MechParser: A Vision-Language Framework for Parsing Chemical Reaction Mechanism Diagrams

Yufan Chen[1] **Shing Hin Lu**[1] **Yujian Yuan**[1] **Ching Ting Leung**[2] **Hanyu Gao**[1*]
[1]The Hong Kong University of Science and Technology [2]University of Notre Dame

Abstract

Reaction mechanisms are central to understanding chemical reactivity, guiding reaction prediction, synthesis design, and selectivity rationalization. In chemistry literature, they are universally depicted as curved electron-pushing arrows that capture stepwise electron flow and bond changes. Yet, this vast knowledge remains trapped in unstructured diagrams, forcing data-driven modeling to rely on small or template-expanded datasets. We introduce MechParser, a vision-language framework that converts mechanism diagrams into SM-SMARTS (Sequential Mechanism-Aware SMARTS), a representation encoding electron flow and atom state changes as a cumulative string sequence. At its core, MechParser-VL applies a left-right visual prompt design (arrow diagram alongside an atom-indexed reference) to separate dynamic arrow reasoning from static atom identification, and is trained via a four-stage curriculum on over 300K samples from a geometry-aware synthetic data pipeline and an expert-annotated real-world dataset. On both synthetic and real-world benchmarks, MechParser-VL substantially outperforms zero-shot proprietary and open-source VLMs, and controlled comparisons with task-adapted baselines on the same backbone and data trace its margin to the visual prompt and staged curriculum. By standardizing visual mechanisms into machine-readable data, MechParser lays the groundwork for constructing large-scale mechanism databases from the chemical literature, which in turn can support the training of advanced mechanism-centric models on authentic mechanistic data. Code, model weights and data are available at `https://github.com/CYF2000127/MechParser`.

1 Introduction

Reaction mechanisms describe the electron-flow narrative of how chemical bonds form and break, and they sit at the heart of organic chemistry. They provide chemists with the predictive power to design new syntheses, rationalize selectivity, and discover new reactivity [14, 37]. Yet, despite their fundamental role, the vast body of mechanistic knowledge accumulated over decades of research remains overwhelmingly inaccessible to machines: it is encoded as curved electron-pushing arrows on diagrams scattered across textbooks, papers, and patents. This data bottleneck has been a persistent obstacle for the nascent field of data-driven mechanism modeling [30, 41, 26, 17]. Unlike adjacent areas of data-driven chemistry such as molecular property prediction [62, 67, 52], retrosynthesis [16, 8, 55, 39, 23], and reaction product prediction [53, 28, 15, 54], where large-scale curated corpora have fueled rapid progress, existing mechanism datasets primarily rely on expert curation of mechanistic reaction templates, which are then used to expand overall reactions into mechanistic steps. The manual curation significantly limits the diversity and coverage of existing reaction mechanism datasets.

Existing automated extraction tools fall short of capturing reaction mechanisms. Optical Chemical Structure Recognition (OCSR) systems [50, 48, 11] excel at converting molecular images into machine-readable structures, but they operate exclusively on static molecules. Reaction-image parsers [47, 12, 57, 58] extract reactant-product pairs from reaction schemes, capturing only the net transformation rather than the electron-pushing logic that connects them. The dynamic essence of

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

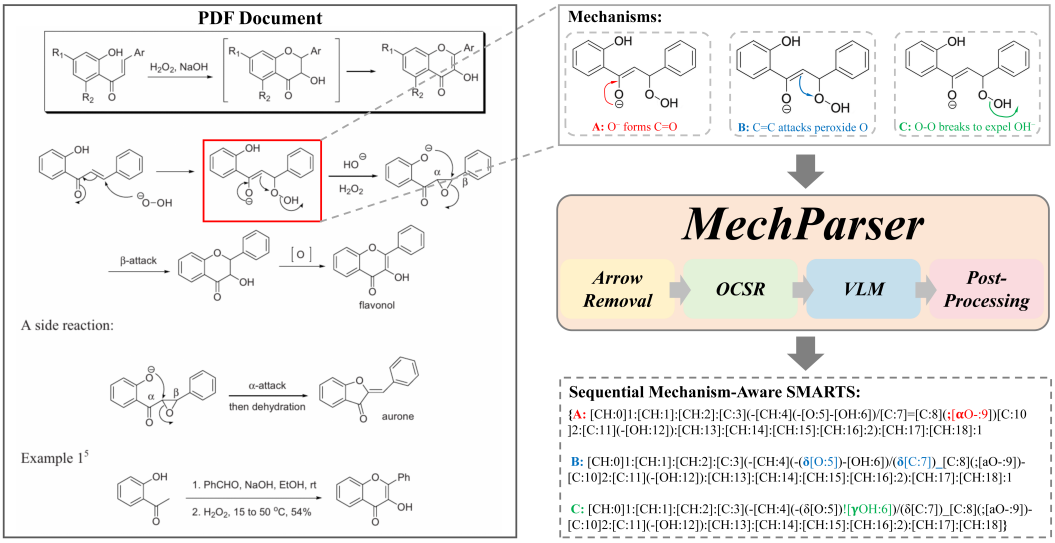

Figure 1: **MechParser overview.** MechParser parses mechanism diagrams from real literature or patents into SM-SMARTS, a sequential string representation that cumulatively encodes each curved-arrow step, including bond breaking, bond formation, and atom state changes. The framework combines arrow removal, OCSR, VLM-based arrow reasoning, and post-processing to produce the final SM-SMARTS output.

mechanism diagrams, namely the curved arrows depicting stepwise electron flow, bond breaking, and bond formation, is therefore treated as visual noise by current pipelines, leaving the true logic of organic chemistry inaccessible. For mechanism-centered modeling, the same representational gap remains: existing molecular linear notations, such as SMILES [61], SMARTS [59], and SELFIES [34], are well suited to static structures or reactant-product transformations, but do not explicitly describe the stepwise atom-state and bond changes that define a reaction mechanism.

Building such an automated mechanism parsing system poses three central challenges. (1) Representation gap: a mechanism-level language must capture dynamic, stepwise electron redistribution rather than only static structures or net transformations. (2) Data scarcity: the combinatorial complexity of multi-step reactions makes curating large-scale annotated data manually prohibitive. (3) Visual complexity: in real-world literature, mechanism diagrams exhibit immense variability in drawing styles, with curved arrows frequently overlapping dense molecular graphs. While Vision-Language Models (VLMs) [40, 4, 49, 1] demonstrate powerful general reasoning capable of handling such variability, they struggle with the precise spatial localization required here: even a minor mislocalization of an arrow's source or sink propagates into catastrophic failure of the entire mechanistic logic. To overcome these limitations, we introduce MechParser, a vision-language framework for parsing chemical reaction mechanism diagrams. As shown in Figure 1, given a raw mechanism diagram from literature, MechParser converts the curved electron-pushing arrows into SM-SMARTS (Sequential Mechanism-Aware SMARTS), a stepwise, machine-readable string representation that encodes dynamic arrow semantics directly into molecular strings. At the core of this framework, MechParser-VL predicts the ordered electron-flow arrow list from our visual prompt. To tackle data scarcity and alignment challenges, we develop a geometry-aware synthetic data pipeline and curate an expert-annotated real-world dataset, coupled with a curriculum learning framework optimized via reinforcement learning. Beyond extraction, MechParser provides foundational infrastructure for the mechanism prediction community: by parsing real-world diagrams into a standardized SM-SMARTS vocabulary, it enables the construction of large-scale mechanism databases, which in turn makes it possible to train robust and generalizable mechanism prediction models on real literature data and to support broader mechanism-centric downstream applications. Our main contributions are:

- **SM-SMARTS:** we extend the SMARTS notation with symbols for bond transformations and atom state changes, encoding stepwise electron flow as a sequential, machine-readable string that also serves as an effective training target for mechanism prediction.
- **Scalable synthetic data generation and real data construction:** we design a geometry-aware synthetic data pipeline yielding 300K+ training samples with zero manual effort, and introduce

a manually annotated dataset of real-world mechanism diagrams for both domain adaptation and community evaluation.
- **Visual prompt and curriculum training:** we train MechParser-VL with a visual prompt (left-right image concatenation with atom-indexed reference) to disentangle arrow reasoning from atom identification, coupled with a four-stage curriculum refined by a chemistry-aware multi-tier reward via reinforcement learning.
- **Strong empirical results:** MechParser-VL substantially outperforms general-purpose VLMs on arrow reasoning; controlled comparisons with task-adapted baselines sharing its backbone and data attribute the margin to the visual prompt and staged curriculum, and the fully automated pipeline reaches 76.2% exact match on real literature diagrams with a 4B backbone that supports practical deployment.

2 Related Work

Extensions of SMILES and SMARTS. SMILES [61] and SMARTS [59] are foundational linear representations in cheminformatics. FG-SMILES in Image2SMILES [32] represents R-groups as pseudo-atoms, and MolParser [19] introduced E-SMILES for Markush structures and polymers. Reaction description languages such as ReactSeq [64] encode molecular editing operations for reaction transformations, but do not explicitly represent the ordered electron-flow semantics of curved arrows. Most recently, MechSMILES [44] encodes arrow-pushing operations for language-model-based mechanism prediction from textual reaction inputs, but is not designed to align these operations with visual diagram elements or cumulative atom-indexed scaffold states. Our SM-SMARTS addresses these gaps by preserving an atom-mapped SMARTS scaffold with stepwise bond and atom-state edits, serving both as a parsing target for mechanism diagrams and as a structured representation for downstream prediction tasks. In a controlled comparison on mech-USPTO-31K [9], a mechanism prediction model trained on SM-SMARTS targets matches one trained on MechSMILES under plain reactant input and leads by 3.51 points of arrow-set accuracy when both are grounded on the atom-mapped scaffold that MechParser supplies at inference time (Appendix N).

Optical Chemical Structure Recognition (OCSR). OCSR translates molecular diagrams into machine-readable formats. Early rule-based systems [20] have been superseded by deep learning approaches in two paradigms: end-to-end methods such as DECIMER [50], SwinOCSR [66], MolParser [19], and MarkushGrapher [43] formulate OCSR as image captioning, while graph-based methods including MolScribe [48], MolGrapher [42], and MolNexTR [11] explicitly reconstruct molecular graphs for improved structural validity. All these paradigms focus exclusively on static molecular backbones, treating curved arrows as noise. Our work is complementary: we leverage existing OCSR to extract the static scaffold and dedicate a VLM to deciphering the superimposed dynamic arrows.

VLMs in Chemical Information Extraction. Vision-Language Models [40, 3, 1] have been increasingly adopted for chemical information extraction. RxnScribe [47], RxnIM [12], RxnCaption [57], and IdtVP [58] parse reaction diagrams into structured reaction schemes, while ChemVLM [38] adapts a multimodal backbone for general chemistry tasks and multi-modal pipelines such as ChemEAGLE [13] and MERMaid [36] extract complete reaction schemes from complex literature figures. However, these systems identify overall transformations but entirely bypass the mechanistic layer: curved electron-pushing arrows remain untranslatable background. Our work explicitly addresses this gap via a spatial-disentangling visual prompt and chemistry-aware RL.

Generative Reaction Mechanism Prediction. Inferring the sequence of elementary mechanistic steps is an emerging frontier. Early efforts framed mechanism prediction over hand-curated reaction families [31] or learned electron paths from atom-mapped reactions [6]. More recently, FlowER [30] employs continuous-time electron flow matching; ArrowFinder [41] scores atom pairs for polar arrow-pushing; ReactAIvate [26] applies attention-based GNNs to reactivity hotspots; Reactron [10] integrates electron movement into reaction outcome prediction; and DeepMech [17] generates intermediates step by step with sequence-to-sequence models. These models depend on mechanistic corpora that are at present produced by template expansion: Chen et al. [9] build mech-USPTO-31K by pairing extracted reaction templates with expert-coded mechanistic templates, and Joung et al. [29] impute intermediates between reported reactants and products to obtain over five million elementary steps, naming dataset diversity, consecutive-step prediction and atom-conservation violations as the principal remaining obstacles. In both, the mechanism is inferred from a net transformation rather than read from a chemist's own depiction, so coverage is bounded by template availability; large-scale

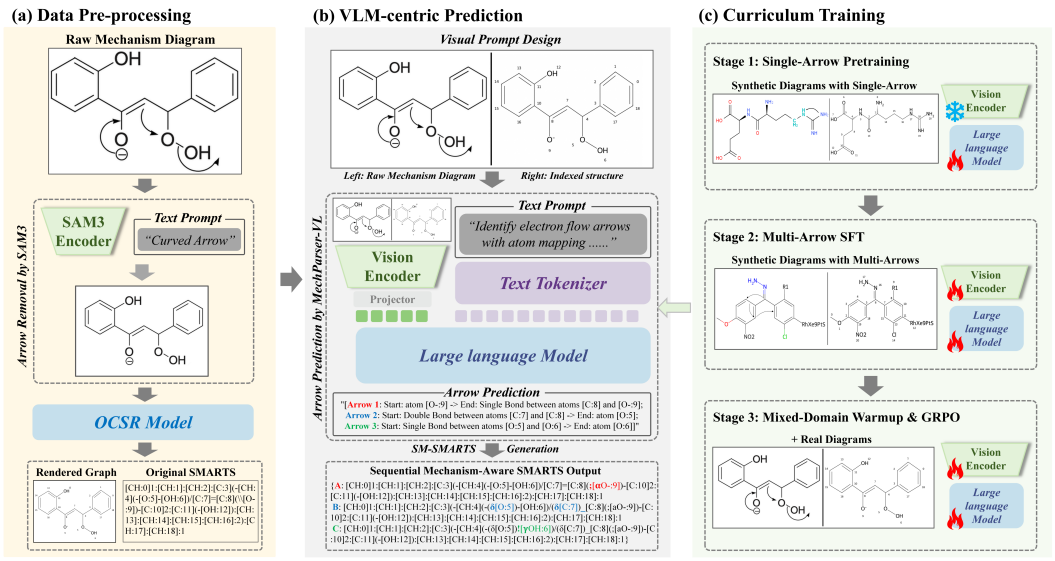

Figure 2: **MechParser pipeline and training curriculum.** MechParser first removes arrows for OCSR while preserving the original arrow-bearing diagram for reasoning: (a) SAM3 cleans the diagram and MolNexTR recovers an atom-indexed molecular scaffold; (b) this scaffold is rendered as the right panel of the visual prompt, paired with the original diagram on the left, so that MechParser-VL can predict the ordered arrow list that is merged with the SMARTS into stepwise SM-SMARTS; (c) we train MechParser-VL with a curriculum that progresses from simple synthetic supervision to real-diagram adaptation and GRPO refinement.

