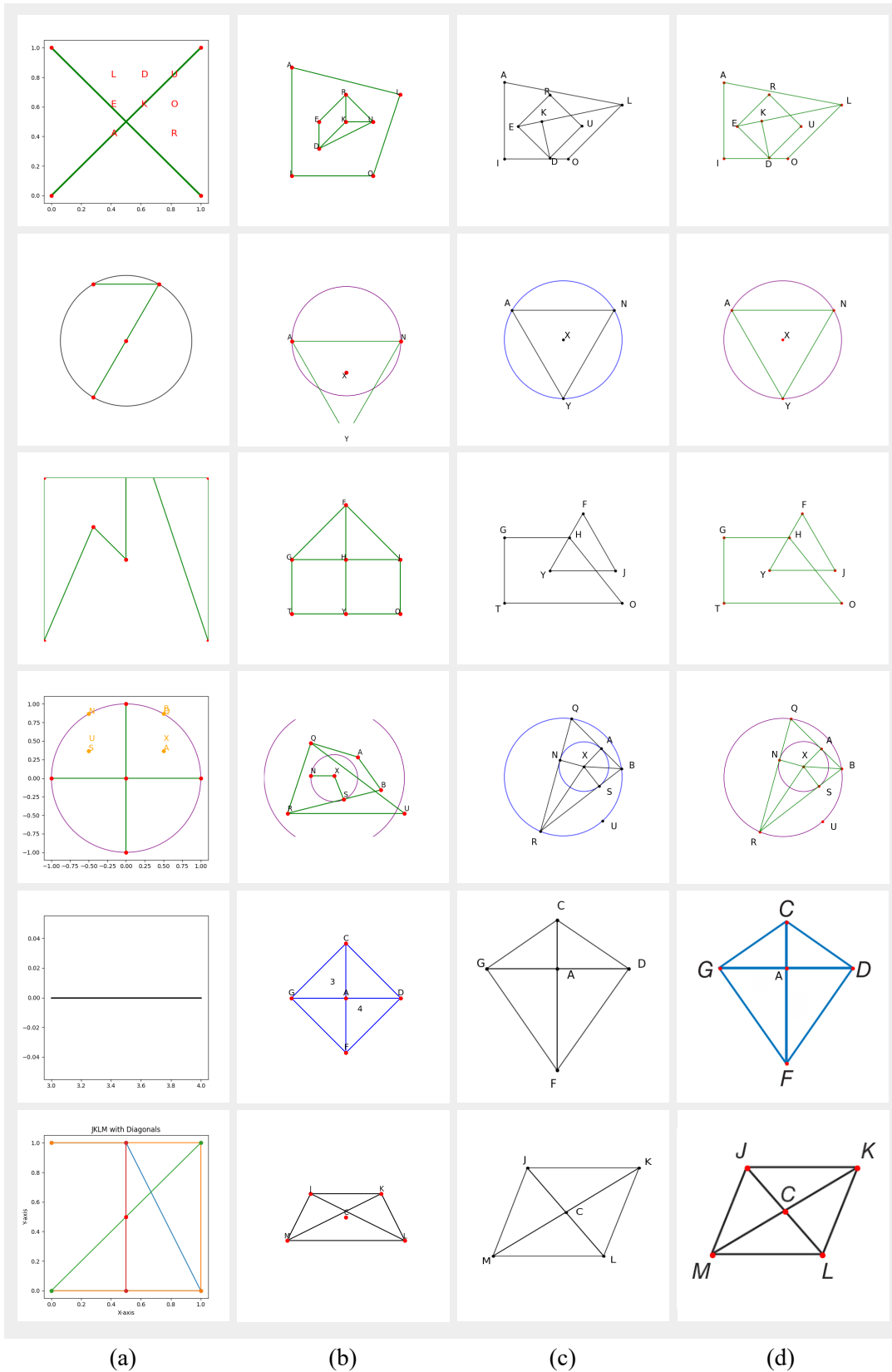
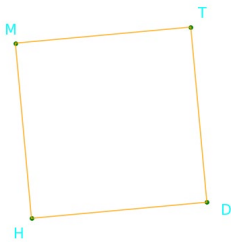


Figure 12. Qualitative visualizations on geomatic diagrams reconstruction w.r.t. Qwen2.5-VL-7B, GPT-4o, and SymVAE-7B. The final column shows the ground-truth inputs.





