PARSING AND BINDING: CHART UNDERSTANDING AS QUERYABLE INSTANCE GRAPH RECOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Chart understanding is commonly scored by answers or recovered tables. These metrics can look strong even when marks are merged or legend bindings are wrong, because they are blind to the underlying instance structure. We propose a structural view: recovery of a queryable instance graph, in which each claim remains grounded in masks and "what is seen" is scored separately from "how instances are bound." We operationalize this in ChartLogic, an ontology-guided Parse-and-Connect model over shared instance anchors. Parse lifts pixels to instances with category, grouping, and identity; Connect binds them into relations among marks, axes, legends, and data structures. A shared generation source yields aligned images, hierarchical label maps, and ontology-typed ABox graphs, providing supervision for both stages. At evaluation, task-conditioned SPARQL templates are run on a graph built only from predicted instances, with a ground-truth skeleton as a diagnostic. Experiments on synthetic charts and on verify-120 (ChartQA-Redraw-120)—a human-calibrated, re-rendered ChartQA validation subset—yield three observations. First, ontology-aligned hierarchical supervision recovers the structured instance identity that enables downstream binding. Second, GT-skeleton scores remain substantially higher than open recovery even when many query subjects are missing from the predicted graph; open recovery failures decompose into unmatched instances and wrong bindings among matched ones, so Parse and Connect must be attributed separately. Third, under transfer, category accuracy stays high while geometric recall falls first, a category-vs-geometry gap that also appears on verify-120. Together, these observations suggest that content-level and GT-skeleton scores can mask an instance-topology failure that only open recovery reveals.

1 INTRODUCTION

Charts are designed artifacts: their meaning comes from recurring roles—marks encode values, axes provide references, legends explain series. Automatically recovering this structure is a prerequisite for using charts as evidence in scientific papers, financial reports, and business dashboards. End-to-end chart models, however, typically map an image directly to a QA answer, a caption, or a recovered table (Masry et al., 2023; Xu et al., 2025; Lee et al., 2023; Liu et al., 2023b;a; Chen et al., 2024; Kondic et al., 2026). The visual evidence behind each claim remains hidden. If the model misread a legend or swapped an axis, the answer is wrong—but there is no way to check. In settings where a chart is one link in an evidence chain, this unauditability is a fatal defect, so we recover an intermediate structure linking every semantic claim to its supporting pixels.

Two existing routes fall short of this requirement. End-to-end answer mapping (Masry et al., 2023; Liu et al., 2023b; Xu et al., 2025) is task-optimized but opaque: errors cannot be traced to a misread mark or a swapped axis. Separate perception-and-parsing pipelines (Siegel et al., 2016; Luo et al., 2021; Lal et al., 2023) expose intermediate objects, yet cannot be trained jointly, and their visual errors propagate through hand-crafted heuristics. We use a lightweight description-logic ontology as a controlled vocabulary for chart design roles.

To this end, we propose ChartLogic, a Parse-and-Connect framework that decomposes chart understanding into two separable sub-problems of equal status. *Parse* performs hierarchical panoptic parsing, mapping pixels to category labels, series factors, and instances. *Connect* performs rela-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

tional grounding, taking anchor-identified instances and predicting object-property relations that link marks, axes, legends, and data structures; the predicted graph is then consumed by task-conditioned queries to produce bindings. A description-logic ontology provides the vocabulary for *both* stages rather than Parse alone. Per-chart ABox assertions attach each instance—identified by a three-level code (category, series, instance)—to organizational and visual relation targets and record its literal attributes. Connect predicts these relations; literals are not Connect targets. ChartLogic evaluates whether a predicted graph, after relation-aware training, can be consumed by the same symbolic task templates as the GT graph, rather than by free-form QA accuracy.

Rather than organizing the paper by components, we organize it around three falsifiable propositions:

P1 (Nested identity): Aligned series supervision recovers the series identity that panoptic PQ cannot see.

P2 (Parse $\neq$ Connect): Strong performance on a ground-truth skeleton (where instance subjects are given) does not guarantee open-recovery binding, because predicted instances may be unmatched.

P3 (Category-vs-geometry gap): Under cross-source transfer, category accuracy holds while geometric recall falls first.

We instantiate these claims with hierarchical Parse metrics, ground-truth-skeleton vs. open-recovery SPARQL protocols, and the same task templates on predicted graphs.

2 Related Work

Chart understanding spans several task formulations that are often conflated. Derendering and table recovery methods (Liu et al., 2023a; Masry et al., 2023) map images to data tables without interpretable intermediates. VQA and captioning systems (Lee et al., 2023; Liu et al., 2023b) encode chart structure only implicitly, and high answer accuracy can overestimate real understanding (Huang et al., 2025). Detection-centric pipelines (Siegel et al., 2016; Luo et al., 2021; Zhu et al., 2026) localize components but resolve relations via heuristics that generalize poorly. Structured intermediates such as scene graphs (Tang et al., 2023) provide relational descriptions but lack formal semantics for consistency checking. Neuro-symbolic methods (Mao et al., 2019; Manhaeve et al., 2018; Yang et al., 2023) typically constrain neural outputs with logical rules; in contrast, ChartLogic uses the TBox as a controlled vocabulary and query interface for ontology-typed ABoxes, not as a training constraint.

3 Method

This section presents our approach: a formal basis and chart representation, a task definition, synthetic supervision, and the Parse–Connect model.