Table 1: SM-SMARTS extension symbols. Bond symbols annotate connections between atoms; atom prefixes encode formal charge changes accompanying electron flow.

Category	Symbol	Meaning	Example
Bond	`^`	Change to triple bond	`[C:2]^[C:3]`
	`;`	Change to double bond	`;[α O⁻:5]`
	`_`	Change to single bond	`_[γ O:3]`
	`!`	Break bond	`[C:2]![N:4]`
	δ	Form new single bond between atoms or fragments	`(δ[C:7])/(δ[C:13])`
Atom	α	Valence +1 (electron donor)	`[α O⁻:3]`: $O^- \rightarrow O$
	γ	Valence −1 (electron acceptor)	`[γ O:4]`: $O \rightarrow O^-$

reaction pre-training [65] likewise operates on net transformations. Our work aims to relieve this bottleneck by extracting mechanisms directly from the diagrams chemists already publish.

3 MechParser

3.1 Sequential Mechanism-Aware SMARTS

Encoding a reaction mechanism requires a representation that captures its dynamic, step-by-step nature. Standard SMARTS [59] provides a rich vocabulary for substructure matching but is inherently *static*: it cannot express the dynamic bond transformations and electron redistribution that curved arrows depict. Reaction SMILES (`reactants»products`) captures only the net change, discarding the stepwise mechanistic logic. We address this gap by introducing SM-SMARTS, an extension of the SMARTS grammar with dedicated symbols for bond-level and atom-level state changes triggered by electron flow. Table 1 summarizes the full symbol set.

The symbols in Table 1 fall into two groups. Bond symbols describe how a connection changes during electron flow: `^`/`;`/`_` update bond order, `!` marks bond cleavage, and δ marks a newly formed bond between atoms or fragments. Atom prefixes record local state changes accompanying electron movement: α denotes an electron-donating site whose valence increases, whereas γ denotes an electron-accepting site whose valence decreases. Crucially, SM-SMARTS encodes the mechanism as an *ordered sequence* of cumulative modifications rather than a single final state. Given an input

SMARTS S_0 and n curved arrows, we produce a sequence $\{S_1, \ldots, S_n\}$, where each S_i applies the i-th arrow's modifications on top of S_{i-1}. This captures the temporal ordering of electron flow, preserving the causal chain of bond-breaking and forming events. Because every edit is written onto an atom-mapped scaffold, SM-SMARTS is also a natural target for downstream mechanism prediction: on mech-USPTO-31K it matches MechSMILES under plain reactant input and surpasses it when both are grounded on the scaffold (Appendix N). Detailed SM-SMARTS definitions and an illustrative example based on Figure 1 are provided in Appendix A.

3.2 Visual Prompt Design

Interpreting curved arrows requires simultaneously understanding (i) *what* the arrows point to and (ii) *which atoms* are involved by index. Performing both tasks from a single image is challenging, as arrows may occlude atom labels and molecular structure.

We propose a *visual prompt* strategy as illustrated in the top of Figure 2(b) that constructs a side-by-side image: the left panel contains the original mechanism diagram with curved arrows, providing electron flow information, while the right panel shows the same molecule re-drawn by RDKit with explicit atom indices, providing a clean reference for atom identification. This design separates the two subtasks: MechParser-VL reads arrows from the left panel and resolves atom identities from the right panel.

The accompanying text prompt defines a structured entity vocabulary: atoms are denoted `atom [Element:ID]` (*e.g.* `atom [C:5]`) and bonds are denoted `<Type> Bond between atoms [E1:ID1] and [E2:ID2]`. MechParser-VL outputs a structured prediction: `[Arrow 1: Start: <entity> -> End: <entity>; Arrow 2: ...]`, where arrows are numbered in the order of electron flow.

3.3 Full MechParser Pipeline

The full MechParser pipeline converts a raw mechanism diagram into a stepwise SM-SMARTS sequence by combining arrow cleanup, molecular scaffold recognition, visual arrow reasoning, and deterministic string parsing. Figure 2(a) and (b) summarize this inference process. Given a raw mechanism diagram, MechParser first queries SAM3 [7] with the text prompt "curved arrow" to segment and remove every curved arrow from the image, replacing the detected regions with white fill. MolNexTR [11] then runs OCSR on the cleaned canvas and returns the SMILES together with atom coordinates, symbols, and bonds, which we serialize into an atom-mapped input SMARTS. The visual prompt needs atom coordinates, which graph-based recognisers provide, and among them MolNexTR leads MolScribe on every standard OCSR benchmark; the interface is model-agnostic (Appendix M). In parallel, the original arrow-bearing diagram is preserved for arrow interpretation. MechParser then constructs the visual prompt, feeds it together with a text prompt to MechParser-VL, and obtains an ordered arrow list. Finally, a deterministic SM-SMARTS bidirectional parser (details described in Appendix B) merges the predicted arrow list with the input SMARTS to generate the stepwise SM-SMARTS output. The separation between arrow removal and arrow prediction is deliberate: curved arrows interfere with molecular structure recognition, so we remove them for OCSR but preserve them for MechParser-VL to interpret.

3.4 MechParser-VL

We build MechParser-VL by adapting Qwen3-VL-4B-Instruct [49] via LoRA [27] to preserve pretrained representations while learning the mechanism-specific output format. MechParser-VL consumes the visual prompt and the structured text prompt, then emits an ordered arrow list in the form `[Arrow 1: Start: <entity> -> End: <entity>; Arrow 2: ...]`. Thus, MechParser-VL is responsible for the visual arrow-reasoning component of MechParser, while OCSR and SM-SMARTS generation remain separate deterministic or externally provided stages. The vision tower is selectively frozen or unfrozen depending on the curriculum stage (Section 3.5).

3.5 Curriculum Training

We train MechParser-VL under a four-stage curriculum [5] (Figure 2(c)) that progressively increases task complexity and bridges the gap between abundant synthetic and scarce real data. Throughout all stages we apply visual data augmentation to the input diagrams, broadening the appearance distribution beyond the rendered synthetic style and improving robustness to real-world drawing

variations. **Stage 1** pretrains the LoRA adapter on 200K single-arrow synthetic diagrams with the vision tower frozen; this establishes the output format, entity vocabulary, and basic visual-textual alignment without disturbing pretrained visual features, since single-arrow examples provide unambiguous start–end supervision. **Stage 2** unfreezes the vision tower and continues supervised fine-tuning (SFT) on 100K multi-arrow (2–3 arrows) synthetic samples, introducing the core challenge of sequential arrow reasoning, where the model must identify multiple arrows, infer their ordering, and emit a chain-coherent output. **Stage 3a** continues SFT on a mixed set of 3,000 synthetic and 620 real diagrams to close the synthetic-to-real domain gap; the synthetic portion acts as regularization against overfitting to the small real set, while the real portion exposes the model to authentic rendering styles, hand-drawn artifacts, and greater chemical complexity. **Stage 3b** applies Group Relative Policy Optimization (GRPO) [56] on the remaining 420 real samples to push beyond the SFT ceiling, under a chemistry-aware multi-tier reward that combines a chemistry-accuracy term and a format-compliance term to address the reward sparsity of binary exact-match signals. Full reward definitions and training details are provided in Appendices C and D, respectively.

4 Dataset Construction

Training MechParser-VL requires large-scale arrow-annotated mechanism diagrams, yet no such dataset exists. To address this, we construct two complementary data resources: a fully automatic synthetic data generator that produces 303,000 diverse training samples at zero annotation cost (§4.1), and a curated real-world dataset of 1,191 expert-annotated diagrams for real-world domain adaptation and evaluation (§4.2).

4.1 Geometry-Aware Synthetic Data Generation

Given a SMILES string from PubChem [33], we render two images: a molecule image via Indigo [18] with stochastic augmentation (rotation, font variation, bond width) and an atom-indexed reference via RDKit [51] with explicit indices; together these form the left and right panels of the visual prompt (Section 3.2). For each molecule we then generate 1–3 curved arrows sequentially, subject to a set of chemical and geometric constraints that collectively yield arrows resembling those drawn by human chemists. The start and end entities of every arrow must be within 1–3 bonds in the molecular graph, reflecting the locality of electron transfer in real mechanisms. For arrows beyond the first, the start entity is chosen with 50% probability from the previous arrow's end entity or an adjacent bond or atom thereof, producing chained electron flow where one arrow naturally leads into the next. Both atoms and bonds can serve as start or end points, covering the two fundamental types of electron movement: lone-pair donation (from an atom) and π-bond donation (from a bond).

Each candidate arrow is rendered as a quadratic Bézier curve whose control point is biased toward the molecular center of mass, mimicking the natural tendency of hand-drawn arrows to curve inward toward the reaction center. The curve is then accepted only if it passes three geometric collision checks: no curve point lies within a distance threshold of non-involved atoms or bonds, no curve segment intersects non-involved bond segments, and no intersection with previously drawn arrows. To increase visual diversity, we randomly vary line width, arrowhead size, curvature parameters, rendering scale, and background shade; the ground-truth annotation is generated simultaneously from the structured entity format and the known atom indices. Full details of the generator are provided in Appendix E. This pipeline produces 303,000 training samples from PubChem molecules with zero manual annotation cost: 200,000 single-arrow diagrams are used for Stage 1 pretraining, 100,000 multi-arrow diagrams (2–3 arrows each) for Stage 2 SFT, and an additional 3,000 multi-arrow samples for the Stage 3a mixed-domain warmup (Section 3.5).

4.2 Real-World Dataset Construction

To evaluate MechParser on authentic data and bridge the synthetic-to-real domain gap, we curate a dedicated dataset of reaction mechanism diagrams from real chemical literature, comprising 1,191 expert-annotated diagrams in total. We extract mechanism diagrams from multiple canonical mechanism and name-reaction resources [37, 14, 35, 60]. We focus on figures that contain at least one curved electron-pushing arrow with clearly identifiable start and end entities; the selected diagrams cover diverse mechanism types including nucleophilic additions, electrophilic aromatic substitutions, eliminations, rearrangements, and pericyclic reactions. Annotations are produced by two PhD students in chemistry with prior experience in OCSR annotation, via a two-stage human-algorithm

workflow. In the first stage, each annotator only labels the *arrow movements*: they identify every curved arrow in a diagram and record its start and end entities (*e.g.* `atom [O-:4]` or `Double Bond between atoms [C:0] and [C:2]`) following our structured entity format, and every diagram is independently cross-checked by the other annotator, with disagreements resolved through joint discussion. In the second stage, the annotated arrow list is fed together with the MolNexTR-derived atom-mapped SMARTS into our SM-SMARTS bidirectional parser to generate the stepwise SM-SMARTS ground truth. The two annotators then jointly verify both artifacts for every sample (the OCSR-derived input SMARTS and the generated SM-SMARTS output). Any case exhibiting OCSR errors or downstream inconsistencies is sent back for correction, only after which the sample enters the final dataset. Of the 1,191 annotated diagrams, 620 are used for Stage 3a mixed-domain warmup, 420 for Stage 3b GRPO, and 151 diagrams (containing 361 annotated arrows) form our *Real-Eval* set, reserved exclusively for final testing and never seen during any training stage. Real-Eval is drawn at the diagram level and stratified by source, and Appendix L evaluates transfer to an entirely held-out source. The annotations and split files are released together with the code. Diagrams span 1–5 arrows per image with an average of 2.4 arrows, covering a broad range of rendering styles, image resolutions, and chemical complexity levels.

5 Experiments

In this section, we evaluate MechParser on both synthetic and real-world mechanism benchmarks, compare MechParser-VL against a broad set of proprietary and open-source VLMs, study the contribution of each design component through ablations, and analyze the dominant error modes.

5.1 Evaluation Benchmarks and Metrics

Evaluation benchmarks. We evaluate on two benchmarks covering in-domain and real-world scenarios. Synthetic-Eval contains 10,000 samples produced by our synthetic data generation pipeline (5000 single-arrow and 5000 multi-arrow with 2–3 arrows). Real-Eval consists of 151 expert-annotated mechanism diagrams, covering diverse reaction types and drawing styles, and serves as the primary benchmark reflecting real-world deployment.

Metrics. Arrow-level Precision, Recall, and F1 measure individual curved arrows with bond-order-invariant matching, and SM-SMARTS-level Precision, Recall, and F1 count a cumulative step as correct when all newly introduced extension symbols match in type and position. At the sample level, the ground truth is a partial order over arrows induced by electron-flow chaining, since a diagram does not always fix a single linear order. Our primary metric, *equivalence-aware Exact Match* (EA-EM), accepts any linear extension of this partial order: independent branches may appear in any relative order and a directed cycle may start at any of its arrows, while arrow identity, direction and count must match exactly. Of the 151 Real-Eval diagrams, 105 are total orders, 30 form a directed cycle, and 16 contain independent branches. We also report the stricter *sequence Exact Match* (Seq-EM), which requires the stored serialisation itself; under the oracle scaffold MechParser-VL reaches 80.1% EA-EM and 70.2% Seq-EM, the gap coming from 15 predictions that recover the complete mechanism in a different valid order (Appendix J). We report arrow-level metrics for the VLM comparison, SM-SMARTS-level metrics for the end-to-end pipeline, and EA-EM in both settings.

