From Natural Language to Geometric Diagrams: A Hybrid LLM–Constraint System for Diagram Construction

Philip Todd[1][0000−0002−7118−8755], Colin Logue[1], and Kavi Wilson[1]

Saltire Software, Portland, OR, USA
{philt, colinl, kaviw}@saltire.com
https://www.saltire.com

Abstract. We present a system for generating high-quality geometric diagrams from short natural-language descriptions. A lightweight language-processing layer extracts a compact intermediate representation, and a deterministic constraint-based geometry engine constructs a fully specified diagram. The system resolves underspecification through explicit heuristics and applies a clarity optimizer to avoid degenerate or misleading layouts. This architecture yields diagrams that are reproducible, mathematically coherent, and suitable for downstream tasks such as automated reasoning or interactive editing.

1 Introduction

Since Euclid's *Elements* more than two millennia ago, diagrams have been fundamental to the communication and development of geometry. They provide a bridge between abstract relational descriptions and concrete spatial understanding, and they remain a primary medium in textbooks, research expositions, and problem–solving contexts. At the same time, many geometry problems—particularly those in textbooks, examinations, and computational benchmarks—are presented without accompanying figures, relying instead on concise natural–language descriptions. Recovering the intended configuration from such text is nontrivial: natural language is flexible and often underspecified, whereas geometric diagrams require precision, internal consistency, and explicit relational structure.

We introduce a system that generates high-quality geometric diagrams directly from natural-language descriptions. The system integrates an LLM-based interpreter, a symbolic geometry engine, and a clarity-driven optimizer that refines initial placements to improve visual clarity. This combination enables reliable conversion of informal geometric descriptions into precise, visually coherent diagrams.

2 Related Work

Research on generating diagrams from natural language draws on contributions from multiple communities, particularly diagram synthesis and geometry prob-

lem understanding. Recent work has begun to treat diagram generation as a structured synthesis problem. Wei et al. introduce *GeoLoom* [1], which translates natural language into a formal geometric language and uses a Monte Carlo coordinate search to produce diagrams; a follow-up system, *MagicGeo* [2], optimizes coordinates directly from text prompts. Both approaches operate purely in coordinate space, do not incorporate symbolic geometry, and do not explicitly optimize for diagram clarity. Their associated datasets (GeoNF and MagicGeoBench) also do not appear to be publicly available.

Beyond geometry, text-to-diagram generation has been explored for flowcharts, mind maps, and other structured visualizations. These systems emphasize logical or semantic structure and typically rely on templates or layout heuristics rather than geometric constraint satisfaction, addressing a different class of diagrams.

Dynamic geometry environments such as GeoGebra [3] and constraint-based systems such as Geometry Expressions [4] provide exact constructions from formal specifications, but they require formal input or direct manipulation and do not accept free-form natural language. They also do not optimize diagrams for clarity.

To our knowledge, no prior system combines (1) natural-language interpretation, (2) exact constraint-based geometry, and (3) clarity-driven optimization. Existing systems address at most two of these dimensions: GeoLoom and MagicGeo combine natural language with Monte Carlo–based geometric solvers; dynamic geometry systems provide exact constructions but require formal input; and clarity optimization has not been addressed in the context of diagram generation. A further distinction concerns solver performance: Monte Carlo methods typically incur higher computational cost during constraint satisfaction, limiting their suitability for iterative refinement. In contrast, the fast symbolic solver used in our system makes repeated constraint resolution feasible within a clarity-optimization loop.

3 System Architecture

The system consists of three components: (1) natural-language interpretation, (2) symbolic geometric construction, and (3) clarity optimization. A key design principle is the separation of linguistic interpretation from geometric correctness, and the separation of correctness from visual clarity.

3.1 Natural Language Interpretation

The first component of the system translates the user's natural-language description into a structured intermediate representation (IR) suitable for symbolic geometric construction. This stage resolves linguistic ambiguity, identifies geometric entities, and extracts the relational structure of the described configuration.