**Question:**  
What is the shape of object DHMT in the image?

- Choices:**  
A: scalene triangle  
B: circle  
C: equilateral triangle  
**D: square**

### SymVAE-7B

Step 1: **Identify the vertices.** The shape has four vertices labeled as D, H, M, and T

Step 2: **Determine the edges.** Four lines are formed: DH, HM, MT, and TD, connecting the vertices to form a closed quadrilateral.

Step 3: **Verify shape properties.** All four sides DH, HM, MT, and TD are of equal length, and all angles at D, H, M, and T are right angles (90 degrees).

Step 4: **Conclude the shape.** Based on four equal sides and four right angles, the shape is a square. The final answer is D.

### Qwen2.5-VL-7B

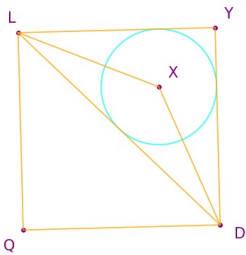
Step 1: Identify the shape: The figure appears to be a quadrilateral with four corners.

Step 2: Examine the sides: The side DH appears to be equal in length to the side MT. The side HT appears to be equal in length to the side MD. All four sides appear to be of equal length.

Step 3: Check the angles: The angles at each vertex (D, H, M, T) appear to be right angles (90 degrees).

Step 4: Conclusion based on properties: Since all four sides are equal and all angles are right angles, the figure is a square. Therefore, the answer is D.

(a)



**Question:**  
Does the image show the relationship between circle X and triangle LYD as incircle?

- Choices:**  
**A: Yes**  
B: No

### SymVAE-7B

Step 1: **Identify the points.** The diagram contains five points: L, Y, D, Q, and X.

Step 2: **Identify the line segments.** The points are connected by seven edges: LY, YD, DQ, QL, DL, XD, and XL.

Step 3: **Recognize the shapes.** Two shapes are identified: a square with parallel opposite sides, perpendicular adjacent sides, and equal side lengths; and a circle centered at point X.

Step 4: **Analyze geometric relations.** The circle at X lies entirely within the triangle formed by D, L, and Y. The circle is tangent to all three sides of this triangle, satisfying the definition of an incircle. So, the answer is A.

### Qwen2.5-VL-7B

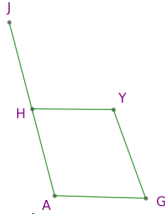
Step 1: Identify the triangle in question. The triangle in question is triangle LYD.

Step 2: Check if circle X is tangent to all three sides of triangle LYD. Circle X appears to be tangent to side LY at point X. Circle X appears to be tangent to side DY at point X. Circle X does not appear to be tangent to side LD at any point other than the center of the circle.

Step 3: Determine if the circle is an incircle. For circle X to be the incircle of triangle LYD, it must be tangent to all three sides of the triangle. Since it is not tangent to side LD, it cannot be the incircle of triangle LYD. Therefore, the answer is B.

(b)

Figure 14. Emergent cross-lingual translation on MathGlance evaluation samples. The translated chain-of-thoughts are not simple mirrors of the priors embedded in the base model (Qwen2.5-VL-7B); instead, they follow the structured reasoning induced by our logic-form representation.



**Question:**

Can the relationship among point J, point A and point H in the image be described as point reflection?

**Choices:**

A: Yes

B: No

**SymVAE-7B**

Step 1: **Identify the points.** Five vertices are detected in the diagram: J, A, H, Y, and G.

Step 2: **Identify the edges.** The points are connected by several edges: JH connects J to H, HA connects H to A, HY connects H to Y, AG connects A to G, and YG connects Y to G, forming a quadrilateral AHYG with an additional point J above.