5.2 Main Results

Comparison with VLMs. Table 2 compares MechParser-VL against proprietary and open-source VLMs on Synthetic-Eval and Real-Eval. All baselines are evaluated zero-shot under the same visual prompt and use the oracle molecular scaffold, so that differences reflect arrow-reasoning ability rather than OCSR quality. Among proprietary models, only Gemini 2.5 Pro performs at a usable level, achieving 51.7% EA-EM and 57.9% F1 on Real-Eval, while GPT-4o, GPT-5, Claude Sonnet 4, and Grok 4 cluster between 17.9% and 24.5% EA-EM. All models score higher on Synthetic-Eval than on Real-Eval because the synthetic engine produces a relatively uniform visual style, whereas literature diagrams exhibit much greater variation in drawing conventions, arrow density, and layout. MechParser-VL reaches 80.1% EA-EM and 88.6% arrow F1 on Real-Eval, substantially above every model in the comparison. Since the baselines are applied zero-shot, Table 3 further compares MechParser-VL with task-adapted baselines trained on the same data. Diagram-level bootstrap intervals (10,000 replicates over the 151 diagrams) give MechParser-VL 88.6 [85.4, 91.8] arrow F1

Table 2: **Comparison with VLMs on Synthetic-Eval and Real-Eval.** Proprietary and open-source baselines are evaluated zero-shot using the same visual prompt and text instructions. All entries use the oracle molecular scaffold (ground-truth atom-indexed reference) to isolate arrow reasoning. Best in **bold**, second best underlined.

	Synthetic-Eval				Real-Eval			
Model	Precision	Recall	F1	EA-EM	Precision	Recall	F1	EA-EM
Proprietary VLMs (zero-shot)								
GPT-4o [45]	26.1	25.2	25.6	27.1	21.1	21.9	21.3	17.9
GPT-5 [46]	39.1	35.9	37.4	32.3	23.8	24.1	23.8	19.9
Claude Sonnet 4 [2]	31.4	33.6	32.5	26.5	31.9	31.0	31.3	24.5
Grok 4 [63]	30.1	32.6	31.3	23.1	20.5	22.7	21.6	18.5
Gemini 2.5 Pro [21]	61.4	66.3	63.8	53.7	57.6	58.4	57.9	51.7
Open-source VLMs (zero-shot)								
Gemma3-27B [22]	22.4	23.9	23.1	17.8	12.5	19.7	15.5	9.3
Qwen3-VL-32B [49]	28.2	26.9	27.5	25.1	10.5	19.4	13.6	7.3
InternVL3-78B [68]	24.1	28.6	26.2	20.8	15.5	21.9	18.3	11.9
GLM-4.6V-106B [24]	35.1	34.3	34.7	29.7	23.0	26.3	24.7	21.2
MechParser-VL (ours, 4B)	**94.5**	**95.1**	**94.8**	**85.2**	**89.5**	**88.1**	**88.6**	**80.1**

Table 3: **Matched task-adapted baselines on Real-Eval and Synthetic-Eval.** All rows share the Qwen3-VL backbone family, LoRA configuration, training pool and splits. "Flat SFT" is single-stage supervised fine-tuning with no curriculum and no reinforcement learning. "Overlay" renders atom indices on the original diagram; "left-right" is our concatenated visual prompt.

			Real-Eval		Synthetic-Eval	
Model / recipe	Visual input	Prompt	F1	EA-EM	F1	EA-EM
4B, flat SFT	Overlay	Minimal	51.6	41.1	60.1	54.8
4B, flat SFT	Left-right	Minimal	62.5	49.0	72.7	61.2
4B, flat SFT	Left-right	Full	71.4	55.6	76.3	65.1
8B, flat SFT	Left-right	Minimal	69.5	57.4	78.2	64.8
MechParser-VL (ours, 4B)	Left-right	Full	**88.6**	**80.1**	**94.8**	**85.2**

and 80.1 [73.8, 86.4] EA-EM against 57.9 [52.8, 63.0] and 51.7 [43.7, 59.7] for Gemini 2.5 Pro, so the intervals do not overlap on either metric (all models in Appendix K).

Matched task-adapted baselines. Table 3 holds the backbone, LoRA configuration, training pool and splits fixed and varies the visual input, the prompt and the training recipe. Single-stage flat SFT with the overlay input reaches 51.6 F1 and 41.1 EA-EM, below Gemini 2.5 Pro. Replacing the overlay with the left-right visual prompt adds 10.9 F1 and 7.9 EA-EM, and the structured prompt a further 8.9 and 6.6, lifting flat SFT above Gemini on both metrics; the input design, rather than fine-tuning alone, closes the gap to the strongest zero-shot model. The staged curriculum with GRPO then adds 17.2 F1 and 24.5 EA-EM. Under the same flat-SFT left-right minimal recipe, a Qwen3-VL-8B backbone gains 7.0 F1 and 8.4 EA-EM over 4B in a single run, comparable to the contribution of the structured prompt.

End-to-end pipeline evaluation. Table 4 evaluates the fully automated pipeline on Real-Eval, where a raw diagram is processed through SAM3 arrow removal, MolNexTR OCSR, visual prompt construction, arrow prediction, and SM-SMARTS generation in sequence. Compared to the oracle scaffold upper bound, the full pipeline drops by only 3.9% in EA-EM and 4.1% in SM-SMARTS F1, indicating that MechParser preserves most of the oracle scaffold performance in the fully automated setting. The oracle arrow removal row is only 0.6% above the full pipeline in EA-EM, so SAM3 arrow removal introduces only minor degradation relative to manually cleaned diagrams. The failure attribution therefore separates upstream preprocessing and OCSR errors from the model-side residual error: OCSR errors corrupt the SMARTS supplied to MechParser-VL and propagate through every downstream stage, while the remaining model-side failures arise from arrow prediction itself.

Qualitative comparison. Figure 3 shows side-by-side outputs from the representative baselines (GLM-4.6V, GPT-5, Gemini 2.5 Pro) and MechParser-VL on the same Real-Eval diagrams, illustrating how the metric gaps in Table 2 manifest qualitatively. The baselines exhibit three recurring failure modes that aggregate metrics obscure: GLM-4.6V hallucinates non-existent bond entities and

Table 4: **End-to-end pipeline evaluation on Real-Eval.** "Oracle scaffold" uses the ground-truth atom-indexed reference and bypasses OCSR, isolating MechParser-VL's arrow reasoning. "Oracle arrow removal" replaces SAM3 with manually cleaned diagrams but still runs MolNexTR and MechParser-VL, isolating whether automatic arrow removal introduces downstream errors. "Full pipeline" runs the complete automated stack.

Configuration	OCSR Accuracy	SM-SMARTS F1	EA-EM
Oracle scaffold	**100.0**	**88.6**	**80.1**
Oracle arrow removal + MolNexTR	94.0	85.0	76.8
Full MechParser pipeline	93.4	84.5	76.2
Failure attribution on Real-Eval (% of samples)			
Arrow removal errors		0.6	
OCSR errors		3.3	
Arrow prediction errors		19.9	
SM-SMARTS generation errors		0	

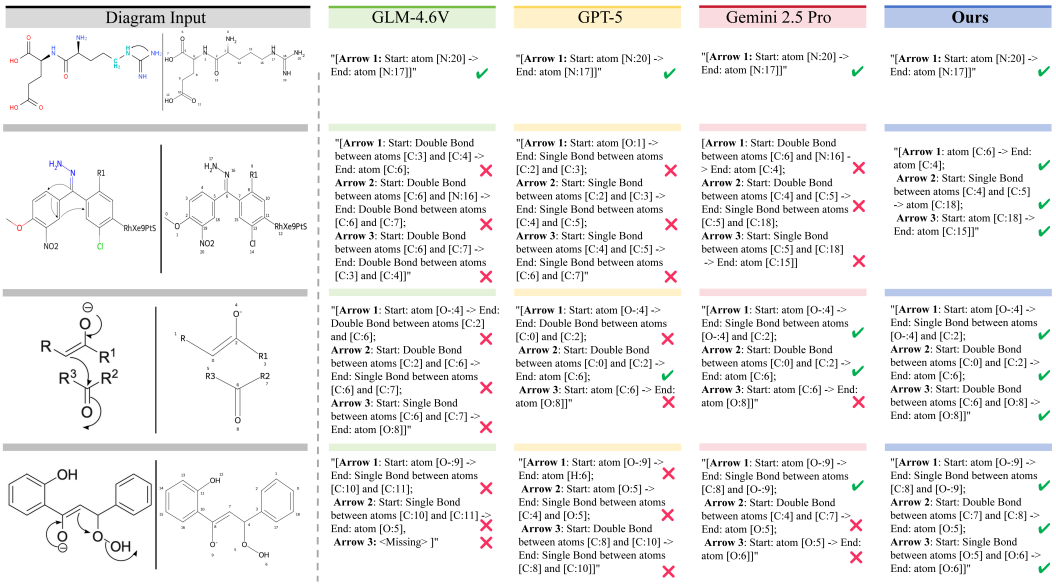

Figure 3: **Qualitative comparison with state-of-the-art VLMs.** Mechanism predictions on representative Real-Eval diagrams from GLM-4.6V, GPT-5, Gemini 2.5 Pro, and MechParser-VL.

truncates the arrow list; GPT-5 suffers from severe global spatial misalignments, grounding arrows to distant, unrelated regions; and Gemini 2.5 Pro struggles with local topological chaining, frequently misidentifying the relay entities between consecutive steps. MechParser-VL produces fully correct mechanisms across these examples, indicating that the gap reflects systematic spatial-grounding errors rather than minor noise. A detailed analysis of MechParser-VL's own residual failures is provided in Appendix H.

5.3 Ablation Studies

We study the contribution of three key design choices on Real-Eval (Table 5) and the importance of arrow removal as a pre-processing step for OCSR (Table 6). Qualitative examples for both ablations are deferred to Appendix I.

Effect of each training component. Table 5a adopts a removal-form layout in which each row removes one component from the full MechParser-VL, so that a larger drop indicates a more important component. The largest drop comes from removing visual data augmentation, with EA-EM falling from 80.1% to 61.6% and arrow F1 from 88.6% to 70.9%, identifying augmentation as the most important factor for cross-domain robustness. The two staged supervision components also contribute substantially: removing the multi-arrow SFT (Stage 2) reduces EA-EM to 64.9%, slightly more than removing the single-arrow pretraining (Stage 1) at 66.2%, reflecting that most literature mechanisms involve two or three coupled arrows. Removing the mixed-domain warmup (Stage 3a) lowers EA-EM to 72.2%, and removing GRPO (Stage 3b) returns the model to its SFT ceiling at 74.2% EA-EM.

Table 5: **Ablation studies on Real-Eval.** (a) Removal-form ablation: starting from full MechParser-VL, we remove one component at a time to isolate its contribution; "−" before a row indicates the corresponding component is removed. (b) Visual prompt design: left-right concatenation vs. overlaying atom indices on a single panel.

Configuration	Arrow F1	EA-EM
(a) Removal-form ablation of training components		
MechParser-VL (full)	**88.6**	**80.1**
− Stage 3b (GRPO)	82.8	74.2
− Stage 3a (mixed warmup)	80.1	72.2
− Stage 2 (multi-arrow SFT)	76.2	64.9
− Stage 1 (single-arrow pretrain)	79.5	66.2
− data augmentation	70.9	61.6
(b) Visual prompt design		
Atom indices overlaid on original diagram	84.2	75.5
Left-right concatenation	**88.6**	**80.1**

Table 6: **Effect of arrow removal before OCSR on downstream prediction.** For both Synthetic-Eval and Real-Eval we report OCSR Accuracy (OCSR Acc) of the molecular structure recognition step together with the resulting end-to-end SM-SMARTS F1 (F1) and EA-EM (EM).

	Synthetic-Eval			Real-Eval		
Configuration	OCSR Acc	F1	EM	OCSR Acc	F1	EM
OCSR on raw diagram	78.7	65.1	51.3	57.0	42.1	33.1
OCSR after arrow removal	**97.1**	**92.4**	**83.0**	**93.4**	**84.5**	**76.2**

Overall, data and supervision design dominate absolute performance, while reinforcement learning still delivers a consistent final gain.

Visual prompt design. A natural alternative to our left-right concatenation is to overlay atom indices directly on the original diagram. Overlaying indices creates visible clutter where they collide with arrows and atom labels near ring junctions, whereas the concatenated layout isolates electron-flow cues from atom identities. Table 5b confirms a measurable drop in this case, from 80.1% to 75.5% EA-EM and 88.6% to 84.2% arrow F1, showing that explicit spatial separation of the two subtasks yields cleaner visual signals. Across five shared seeds, this contrast and the GRPO gain, the two smallest margins in Table 5, are positive in every seed with paired 95% intervals excluding zero (4.90 and 5.43 EA-EM on average; Appendix K); the remaining single-run effects (7.9–18.5 EA-EM) are 4–10$\times$ the five-seed standard deviation of 1.79.

Arrow removal before OCSR. Table 6 shows that arrow removal yields substantial gains on both benchmarks, most dramatically on Real-Eval where OCSR Accuracy rises from 57.0% to 93.4%. The downstream effect is amplified further: SM-SMARTS F1 climbs from 42.1% to 84.5% and EA-EM from 33.1% to 76.2%, confirming that an OCSR failure is rarely local but instead disables every subsequent stage of the pipeline. Mis-recognized arrows either produce an *Invalid* SMILES that blocks visual prompt construction entirely, or corrupt the input SMARTS so that downstream prediction drifts away from the ground truth; SAM3 cleanup removes both failure modes.

6 Conclusion

This paper presents MechParser, a vision-language framework that parses chemical reaction mechanism diagrams into SM-SMARTS, a sequential extension of SMARTS that encodes electron flow, bond transformations, and atom state changes as a machine-readable string. Around this representation we build a geometry-aware synthetic data pipeline and an expert-annotated real-world dataset, and train MechParser-VL with a left-right visual prompt that separates arrow reasoning from atom identification and a four-stage curriculum; controlled comparisons attribute its margin over task-adapted baselines to these two design choices. MechParser provides the diagram-parsing stage of literature-scale mechanism extraction; combined with figure detection and cross-panel atom mapping, it opens the way to large-scale mechanism databases, mechanism prediction models trained on real data, and mechanism-aware retrosynthesis. Limitations and future directions are discussed in Appendix O.