3.1 Formal Basis and Chart Representation

A chart comprises elements such as bars, axes, legends, and labels, with attributes (color, text, numeric quantities) and relations among them. These form a recurring chart schema, while each image provides concrete content. Description logic (DL) captures this distinction: a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a TBox $\mathcal{T}$ defining concepts and properties, and an ABox $\mathcal{A}$ recording assertions about specific instances. Here, the chart schema is represented in $\mathcal{T}$, and concrete content in $\mathcal{A}$.

The chart TBox $\mathcal{T}$ comprises concepts for entity types (e.g., `Bar`, `Line`, `XAxis`, `Legend`), datatype properties for literal attributes (color, text, numeric quantities), and object properties for relations linking entities to other visual or organizational instances. Each chart is an ABox $\mathcal{A}$ over $\mathcal{T}$; since $\mathcal{T}$ is shared, all image-specific information resides in $\mathcal{A}$. To recover $\mathcal{A}$ from pixels, we introduce an intermediate representation with two components: an instance anchor α and a hierarchical label map M.

Instance anchor. An instance anchor is a triple $\alpha = (c, s, i)$: c the category (from $\mathcal{T}$), s a chart-local group (e.g., series or axis side), and i the index within a (c, s) group.

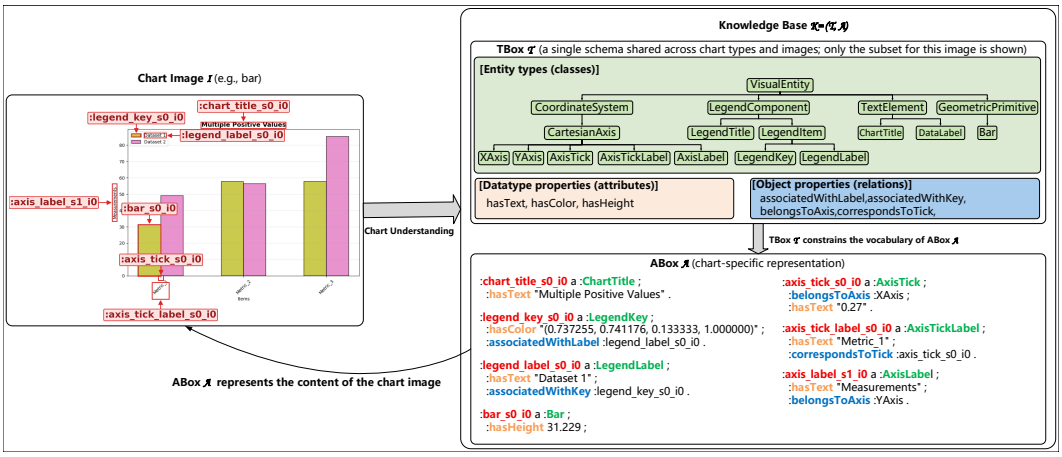


Figure 1: From a chart image to its ABox under a single TBox $\mathcal{T}$ shared across chart types. Left: a bar-chart image with selected individuals. Right: the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where only the subset of $\mathcal{T}$ relevant to this image is shown, and $\mathcal{A}$ instantiates the corresponding concepts and properties for this image. In the ABox excerpts, text colors indicate term types: red for instance names, green for their corresponding TBox class names, orange for datatype properties, and blue for object properties. $\mathcal{T}$ constrains the vocabulary of $\mathcal{A}$, and $\mathcal{A}$ represents the content of image I. Although the example is a bar chart, $\mathcal{T}$ is a single schema shared across chart types and independent of the particular image, while each image I has its own ABox $\mathcal{A}$.

Hierarchical label map. Let Λ be the pixel domain. The label map $M : \Lambda \to C \times S \times N$ assigns to each pixel p its anchor $M(p) = (c_p, s_p, i_p)$, with C, S, N the sets of categories, groups, and indices. Equivalently, M has three channels storing c, s, and i, respectively.

ABox. The ABox $\mathcal{A}$ is partitioned into a disjoint union

$$\mathcal{A} = \mathcal{A}^{\text{lit}} \uplus \mathcal{A}^{\text{org}} \uplus \mathcal{A}^{\text{vis}},$$

where $\mathcal{A}^{\text{lit}}$ records literal attributes, $\mathcal{A}^{\text{org}}$ organizational relations (e.g., to an axis or data mapping), and $\mathcal{A}^{\text{vis}}$ visual bindings between instances (e.g., legend key and label). The property `hasMaskEncoding` links each instance to its anchor in M. Since $\mathcal{T}$ constrains property domains and ranges, $\mathcal{A}$ can be checked for consistency.

3.2 Problem Definition

Given the representation in Section 3.1, we define the task as recovering the ABox $\mathcal{A}$ from a chart image I under a fixed ontology $\mathcal{T}$.