Step 3: **Classify the shapes.** The diagram contains a quadrilateral formed by A, H, Y, G with parallel sides HY and AG, indicating a trapezoid structure.

Step 4: **Verify the point reflection condition.** Examining points J, A, and H: point H serves as the center of reflection. The distances from H to J and from H to A are equal, and J, H, A are collinear. This satisfies the point reflection relationship where A is the reflection of J through H. So, the answer is A.

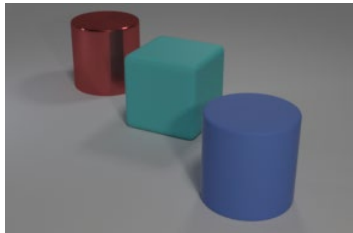
**Qwen2.5-VL-7B**

Step 1: Identify the given points and lines. Points: J, A, H, Y, G. Lines: JH, AH, HY, AG.

Step 2: Check for symmetry or reflection. Point reflection implies that there is a line of symmetry such that each point has a corresponding point that is equidistant from this line but on the opposite side. In the image, point J appears to be above point H, and point A is below point H. There is no clear line of symmetry that would make J and A symmetric with respect to H.

Step 3: Analyze the positions of the points. Point J is not directly related to point A through a simple reflection over any line passing through H. The distances and angles do not suggest a reflection pattern. Therefore, the final answer is B.

Figure 15. Emergent cross-lingual translation on MathGlance evaluation samples. The translated chain-of-thoughts are not simple mirrors of the priors embedded in the base model (Qwen2.5-VL-7B); instead, they follow the structured reasoning induced by our logic-form representation.



**Question:**

Is it correct that the large red metal cylinder is described as being in the 'top right' relative to the large cyan rubber cube in the image?

**Choices:**

A: Yes

B: No

**SymVAE-7B**

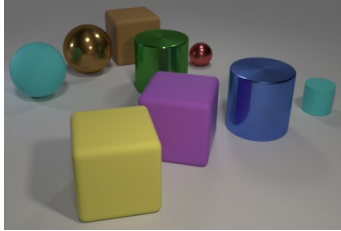
Step 1: **Point instances:** [cylinder\_red, cube\_cyan, cylinder\_blue]

Step 2: **Shape instances:** cylinder(red), cube(cyan), cylinder(blue)

Step 3: **Position of shape centers:** cylinder\_red=(0.342, 0.221), cube\_cyan=(0.457, 0.262), cylinder\_blue=(0.598, 0.367)

Step 4: **Compare the positions** of the cylinder\_red and the cube\_cyan. The cylinder\_red is to the top left of the cube\_cyan. So, the answer is B.

Figure 16. Emergent cross-modal translation on MathGlance evaluation samples. The model can translate planar geometric vertex points into 3D object attributes.



**Question:**

Are there 6 shapes in the picture

**Choices:**

A: Yes

B: No

**SymVAE-7B**

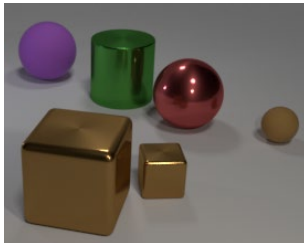
Step 1: **Point instances:** [cyan\_1, gold, red, green, blue, yellow, purple, brown, cyan\_2]

Step 2: **Shape instances:** sphere(cyan\_1), sphere(gold), sphere(red), cylinder(green), cylinder(blue), cube(yellow), cube(purple), cube(brown), cylinder(cyan\_2)

Step 3: **Point positions:** cyan\_1=(0.221, 0.258), gold=(0.293, 0.221), red=(0.598, 0.221), green=(0.479, 0.234), blue=(0.736, 0.307), yellow=(0.359, 0.498), purple=(0.551, 0.379), brown=(0.391, 0.221), cyan\_2=(0.891, 0.314)

Step 4: **Count the total number of objects.** There are nine objects in total. So, the answer is B.

(a)



**Question:**

Is it correct that the large brown metal cube is described as being in the 'bottom right' relative to the small brown metal cube in the image?