Acknowledgments

We thank the Information Technology Services Center (ITSC) of HKUST for providing the HPC4 and SuperPod clusters as computational resources.

Funding. We acknowledge financial support from the HKUST Start-up Fund (project no. R9251).

Competing interests. The authors declare no competing interests.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Anthropic. Claude 4 system card. `https://www.anthropic.com/claude-4-system-card`, 2025.

[3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

[4] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, pages 41–48, 2009.

[6] John Bradshaw, Matt J Kusner, Brooks Paige, Marwin H S Segler, and José Miguel Hernández-Lobato. A generative model for electron paths. In *International Conference on Learning Representations (ICLR)*, 2019.

[7] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, et al. SAM 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

[8] Binghong Chen, Chengtao Li, Hanjun Dai, and Le Song. Retro*: Learning retrosynthetic planning with neural guided A* search. In *International Conference on Machine Learning*, pages 1608–1616, 2020.

[9] Shuan Chen, Ramil Babazade, Taewan Kim, Sunkyu Han, and Yousung Jung. A large-scale reaction dataset of mechanistic pathways of organic reactions. *Scientific Data*, 11(1):863, 2024. doi: 10.1038/s41597-024-03709-y.

[10] Shuan Chen, Kye Sung Park, Taewan Kim, Sunkyu Han, and Yousung Jung. Predicting chemical reaction outcomes based on electron movements using machine learning. *arXiv preprint arXiv:2503.10197*, 2025.

[11] Yufan Chen, Ching Ting Leung, Yong Huang, Jianwei Sun, Hao Chen, and Hanyu Gao. MolNexTR: A generalized deep learning model for molecular image recognition. *Journal of Cheminformatics*, 16(1):141, 2024. doi: 10.1186/s13321-024-00926-w.

[12] Yufan Chen, Ching Ting Leung, Jianwei Sun, Yong Huang, Linyan Li, Hao Chen, and Hanyu Gao. Towards large-scale chemical reaction image parsing via a multimodal large language model. *Chemical Science*, 16(45):21464–21474, 2025. doi: 10.1039/D5SC04173B.

[13] Yufan Chen, Ching Ting Leung, Bowen Yu, Jianwei Sun, Yong Huang, Linyan Li, Hao Chen, and Hanyu Gao. A multi-agent system enables versatile information extraction from the chemical literature. *arXiv preprint arXiv:2507.20230*, 2025.

[14] Jonathan Clayden, Nick Greeves, and Stuart Warren. *Organic Chemistry*. Oxford University Press, 2nd edition, 2012.

[15] Connor W Coley, Regina Barzilay, Tommi S Jaakkola, William H Green, and Klavs F Jensen. Prediction of organic reaction outcomes using machine learning. *ACS Central Science*, 3(5): 434–443, 2017. doi: 10.1021/acscentsci.7b00064.

[16] Connor W Coley, Luke Rogers, William H Green, and Klavs F Jensen. Computer-assisted retrosynthesis based on molecular similarity. *ACS Central Science*, 3(12):1237–1245, 2017. doi: 10.1021/acscentsci.7b00355.

[17] Manajit Das, Ajnabiul Hoque, Mayank Baranwal, and Raghavan B Sunoj. DeepMech: A machine learning framework for chemical reaction mechanism prediction. *arXiv preprint arXiv:2509.15872*, 2025.

[18] EPAM Systems. Indigo toolkit. `https://lifescience.opensource.epam.com/indigo/`, 2024.

[19] Xi Fang, Jiankun Wang, Xiaochen Cai, Shangqian Chen, Shuwen Yang, Haoyi Tao, Nan Wang, Lin Yao, Linfeng Zhang, and Guolin Ke. MolParser: End-to-end visual recognition of molecule structures in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 24528–24538, 2025.

[20] Igor V Filippov and Marc C Nicklaus. Optical structure recognition software to recover chemical information: OSRA, an open source solution. *Journal of Chemical Information and Modeling*, 49(3):740–743, 2009. doi: 10.1021/ci800067r.

[21] Gemini Team, Google DeepMind. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[22] Gemma Team, Google DeepMind. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[23] Samuel Genheden, Amol Thakkar, Veronika Chadimová, Jean-Louis Reymond, Ola Engkvist, and Esben Bjerrum. AiZynthFinder: A fast, robust and flexible open-source software for retrosynthetic planning. *Journal of Cheminformatics*, 12(1):70, 2020. doi: 10.1186/s13321-020-00472-1.

[24] GLM-V Team, Zhipu AI & Tsinghua University. GLM-4.5V and GLM-4.1V-Thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025.

[25] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[26] Ajnabiul Hoque, Manajit Das, Mayank Baranwal, and Raghavan B Sunoj. ReactAIvate: A deep learning approach to predicting reaction mechanisms and unmasking reactivity hotspots. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI)*, volume 392, pages 2645–2652, 2024. doi: 10.3233/FAIA240796.

[27] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[28] Wengong Jin, Connor W Coley, Regina Barzilay, and Tommi S Jaakkola. Predicting organic reaction outcomes with Weisfeiler–Lehman network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[29] Joonyoung F Joung, Mun Hong Fong, Jihye Roh, Zhengkai Tu, John Bradshaw, and Connor W Coley. Reproducing reaction mechanisms with machine-learning models trained on a large-scale mechanistic dataset. *Angewandte Chemie International Edition*, page e202411296, 2024. doi: 10.1002/anie.202411296.

[30] Joonyoung F Joung, Mun Hong Fong, Nicholas Casetti, Jordan P Liles, Ne S Dassanayake, and Connor W Coley. Electron flow matching for generative reaction mechanism prediction. *Nature*, 645:115–123, 2025. doi: 10.1038/s41586-025-09426-9.

[31] Matthew A Kayala, Chloé-Agathe Azencott, Jonathan H Chen, and Pierre Baldi. Learning to predict chemical reactions. *Journal of Chemical Information and Modeling*, 51(9):2209–2222, 2011. doi: 10.1021/ci200207y.

[32] Ivan Khokhlov, Lev Krasnov, Maxim V Fedorov, and Sergey Sosnin. Image2smiles: Transformer-based molecular optical recognition engine. *Chemistry–Methods*, 2(1):e202100069, 2022. doi: 10.1002/cmtd.202100069.

[33] Sunghwan Kim, Jie Chen, Tiejun Cheng, et al. PubChem 2023 update. *Nucleic Acids Research*, 51(D1):D1373–D1380, 2023. doi: 10.1093/nar/gkac956.

[34] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alán Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020. doi: 10.1088/2632-2153/aba947.

[35] László Kürti and Barbara Czakó. *Strategic Applications of Named Reactions in Organic Synthesis: Background and Detailed Mechanisms*. Elsevier Academic Press, 2005.

[36] Shi Xuan Leong, Sergio Pablo-García, Brandon Wong, and Alán Aspuru-Guzik. MERMaid: Universal multimodal mining of chemical reactions from PDFs using vision-language models. *ChemRxiv preprint*, 2025. doi: 10.26434/chemrxiv-2025-8z6h2.

[37] Jie Jack Li. *Name Reactions: A Collection of Detailed Mechanisms and Synthetic Applications*. Springer, 5th edition, 2014.

[38] Junxian Li, Di Zhang, Xunzhi Wang, Zeying Hao, Jingdi Lei, Qian Tan, Cai Zhou, Wei Liu, Yaotian Yang, Xinrui Xiong, Weiyun Wang, Zhe Chen, Wenhai Wang, Wei Li, Mao Su, Shufei Zhang, Wanli Ouyang, Yuqiang Li, and Dongzhan Zhou. ChemVLM: Exploring the power of multimodal large language models in chemistry area. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 415–423, 2025.

[39] Bowen Liu, Bharath Ramsundar, Prasad Kawthekar, Jade Shi, Joseph Gomes, Quang Luu Nguyen, Stephen Ho, Jack Sloane, Paul Wender, and Vijay Pande. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS Central Science*, 3(10):1103–1113, 2017. doi: 10.1021/acscentsci.7b00303.

[40] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916, 2023.

[41] Ryan J Miller, Alexander E Dashuta, Brayden Rudisill, David Van Vranken, and Pierre Baldi. Mechanism-aware deep learning for polar reaction prediction. *Journal of the American Chemical Society*, 147(44):41168–41176, 2025. doi: 10.1021/jacs.5c16838.

[42] Lucas Morin, Martin Danelljan, Maria Isabel Agea, Ahmed Nassar, Valery Weber, Ingmar Meijer, Peter Staar, and Fisher Yu. MolGrapher: Graph-based visual recognition of chemical structures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19552–19561, 2023.

[43] Lucas Morin, Valéry Weber, Ahmed Nassar, Gerhard Ingmar Meijer, Luc Van Gool, Yawei Li, and Peter Staar. MarkushGrapher: Joint visual and textual recognition of Markush structures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14505–14515, 2025.

[44] Théo A Neukomm, Zlatko Jončev, and Philippe Schwaller. Teaching language models mechanistic explainability through MechSMILES. *arXiv preprint arXiv:2512.05722*, 2025.

[45] OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[46] OpenAI. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2026.

[47] Yujie Qian, Jiang Guo, Zhengkai Tu, Connor W Coley, and Regina Barzilay. RxnScribe: A sequence generation model for reaction diagram parsing. *Journal of Chemical Information and Modeling*, 63(13):4030–4041, 2023. doi: 10.1021/acs.jcim.3c00439.

[48] Yujie Qian, Jiang Guo, Zhengkai Tu, Zhening Li, Connor W Coley, and Regina Barzilay. MolScribe: Robust molecular structure recognition with image-to-graph generation. *Journal of Chemical Information and Modeling*, 63(7):1925–1934, 2023. doi: 10.1021/acs.jcim.2c01480.

[49] Qwen Team. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[50] Kohulan Rajan, Achim Zielesny, and Christoph Steinbeck. DECIMER: towards deep learning for chemical image recognition. *Journal of Cheminformatics*, 12:65, 2020. doi: 10.1186/s13321-020-00469-w.

[51] RDKit community. RDKit: Open-source cheminformatics software. `https://www.rdkit.org`, 2024.

[52] Nadine Schneider, Daniel M Lowe, Roger A Sayle, and Gregory A Landrum. Development of a novel fingerprint for chemical reactions and its application to large-scale reaction classification and similarity. *Journal of Chemical Information and Modeling*, 55(1):39–53, 2015. doi: 10.1021/ci5006614.

[53] Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS Central Science*, 5(9):1572–1583, 2019. doi: 10.1021/acscentsci.9b00576.

[54] Philippe Schwaller, Benjamin Hoover, Jean-Louis Reymond, Hendrik Strobelt, and Teodoro Laino. Extraction of organic chemistry grammar from unsupervised learning of chemical reactions. *Science Advances*, 7(15):eabe4166, 2021. doi: 10.1126/sciadv.abe4166.

[55] Marwin HS Segler, Mike Preuss, and Mark P Waller. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555(7698):604–610, 2018. doi: 10.1038/nature25978.

[56] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y K Li, Y Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[57] Jiahe Song, Chuang Wang, Bowen Jiang, Yinfan Wang, Hao Zheng, Xingjian Wei, Chengjin Liu, Rui Nie, Junyuan Gao, Jiaxing Sun, Yubin Wang, Zhenhua Huang, Jiang Wu, Qian Yu, and Conghui He. RxnCaption: Reformulating reaction diagram parsing as visual prompt guided captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2511.02384.

[58] Jiahe Song, Chuang Wang, Yinfan Wang, Hao Zheng, Rui Nie, Bowen Jiang, Xingjian Wei, Junyuan Gao, Yubin Wang, Bin Wang, Lijun Wu, Jiang Wu, Qian Yu, and Conghui He. Molecular identifier visual prompt and verifiable reinforcement learning for chemical reaction diagram parsing. *arXiv preprint arXiv:2603.15011*, 2026.

[59] Daylight Chemical Information Systems. Smarts – a language for describing molecular patterns. `https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html`, 2007.

[60] The Society of Synthetic Organic Chemistry, Japan, editor. *Enshu de Manabu Yuki Hanno Kiko: Daigakuin Nyushi kara Saisentan made*. Kagaku-Dojin, Kyoto, Japan, October 2005. ISBN 978-4-7598-1045-5. Commonly known as the "Fukuyama Mechanism Book".

[61] David Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28 (1):31–36, 1988.

[62] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018. doi: 10.1039/C7SC02664A.

[63] xAI. Grok 4 model card. https://data.x.ai/2025-08-20-grok-4-model-card.pdf, 2025.

[64] Jiacheng Xiong, Wei Zhang, Yinquan Wang, Jiatao Huang, Yuqi Shi, Mingyan Xu, Manjia Li, Zunyun Fu, Xiangtai Kong, Yitian Wang, Zhaoping Xiong, and Mingyue Zheng. Bridging chemistry and artificial intelligence by a reaction description language. *Nature Machine Intelligence*, 7(5):782–793, 2025. doi: 10.1038/s42256-025-01032-8.

[65] Li-Cheng Xu, Miao-Jiong Tang, Junyi An, Fenglei Cao, and Yuan Qi. A unified pre-trained deep learning framework for cross-task reaction performance prediction and synthesis planning. *Nature Machine Intelligence*, 7(9):1561–1571, 2025. doi: 10.1038/s42256-025-01098-4.

[66] Zhanpeng Xu, Jianhua Li, Zhaopeng Yang, Shiliang Li, and Honglin Li. SwinOCSR: End-to-end optical chemical structure recognition using a Swin Transformer. *Journal of Cheminformatics*, 14:41, 2022. doi: 10.1186/s13321-022-00624-5.

[67] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, 2019. doi: 10.1021/acs.jcim.9b00237.

[68] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Supplementary Materials for

MechParser: A Vision-Language Framework for Parsing Chemical Reaction Mechanism Diagrams

A SM-SMARTS Detailed Specification

This appendix provides the formal definition of the sequential representation introduced in Section 3.1, followed by symbol semantics, conservation constraints, and an illustrative example.