Formalization. We pack the anchors and ABox into $\Phi = (\{\alpha_k\}_{k=1}^{K}, \mathcal{A})$, where K is the number of instances, α_k are recorded in M, and I is the input. Given I, the objective is

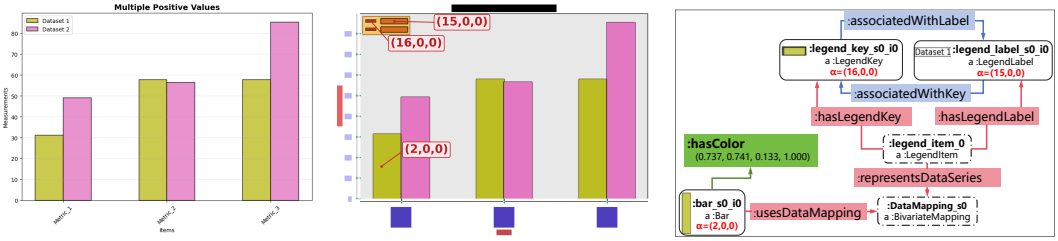
$$f : I \mapsto \hat{\Phi} = (\{\hat{\alpha}_k\}_{k=1}^{K}, \hat{\mathcal{A}}), \tag{1}$$

where $\hat{\mathcal{A}}$ is grounded on the same anchors.

We decompose f into two subtasks. *Parse* recovers the visual side: the anchors $\{\hat{\alpha}_k\}$, the label map $\hat{M}$ evaluated against $M^\star$, and the literal attributes $\hat{\mathcal{A}}^{\text{lit}}$. *Connect* predicts the relations $\hat{\mathcal{A}}^{\text{org}} \cup \hat{\mathcal{A}}^{\text{vis}}$ on the same anchors. Evaluation queries $\hat{\mathcal{A}}$ and $\mathcal{A}^\star$ with the same task templates and compares the resulting bindings. Neither task suffices alone: accurate localization does not guarantee correct relations, and correct relations do not compensate for inaccurate localization.

3.3 Logic-Guided Synthetic Chart Generation

The task requires a label map and a grounded ABox sharing instance anchors. Learning this recovery needs aligned supervision: both must refer to the same instances at pixel level. Real chart images



(a) Chart image sample (b) Hierarchical label map with (c, s, i) samples (c) Illustration of an ABox fragment

Figure 2: From a chart image to an aligned hierarchical label map and ABox. (a) Chart image I. (b) Hierarchical label map M, where each instance anchor $\alpha = (c, s, i)$ encodes category c, group s, and instance index i; the highlighted codes $(2, 0, 0)$, $(15, 0, 0)$, and $(16, 0, 0)$ correspond to the bar, legend label, and legend key expanded in (c). (c) ABox fragment for these three visual instances (solid rounded rectangles) and their associated organizational instances (dashed rounded rectangles). Sharp rectangles denote relation predicates, colored by their semantic role: green for literal attributes, red for organizational relations, and blue for visual bindings, corresponding to $\mathcal{A}^{\text{lit}}$, $\mathcal{A}^{\text{org}}$, and $\mathcal{A}^{\text{vis}}$ defined in the text. The anchor α links each visual instance in an ABox assertion to the corresponding region in the label map M, making each assertion traceable to pixels.

rarely come with a generating program, so such alignment cannot be read from pixels. We therefore synthesize charts. Under our control, the object tree drawn as I records category, series, geometry, and data bindings. We treat it as a single generation source Ω and derive I, $M^\star$, and $\mathcal{A}^\star$ from it, so visual identity and symbolic assertions share anchors by construction (Fig. 2).

Shared generation source. A controlled generator covers five chart types (bar, line, pie, scatter, histogram), samples structured data D and metadata (titles, axis labels), and writes a rendering specification. During rendering of I, we traverse the object tree and record geometry, style, labels, and data bindings. The resulting tree is Ω; each object carries a visual category c, a series factor s, and an instance index i from its metadata.

From Ω to aligned supervision. Using Ω as the single source of truth, we rasterize each object's geometry onto $M^\star$ and populate $\mathcal{A}^\star$ with literal, organizational, and visual-binding facts. Painting an object writes its anchor $\alpha = (c, s, i)$ into the three channels of $M^\star$. For storage, the channels are packed as

$$M^{\text{packed}}[y, x] = c_{y,x} \cdot 2^{24} + s_{y,x} \cdot 2^{16} + i_{y,x}, \tag{2}$$

where $c_{y,x} \in [0, 2^8 - 1]$, $s_{y,x} \in [0, 2^8 - 1]$, and $i_{y,x} \in [0, 2^{16} - 1]$. Pixels with the same α form one instance. Each object becomes an ontology instance whose `hasMaskEncoding` stores the packed α, and its facts are asserted from Ω. Each training sample is $(I, M^\star, \mathcal{A}^\star)$; corpus size and splits are in the experiments.

3.4 Parse–Connect Model

Given the aligned supervision above, we realize f as two stages over the same instance anchors (Fig. 3): Parse and Connect.

3.4.1 Parse: Hierarchical Panoptic Parsing

Parse performs hierarchical panoptic parsing to recover visual instances. Connect requires an addressable instance vector for each element, not only a mask; pixel-wise classifiers lack instance identity, and box detectors need a separate pooling step. A query-based set predictor associates each instance with a decoder embedding, so the query that predicts a mask can serve as the feature for relational heads. We therefore instantiate Parse with Mask2Former (Cheng et al., 2022).