**Choices:**

A: Yes

B: No

**SymVAE-7B**

Step 1: **Point instances:** [purple, red, green, large\_brown, small\_brown\_1, small\_brown\_2]

Step 2: **Shape instances:** sphere(purple), sphere(red), cylinder(green), cube(large\_brown), cube(small\_brown\_1), sphere(small\_brown\_2)

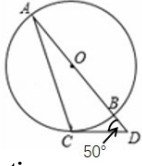
Step 3: **Material instances:** rubber(sphere(purple)), metal(sphere(red)), metal(cylinder(green)), metal(cube(large\_brown)), metal(cube(small\_brown\_1)), rubber(sphere(small\_brown\_2))

Step 4: **Position of objects:** sphere(purple)=(0.367, 0.221), sphere(red)=(0.648, 0.262), cylinder(green)=(0.500, 0.221), cube(large\_brown)=(0.391, 0.389), cube(small\_brown\_1)=(0.586, 0.391), sphere(small\_brown\_2)=(0.837, 0.289)

Step 5: **Compare the positions** of the cube(large\_brown) and the cube(small\_brown\_1). The cube(large\_brown) is to the bottom left of the cube(small\_brown\_1). So, the answer is B.

(b)

Figure 17. Emergent cross-modal translation on MathGlance evaluation samples. The model can translate planar geometric vertex points into 3D object attributes.



**Question:**  
The degree of angle A is?

**Answer:**  
20°

**Ours**

- ✓ Since CD is a tangent to circle O, we have OC perpendicular to CD, so  $\angle OCD = 90^\circ$
- ✓ Given that  $\angle D = 50^\circ$ , we can find  $\angle COD$  using the fact that the sum of angles in a triangle is  $180^\circ$ . Therefore,  $\angle COD = 180^\circ - \angle OCD - \angle D = 180^\circ - 90^\circ - 50^\circ = 40^\circ$
- ✓ Since  $OA = OC$ , we know that  $\angle A = \angle OCA$
- ✓ Also,  $\angle COD = \angle A + \angle OCA$ . Thus,  $\angle A = \frac{1}{2} \times \angle COD = \frac{1}{2} \times 40^\circ = 20^\circ$ .

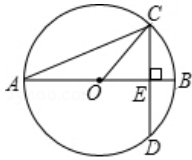
**ChatGPT-4o**

- ✗ Notice that  $\angle CBD = 50^\circ$  is an exterior angle to triangle  $\triangle ABC$
- ✓ Points A, B, C lie on the circle, so  $\angle ABC$  is an inscribed angle.
- ✗ Since  $\angle CBD = 50^\circ$  is the exterior angle to triangle ABC, it equals the sum of the two opposite interior angles, i.e.:  $\angle CBD = \angle CAB + \angle ACB$
- ✗ But more importantly, in this specific circle configuration, if AB and CD intersect at point B, and you are given that  $\angle CBD = 50^\circ$ , and if AB is a chord and CD is a tangent to the circle at point B, then:  $\angle CAB = \angle CBD = 50^\circ$

**Qwen2.5-VL-7B**

- ✗ Identify the given information:  $\angle CBD = 50^\circ$ , O is the center of the circle. AB is a diameter of the circle, so  $\angle ACB$  is a right angle.
- ✗ Use the Inscribed Angle Theorem: An inscribed angle is half the measure of the central angle that subtends the same arc. Here,  $\angle BCD$  is an inscribed angle subtending arc BD. The corresponding central angle  $\angle BOD$  will be twice the measure of  $\angle BCD$ .  $\angle BOD = 2 \times \angle BCD = 2 \times 50^\circ = 100^\circ$
- ✗  $\angle ABC$  is an inscribed angle subtending arc AD. The corresponding central angle  $\angle AOD$  will be twice the measure of  $\angle ABC$ :  $\angle AOD = 2 \times \angle ABC$ . From the previous calculation,  $\angle BOD = 100^\circ$ . Therefore,  $\angle AOD = 100^\circ$ . So  $\angle ABC = \frac{\angle AOD}{2} = 50^\circ$ . Then  $\angle A = 180^\circ - \angle ACB - \angle ABC = 90^\circ - 50^\circ = 40^\circ$