The IR is a deterministic, machine-readable specification of geometric objects and constraints. It includes declarations of points, lines, segments, circles, and other primitives, together with relations such as incidence, perpendicularity,

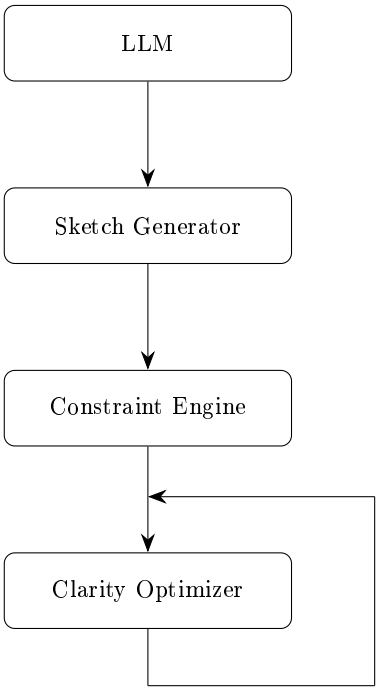

Fig. 1. System architecture showing the flow from language interpretation to sketch generation, constraint solving, and clarity optimization, with a feedback loop from clarity optimization back toward constraint solving.

parallelism, and equality of lengths. The IR is intentionally minimal: it captures only the information required for geometric construction, without committing to any particular coordinate realization.

A large language model (LLM) maps free-form text to the IR. The model is prompted with a schema describing the IR and a small set of illustrative examples. Given an input such as "Construct a triangle ABC with $AB = AC$ and let D be the midpoint of BC," the LLM identifies the geometric primitives (triangle, points A, B, C, midpoint D) and the associated constraints (isosceles condition, midpoint relation), and emits a structured IR encoding these relationships.

As illustrated in Example 4.1, the LLM frequently introduces its own naming conventions for geometric objects, even when the natural-language input does not supply explicit labels. This behaviour reflects induced lexicalization: the model assigns systematic names (for example, for an altitude foot or a median point) to satisfy the structural requirements of the IR. These labels are treated as arbitrary and normalized during IR processing; the solver relies only on the underlying geometric relations, not on the specific names chosen by the model.

The LLM also shows uneven grounding across higher-order geometric terms. Some concepts, such as *orthocenter* or *nine-point circle*, are treated as atomic

and are mapped cleanly into the IR. Others, such as *radical axis*, appear less consistently in the model's training data and are therefore interpreted with mixed granularity: the model may expand them into a sequence of constructions, partially reconstruct their definition, or treat them as atomic depending on context. Our system accepts these variations as produced, and the solver operates directly on the resulting IR. Introducing canonical constructions for such terms is a natural extension for future versions.

Natural language frequently contains implicit or ambiguous references, such as "the foot of the perpendicular," "the intersection of the two lines," or "extend the segment." The LLM resolves these expressions into explicit IR constructs. It also handles anaphora (for example, "this point," "the other side") and ordering expressions (for example, "B between A and C"). When the description is underspecified, the IR captures only the constraints that are explicitly stated, leaving degrees of freedom to be resolved during sketch generation and clarity optimization.

3.2 Constraint Based Geometry Resolution

The second component of the system constructs a mathematically correct geometric model from the intermediate representation (IR). This stage is implemented using the Geometry Expressions engine, which provides exact symbolic constraint solving. The construction process consists of two substeps: generating an initial sketch and resolving the IR against this sketch to obtain a fully determined configuration.

The IR typically underdetermines the geometry: free points may have no specified coordinates, and many constructions admit multiple valid realizations. To provide the symbolic solver with a concrete starting point, the system generates an initial *sketch* that assigns provisional coordinates to all free objects. The sketch also encodes *sidedness* information, such as which side of a line a point should lie on or which branch of an intersection should be selected. These choices determine the topological structure of the configuration and ensure that the solver converges to a realization consistent with the user's intent.

Sketch generation uses lightweight stochastic sampling to place free points in a generic position. These initial placements do not need to be accurate or aesthetically pleasing; they serve only to provide the symbolic solver with a non-degenerate starting configuration. All issues of correctness are handled in the subsequent constraint-solving step.

Given the IR and the sketch, the Geometry Expressions engine resolves all geometric constraints exactly. The solver treats the sketch as an initialization rather than a numerical approximation: it uses the sketch's coordinates and sidedness choices to select a branch of the solution space, then computes the precise positions of all objects using symbolic and algebraic methods. The result is a mathematically correct diagram in which all constraints in the IR are satisfied by construction.