A.1 Formal Sequential Definition

Given an input SMARTS string S_0 describing the molecule before any electron movement, and an ordered sequence of n curved arrows $\{a_1, a_2, \ldots, a_n\}$, SM-SMARTS produces:

$$S_i = \textsc{Apply}(S_{i-1}, a_i), \quad i = 1, \ldots, n \tag{1}$$

where APPLY modifies the SMARTS string by inserting the appropriate bond and atom symbols (Table 1) at positions corresponding to the atoms involved in arrow a_i. The output is the full sequence $\{S_1, S_2, \ldots, S_n\}$, labeled `A`, `B`, `C`, etc. Each step cumulatively inherits all modifications from previous steps, creating a complete audit trail of the mechanism.

This sequential structure captures the *temporal ordering* of electron flow, which is chemically meaningful: in many mechanisms, the order in which arrows are drawn reflects the causal chain of bond-breaking and bond-forming events. Unlike Reaction SMILES (`reactants»products`), which collapses the entire transformation into a single before-after snapshot, SM-SMARTS preserves every intermediate state.

The following EBNF specifies the extension layer used by SM-SMARTS. The base molecular scaffold remains an atom-mapped SMARTS string, while mechanistic symbols are parsed by our SM-SMARTS parser rather than by native SMARTS toolkits such as RDKit.

```
smsmarts_sequence ::= smsmarts_step+
smsmarts_step     ::= step_label ":" extended_smarts
step_label        ::= "A" | "B" | "C" | ...
extended_smarts   ::= smarts_token+
smarts_token      ::= native_smarts_token
                    | bond_edit
                    | new_bond_marker
bond_edit         ::= "^" | ";" | "_" | "!"
new_bond_marker   ::= "\delta" atom_bracket
atom_bracket      ::= "[" atom_prefix? atom_spec ":" atom_map "]"
atom_prefix       ::= "\alpha" | "\gamma"
atom_spec         ::= element charge? hydrogen_count?
atom_map          ::= integer
```

Operationally, the parser treats `\alpha`, `\gamma`, and `\delta` as mechanism-level annotations attached to native atom-mapped SMARTS tokens. Downstream tools can therefore recover the standard molecular scaffold by stripping or interpreting the extension layer before calling a native SMARTS/SMILES parser.

A.2 Symbol Semantics

SM-SMARTS extends SMARTS with two classes of symbols: bond-level symbols placed on connections and atom-level prefixes placed inside atom brackets. The bond-order symbols `^`, `;`, and `_` indicate that the corresponding connection changes to a triple, double, or single bond, respectively. The cleavage symbol `!` marks a bond that is broken by the arrow step. The formation symbol δ marks atoms that become connected by a newly formed single bond; when the bond forms between two previously separated atoms or fragments, both endpoints receive the δ annotation.

The atom prefixes describe local state changes caused by electron movement. Prefix α marks an electron-donating atom whose valence increases after donating electron density into a bond or adjacent center. Prefix γ marks an electron-accepting atom whose valence decreases after receiving electron density. These atom-level annotations make formal charge and valence changes explicit rather than leaving them implicit in the final molecular graph.

A.3 Conservation Constraints

The extension symbols obey strict pairing rules that reflect electron conservation:

- **One source event and one destination event per arrow:** Every arrow records where electron density leaves and where it arrives. The source side is either a valence increase α on a donating atom, a bond-order reduction `_`, or a cleavage `!`; the destination side is either a valence decrease γ on a receiving atom, a bond-order increase (`;` for double, `^` for triple), or a newly formed bond marked by a δ pair. The two sides are chosen independently, so an arrow may combine α with `;` (e.g., `;[`α`O⁻:5]`), `!` with γ (e.g., `!_[`γ`O:3]`), or `_` with a δ pair.
- **Concerted shifts:** In concerted mechanisms where electrons transfer strictly from bond to bond (e.g., conjugated π-shifts), both events are bond edits, so the step carries no atom-level symbol and is checked by the same rule with $n_\alpha = n_\gamma = 0$.
- **New bond formation:** The δ symbol marks the two endpoints of a bond that did not exist in the preceding step. Both endpoints receive the annotation; the bond may be intramolecular, since the criterion is that the two atoms were not already bonded.

These constraints ensure that for each arrow, the total electron count is preserved: electrons lost from one site are gained at another. The check is formalised in Appendix B.

A.4 Illustrative Example

Consider the intramolecular epoxidation of a chalcone enolate intermediate, corresponding to the reaction mechanism shown in Figure 1. The relevant active portion of the input SMARTS is:

```
...-[CH:4](-[O:5]-[OH:6])=[C:7]-[C:8](-[O-:9])-...
```

Three curved arrows encode the reaction cascade: (A) the enolate oxygen (O^{9-}) donates a lone pair to form a carbonyl, (B) the C^7=C^8 π-electrons attack the peroxide oxygen (O^5), and (C) the peroxide O^5–O^6 bond breaks to expel the hydroxide leaving group. The corresponding SM-SMARTS sequence is:

```
A: ...-[CH:4](-[O:5]-[OH:6])=[C:7]-[C:8](;[αO-:9])-...
B: ...-[CH:4](-(δ[O:5])-[OH:6])(δ[C:7])_[C:8](;[αO-:9])-...
C: ...-[CH:4](-(δ[O:5])![γOH:6])(δ[C:7])_[C:8](;[αO-:9])-...
```

Step A: The enolate oxygen (O^-:9) donates its lone pair, increasing its valence (α), and the C:8–O:9 single bond is upgraded to a double bond (`;`). Step B: The C:7=C:8 double bond is downgraded to a single bond (`_`), and the freed π-electrons form a new intramolecular bond between C:7 and O:5, which is denoted by the δ symbols wrapping both participating atoms. Step C: The peroxide O:5–O:6 single bond is completely cleaved (`!`), and the bonding electrons are transferred to the hydroxide leaving group (O:6), decreasing its valence (γ).

Note how the cumulative sequence elegantly tracks the continuous electron cascade: the initial electron donor (α), the intermediate bond shifts (`;`, `_`, δ), and the ultimate electron sink (γ) are all explicitly recorded, maintaining strict electron conservation throughout the mechanism.

B SM-SMARTS Bidirectional Parser

This appendix details the symbolic parser used by MechParser at the last step of Figure 2(b). The parser is bidirectional. In the *forward* direction, it converts the arrow list predicted by MechParser-VL into the cumulative SM-SMARTS sequence. In the *reverse* direction, it reads a SM-SMARTS sequence, recovers the native atom-mapped SMARTS scaffold together with mechanism-level edits, and reconstructs the corresponding ordered arrow operations. Both directions are deterministic: no learnable parameters are involved beyond what has already been used to produce the arrow list.

Inputs and outputs. Let S_0 denote the atom-mapped input SMARTS emitted by the OCSR stage, and let $A = [(s_1, e_1), \ldots, (s_n, e_n)]$ denote the ordered list of curved arrows predicted by MechParser-VL, where each endpoint s_i or e_i is either an atom `[Element:ID]` or a bond `<Type> Bond between atoms [`E_1`:`ID_1`] and [`E_2`:`ID_2`]`. Forward parsing emits the cumulative sequence $(S_1, S_2, \ldots, S_n)$ defined by Eq. (1) in Appendix A. Reverse parsing takes $(S_1, S_2, \ldots, S_n)$ and emits (S_0, A, E), where E is the ordered list of explicit bond and atom edits encoded by the extension symbols.

Forward symbolic edits. For each arrow a_i, the source entity s_i (electrons *leaving*) and the target entity e_i (electrons *arriving*) determine which symbol is inserted:

- If s_i is a *bond*, it is annotated in-place with `_` (change to a single bond) or `!` (full cleavage); if s_i is an *atom*, it is annotated with α to record its valence increase.
- If e_i is a *bond*, it is annotated with `;` (change to a double bond) or `^` (change to a triple bond); if e_i is an *atom* already connected to s_i, it is annotated with γ to record its valence decrease.
- If both s_i and e_i are atoms that are not already bonded in S_{i-1}, the procedure wraps *both* endpoints with δ to mark the newly formed bond. The test is adjacency, so intramolecular ring closures are marked the same way.

Every edit is applied on top of S_{i-1}, so modifications accumulate: downstream arrows see all prior symbol insertions and act on the already-modified string.

Reverse parsing. Given a SM-SMARTS sequence, the parser first tokenizes each step according to the EBNF in Appendix A. It then strips extension symbols to recover the native atom-mapped SMARTS scaffold and records each removed symbol as a typed edit. Comparing consecutive steps (S_{i-1}, S_i) identifies the new edits introduced at step i: α marks an electron donor atom, γ marks an electron acceptor atom, `_/!` mark bond-order decreases or cleavage events, `;/^` mark bond-order increases, and paired δ markers define a newly formed bond. These edits are then paired by atom-map indices and bond endpoints to reconstruct the ordered arrow operation $a_i = (s_i, e_i)$. If several pairings are possible, the parser chooses the chemically local pairing with the shortest graph distance; unresolved ambiguity is reported as a parse error rather than silently producing a mechanism.

Conservation check. The invariant is asserted per arrow, over the edit set Δ_i introduced at step i. Each arrow contributes exactly one source-side event and exactly one destination-side event, chosen independently. Writing n_α, n_{red}, n_{clv} for the counts of α, `_` and `!`, and n_γ, n_{dbl}, n_{tpl}, n_δ for the counts of γ, `;`, `^` and δ, let

$$N_{\mathrm{src}} = n_\alpha + n_{\mathrm{red}} + n_{\mathrm{clv}}, \qquad N_{\mathrm{dst}} = n_\gamma + n_{\mathrm{dbl}} + n_{\mathrm{tpl}} + n_\delta/2.$$

Step i is accepted if and only if $N_{\mathrm{src}} = 1$, $N_{\mathrm{dst}} = 1$, $n_\delta \in \{0, 2\}$, and, when $n_\delta = 2$, the two δ-marked atoms are distinct and were not already bonded in S_{i-1}. The three steps of the worked example in Appendix A realise three different admissible combinations: step A pairs α with `;`, step B pairs `_` with a δ pair, and step C pairs `!` with γ. Concerted bond-to-bond shifts are covered by the same rule, with both events supplied by bond edits. A step violating any condition is reported as a conservation error and the corresponding S_i is rejected, which serves both as a guard at inference time and as a correctness filter on synthetic training data. Algorithm 1 provides the full pseudocode.

Complexity and determinism. For a mechanism with n arrows, forward parsing performs n string edits, each scoped to a constant-bounded substring around the involved atoms/bonds, while reverse parsing scans the cumulative sequence and diffs adjacent steps. Both directions cost $O(n \cdot L)$ where L is the length of the scaffold string. Because the parser is rule-based and free of sampling, SM-SMARTS post-processing never introduces stochasticity on top of the arrow list emitted by MechParser-VL; any variance observed between runs therefore originates entirely from MechParser-VL's arrow prediction.

C Reward Function Details

C.1 Chemistry-Aware Multi-Tier Accuracy Reward

For each arrow i, we compare the predicted start/end entities against the reference using positional matching (Arrow 1 vs. Arrow 1). Each entity is scored by a tiered function:

$$R_{\text{entity}}(p, r) = \begin{cases} 1.0 & \text{bond-order-invariant exact match} \\ 0.85 & \text{charge-agnostic match (element+ID correct)} \\ 0.4 & \text{bond: atoms correct, bond type wrong} \\ 0.3 & \text{charge-agnostic atoms correct, bond type wrong} \\ 0.25 & \text{atom ID overlap but entity type mismatch} \\ 0.0 & \text{no structural overlap} \end{cases} \tag{2}$$

Algorithm 1 SM-SMARTS Bidirectional Parsing

Require: Mode $m \in \{\text{ENCODE}, \text{DECODE}\}$; input X
Ensure: Cumulative SM-SMARTS sequence or recovered (S_0, A, E)

if $m = \text{ENCODE}$ **then**
  Parse X as $(S_0, A = [(s_1, e_1), \ldots, (s_n, e_n)])$
  for $i = 1$ to n **do**
    $S_i \leftarrow S_{i-1}$
    if s_i is a bond **then**
      Annotate bond s_i with `_` or `!`
    else
      Annotate atom s_i with α
    end if
    if e_i is a bond **then**
      Annotate bond e_i with `;` or `^`
    else if s_i and e_i are already bonded in S_{i-1} **then**
      Annotate atom e_i with γ
    else
      Annotate both endpoint atoms with δ
    end if
    Assert electron-conservation invariants
  end for
  return $(S_1, \ldots, S_n)$
else
  Parse X as SM-SMARTS sequence $(S_1, \ldots, S_n)$
  $S_0 \leftarrow \text{STRIPEXTENSIONS}(S_1)$; $A \leftarrow [\,]$; $E \leftarrow [\,]$
  for $i = 1$ to n **do**
    $\Delta_i \leftarrow \text{DIFFEXTENSIONS}(S_{i-1}, S_i)$
    Record typed edits in Δ_i into E
    Pair donor, acceptor, bond-change, and δ edits by atom-map indices
    Recover arrow operation $a_i = (s_i, e_i)$; append a_i to A
    Assert electron-conservation invariants
  end for
  return (S_0, A, E)
end if

Key design choices: (1) **Bond-order invariance**: for bond entities, atom ordering is canonicalized (*i.e.* `[C:5] and [C:4]` $\equiv$ `[C:4] and [C:5]`), since bond descriptions should be order-independent. (2) **Charge tolerance**: Tier 2 strips charge symbols before comparison (`[N+:3]` $\rightarrow$ `[N:3]`), so a prediction that identifies the correct atom but misassigns its charge receives 0.85 instead of 0. This is a deliberate gradient-shaping choice: under exact matching, "correct atom but wrong charge" would be a hard binary failure that yields no learning signal, whereas Tier 2 turns it into a dense $0.85 \rightarrow 1.0$ improvement signal that the GRPO policy can climb step by step. (3) **Structural partial credit**: Tiers 3–5 reward the model for getting the right atoms even when the bond type or entity type is wrong, providing gradient signal for structural understanding.