(a)



**Question:**  
As shown in the figure, angle  $\angle CAO = 22.5^\circ$ ,  $OC = 6.0$ , then the length of CD is ()?

**Answer:**  
 $6\sqrt{2}$

**Ours**

- ✓ From the given information, we have  $\angle COE = 2 \angle CAO = 45^\circ$
- ✓ Since CE is perpendicular to AB, we have  $\angle OCE = 90^\circ - \angle COE = 45^\circ$
- ✓ Therefore, triangle COE is an isosceles right triangle.
- ✓ Thus,  $OE = CE = \frac{\sqrt{2}}{2} \times OC = 3\sqrt{2}$ . Therefore,  $CD = 2CE = 6\sqrt{2}$ .

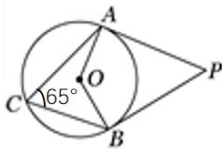
**ChatGPT-4o**

- ✓ In triangle COE,  $\angle CAO = 22.5^\circ$  is an angle between line segment CA and diameter AB. Triangle COA is within a circle with diameter AB, so AO is the radius.
- ✗ Use trigonometry in triangle COA,  $\sin(\angle CAO) = \frac{OE}{OC}$ . So,  $OE = OC \times \sin(\angle CAO) = 6 \times \sin(22.5^\circ) \approx 6 \times 0.3827 \approx 2.296$
- ✗ In triangle CED (right triangle), the vertical leg from E to D is the same as OE. So,  $CD = OE \approx 2.296$

**Qwen2.5-VL-7B**

- ✓ Identify Key Properties: Since AB is a diameter of the circle,  $\angle ACB = 90^\circ$  (by the Inscribed Angle Theorem). OC is a radius of the circle, so  $OC = OA = OB = R$ , where R is the radius. Given  $OC = 6.0$ , the radius  $R = 6.0$ .
- ✗ Place the center O at the origin (0,0) of a coordinate system. Let  $A = (-6, 0)$  and  $C = (6, 0)$  since  $OA = OC = 6$ . Since  $\angle CAO = 22.5^\circ$ ,  $C = (6 \cos(22.5^\circ), 6 \sin(22.5^\circ)) = (4.87, 2.42)$ .
- ✗ Since CD is the vertical distance from C to the x-axis, which is simply the y-coordinate of C:  $CD = 6 \sin(22.5^\circ) = 6 \cdot \frac{\sqrt{2}-\sqrt{2}}{2} = 3\sqrt{2-\sqrt{2}} \approx 3$

(b)



**Question:**  
Find angle P

**Answer:**  
50°

**Ours**

- ✓ Given that PA and PB are tangents to circle O at points A and B respectively, we can conclude that OA is perpendicular to AP and OB is perpendicular to BP. Therefore,  $\angle OAP = \angle OBP = 90^\circ$
- ✓ Since  $\angle ACB = 65^\circ$ , we have  $\angle AOB = 2\angle ACB = 130^\circ$ .
- ✓ Using the fact that the sum of angles in a quadrilateral is  $360^\circ$ , we can find  $\angle APB$ :  $\angle APB = 360^\circ - \angle OAP - \angle OBP - \angle AOB = 360^\circ - 90^\circ - 90^\circ - 130^\circ = 50^\circ$
- ✓ Therefore, the measure of  $\angle P$  is  $50^\circ$ .

**ChatGPT-4o**

- ✓ We are given a circle with center O, and triangle ABC inscribed in it. AP and BP are extended such that they intersect at point P outside the circle.
- ✗ In circle geometry, the angle formed by two chords intersecting outside the circle (like  $\angle APB$ ) equals the difference between the arcs they intercept divided by 2:  $\angle APB = \frac{1}{2}(\widehat{AB} - \widehat{CD})$
- ✗ But in this case, since APB subtends the arc AB, and triangle ABC is inscribed in the circle:  $\angle APB = 180^\circ - \angle ACB$