Mask2Former uses a set of queries. We index queries by j and regions by k; k denotes a ground-truth region during training and a kept region at inference. Each query j predicts a class distribution $\hat{p}_j$ (with a no-object class) and a mask $\hat{m}_j$. During training, queries are matched to ground-truth regions in $M^\star$ by category and mask costs, and we compute classification and mask losses on that

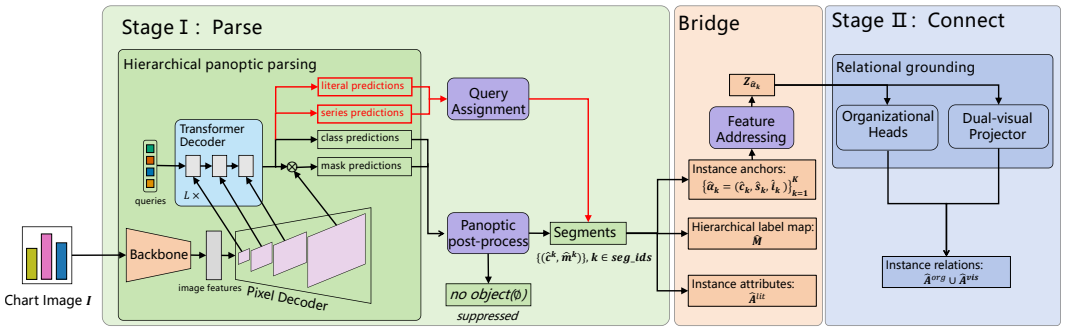


Figure 3: The architecture of ChartLogic. Parse (left) performs hierarchical panoptic parsing to recover instance anchors $\{\hat{\alpha}_k\}$, the hierarchical label map $\hat{M}$, and instance attributes $\hat{\mathcal{A}}^{\text{lit}}$. Feature addressing (center, Bridge) gathers instance features $z_{\hat{\alpha}_k}$ from the query assigned to each anchor $\hat{\alpha}_k$. Connect (right) predicts instance relations $\hat{\mathcal{A}}^{\text{org}} \cup \hat{\mathcal{A}}^{\text{vis}}$.

matching; the matched query is $\sigma(k)$. At inference, panoptic post-processing discards no-object queries and resolves overlaps, keeping regions as instances. The category of a region is the class of its query.

We attach a linear series head to the same query embedding q_j:

$$\hat{p}_{s,j} = \text{Softmax}(W_s q_j) \in \Delta^{S_{\max}}, \tag{3}$$

where $S_{\max}$ is the number of series labels. The series head has no no-object class. Each query emits $(\hat{p}_j, \hat{m}_j, \hat{p}_{s,j})$. Series is not part of the matching cost, and object properties in $\mathcal{A}^{\text{org}} \cup \mathcal{A}^{\text{vis}}$ are predicted by Connect, not Parse. The within-group index i in $M^\star$ is not a third class label. Training reads $\hat{p}_{s,\sigma(k)}$ and supervises it with $s_k^\star$; inference reads the series distribution from the query linked to each kept region.

Training. Ground-truth segments come from $M^\star$: each is the binary mask of one instance, labeled by $c^\star$ and $s^\star$. The within-group index i is not used in matching or series loss. Query–GT matching follows Mask2Former and uses category and mask costs only; s is not included. Let σ denote this matching, and $\mathcal{M}$ the matched ground-truth regions. On matched queries we add a series cross-entropy

$$\mathcal{L}_{\text{series}} = \sum_{k \in \mathcal{M}} \text{CE}\big(\hat{p}_{s,\sigma(k)},\, s_k^\star\big), \tag{4}$$

where $\hat{p}_{s,j}$ is the series distribution defined in Eq. equation 3, and keep the standard classification and mask losses $\mathcal{L}_{\text{m2f}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{mask}}$. We additionally attach a lightweight literal head to the same query embedding, which outputs a set of attribute–value pairs, and optionally add a query-level visual-literal term $\mathcal{L}_{\text{literal}}$ (e.g., `hasColor`). The Parse objective is

$$\mathcal{L}_{\text{parse}} = \mathcal{L}_{\text{m2f}} + \lambda_{\text{ser}}\, \mathcal{L}_{\text{series}} + \lambda_{\text{lit}}\, \mathcal{L}_{\text{literal}}. \tag{5}$$

Inference. Each query emits class, mask, series, and literal predictions. Panoptic post-processing uses class and mask only: it discards no-object queries and yields segments $\{(\hat{c}_k, \hat{m}_k)\}$, where k indexes the segments. Since post-processing discards the originating queries, we recover the series and literal predictions by matching each segment back to its query via mask IoU:

$$q^*(k) = \arg\max_j \text{IoU}(\hat{m}_k, \hat{m}_j). \tag{6}$$

We then read the series factor and instantiate the instance attributes $\hat{\mathcal{A}}^{\text{lit}}$ from that query:

$$\hat{s}_k = \arg\max_s \hat{p}_{s,q^*(k)}. \tag{7}$$

Post-processing assigns each kept region a chart-unique segment id. We enumerate an instance index $\hat{i}_k$ within each $(\hat{c}_k, \hat{s}_k)$ group, record $(\hat{c}_k, \hat{s}_k, \hat{i}_k)$ into $\hat{M}$, and match predicted regions to $M^\star$ by mask overlap for evaluation. In sum, s and literals are trained through query–GT matching, while i is assigned at inference by enumeration within each $(\hat{c}, \hat{s})$ group.