The per-arrow score and final accuracy reward are:

$$R_{\text{arrow}_i} = \frac{R_{\text{entity}}(p_i^{\text{start}}, r_i^{\text{start}}) + R_{\text{entity}}(p_i^{\text{end}}, r_i^{\text{end}})}{2} \tag{3}$$

$$R_{\text{acc}} = \text{clamp}\left(\frac{\sum_{i=1}^{N_{\text{ref}}} R_{\text{arrow}_i}}{N_{\text{ref}}} - 0.2 \cdot |N_{\text{pred}} - N_{\text{ref}}|,\ 0,\ 1 \right) \tag{4}$$

where the $0.2 \cdot |\Delta N|$ term penalizes predicting the wrong number of arrows.

C.2 Format Reward

The format reward R_{fmt} encourages well-structured output:

$$R_{\text{fmt}} = \begin{cases} 1.0 & \text{valid bracket structure with parseable arrows} \\ 0.5 & \text{brackets present, arrow pattern found, but parse fails} \\ 0.3 & \text{starts with [, contains "Arrow" keyword} \\ 0.1 & \text{starts with [only} \\ 0.0 & \text{no structural formatting} \end{cases} \tag{5}$$

D Training Details for Each Stage

This appendix details the training configuration of MechParser-VL for each curriculum stage introduced in Section 3.5. All stages adapt Qwen3-VL-4B-Instruct with LoRA [27] (rank 32, $\alpha = 64$, dropout 0.05); the base model weights are kept frozen, and only the LoRA adapters (plus, when unfrozen, the vision tower) are updated. All experiments are performed on 6×NVIDIA RTX 5880 GPUs with DeepSpeed ZeRO-2, a per-device micro-batch of 8, and an effective global batch size of 48. Visual data augmentation (Appendix F) is applied throughout.

Stage 1: Single-Arrow Pretraining. The first stage trains on 200,000 synthetic single-arrow diagrams. The vision tower is *frozen*, so only the language-side LoRA adapter is updated, forcing the model to internalize the entity vocabulary and the structured output format without perturbing pretrained visual features. We use AdamW with a cosine learning-rate schedule (peak learning rate 1×10^{-4} with 3% warmup) and train for 2 epochs. Because each diagram contains exactly one start–end pair, supervision is unambiguous and the model rapidly converges to the `[Arrow 1: Start: ...-> End: ...]` template.

Stage 2: Multi-Arrow SFT. Starting from Stage 1, we further fine-tune on 100,000 synthetic multi-arrow diagrams (2–3 arrows each) with the vision tower *unfrozen*, so that LoRA adapters are inserted in both the vision and language modules. This stage introduces the key challenge of *sequential* arrow prediction and requires the vision encoder to adapt to overlapping curves and dense arrowheads. We use AdamW with peak learning rate 5×10^{-5} for LoRA parameters (and a $2\times$-smaller learning rate for vision-side LoRA), cosine schedule with 3% warmup, and train for 2 epochs. Maximum prompt and completion lengths are set to 4,096 and 512 tokens, respectively.

Stage 3a: Mixed-Domain Warmup. To bridge the synthetic-to-real domain gap, we construct a mixed training set of 3,620 samples: 3,000 multi-arrow synthetic diagrams (drawn disjointly from the Stage 2 pool) and 620 expert-annotated real mechanism diagrams from our real dataset (Section 4.2). The synthetic portion acts as regularization, preventing overfitting to the small real subset, while the real portion exposes the model to authentic drawing styles, image quality, and chemical complexity. Continuing from Stage 2, we use AdamW with a reduced peak learning rate of 2×10^{-5}, cosine schedule with 5% warmup, and train for 3 epochs.

Stage 3b: GRPO with Chemistry-Aware Rewards. The final stage applies Group Relative Policy Optimization (GRPO) [56, 25] on 420 real-only samples to push beyond the SFT ceiling. GRPO draws 6 candidate completions per prompt and updates the policy using their group-relative reward ranking, with a KL-divergence coefficient $\beta_{\text{KL}} = 0.04$ against a frozen reference model initialized from Stage 3a. Sampling uses temperature 1.1, maximum completion length 512 tokens, and maximum prompt length 4,096 tokens. We optimize with AdamW at learning rate 1×10^{-6} for 8 epochs. The total reward combines the multi-tier accuracy reward R_{acc} (Eq. 4) and the format reward R_{fmt} (Eq. 5) with asymmetric weighting $R = 3\, R_{\text{acc}} + R_{\text{fmt}}$, emphasizing chemical correctness over surface-form compliance.

E Synthetic Data Generation Details

Algorithm 2 summarizes the geometry-aware arrow generator used for Stage 1 and Stage 2 synthetic data (Section 4.1). The procedure iterates over n arrows: at each step it samples candidate start/end entities, enforces the topological distance filter, constructs a quadratic Bézier curve with an inward-biased control point, and accepts the curve only if it passes all three collision checks against the molecular skeleton and previously drawn arrows.

Algorithm 2 Geometry-Aware Arrow Generation

Require: Molecule graph $G = (V, E)$, coordinates $\mathbf{c}$, number of arrows n
Ensure: Arrow annotations $\{(s_i, e_i)\}_{i=1}^{n}$, annotated diagram

```
Compute adjacency list and bond segments from G
prev_end ← None; drawn_curves ← {}
for i = 1 to n do
    for trial = 1 to T_max do
        Sample start entity s_i (from prev_end with prob 0.5 if i > 1)
        Sample end entity e_i (atom or bond)
        if topo_dist(s_i, e_i) ∉ [1, 3] then
            continue
        end if
        Compute Bézier curve with inward-biased control point
        if collision_free(curve, G, drawn_curves) then
            Draw curve with arrowhead on diagram
            drawn_curves ← drawn_curves ∪ {curve}
            prev_end ← e_i
            break
        end if
    end for
end for
return Arrow annotations and annotated diagram
```

Collision detection parameters. Point-to-segment distance threshold $d_{\text{bond}} = d_{\text{atom}} = 8$ pixels. Arrow Bézier curves are sampled at 50 equidistant points for collision checking. The offset from start/end atoms along the tangent direction is $0.12\times$ median bond length.

Arrow style randomization. Line width $\in \{1, 2\}$ pixels (weighted equally). Arrowhead length $\in [8, 11]$ pixels. Arrowhead spread angle parameter $\in [0.30, 0.45]$. Control point curvature ratios tried in order: $\{0.40, 0.30, 0.50, 0.25\}$ of start-end distance.

Dataset statistics. Total synthetic diagrams: 303,000 (200,000 single-arrow for Stage 1 + 100,000 multi-arrow for Stage 2 + 3,000 multi-arrow for Stage 3a). Synthetic-Eval set: 10,000 held-out samples (5,000 single-arrow + 5,000 multi-arrow), disjoint from all training subsets. Arrow type distribution (over multi-arrow set): $\sim$40% atom$\rightarrow$atom, $\sim$30% bond$\rightarrow$atom, $\sim$20% atom$\rightarrow$bond, $\sim$10% bond$\rightarrow$bond.

F Visual Data Augmentation

Augmentations are applied independently to each input mechanism diagram at training time and are designed to mimic the photometric and geometric perturbations observed in real literature crops. The parameters are as follows: random rotation $\in [-10^\circ, 10^\circ]$; random scaling $\in [0.9, 1.1]\times$; random affine transform (rotation $\in [-5^\circ, 5^\circ]$, x-shear $\in [-10^\circ, 10^\circ]$, translation $\in [-5\%, +5\%]$ of the image size, probability 0.3); color jitter (brightness ± 0.15, contrast ± 0.15, saturation ± 0.10); Gaussian noise $\sigma \in [0, 0.02]$; random JPEG compression with quality $\in [60, 95]$ (probability 0.3); downscale then bicubic-upscale with scale factor $\in [0.25, 0.50]$ (probability 0.2); RDKit re-rendering of the right reference panel with stochastic atom-label fonts and bond widths (probability 0.5). Augmentations are disabled at evaluation time. Figure 4 visualizes representative outcomes of each augmentation, including the synchronized left–right panel re-rendering that keeps arrow coordinates consistent with the atom-indexed reference.

G Full Inference Prompt

Listing 1 shows the complete text instruction used for both training and inference. The prompt is paired with the left–right concatenated visual prompt image (Section 3.2) as a single multi-modal input, and the expected response is a flat `[Arrow i: Start: ...-> End: ...; ...]` string enumerating every curved arrow in the left panel.

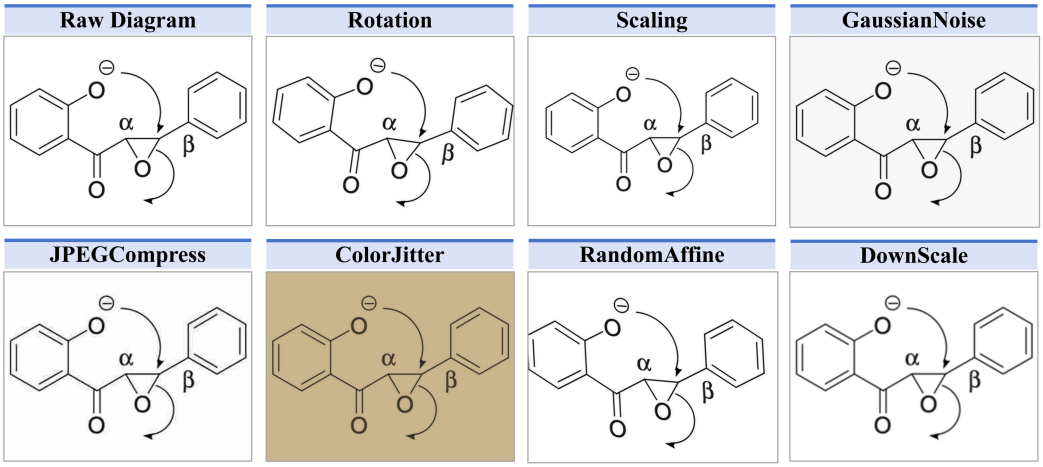

Figure 4: **Visual data augmentation.** Examples of a mechanism diagram rendered under the various geometric and photometric augmentation policies applied during MechParser-VL training.

Listing 1: Full text prompt used by MechParser-VL for arrow extraction.

```
Task: Analyze the provided chemical image.
- The left panel shows a molecule with curved arrows indicating sequential electron flow.
     If the left panel contains no curved arrows or lacks complete arrows, strictly
    output: "Error: No suitable curved arrows found in the source image."
- The right panel shows the same molecule with atom indices for reference.

Your goal is to identify the Start and End points of each curved arrow in the left panel
    by matching them with the atom indices in the right panel. The arrows are numbered
    in the order of electron flow (the end of one arrow connects to the start of the
    next).

Entity naming rules:
- For an atom, use: atom [Element:ID]. Include formal charges if present (e.g., atom [N
    +:5], atom [O-:3]). If an entity is present in the left panel but missing from the
    right panel, use "unknown" as its ID (e.g., atom [-O2CCF3:unknown]).
- For a bond, use: <Type> Bond between atoms [Element:ID] and [Element:ID] (Types: Single
    , Double, Triple, Aromatic)

Strictly output your answer in the following format, with no extra text or explanations:
[Arrow 1: Start: <Start_Entity> -> End: <End_Entity>; Arrow 2: Start: <Start_Entity> ->
    End: <End_Entity>; ...]

Example of expected output:
[Arrow 1: Start: atom [O-:3] -> End: atom [C:4]; Arrow 2: Start: Double Bond between
    atoms [C:4] and [C:5] -> End: atom [N+:6]]
```

Three design aspects are worth emphasizing. First, the *no-arrow sentinel* (“`Error: No suitable curved arrows found in the source image.`”) lets the model explicitly reject inputs that contain no mechanistic arrows, which is essential when the pipeline is applied to real-world figure crops whose classification as mechanism vs. reaction diagram may be imperfect. Second, the *unknown-ID escape hatch* covers cases in which an entity is visible in the left panel (*e.g.* an off-scaffold leaving group labelled as a Lewis structure) but does not appear in the MolNexTR-derived right panel, avoiding hallucinated atom indices. Third, the explicit reminder that “the end of one arrow connects to the start of the next” biases the model toward chemically coherent sequential chains rather than treating arrows as independent predictions.

H Error Analysis

To understand the remaining failure modes of MechParser-VL on Real-Eval, we manually categorize every incorrectly predicted entity from the arrow-level predictions. Table 7 summarizes the distribution across five mutually exclusive error types, and Figure 5 illustrates one representative case for each of the three most frequent categories.

Entity type confusion (30.5%) is the largest category, where the model predicts an atom when the arrow targets a bond or vice versa. These errors tend to occur at arrow endpoints that lie close to, but

Table 7: **The error type distribution.** We categorize the 82 incorrect entities predicted by MechParser-VL on the Real-Eval benchmark.

Error Type	Count	Percentage (%)
Entity type confusion (atom ↔ bond)	25	30.5
Atom element error	23	28.0
Atom index error	19	23.2
Formal charge error	8	9.8
Bond type error	7	8.5
Total incorrect entities	82	100

not exactly on, a bond midpoint, reflecting genuine visual ambiguity in literature drawings. **Atom element errors (28.0%)** suggest limitations in reading labels from the reference panel, most often involving visually similar elements or dense regions where disparate labels (e.g., C vs. N, or H vs. N) are crowded together. **Atom index errors (23.2%)** involve confusing adjacent atoms in the molecular graph, where neighboring atoms of the same element (such as identical carbons or a chain of sulfurs) share almost identical visual context. **Formal charge errors (9.8%)** occur when the model correctly identifies the participating atom but assigns an incorrect formal charge (e.g., predicting a neutral oxygen instead of an oxygen anion), reflecting a failure to notice small superscript charge symbols in the input diagram. **Bond type errors (8.5%)** are the smallest category and generally concern aromatic-vs-double disambiguation inside ring systems, which is inherently ambiguous from a purely visual perspective.