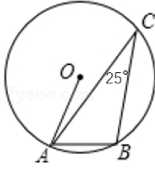
**Qwen2.5-VL-7B**

- ✗ Given:  $\angle ACB = 65^\circ$ . Since O is the center of the circle, OA, OB, and OC are radii of the circle, making triangle OAB an equilateral triangle (since all sides are equal).
- ✓ The central angle subtended by arc AB at the center O is twice the inscribed angle subtended by the same arc at any point on the circumference. Therefore:  $\angle AOB = 2 \times \angle ACB = 2 \times 65^\circ = 130^\circ$
- ✗ Now consider triangle OAP. Since OA and OP are both radii of the circle, they form two equal sides of this triangle. Thus, triangle OAP is isosceles with  $OA = OP$ .
- ✗  $\angle OAP + \angle OPA + \angle AOP = 180^\circ$ , then  $2 \angle OAP + 130^\circ = 180^\circ$ . So  $\angle OAP = 25^\circ$ . Therefore,  $\angle APB = 2 \angle OPA = 50^\circ$ .

(c)

Figure 18. Response comparisons between SymVAE+CoTs-7B, GPT-4o, and Qwen-2.5-VL-7B in MathVerse.

As shown in the figure, then the degree of angle BAO is ( )  
 Choices: A:50° B:55° C:60° D:65°



**Answer:**  
D

Ours	ChatGPT-4o	Qwen2.5-VL-7B
<ul style="list-style-type: none"> <li>✓ Given that <math>\angle ACB = 25^\circ</math>, we can use the property of angles subtended by the same arc to find <math>\angle AOB</math>.</li> <li>✓ Step 1: Since <math>\angle ACB = 25^\circ</math>, <math>\angle AOB = 2 \angle ACB = 2 \times 25^\circ = 50^\circ</math>.</li> <li>✓ Step 2: In triangle OAB, <math>OA = OB</math> because they are radii of the circle. Therefore, <math>\angle OAB = \angle OBA</math>.</li> <li>✓ Step 3: The sum of angles in a triangle is <math>180^\circ</math>. So, <math>\angle OAB + \angle OBA + \angle AOB = 180^\circ</math>.</li> <li>✓ Step 4: Substituting the known values, we get <math>2\angle OAB + 50^\circ = 180^\circ</math>.</li> <li>✓ Step 5: Solving for <math>\angle BAO</math>, we get <math>\angle BAO = (180^\circ - 50^\circ) / 2 = 65^\circ</math>. Therefore, the degree of <math>\angle BAO</math> is <math>65^\circ</math>. The answer is D.</li> </ul>	<ul style="list-style-type: none"> <li>✗ We are given a circle with center O, and triangle ABC inscribed in the circle. The <math>\angle BCO = 25^\circ</math>. We are to find the <math>\angle BAO</math>.</li> <li>✗ Since O is the center of the circle and C lies on the circle, segments OC and OB are radii of the circle. Therefore, triangle BOC is isosceles with <math>OB = OC</math>. In triangle BOC, two sides are equal, and the angle between them is at vertex C.</li> <li>✗ In triangle BOC, <math>\angle BCO = 25^\circ</math>. Let <math>\angle OBC = 25^\circ</math>, since triangle BOC is isosceles. Then, <math>\angle BCO = 180^\circ - 25^\circ - 25^\circ = 130^\circ</math>.</li> <li>✓ Now consider triangle BAO. Since O is the center, <math>OA=OB</math>, and triangle ABO is isosceles with equal angles at A and B.</li> <li>✗ The <math>\angle AOB</math> is the same <math>130^\circ</math> obtained from the central angle subtended by arc AB. Let <math>\angle BAO = \angle ABO = x</math>. Using the triangle angle sum, we have <math>x + x + 130^\circ = 180^\circ</math>, so <math>2x = 50^\circ</math>, and therefore <math>x = 25^\circ</math>. Hence <math>\angle BAO = 25^\circ</math>.</li> </ul>	<ul style="list-style-type: none"> <li>✗ Given: <math>\angle COB</math> is <math>25^\circ</math>. O is the center of the circle. AOB is a straight line (<math>180^\circ</math>).</li> <li>✓ Since O is the center of the circle, triangle OAB is an isosceles triangle with <math>OA = OB</math> (radii of the circle). Therefore, angles OAB and OBA are equal. Let's denote these angles as x.</li> <li>✗ The sum of the angles in triangle OAB is <math>180^\circ</math>. <math>\angle OAB + \angle OBA + \angle AOB = 180^\circ</math>. Then <math>x + x + 25^\circ = 180^\circ</math>. <math>x = 77.5^\circ</math>. So, angle BAO is <math>77.5^\circ</math>. However, this does not match any of the provided choices.</li> </ul>