3.4.2 FEATURE ADDRESSING

Feature addressing is the bridge from Parse outputs to Connect heads: for each instance anchor $\hat{\alpha}_k$, it materializes a feature vector $z_{\hat{\alpha}_k}$ and does not learn relations. $z_{\hat{\alpha}_k}$ is the decoder embedding of the query bound to that anchor. At inference, Parse has already bound the kept region k to $q^*(k)$, so

$$k \;\mapsto\; q^*(k) \;\mapsto\; z_{\hat{\alpha}_k}, \tag{8}$$

without a second matching step. During training, for each ground-truth region k, $z_{\hat{\alpha}_k}$ is the embedding of its matched query $\sigma(k)$; if no query is matched, we pool backbone features over the ground-truth mask region and use that vector instead.

3.4.3 CONNECT: RELATIONAL GROUNDING

Connect predicts $\mathcal{A}^{\text{org}}$ and $\mathcal{A}^{\text{vis}}$ (Sec. 3.1) from these features. We name the heads after the ABox partition of Section 3.1: *Type-A* for literal attributes, *Type-B* for organizational object properties, and *Type-C* for dual-visual bindings. For a fixed organizational predicate such as `belongsToAxis`, a *Type-B* object head classifies the relational object (e.g., `XAxis` vs. `YAxis`). A *Type-C* dual-visual projector—hereafter *G2L*, a legend–mark binder trained on `correspondsTo`—is trained with a contrastive (InfoNCE) objective: for each legend key, the positive is its corresponding data mark, and negatives are other data marks in the same chart, together with data marks from other charts in the batch. Legend key–label bindings (i.e., `LegendKey` to `LegendLabel` correspondences) also appear in $\mathcal{A}^{\text{vis}}$ (Fig. 2); we keep them for a complete ABox but do *not* train a dedicated head for them, since they are largely fixed by layout rather than by data correspondence. In our main Connect configuration—*G2L-joint*—we jointly fine-tune Parse with both the G2L projector and the Type-B axis head; the latter is used for axis-membership queries at evaluation.

Training objective. Each addressed feature $z_{\hat{\alpha}}$ is trained to predict the organizational target and dual-visual partner specified in $\mathcal{A}^\star$. The Connect loss is

$$\mathcal{L}_{\text{connect}} = \mathcal{L}_{\text{org}} + \mathcal{L}_{\text{vis}} + \mathcal{L}_{\text{aux}}, \tag{9}$$

where $\mathcal{L}_{\text{org}}$ covers organizational targets, $\mathcal{L}_{\text{vis}}$ covers dual-visual bindings (primarily key–mark `correspondsTo`), and $\mathcal{L}_{\text{aux}}$ collects auxiliary consistency terms and does not repeat the series or literal losses in $\mathcal{L}_{\text{parse}}$. Relative loss weights are selected on validation and reported in the experiments. Connect is evaluated by the SPARQL-binding protocol; the predicate profile and closed/open settings are defined in the experiments.

3.4.4 JOINT TRAINING

Parse and Connect share the kept regions and their query embeddings but keep their objectives separable. Literal supervision (color, geometry) is handled in $\mathcal{L}_{\text{parse}}$ when the literal head is used, and not duplicated in Connect. During joint training,

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{parse}} + \lambda_{\text{conn}}\, \mathcal{L}_{\text{connect}}, \tag{10}$$

with λ_{conn} reported in the experiments. Feature addressing is the only bridge: Connect operates on addressed features and does not rewrite per-pixel Parse labels.

4 EXPERIMENTS

Our goal is not to maximize a single panoptic score, but to test whether ChartLogic produces an ontology-typed, queryable instance graph $\hat{\Phi} = (\{\hat{\alpha}_k\}_{k=1}^{K}, \hat{\mathcal{A}})$. We organize the experiments around three research questions:

RQ1: How well does Parse recover instance identities? We decompose Parse with nested identity metrics aligned to $\alpha = (c, s, i)$, covering category, series, and instance index.

RQ2: Can the recovered graph answer the same queries as a ground-truth graph? We query either a GT skeleton or a graph built only from predicted instances with the same SPARQL templates, covering both local attribute queries and global topology queries.

RQ3: Are the gains attributable to the method rather than to the renderer? We ablate series supervision and replace learned edges by heuristics, so that gains can be attributed to the method rather than to the renderer.

Table 1: Hierarchical Parse on held-out synthetic **test** (n=400). Checkpoints selected by validation native PQ. Dash: without a series coordinate there is no $\mathrm{Acc}_{cs|\mathrm{geo}}$ to compute.

Model	PQ	R_{geo}	$\mathrm{Acc}_{c\|\mathrm{geo}}$	$\mathrm{Acc}_{cs\|\mathrm{geo}}$	$\mathrm{InstRec}_{cs}$
No series head	0.601 ± 0.002	0.672 ± 0.002	1.000 ± 0.001	—	—
Shuffled series	0.606 ± 0.003	0.679 ± 0.002	1.000 ± 0.000	0.039 ± 0.018	0.027 ± 0.013
Real series (λ=0.3)	0.612 ± 0.004	0.686 ± 0.005	1.000 ± 0.000	0.959 ± 0.005	0.657 ± 0.008
+ literal (`lit005`)	0.607 ± 0.005	0.682 ± 0.003	1.000 ± 0.000	0.933 ± 0.002	0.635 ± 0.006
+ G2L-joint	0.606 ± 0.003	0.678 ± 0.004	1.000 ± 0.000	0.944 ± 0.003	0.638 ± 0.005