A practical advantage of this approach is solver performance. The symbolic engine resolves typical school-level and contest-level geometric configurations

quickly enough to support repeated calls during clarity optimization, enabling the system to explore multiple realizations of the same IR. This stands in contrast to Monte-Carlo-based approaches, where constraint satisfaction is substantially more expensive and therefore unsuitable for iterative refinement.

The solver is also a deterministic subroutine: given the same IR and sketch, it always produces the same geometric model. It is invoked once to construct the initial diagram and repeatedly during clarity optimization when alternative sketches are explored.

3.3 Clarity Optimization

The final component of the system improves the visual quality of the diagram produced by symbolic construction. While the symbolic solver guarantees mathematical correctness, the resulting diagram may contain undesirable visual artifacts such as points that are too close or too far apart, nearly coincident lines, or accidental symmetries that are not implied by the intermediate representation (IR). The clarity optimization component addresses these issues by searching over alternative initial sketches and selecting the diagram that maximizes a clarity metric.

Geometric descriptions often admit many valid realizations. For example, an isosceles triangle may be tall and narrow or short and wide; a midpoint construction may place the midpoint in a region of the diagram that is visually cluttered; and an intersection point may lie extremely close to another object, creating the appearance of an unintended relationship. These issues arise because the IR specifies relational constraints but does not prescribe a particular layout. The clarity optimization component exploits this freedom to choose a realization that is visually and pedagogically effective.

Starting from the initial sketch generated in the symbolic construction stage, the system explores the space of possible realizations using a hybrid simulated annealing and steepest-descent optimizer. The optimizer perturbs only the *locations* of free points; no explicit sidedness choices are modified. However, because sidedness is implicit in the relative positions of points and lines, sufficiently large perturbations may cause the solver to select a different topological branch. Large perturbations proposed during the annealing phase allow the optimizer to escape local minima and explore qualitatively different configurations, while smaller perturbations refine promising candidates.

Each perturbed sketch is passed to the symbolic solver, which produces a mathematically correct diagram. Because sidedness is inferred from the sketch rather than explicitly encoded, the solver may adopt different valid realizations when perturbations move points across implicit boundaries. The resulting diagram is then evaluated by a clarity metric that penalizes several undesirable visual features: points or intersections that lie uncomfortably close to one another, points that are excessively far from the rest of the configuration, near-equalities of lengths or angles that are not implied by the IR, and incidental alignments in which points appear nearly collinear or nearly concurrent without geometric justification.

The optimizer evaluates the collection of candidate diagrams generated during the annealing and descent phases and selects the one with the highest clarity score. Because the symbolic solver enforces correctness for every candidate, the optimization process operates entirely within the space of valid realizations. The result is a diagram that is both mathematically correct and visually clear, avoiding misleading coincidences and presenting the intended geometric relationships in a clean and interpretable form.

Sketch perturbations are generated stochastically during the annealing phase. When a random seed is fixed, the clarity optimization process is fully deterministic; otherwise, it explores the sketch space stochastically while still converging to a clarity-optimized diagram.

The final output is a fully constrained dynamic geometry model, allowing the diagram to be edited, manipulated, or further explored within its native environment. In addition, the system produces static SVG or PDF renderings for use in documents or other non-interactive settings.

4 Examples

We illustrate the behavior of the system with two examples that show the full sequence from natural language to the final clarity-optimized diagram.

4.1 Example 1: A triangle with an altitude and a median

Natural language input. Draw a triangle along with its altitude and the median through the same vertex.

Intermediate representation.

```
triangle A B C
altitude D A B C
median M1 A B C
```

The results are shown in Figure 2

We see here lexification by the LLM. It was not told what to name the points, though it was given naming conventions. We also observe that the clarity optimiser chose a triangle where the base of the altitude and the base of the median had a reasonable separation.

4.2 Example 2: A symmedian

Natural language input. A symmedian is a cevian in a triangle formed by reflecting a median across the corresponding internal angle bisector. Draw triangle ABC. Construct the symmedian from vertex A and mark its intersection with side BC. Draw the circle with diameter from A to this intersection point.