Figure 19. Response comparisons between SymVAE+CoTs-7B, GPT-4o, and Qwen-2.5-VL-7B in MathVerse.

Table 7. Performance comparison on MathVerse testmini set. We evaluate models across different visual dependency levels: *Text Dominant* (text-heavy problems), *Text Lite* (moderate text), *Vision Intensive* (requires visual understanding), *Vision Dominant* (primarily visual), and *Vision Only* (purely visual reasoning). Results show accuracy (%) for each category.

Model	Base LLM	All	Text Dom.	Text Lite	Vision Int.	Vision Dom.	Vision Only
<i>Baselines</i>							
Random Chance	-	12.4	12.4	12.4	12.4	12.4	12.4
Human	-	67.7	71.2	70.9	61.4	68.3	66.7
<i>Large Language Models</i>							
ChatGPT	-	26.1	33.3	18.9	-	-	-
GPT-4	-	33.6	46.5	46.5	-	-	-
<i>Closed-Source Multimodal LLMs</i>							
Qwen-VL-Plus [1]	-	11.8	15.7	11.1	9.0	13.0	10.0
Qwen-VL-Max [1]	-	25.3	30.7	26.1	24.1	24.1	21.4
<i>Open-Source Multimodal LLMs</i>							
LLaMA-Adapter V2 [7]	LLaMA-7B	5.7	6.2	5.9	6.1	4.2	6.1
ImageBind-LLM [9]	LLaMA-7B	9.2	11.4	11.3	8.9	11.2	3.4
mPLUG-Owl2 [37]	LLaMA-7B	5.9	6.6	6.3	6.3	5.6	4.9
SPHINX-Plus [17]	LLaMA2-13B	12.2	13.9	11.6	11.6	13.5	10.4
SPHINX-MoE [18]	Mixtral-8x7B	15.0	22.2	16.4	14.8	12.6	9.1
G-LLaVA [6]	LLaMA2-7B	16.6	20.9	20.7	17.2	14.6	9.4
LLaVA-1.5 [19]	Vicuna-13B	7.6	8.8	7.6	7.4	7.4	6.9
ShareGPT4V [4]	Vicuna-13B	13.1	16.2	16.2	15.5	13.8	3.7
Math-LLaVA [28]	Vicuna-13B	19.0	21.2	19.8	20.2	17.6	16.4
LLaVA-NeXT [20]	LLaMA3-8B	19.3	24.9	20.9	20.8	16.1	13.8
Qwen2.5-VL-7B [1]	Qwen2.5-7B	49.2	58.4	52.7	33.2	30.6	21.1
<i>Our Models</i>							
<b>SymVAE+CoTs-7B</b>	Qwen2.5-7B	<b>51.8</b>	<b>61.3</b>	<b>55.2</b>	<b>35.2</b>	<b>33.4</b>	<b>24.9</b>

Table 8. Comparison between SymVAE-7B and the fine-tuned Qwen2.5-VL-7B (Qwen2.5+) using the same data and training settings.

Method	Avg.	Plane Geo.	Solid Geo.	Graphs	Layout	Type	Avg.	High-Level
Qwen2.5+	57.4	59.3	52.8	60.2	90.0	70.8	71.5	72.4
<b>SymVAE+</b>	<b>72.6</b>	<b>77.9</b>	<b>67.6</b>	<b>72.2</b>	<b>96.3</b>	<b>76.8</b>	<b>79.6</b>	<b>77.3</b>