4.1 SETUP

Dataset. The main dataset contains 3,200 training, 400 validation, and 400 test charts (bar, line, pie, scatter, histogram) at 448×448, generated from the shared generation source of Section 3.3. Each chart includes an image, a hierarchical map $M^\star$, and a core-profile ABox $\mathcal{A}^\star$. This synthetic split is fully automatic from Ω and requires no human annotation. As an external content-source stress test, we build verify-120 (ChartQA-Redraw-120): 120 ChartQA validation charts (Masry et al., 2022). Constructing verify-120 involves two human steps, neither of which produces pixel-level supervision. First, we calibrate the original ChartQA tables and annotate light style attributes (chart type, layout cues, and series names). Second, these calibrated tables and style annotations are fed into our generation pipeline, which re-draws each chart and produces the corresponding $M^\star/\mathcal{A}^\star$. We then visually compare the re-drawn image against the original to confirm that the style is preserved; no pixel-wise hand labeling is performed. verify-120 is therefore ChartQA-sourced under our renderer, not an evaluation on original ChartQA pixels, and is used only as a Parse-level content-source stress test, not as an open-recovery benchmark.

Implementation details. We use the Parse–Connect model of Section 3.4 with the following hyperparameters: λ_{ser}=0.3, λ_{lit}=0.05, and λ_{conn}=0.15. Within $\mathcal{L}_{\mathrm{connect}}$, the Type-C/G2L InfoNCE term is weighted 1.0 and the Type-B object CE term 0.5; color-contrast auxiliaries are disabled. Training uses AdamW with cosine decay and batch size 2. Model selection uses validation native PQ.

Parse metrics. Native PQ is used for model selection but does not score series membership. We match predicted and GT instances one-to-one by mask IoU (IoU$\geq$ 0.5, Hungarian) and report geometric recall R_{geo} (the fraction of GT instances matched), conditional category accuracy $\mathrm{Acc}_{c|\mathrm{geo}}$, ID-strict group accuracy $\mathrm{Acc}_{cs|\mathrm{geo}}$ (c and series id s both correct on matched pairs), and instance recovery $\mathrm{InstRec}_{cs}$ (the fraction of GT instances with both c and s correct). Matching and set-based binding recall do not use list indices, so scores are invariant to the order of predicted instances. In contrast, $\mathrm{Acc}_{cs|\mathrm{geo}}$ is *not* invariant to relabeling s: it tests whether the ontology coordinate recovered by Parse matches the vocabulary that Connect and SPARQL consume. As a diagnosis, we also report chart-local Hungarian remapping ($\mathrm{Acc}_{cs}^{\mathrm{perm}}$) and $\mathrm{ARI}(s)$ on category-correct matches. Because $\mathrm{Acc}_{cs|\mathrm{geo}}$ compares the predicted series coordinate against the vocabulary that Connect and SPARQL consume, it determines whether a predicted instance can participate in binding queries; native PQ, which scores class and region only, cannot see it.

Connect protocols. *GT-skeleton probe:* GT subjects with predicted edges inserted into $\mathcal{A}^\star$—tests whether relation heads are readable when individuals are given. *Open recovery (primary):* predicted Parse instances as individuals, predicted or heuristic edges, same SPARQL templates—tests whether the predicted graph is queryable. We report binding-tuple recall and decompose failures into unmatched subjects (`parse_miss`; tuple-level $\sum n_{\mathrm{miss}}/\sum n_{\mathrm{gt}}$) vs. wrong bindings among matched subjects (`connect_wrong`; same denominator). Recall is macro-averaged over charts; miss/wrong rates are tuple-level aggregates. We focus on three structural tasks: axis membership (`belongsToAxis`, measured on axis tick labels), mark–series grouping (`usesDataStructure`), and legend–mark correspondence (`correspondsTo`).

4.2 RQ1: DOES PARSE RECOVER INSTANCE IDENTITY?

We decompose Parse along the instance anchor $\alpha = (c, s, i)$.

Panoptic PQ does not see series identity. Table 1 decomposes Parse along α. Whenever a GT instance is geometrically matched, its fine category is almost always correct ($\mathrm{Acc}_{c|\mathrm{geo}} \approx 1$). The dominant Parse error is missing instances: R_{geo} remains around 0.67–0.69. Native PQ tracks this localization ceiling and does not score series membership: shuffled vs. real series remain tied at 0.606 vs. 0.612. Thus PQ alone does not reveal series identity.

Nested metrics do. ID-strict $\mathrm{Acc}_{cs|\mathrm{geo}}$ rises from 0.039 to 0.959 and $\mathrm{InstRec}_{cs}$ from 0.027 to 0.657 under aligned series supervision. Table 2 rules out a global permutation: shuffled series reaches only 0.214 under chart-local remapping with $\mathrm{ARI}(s)$=0.007, so the failure is a collapsed partition rather than a relabeled one. Real series is essentially invariant (0.959 → 0.962, $\mathrm{ARI}(s)$=0.924).

Table 2: Series-label diagnosis under the same geometric matches as Table 1 (held-out test, n=400).