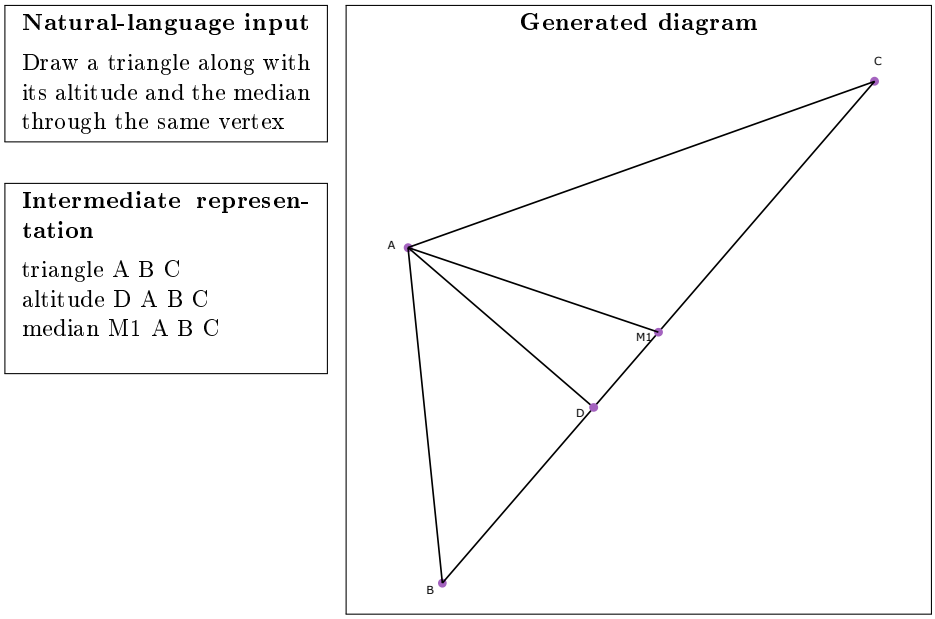

Fig. 2. Example illustrating the three stages of the system: natural-language input (top left), intermediate representation (bottom left), and the resulting geometric diagram (right).

Intermediate representation

```
triangle A B C
midpoint D B C
anglebisector bisA B A C
reflection E D bisA
segment A E
intersection K AE BC
midpoint O A K
circle circleAK O A
```

This example illustrates uneven lexical grounding. Without the explicit definition of symmedian, the LLM was unable to give meaningful output (it created a 'symmedian' command). With the addition of the definition, however, it was able to create a valid IR model. The concept of a circle with a diameter between A and K, on the other hand, without further explanation is converted into a circle whose center is the midpoint of A and K, and runs through A . This is clearly an atomic concept. We also note the handling of anaphora in the resolution of the sentence "Draw the circle with diameter from A to this intersection point".

We illustrate the concept of clarity by example. Figure 3 shows four different outputs from the clarity optimizer for this example. Though each output is quite