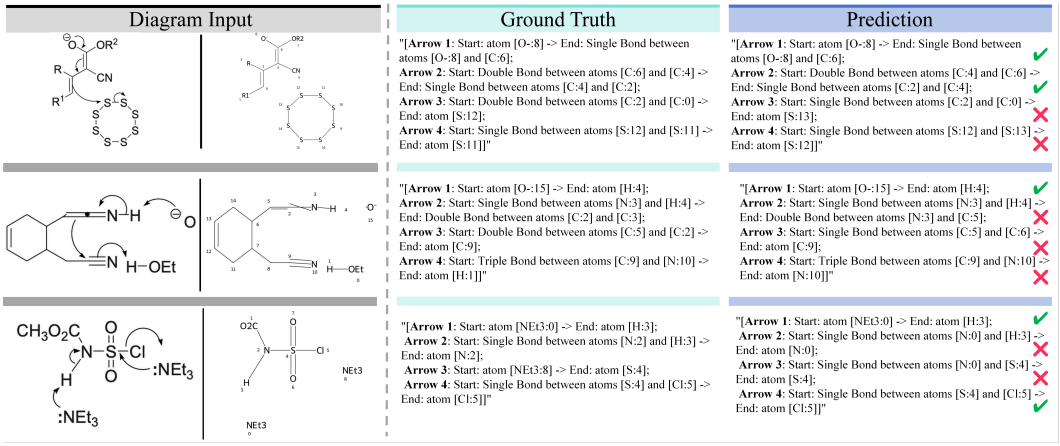

Figure 5: **Representative failure cases.** Examples of incorrect mechanism predictions by MechParser-VL are shown alongside their ground-truth annotations for the three most frequent error categories identified in Table 7.

I Additional Qualitative Results

This appendix provides additional qualitative results supporting the ablation studies in Section 5.3. Figure 6 illustrates the two visual prompt designs compared in Table 5b, and Figure 7 supports the arrow-removal ablation in Table 6.

I.1 Visual Prompt Design Comparison

The three rows in Figure 6 make concrete the visual interference that drives the 4.6-point Exact-Match gap reported in Table 5b. Under the overlay variant, two distinct failure modes emerge. First, as seen in the first row, atom indices stamped directly on the diagram collide with curved-arrow shafts, causing the model to misidentify the start of arrow 2 as an atom (C:5) rather than the correct double bond due to the overlapping clutter. Second, as seen in the third row, overlaying indices relies on the absolute pixel coordinates extracted by the OCSR model; when these coordinates suffer from slight scaling or translation offsets, the overlaid indices undergo a global shift (e.g., labels drifting away from their respective atoms), causing the VLM to systematically misread the atom indices entirely (predicting N:5 instead of N:6, and C:3 instead of C:4). The concatenation variant avoids both failure

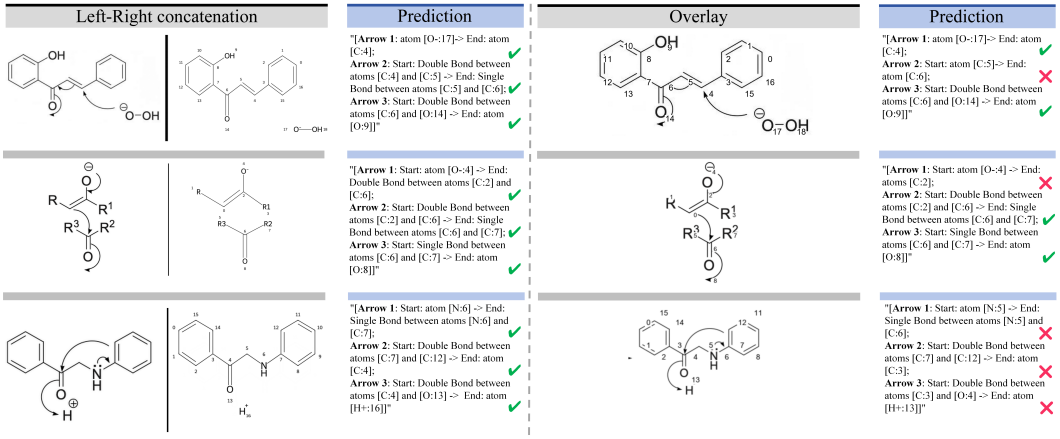

Figure 6: **Two visual prompt designs.** Examples of mechanism predictions are shown for the *left–right concatenation* variant versus the *overlay* variant when processed by MechParser-VL.

modes: it prevents visual occlusion because the original diagram and the reference occupy disjoint regions, and it is immune to OCSR coordinate drift because the right panel is a clean, topology-based re-rendering by RDKit. This pattern confirms that explicit spatial separation yields cleaner visual signals and greater robustness.

I.2 OCSR Accuracy With And Without Arrow Removal

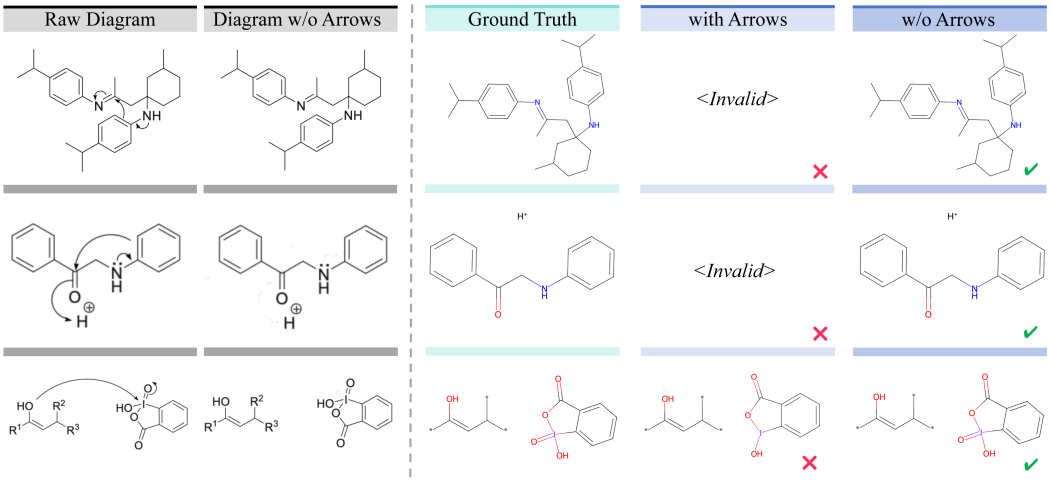

Figure 7: **Effect of arrow removal on OCSR.** Examples of molecular structure predictions are shown for OCSR applied directly to raw arrow-bearing diagrams versus diagrams pre-processed with SAM3 arrow removal, compared against the ground truth.

The three rows in Figure 7 expose two qualitatively distinct OCSR failure modes that drive the large performance gap in Table 6. In the first two rows, MolNexTR mistakenly traces the curved arrows as structural bonds; the resulting valence violations cause the SMILES sanitizer to reject the molecule entirely. Consequently, the pipeline halts because no atom-indexed reference can be generated for the visual prompt, yielding no SM-SMARTS output. The third row illustrates a more subtle failure where MolNexTR returns a syntactically valid SMILES, but omits atoms or bonds that were occluded by the arrows. In this case, the visual prompt still renders and MechParser-VL successfully produces a prediction, but all downstream metrics are inevitably evaluated against a flawed scaffold. After applying SAM3 for arrow removal, both types of errors are eliminated, enabling the OCSR model to recover the correct structure across all three diagrams. Because any scaffold error inherently guarantees zero credit for Exact Match and SM-SMARTS F1, the improvements in OCSR Accuracy are disproportionately amplified downstream. For instance, an 18.4 point accuracy gain on Synthetic-Eval (78.7% to 97.1%) translates into a 31.7 point increase in Exact Match (51.3% to 83.0%), as detailed in Table 6.

J Equivalence-Aware and Sequence Exact Match

Section 5.1 defines EA-EM as acceptance of any linear extension of the ground-truth partial order, and Seq-EM as the stricter variant that additionally requires the stored serialisation. This appendix gives the equivalence classes and the resulting breakdown.

Table 8: **Equivalence classes over the 151 Real-Eval diagrams.** Arrow identity, direction and count must always match.

Partial-order shape	Equivalence admitted by EA-EM	Diagrams
Total order (chain)	none; exact positional match	105
Directed cycle	cyclic rotation, never reversal	30
Disconnected / branching	permutation of independent parts	16

Table 9: **EA-EM versus Seq-EM by partial-order shape.**

Real-Eval subset	Diagrams	EA-EM	Seq-EM	Difference
Total order	105	88/105 (83.8%)	88/105 (83.8%)	0.0
Directed cycle	30	22/30 (73.3%)	13/30 (43.3%)	30.0
Disconnected / branching	16	11/16 (68.8%)	5/16 (31.3%)	37.5
All	151	121/151 (80.1%)	106/151 (70.2%)	9.9

The two metrics coincide on all 105 total-order diagrams. The 9.9-point aggregate gap comes from 15 predictions that recover the complete mechanism under a different valid order: 9 start a directed cycle at a different arrow, and 6 order independent branches differently from the stored serialisation. Seq-EM thus additionally measures agreement with a serialisation convention that the diagram itself does not determine.

K Statistical Protocol

Bootstrap intervals. To quantify sampling uncertainty on the 151-diagram Real-Eval, we draw 10,000 bootstrap replicates using the diagram as the resampling unit, which preserves the dependence between arrows belonging to the same diagram, re-aggregate true positives, false positives and false negatives within each replicate, and report the 2.5th and 97.5th percentiles.

Table 10: **Diagram-level bootstrap intervals on Real-Eval.** The lower bound of MechParser-VL exceeds the upper bound of the strongest baseline on both metrics.

Model	Arrow F1 (%)	95% CI	EA-EM (%)	95% CI
GPT-4o	21.3	[17.1, 25.5]	17.9	[11.8, 24.0]
GPT-5	23.8	[19.4, 28.2]	19.9	[13.5, 26.3]
Claude Sonnet 4	31.3	[26.5, 36.1]	24.5	[17.6, 31.4]
Grok 4	21.6	[17.4, 25.8]	18.5	[12.3, 24.7]
Gemini 2.5 Pro	57.9	[52.8, 63.0]	51.7	[43.7, 59.7]
Gemma3-27B	15.5	[11.8, 19.2]	9.3	[4.7, 13.9]
Qwen3-VL-32B	13.6	[10.1, 17.1]	7.3	[3.1, 11.5]
InternVL3-78B	18.3	[14.3, 22.3]	11.9	[6.7, 17.1]
GLM-4.6V-106B	24.7	[20.3, 29.1]	21.2	[14.7, 27.7]
MechParser-VL	**88.6**	[85.4, 91.8]	**80.1**	[73.8, 86.4]

Seed variance. We repeated three configurations across five shared seeds, where a seed fixes LoRA initialisation, data order, augmentation sampling and GRPO rollout sampling; splits and hyperparameters are held constant. Per-seed EA-EM counts out of 151, reported as full / without GRPO / overlay, are 121/112/114, 116/109/112, 122/114/110, 117/108/108, and 121/113/116.

All numbers in the main text use the released checkpoint, seed 1 (121/112/114 of 151); across seeds the full model ranges from 116 to 122. The remaining rows of Table 5a are single runs, with effects (7.9–18.5 EA-EM) of 4–10$\times$ the five-seed standard deviation.

Table 11: **Five-seed variance for the GRPO and visual-prompt ablations.**

Configuration	Arrow F1 (%), mean ± SD	EA-EM (%), mean ± SD
Full, left-right + GRPO	87.82 ± 1.13	79.07 ± 1.79
Without GRPO	82.37 ± 1.47	73.64 ± 1.71
Atom-index overlay + GRPO	83.89 ± 1.21	74.17 ± 2.09

Table 12: **Paired seed contrasts.** All four are positive in every seed with intervals excluding zero.

Paired contrast	Mean gain ± paired SD	95% paired CI	Positive seeds
GRPO gain, EA-EM	5.43 ± 0.55	[4.74, 6.12]	5/5
Left-right vs. overlay, EA-EM	4.90 ± 2.13	[2.26, 7.54]	5/5
GRPO gain, arrow F1	5.45 ± 0.69	[4.60, 6.30]	5/5
Left-right vs. overlay, arrow F1	3.93 ± 1.52	[2.05, 5.82]	5/5

L Leave-One-Source-Out Re-evaluation

Composition of the annotated pool. The 1,191 expert-annotated diagrams come from four printed sources spanning two genres. Real-Eval was drawn at the diagram level with proportional stratification: per-source sampling rates fall within a 12.6–12.9% band, and compendia constitute 62.9% of Real-Eval against 63.1% of the pool. All four sources therefore contribute to both training and Real-Eval, which the leave-one-source-out study below addresses.

Table 13: **Source composition of the annotated corpus.**

Source	Genre	Pool	Real-Eval	Train	Sampling rate
Li, *Name Reactions*	compendium	398	50	348	12.6%
Kürti & Czakó, *Strategic Appl.*	compendium	353	45	308	12.7%
Clayden et al., *Organic Chemistry*	textbook	246	31	215	12.6%
Problem book (Kagaku-Dojin)	worked problems	194	25	169	12.9%
Total		1,191	151	1,040	12.7%

Protocol. For each source we ran a pair of trainings. In the source-isolated arm (S) the test set is 151 diagrams from the held-out source and training uses 793 diagrams from the other three sources only. In the matched control arm (R) the test set is the same 151 diagrams and training uses 793 diagrams drawn from all four. Training size, stage schedule, hyperparameters and seed are identical within a pair, so the only difference is whether the test source appears in training. The figure 793 is the smallest training pool available across the four folds. Stages 1–2 use PubChem molecules with geometrically generated arrows and are source-agnostic, so all eight runs start from the shared Stage-2 checkpoint and retrain only Stages 3a and 3b.

Result. Fold-by-fold paired differences (R − S) are 4.34 ± 2.01 arrow F1, 95% CI $[1.14, 7.55]$, and 3.97 ± 1.43 EA-EM, 95% CI $[1.70, 6.25]$, positive in all four folds. Because S and R share test diagrams, training budget and seed, the difference is attributable to source exposure rather than to a change of test set. The cost is not uniform: the worked-problem collection loses the most, 6.0 EA-EM points, and has the lowest source-isolated score of the four; it is also the most distinct of the sources in drawing convention. The remaining three cluster between 2.6 and 4.0 points without separating by genre. The controls are considerably more stable than the isolated runs, with an F1 standard deviation of 1.5 against 2.9, so once a source is represented in training, performance is largely insensitive to which source is tested.