Model	Acc_{cs}	$\mathrm{Acc}_{cs}^{\mathrm{perm}}$	$\mathrm{ARI}(s)$
Shuffled	0.039 ± 0.018	0.214 ± 0.012	0.007 ± 0.003
Real	0.959 ± 0.005	0.962 ± 0.004	0.924 ± 0.018

Downstream queries consume the coordinate. Under open recovery, mark–series binding reads the Parse series field (Table 3). Real series (λ=0.3) recovers usable group ids, while shuffled series collapses to near-chance recall with matched-but-wrong errors; unmatched-subject rates remain comparable, so the gap is not localization. On a controlled subset where all subjects are matched, ablating predicted edges (identity / shuffle / drop-series) drives series binding from 1.0 to 0.0, confirming that SPARQL templates read ABox structure rather than image shortcuts. On line/scatter, even true series supervision yields low open series binding because mark instances are rarely recovered: a Parse-coverage failure, not a series-coordinate failure.

Table 3: Open-recovery mark–series binding on held-out synthetic **test** (n=400). Series edges come from the Parse series field. Rows correspond to the shuffled / real series entries of Table 1.

Model	Recall	Matched-wrong	Unmatched
Shuffled	0.036	0.780	0.174
Real	0.529	0.119	0.173

4.3 RQ2: Is the recovered graph queryable?

To address RQ2, we compare two protocols: a GT-skeleton probe, which evaluates Connect when instance subjects are given, and open recovery on predicted instances. A natural failure mode is that relation heads remain higher under the GT skeleton, yet the predicted ABox degrades when subjects come from Parse, because individuals may be missing. Table 4 contrasts the two protocols on G2L-joint.

On open recovery (Table 4), axis membership reaches 0.920, mark–series grouping 0.484, and legend–mark correspondence 0.474. Same-mask heuristics are slightly higher on axis (0.932) and series (0.504), while Open improves legend–mark (0.474 vs. 0.319). Series binding fails mainly by unmatched subjects on line/scatter, while legend–mark fails mainly by wrong correspondences among matched subjects, separating a Parse-coverage limit from a Connect-binding limit.

Table 5 makes the attribution checkable. Axis membership is high on all chart types (R_{geo} 0.89–0.97), so its open-recovery success is not the bottleneck. On line/scatter *series* binding, mark R_{geo} is low and almost all errors are unmatched subjects (**Parse**). On the *same* chart types, legend-key R_{geo} remains high (≈0.88–0.99) while matched-but-wrong dominates (**Connect**): the failure is relational, not perceptual. Series binding on line/scatter fails mainly by unmatched subjects, tracking mark R_{geo}; it is a Parse-coverage limit, not a separate Connect claim. Macro averages therefore hide a task×type structure: Parse-limited and Connect-limited failures can coexist on the same chart family.

4.4 RQ3: Are gains attributable to the method?

This section addresses RQ3 by testing whether the gains come from learned relations rather than heuristics or detector coverage. Relative to same-mask heuristics (Table 4, Heuristic), Open shows a clear gain only on legend–mark (0.474 vs. 0.319); axis (0.920 vs. 0.932) and series (0.484 vs. 0.504)

Table 4: Structural queries on the held-out synthetic test set (n=400), G2L-joint. All are binding-tuple recall on the same scale. Open uses learned binders; Heuristic freezes color/geometry binders on the same instances; Skeleton uses GT subjects with predicted edges. Unmatched / Matched-wrong decompose Open failures only.

	Tuple recall			Open error	
Task	Open	Heuristic	Skeleton	Unmatched	Matched-wrong
Axis membership	0.920	0.932	0.980	0.069	0.013
Mark–series grouping	0.484	0.504	0.620	0.197	0.157
Legend–mark correspondence	0.474	0.319	0.662	0.041	0.539

Table 5: Task×type attribution on open recovery (G2L-joint, test). R_{geo}: instance-level subject recall. Binding / miss / wrong: tuple-level, as in Table 4. Attr.: miss$\geq 0.5 \rightarrow$ Parse; wrong$\geq 0.3 \rightarrow$ Connect; else mixed. Pie has no axis task. Values are macro-averaged within each chart type and are not directly comparable to the chart-level macro in Table 4.

	Axis membership					Series grouping					Legend–mark				
Type	R_{geo}	Binding	miss	wrong	Attr.	R_{geo}	Binding	miss	wrong	Attr.	R_{geo}	Binding	miss	wrong	Attr.
Bar	0.971	0.971	0.030	0.001	mixed	0.785	0.655	0.217	0.277	mixed	0.878	0.348	0.122	0.647	Connect
Hist.	0.956	0.952	0.044	0.002	mixed	0.975	0.924	0.074	0.000	mixed	0.988	0.812	0.013	0.175	mixed
Line	0.911	0.920	0.089	0.002	mixed	0.174	0.031	0.826	0.153	Parse	0.931	0.191	0.045	0.880	Connect
Pie	—	—	—	—	—	0.938	0.662	0.063	0.321	Connect	0.998	0.808	0.002	0.187	mixed
Scatter	0.887	0.884	0.113	0.002	mixed	0.200	0.147	0.800	0.111	Parse	0.946	0.211	0.036	0.850	Connect

are slightly lower. Series open recall is therefore driven by the Parse series head, and the learned Connect gain is concentrated on legend–mark.

4.5 Cross-source transfer (supplementary to RQ1/RQ2)

As an external-validity check for RQ1 and RQ2, we evaluate zero-shot transfer on verify-120 (Sec. 4.1).