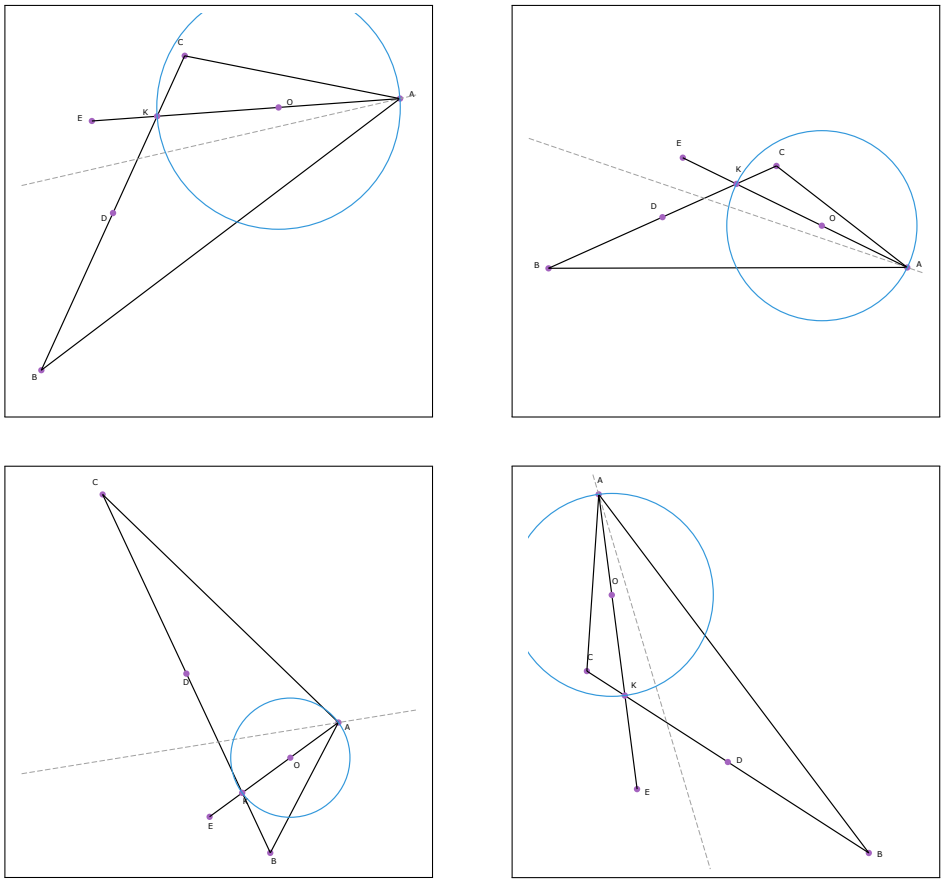

Fig. 3. Four realizations of the same geometric description, illustrating variation in clarity. The diagrams differ in layout, proximity, and incidental alignments, demonstrating the motivation for clarity optimization.

different, the common thread is that the points are reasonably spaced out, no pair too close, no individual too distant.

5 Discussion

The system presented here highlights both the promise and the challenges of combining neural interpretation with symbolic geometric reasoning. While the modular design ensures mathematical correctness and improves transparency, it also exposes several open questions. Natural–language descriptions vary widely in specificity, and some problems require disambiguation that exceeds what can be inferred from text alone. Similarly, clarity optimization raises interesting issues about the trade–off between geometric generality and diagram readability. These considerations suggest opportunities for further refinement, includ-

ing richer intermediate representations, adaptive disambiguation strategies, and broader evaluations across diverse problem sources. More generally, the work illustrates how hybrid AI systems can leverage the complementary strengths of language models and formal reasoning tools, offering a path toward reliable, interpretable geometric computation.

6 Conclusion

We have presented a system that generates mathematically correct and visually clear geometric diagrams directly from natural language descriptions. The system combines three components—natural-language interpretation, symbolic geometric construction, and clarity-driven optimization. The intermediate representation provides a transparent bridge between linguistic descriptions and symbolic reasoning; the Geometry Expressions engine ensures exact correctness; and the clarity optimizer selects a visually effective realization from the space of valid diagrams.

Our approach demonstrates that natural language, constraint based geometry, and optimization can be integrated in a way that preserves the strengths of each. The system produces diagrams that faithfully reflect the user's intent while avoiding misleading coincidences, accidental symmetries, and other visual artifacts that commonly arise in automated constructions.

There are several promising directions for future work. One is to broaden the range of geometric concepts supported by the parser and intermediate representation. Another is to refine the clarity metric to capture additional aspects of diagram quality, including aesthetic criteria and pedagogical conventions. Finally, integrating multimodal input—such as combining natural language with partial sketches—may further enhance usability in interactive settings.

Overall, this work provides a foundation for natural-language-driven geometric diagram generation and highlights the value of combining symbolic correctness with clarity optimization in diagram synthesis.

Reproducibility and availability. The system is fully reproducible given the intermediate representation, the sketch initialization, and the random seed. Upon acceptance, we will make an online demonstration and accompanying resources available to support further use and evaluation.

References

1. Wei, Y., Zhang, H., Wang, Y., Huang, S.: GeoLoom: High-Quality Geometric Diagram Generation from Textual Input. arXiv preprint arXiv:2512.08180 (2025)
2. Wei, Y., Zhang, H., Wang, Y., Huang, S.: MagicGeo: Training-Free Text-Guided Geometric Diagram Generation. arXiv preprint arXiv:2502.13855 (2025)
3. Hohenwarter, J., Hohenwarter, M., Lavicza, Z.: Introducing dynamic mathematics software to secondary school teachers: The case of GeoGebra. Journal of Computers in Mathematics and Science Teaching, 28(2), 135-146. (2009)

4. Todd, P.: Geometry Expressions: A Constraint-Based Interactive Symbolic Geometry System. In: Wang, D. (eds.) Automated Deduction in Geometry (ADG 2006). Lecture Notes in Computer Science, vol. 4530, pp. 189–202. Springer, Berlin, Heidelberg (2007)