Comparability. Absolute scores are lower than the 88.6 / 80.1 of Table 2 because both arms train on 793 diagrams instead of 1,040, and each test set removes 151 diagrams of a single source, leaving the control only 33–188 diagrams of the tested source against 169–348 in the main split. The quantity of interest is the within-fold contrast, which holds everything else constant. The study varies the source axis; since the two compendia cover partly the same named reactions, a mechanism-type-disjoint split would complement it.

Table 14: **Leave-one-source-out re-evaluation with a matched control.** EA-EM is reported out of 151 diagrams in every cell.

	Source-isolated (S)				Matched control (R)			
Held-out source	P	R	F1	EA-EM	P	R	F1	EA-EM
Li	78.1	77.0	77.5	72.2	82.0	81.4	81.7	76.2
Kürti & Czakó	79.7	77.9	78.8	71.5	81.2	80.6	80.9	74.2
Clayden	75.1	73.5	74.3	68.9	79.0	77.8	78.4	72.2
Problem book	73.3	71.8	72.5	65.6	80.0	79.1	79.5	71.5
mean $\pm$ SD			75.8 ± 2.9	69.5 ± 3.0			80.1 ± 1.5	73.5 ± 2.1

M Choice of OCSR Backbone

The visual prompt requires a re-rendered panel carrying atom indices, so the OCSR stage must return atoms with coordinates and bonds rather than a string alone. That requirement excludes caption-style models but does not by itself separate MolScribe from MolNexTR. Empirically, MolNexTR is stronger on every standard OCSR benchmark [11]:

Table 15: **OCSR accuracy on standard benchmarks.**

	Indigo	ChemDraw	RDKit	CLEF	UOB	JPO	USPTO	Staker	ACS
MolScribe [48]	97.5	93.8	94.6	88.3	87.9	77.7	92.6	86.9	71.9
MolNexTR [11]	**97.8**	**95.1**	**96.4**	**90.4**	**88.5**	**82.1**	**93.8**	**88.3**	**81.9**

The margin is largest where it matters here: 10.0 points on ACS and 4.4 on JPO, the two benchmarks with the most diverse drawing styles and the heaviest use of abbreviated groups, which is what literature mechanism diagrams look like. The pipeline is not tied to one OCSR model: the interface is SMILES together with atoms and bonds, so MolScribe substitutes directly. Table 6 indicates that the dominant factor is not the choice of OCSR model but whether arrows are removed first, which moves Real-Eval OCSR accuracy from 57.0% to 93.4%, far more than the gap between any two current models.

N SM-SMARTS versus MechSMILES as a Downstream Target

This appendix evaluates SM-SMARTS as a training target for downstream mechanism prediction, the second role named in Section 2. The study uses the public mech-USPTO-31K corpus [9] and isolates the representation from the parsing pipeline.

Conversion and splits. We converted every MechSMILES mechanism in the corpus to SM-SMARTS and retained only those examples whose round trip MechSMILES $\rightarrow$ SM-SMARTS $\rightarrow$ MechSMILES exactly recovered the canonical electron-arrow set, giving 109,196 examples split into 87,297 train, 10,853 validation and 11,046 test. All arms use the same retained examples in the same order.

Arms. Two factors are crossed: the target representation, and whether the input carries a mapped scaffold in addition to the product and reactant SMILES. The scaffold is the reactant SMARTS, that is, the same molecules written with atom maps rather than as SMILES.

Table 16: **Five matched arms.** Arm A reproduces the published MechSMILES setting with its input and target formats unchanged. B and D are minimally mapped SM-SMARTS variants, introduced so that the symbol scheme can be separated from the token cost of mapping every atom: A–B and C–D compare representations at matched mapping density, while D–E isolates mapping density.

Arm	Input	Target	Mapping
A	product + reactants	MechSMILES tuples	minimal
B	product + reactants	SM-SMARTS symbols	minimal
C	+ reactant SMARTS	MechSMILES tuples	minimal
D	+ reactant SMARTS	SM-SMARTS symbols	minimal
E	+ reactant SMARTS	SM-SMARTS symbols	full

Training. All arms share a randomly initialised 17.9M-parameter T5, a 60-token chemical vocabulary, AdamW, effective batch size 256, learning rate 6×10^{-4}, 30 epochs, beam size 5, and three seeds.

Representation-neutral evaluation. String exact match is not comparable across formats, because each arm is asked for a different string. We therefore decode every prediction back into the same canonical set of MechSMILES arrow triples and score that set. The resulting *arrow-set match* ignores arrow order within a step, normalises equivalent ionisation tuple orientations, and counts any undecodable prediction as incorrect.

Table 17: **Results,** mean $\pm$ sample standard deviation over three seeds.

Arm	Arrow-set match (%)	Decodable (%)
A	90.49 ± 1.21	99.29 ± 0.12
B	92.30 ± 2.10	93.74 ± 1.64
C	87.69 ± 1.58	99.18 ± 0.11
D	91.20 ± 3.11	93.31 ± 2.48
E	93.78 ± 1.76	94.60 ± 1.48

Matched contrasts. Under plain product/reactant input at matched mapping density, B–A comes out at -1.70, $+3.61$ and $+3.50$ across the three seeds, a mean of $+1.80 \pm 3.03$, i.e. parity, with one seed reversing the sign. Given the same scaffold, D–C averages $+3.51$. Full versus minimal mapping, E–D, gives $+0.63$, $+5.55$ and $+1.55$, a mean of $+2.58 \pm 2.62$, bought with sequence length: E's target runs $3.24\times$ longer than D's.

Interpretation. The design difference is where the edit is written. SM-SMARTS marks each bond and atom-state change on the atom it applies to, a local edit to a graph already in view, whereas MechSMILES must re-emit a globally consistent index space and then refer into it with detached tuples. When the graph has to be generated rather than supplied, both formats must establish it first and the advantage narrows, which is what the plain-input parity reflects. The two families also fail in different ways: MechSMILES output is almost always decodable, yet only 88–91% of what decodes from A and C carries the correct arrow set, whereas for B, D and E that figure is 98–99% and the residual failures are invalid SM-SMARTS syntax rather than incorrect chemistry. Grammar-constrained decoding is therefore a natural next step for the SM-SMARTS arms.

Scope of this comparison. The scaffold supplied to arms C, D and E is a deterministic re-encoding of the reactant SMILES that fixes an atom indexing without indicating which atoms react, reproducing the grounding condition of MechParser, where OCSR supplies an atom-indexed scaffold before any arrow is predicted. The study uses one template-expanded corpus and one small architecture: it establishes SM-SMARTS as a viable training target, and extending the comparison to more models, more datasets, and corpora extracted by MechParser is future work (Appendix O).

O Limitations and System Roadmap

This appendix discusses the limitations of the present system and outlines the roadmap toward literature-scale extraction.

Supported scope. MechParser handles printed, publication-style mechanism diagrams in which electron flow is drawn with curved arrows over a single connected scheme, at the level of bond-order and formal-charge changes. Cyclic electron flow *within* one diagram is common and supported: 30 of the 151 Real-Eval diagrams contain a directed arrow cycle without a chemically determined starting arrow, evaluated modulo rotation (Section 5.1).

Beyond the current scope. Stereochemical changes such as Walden inversion, reagent and condition annotations, and mechanisms distributed across several panels are not yet modelled. The last case covers both one structure per elementary step and catalytic cycles drawn as separate species around a ring. MechParser currently parses each panel independently into its own SM-SMARTS sequence; linking panels requires matching the product of one panel to the reactant of the next and recognising a ring layout as a single cycle.

Pipeline boundaries. The pipeline starts from a cropped single-scheme diagram. Locating and cropping mechanism figures in PDF pages is a separate component, and end-to-end yield from raw documents is not measured here. The inference prompt defines a no-arrow sentinel (Appendix G), so crops without curved arrows, the typical error of an imperfect figure detector, are rejected with a dedicated error string. Accuracy also depends on the upstream OCSR stage (Table 6).

Training signal and downstream use. The synthetic training arrows are geometrically valid but not chemically validated, so MechParser-VL is trained to transcribe drawn mechanisms rather than to assess their chemistry. Appendix N shows that SM-SMARTS is a viable training target for mechanism prediction on an existing template-expanded corpus; training such a model on MechParser-extracted mechanisms is the next step.

Generalisation. The 1,191 annotated diagrams are drawn from four printed sources, all of which contribute to both training and Real-Eval. The leave-one-source-out re-evaluation of Appendix L shows that holding out an entire source costs 4.34 ± 2.01 arrow F1 and 3.97 ± 1.43 EA-EM, positive in 4/4 folds with paired intervals excluding zero, so cross-source transfer is the principal generalisation limit. A mechanism-type-disjoint split, based on an expert grouping of reactions into mechanistic classes, would complement this source-level analysis.

Residual error modes. Our error analysis indicates that entity-type confusion and atom identification account for the majority of remaining failures, which suggests that stronger visual grounding for arrow endpoints and higher-resolution reference panels are the most direct routes to further gains. The expert-annotated corpus of 1,191 diagrams is the largest of its kind, but it remains small in absolute terms; expanding it would improve both the reinforcement-learning signal and the statistical resolution of the ablations.

Components of a literature-scale extraction system. Extraction at the scale of the published literature requires four components. First, a detector that locates and crops mechanism figures within PDF pages. Second, molecule detection together with OCSR, recovering every molecule in a cropped scheme as an atom-mapped SMARTS. Third, an atom-mapping step that reconciles indices across consecutive panels so that one atom keeps a single identity throughout a multi-panel mechanism; RxnMapper [54] is our intended starting point. Fourth, MechParser itself, turning each arrow-bearing panel into SM-SMARTS that the shared indexing can stitch into one mechanism.

This paper delivers the fourth component, where the arrow semantics live; the first and third turn a per-panel parser into a whole-figure one and are our planned next steps. Because SM-SMARTS carries atom maps, linking panels reduces to cross-panel atom mapping rather than re-parsing, which makes multi-panel and catalytic-cycle support a direct extension.

Downstream case study. The natural end-to-end demonstration starts from raw PDFs, extracts mechanism figures, builds a corpus with MechParser, and trains a mechanism prediction model on it, compared against the same model trained on template-imputed steps [29]. It requires figure detection, licence-compliant corpus acquisition, expert spot-checking at scale, and a full downstream training and evaluation cycle, and it is the central goal of our future work.

P Societal Impacts

MechParser is intended to make mechanistic knowledge in the chemical literature more accessible to machine learning systems and practicing chemists. Its primary positive impacts are scientific: constructing large-scale mechanism databases, supporting mechanism prediction models trained on authentic literature data, and improving access to mechanistic reasoning for synthesis planning, education, and chemical knowledge discovery. Because the system operates on published chemical diagrams and does not involve human subjects or personal data, we do not foresee privacy or fairness risks. Potential misuse is limited by the domain-specific nature of the task, but as with other tools that accelerate chemical knowledge extraction, downstream applications should follow existing norms for responsible chemical research and safety review.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the contributions made in the paper and distinguish the full MechParser framework from the core MechParser-VL component.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Conclusion summarises the scope of the system, and Appendix O discusses the limitations in full: supported and unsupported diagram types, pipeline boundaries, the training signal, cross-source generalisation (with the leave-one-source-out result of Appendix L), residual error modes, and the components of a literature-scale system that remain to be built.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper is primarily empirical and does not include formal theoretical results or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe the pipeline, model components, datasets, prompts, training curriculum, rewards, and evaluation protocol in Sections 3–5 and the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The full implementation (end-to-end inference pipeline, bidirectional SMSMARTS parser with regression tests, synthetic data generators, four-stage training scripts with the hyperparameters of Appendix D, the GRPO reward, and evaluation code reporting EA-EM and Seq-EM) and the trained weights are released at `https://github.com/CYF2000127/MechParser`, together with the expert annotations, split files and leave-one-source-out folds.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Dataset splits, training stages, hyperparameters, reward definitions, and evaluation metrics are described in Sections 4–5 and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix K reports diagram-level bootstrap 95% percentile intervals (10,000 replicates resampling the 151 Real-Eval diagrams) for every model in Table 2, and five-seed means $\pm$ standard deviations with paired 95% intervals for the GRPO and visual-prompt ablations, where a seed fixes LoRA initialisation, data order, augmentation sampling and GRPO rollout sampling. The leave-one-source-out study (Appendix L) reports fold-level means $\pm$ standard deviations and paired intervals, and the representation comparison

(Appendix N) reports means ± sample standard deviations over three seeds. The remaining ablation rows, the flat-SFT baselines and the 8B control are reported as single point estimates.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the GPU type, number of GPUs, and training setup in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses published chemical diagrams and does not involve human subjects, personal data, or high-risk deployment.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss potential positive impacts and limited misuse risks in Appendix P.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released assets are domain-specific mechanism parsing tools and datasets based on published chemical diagrams, and do not pose the high-risk misuse scenarios described in the question.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing assets are cited and used within their terms: Qwen3-VL-4B-Instruct (Apache 2.0), SAM 3 (SAM License), MolNexTR, RDKit (BSD 3-Clause), Indigo (Apache 2.0), PubChem (public NCBI data), and mech-USPTO-31K. Proprietary baselines were queried through their public APIs under the providers' terms. The real-world diagrams are taken from four printed sources [37, 35, 14, 60], which are cited and remain under their publishers' copyright.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: This paper introduces SM-SMARTS, the synthetic mechanism-diagram generation pipeline, and an expert-annotated real-world mechanism dataset. These assets are documented in Sections 3–4 and Appendices A–E; code, weights and annotations are released with documentation at `https://github.com/CYF2000127/MechParser`.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work does not involve crowdsourcing or human-subject research.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human-subject research and therefore does not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The core method fine-tunes MechParser-VL from Qwen3-VL-4B for chemical mechanism interpretation. The model architecture, training procedure, and evaluation are fully described in Sections 3 and 5.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.