On verify-120, category accuracy remains high (0.968) while R_{geo} and PQ drop to 0.505 and 0.420. Because charts are re-drawn from calibrated tables, this measures a content-source shift under a familiar graphics stack, not transfer to unconstrained screenshots. Within verify-120, bar R_{geo} (0.439) is close to line (0.470), contrasting with the in-domain ordering where bar is the easier type; we therefore do not transfer the synthetic per-type ordering to the redraw. We do not run open recovery on verify-120: with $R_{\text{geo}} = 0.505$, the same SPARQL templates would mainly restate Parse misses rather than reveal a new failure mode. We therefore report verify-120 as a Parse-level content-source stress test: a category-vs-geometry gap, not a full open-recovery measurement.

Table 6: Zero-shot G2L-joint on verify-120 (ChartQA-Redraw-120) vs. synthetic test; not an evaluation on original ChartQA pixels.

Split	PQ	R_{geo}	$\text{Acc}_{c\|\text{geo}}$	$\text{Acc}_{cs\|\text{geo}}$
Synthetic	0.606	0.678	1.000	0.944
verify-120	0.420	0.505	0.968	0.864

5 Conclusion

We present ChartLogic, a Parse-and-Connect framework that recovers a queryable instance graph $\hat{\Phi}$ from pixels under a shared TBox $\mathcal{T}$, with an open-recovery protocol and a synthetic generator for aligned supervision. Three propositions organize the evidence. P1: aligned series supervision recovers a nested identity invisible to panoptic PQ but consumed by series binding. P2: open recovery failures split into unmatched instances and wrong bindings among matched ones, so Parse and Connect need separate attribution. P3: cross-source transfer shows a category-vs-geometry gap: category accuracy holds while geometric recall falls first. End-to-end chart QA models output answers or tables without instance anchors and cannot participate in binding-level queries; we therefore compare against structural baselines and argue for evaluating intermediate visual-symbolic structure with bounded, inspectable symbolic queries rather than final answers alone.

ACKNOWLEDGMENTS

We thank anonymous reviewers for feedback. Funding details omitted for anonymous review.

ETHICS STATEMENT

This work studies chart parsing and relational grounding on synthetic charts and verify-120 (ChartQA-Redraw-120). It does not collect personal data or deploy decision systems affecting individuals. Residual risks include misuse of structured chart recovery for automated extraction from proprietary documents; we release methods for research on verifiable multimodal understanding and encourage responsible use under existing data licenses.

REPRODUCIBILITY STATEMENT

For reproducibility, code, documentation, and data samples are provided as supplementary material. The complete dataset will be released publicly upon publication. Hyperparameters, splits, and checkpoint selection criteria are reported in Section 4.1, where the construction of verify-120 is also described.

AI USE STATEMENT

We used generative AI tools for language editing and drafting LaTeX/formatting assistance. We did not use generative AI tools to invent experimental results, fabricate citations, or generate core scientific claims. All technical content, equations, and reported numbers were authored and verified by the authors. We take responsibility for the final content of this work.

REFERENCES

Jinyue Chen, Lingyu Kong, Haoran Wei, Chenglong Liu, Zheng Ge, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. Onechart: Purify the chart structural extraction via one auxiliary token. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 147–155, 2024.

Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1280–1289. IEEE, 2022.

Muye Huang, Han Lai, Xinyu Zhang, Wenjun Wu, Jie Ma, Lingling Zhang, and Jun Liu. Evochart: A benchmark and a self-training approach towards real-world chart understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 3680–3688, 2025.

Jovana Kondic, Pengyuan Li, Dhiraj Joshi, Isaac Sanchez, Ben Wiesel, Shafiq Abedin, Amit Alfassy, Eli Schwartz, Daniel Caraballo, Yagmur Gizem Cinar, et al. Chartnet: A million-scale, high-quality multimodal dataset for robust chart understanding. *arXiv preprint arXiv:2603.27064*, 2026.

Jay Lal, Aditya Mitkari, Mahesh Bhosale, and David Doermann. Lineformer: Line chart data extraction using instance segmentation. In *International conference on document analysis and recognition*, pp. 387–400. Springer, 2023.

Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pp. 18893–18912. PMLR, 2023.

Fangyu Liu, Julian Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. Deplot: One-shot visual language reasoning by plot-to-table translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10381–10399, 2023a.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12756–12770, 2023b.

Junyu Luo, Zekun Li, Jinpeng Wang, and Chin-Yew Lin. Chartocr: Data extraction from charts images via a deep hybrid framework. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1916–1924. IEEE, 2021.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

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

Ahmed Masry, Parsa Kavehzadeh, Enamul Hoque, Shafiq Joty, et al. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 14662–14684, 2023.

Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. Figureseer: Parsing result-figures in research papers. In *European Conference on Computer Vision*, pp. 664–680. Springer, 2016.

Benny Tang, Angie Boggust, and Arvind Satyanarayan. Vistext: A benchmark for semantically rich chart captioning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7268–7298, 2023.

Zhengzhuo Xu, Bowen Qu, Yiyan Qi, Sinan Du, Chengjin Xu, Chun Yuan, and Jian Guo. Chartmoe: Mixture of diversely aligned expert connector for chart understanding. In *International Conference on Learning Representations*, volume 2025, pp. 78550–78572, 2025.

Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. *arXiv preprint arXiv:2307.07700*, 2023.

Jiangning Zhu, Yuxing Zhou, Zheng Wang, Juntao Yao, Yima Gu, Yuhui Yuan, and Shixia Liu. Infodet: A dataset for infographic element detection. In *International Conference on Learning Representations*, volume 2026, pp. 29158–29189, 2026.