Automatically Building Diagrams for Olympiad Geometry Problems

Ryan Krueger\textsuperscript{1}, Jesse Michael Han\textsuperscript{2}, and Daniel Selsam\textsuperscript{3}

\textsuperscript{1} University of Oxford, Oxford, UK
\textsuperscript{2} University of Pittsburgh, Pittsburgh PA, USA
\textsuperscript{3} Microsoft Research, Redmond WA, USA

Abstract. We present a method for automatically building diagrams for olympiad-level geometry problems and implement our approach in a new open-source software tool, the Geometry Model Builder (GMB). Central to our method is a new domain-specific language, the Geometry Model-Building Language (GMBL), for specifying geometry problems along with additional metadata useful for building diagrams. A GMBL program specifies (1) how to parameterize geometric objects (or sets of geometric objects) and initialize these parameterized quantities, (2) which quantities to compute directly from other quantities, and (3) additional constraints to accumulate into a (differentiable) loss function. A GMBL program induces a (usually) tractable numerical optimization problem whose solutions correspond to diagrams of the original problem statement, and that we can solve reliably using gradient descent. Of the 39 geometry problems since 2000 appearing in the International Mathematical Olympiad, 36 can be expressed in our logic and our system can produce diagrams for 94% of them on average. To the best of our knowledge, our method is the first in automated geometry diagram construction to generate models for such complex problems.

1 Introduction

Automated theorem provers for Euclidean geometry often use numerical models (\emph{i.e.} diagrams) for heuristic reasoning, \emph{e.g.} for conjecturing subgoals, pruning branches, checking non-degeneracy conditions, and selecting auxiliary constructions. However, modern solvers rely on diagrams that are either supplied manually \cite{7,23} or generated automatically via methods that are severely limited in scope \cite{12}. Motivated by the IMO Grand Challenge, an ongoing effort to build an AI that can win a gold medal at the International Mathematical Olympiad (IMO), we present a method for expressing and solving olympiad-level systems of geometric constraints.

Historically, algebraic methods are the most complete and performant for automated geometry diagram construction but suffer from degenerate solutions...
and, in the numerical case, non-convexity. These methods are restricted to relatively simple geometric configurations as poor local minima arise via large numbers of parameters. Moreover, degenerate solutions manifest as poor distributions for the vertices of geometric objects (e.g., a non-sensible triangle) as well as intersections of objects at more than one point (e.g., lines and circles, circles and circles).

We constructed a domain-specific language (DSL), the Geometry Model-Building Language (GMBL), to express geometry problems whose semantics induce tractable numerical optimization problems. The GMBL includes a set of commands with which users introduce geometric objects and constraints between these objects. There is a direct interpretation from these commands to the parameterization of geometric objects, the computation of geometric quantities from existing ones, and additional numerical constraints. The GMBL employs root selector declarations to disambiguate multiple solution problems, reparameterizations both to reduce the number of parameters and increase uniformity in model variance, and joint distributions for geometric objects that are susceptible to degeneracy (i.e., triangles and polygons). Our DSL treats points, lines, and circles as first-class citizens, and the language can be easily extended to support additional high-level features in terms of these primitives.

We provide an implementation of our method, the Geometry Model Builder (GMB), that compiles GMBL programs into Tensorflow computation graphs and generates models via off-the-shelf, gradient-based optimization. Figure 2 demonstrates an overview of this implementation. Experimentally, we find that the GMBL sufficiently reduces the parameter space and mitigates degeneracy to make our target geometry amenable to numerical optimization. We tested our method on all IMO geometry problems since 2000 (n = 39), of which 36 can be expressed as GMBL programs. Using default parameters, the GMB finds a single model for 94% of these 36 problems in an average of 27.07 seconds. Of the problems for which our program found a model and the goal of the problem could be stated in our DSL, the goal held in the final model 86% of the time.
All code is available on GitHub with which users can write GMBL programs and generate diagrams. Our program can be run both as a command-line tool for integration with theorem provers or as a locally-hosted web server.

2 Background

Here we provide an overview of olympiad-level geometry problem statements, as well as several challenges presented by the associated constraint problems.

2.1 Olympiad-Level Geometry Problem Statements

IMO geometry problems are stated as a sequential introduction of potentially-constrained geometric objects, as well as additional constraints between entities. Such constraints can take one of two forms: (1) geometric constraints describe the relative position of geometric entities (e.g. two lines are parallel) while (2) dimensional constraints enforce specific numerical values (e.g. angle, radius). Lastly, problems end with a goal (or set of goals) typically in the form of geometric or dimensional constraints. The following is an example from IMO 2009:

Let $ABC$ be a triangle with circumcentre $O$. The points $P$ and $Q$ are interior points of the sides $CA$ and $AB$, respectively. Let $K$, $L$, and $M$ be the midpoints of the segments $BP$, $CQ$, and $PQ$, respectively, and let $\Gamma$ be the circle passing through $K$, $L$, and $M$. Suppose that the line $PQ$ is tangent to the circle $\Gamma$. Prove that $OP = OQ$.

( IMO 2009 P2)

This problem introduces ten named geometric objects and has a single goal.

Note that this class of problems does not admit a mathematical description but rather is defined empirically (i.e. as those problems selected for olympiads). The overwhelming majority of these problems are of a particular type – plane geometry problems that can be expressed as problems in nonlinear real arithmetic (NRA). However, while NRA is technically decidable, olympiad problems tend to be littered with order constraints and complex constructions (e.g. mixtilinear incircle) and be well beyond the capability of existing algebraic methods. On the other hand, they are selected to admit elegant, human-comprehensible proofs. It is this class of problems for which the GMBL was designed to express; though rare, any particular olympiad geometry problem is not guaranteed to be of this type and therefore is not necessarily expressible in the GMBL.

2.2 Challenge: Globally Coupled Constraints

A naïve approach to generate models would incrementally instantiate objects via their immediate constraints. For (IMO 2009 P2), this would work as follows:

1. Sample points $A$, $B$, and $C$.  

4 https://github.com/rkruegs123/geo-model-builder
2. Compute $O$ as the circumcenter of $\triangle ABC$.
3. Sample $P$ and $Q$ on the segments $CA$ and $AB$, respectively.
4. Compute $K$, $L$, and $M$ as the midpoints of $BP$, $CQ$, and $PQ$, respectively.
5. Compute $\Gamma$ as the circle defined by $K$, $L$, and $M$.

Immediately we see a problem – there is no guarantee that $PQ$ is tangent to $\Gamma$ in the final model. Indeed, the constraints of [IMO 2009 P2] are quite globally coupled – the choice of $P$ partially determines the circle $\Gamma$ to which $PQ$ must be tangent, and every choice of $\triangle ABC$ does not even admit a pair $P$ and $Q$ satisfying this constraint. This is an example of the frequent non-constructive nature of IMO geometry problems. When there is no obvious reparameterization to avoid downstream effects, all constraints must be considered simultaneously rather than incrementally or as a set of smaller local optimization problems.

### 2.3 Challenge: Root Resolution

Even in the constructive case, local optimization is not necessarily sufficient given that multiple solutions can exist for algebraic constraints. More specifically, two circles or a circle and a line intersect at up to two distinct points and in a problem that specifies each distinct intersection point, the correct root to assign is generally not locally deducible. Without global information, this can lead to poor initializations becoming trapped in local minima. The GMBL accounts for this by including a set of explicit root selectors as described in Section 3.3. These root selectors provide global information for selecting the appropriate point from a set of multiple solutions to a system of equations. The prototypical example of this challenge is presented in Appendix D.

### 3 Methods

In this section we present the GMBL and GMB in detail. In our presentation, we make use of the following notation and definitions:

- The type of a geometric object can be one of (1) point, (2) line, or (3) circle. We denote the type of a real-valued number as number.
– We use <> to denote an instance of a type.
– A name is a string value that refers to a geometric object.

3.1 GMBL: Overview

The GMBL is a DSL for expressing olympiad-level geometry problems that losslessly induces a numerical optimization problem. It consists of four commands, each of which has a direct interpretation regarding the accumulation of (1) real-valued parameters and (2) differentiable losses in terms of these parameters:

1. param: assigns a name to a new geometric object parameterized either by a default or optionally supplied parameterization
2. define: assigns a name to an object computed in terms of existing ones
3. assert: imposes an additional constraint (i.e. differentiable loss value)
4. eval: evaluates a given constraint in the final model(s)

Table 1 provides a summary of their usage. The GMBL includes an extensible library of functions and predicates with which commands are written. Notably, this library includes a notion of root selection to explicitly resolve the selection of roots to systems of equations with multiple solutions.

3.2 GMBL: Commands

In the following, we describe in more detail the usage of each command and their roles in constructing a tractable numerical optimization problem.

param accepts as arguments a string, a type, and an optional parameterization. This introduces a geometric object that is parameterized either by the default parameterization for <type> or by the supplied method. Each primitive geometric type has the following default parameterization:

– point: parameterized by its x- and y-coordinates
– line: parameterized by two points that define the line
– circle: parameterized by its origin and radius

Optional parameterizations embody our method’s use of reparameterization to decrease the number of parameters and increase model diversity. For example, consider a point $C$ on the line $\overrightarrow{AB}$ that is subject to additional constraints. Rather than optimizing over the x- and y-coordinates of $C$, we can express $C$ in terms of a single value $z$ that scales $C$’s placement on the line $\overrightarrow{AB}$.

In addition to the standard usage of param outlined above, the GMBL includes an important variant of this command to introduce sets of points that form triangles and polygons. This variant accepts as arguments (1) a list of point names, and (2) a required parameterization (see Table 1). This joint parameterization of triangles and polygons further prevents degeneracy. For example, to initialize a triangle $\triangle ABC$, we can sample the vertices from normal distributions with means at distinct thirds of the unit circle. This method minimizes the
Table 1: An overview of usage for the four commands.

| Command   | Usage                                                                 |
|-----------|-----------------------------------------------------------------------|
| param     | (param <string> <type> <optional-parameterization>)                   |
|           | or                                                                     |
|           | (param (<string>, ..., <string>) <parameterization>)                  |
| define    | (define <string> <type> <value>)                                      |
| assert    | (assert <predicate>)                                                  |
| eval      | (eval <predicate>)                                                   |

Sampling of triangles with extreme angle values, as well as allows for explicit control over the distribution of acute vs. obtuse triangles by adjusting the standard deviations. Appendix C includes a list of all available parameterizations.

`define` accepts as arguments a string, a type, and a value that is one of `<point>`, `<line>`, or `<circle>`. This command serves as a basic assignment operator and is useful for caching commonly used values. The functions described in Section 3.3 are used to construct `<value>` from existing geometric objects.

`assert` accepts a single predicate and imposes it as an additional constraint on the system. This is achieved by translating the predicate to a set of algebraic values and registering them as losses. This command does not introduce any new geometric objects and can only refer to those already introduced by `param` or `define`. Notably, dimensional constraints and negations are always enforced via `assert`. Detail on supported predicates is presented in Section 3.3.

`eval`, like `assert`, accepts a single predicate and therefore does not introduce any new geometric objects. However, unlike `assert`, the corresponding algebraic values are evaluated and returned with the final model rather than registered as losses and enforced via optimization. This command is most useful for those interested in integrating the GMBL with theorem provers.

3.3 GMBL: Functions and Predicates

The second component of our DSL is a set of functions and predicates for constructing arguments to the commands outlined above. Functions construct new geometric objects and numerical values whereas predicates describe relationships between them. Our DSL includes high-level abstractions for common geometric concepts in olympiad geometry (e.g. excircle, isotomic conjugate).

Functions in the GMBL employ a notion of root selectors to address the “multiple solutions problem” described in Section 2.3. In plane geometry, this problem typically manifests with multiple candidate point solutions, such as the intersection between a line and a circle. Root selectors control for this by allowing users to specify the appropriate point for functions with multiple solutions. Figure 3 demonstrates their usage in the functions `inter-lc` (intersection of a line and circle) and `inter-cc` (intersection of two circles).
3.4 Auxiliary Losses

The optimization problem encoded by a GMBL program includes three additional loss values. Foremost, for every instance of a circle intersecting a line or other circle, we impose a loss value that ensures the two geometric objects indeed intersect. The final two, albeit opposing losses are intended to minimize global degeneracy. We impose one loss that minimizes the mean of all point norms to prevent exceptionally separate objects and a second to enforce a sufficient distance between points to maintain distinctness.

3.5 Implementation

We built the GMB, an open-source implementation that compiles GMBL programs to optimization problems and generates models. The GMB takes as input a GMBL program and processes each command in sequence to accumulate real-valued parameters and differentiable losses in a Tensorflow computation graph. After registering auxiliary losses, we apply off-the-shelf gradient-based local optimization to produce models of the constraint system. In summary, to generate N numerical models, our optimization procedure works as follows:

1. Construct computation graph by sequentially processing commands.
2. Register auxiliary losses.
3. Sample sets of initial parameter values and rank via loss value.
4. Choose (next) best initialization and optimize via gradient descent.
5. Repeat (4) until obtaining N models or the maximum # of tries is reached.
Table 2: An evaluation of our method’s ability to generate a single model for each of the 36 IMO problems encoded in our DSL. For each problem, 10 sets of initial parameters were sampled over which our program optimized up to three. All data shown are the average of three trials. The first row demonstrates results using default parameters ($\epsilon = 0.001$, learning rate = 0.1, # iterations = 5,000).

| $\epsilon$ | Learning Rate | Iterations | % Success | % Goal Satisfaction | Time per Problem (s) |
|------------|---------------|------------|-----------|---------------------|----------------------|
| 0.001      | 0.1           | 5,000      | 93.52     | 85.84               | 27.07 223.51 14.72   |
| 0.01       | 0.1           | 5,000      | 92.60     | 84.71               | 26.86 229.71 14.43   |
| 0.001      | 0.01          | 5,000      | 88.88     | 86.32               | 27.54 137.85 14.33   |
| 0.001      | 0.1           | 10,000     | 92.59     | 86.02               | 34.78 287.51 15.43   |

Our program accepts as arguments (1) the # of models desired (default = 1), (2) the # of initializations to sample (default = 10), and (3) the max # of optimization tries (default = 3). Our program also accepts the standard suite of parameters for training a Tensorflow model, including an initial learning rate (default = 0.1), a decay rate (default = 0.7), the max # of iterations (default = 5000), and an epsilon value (default = 0.001) to determine stopping criteria.

4 Results

In this section, we present an evaluation of our method’s proficiency in three areas of expressing and solving olympiad-level geometry problems:

1. Expressing olympiad-level geometry problems as GMBL programs.
2. Generating models for these programs.
3. Preserving truths (up to tolerance) that are not directly optimized for.

Table 2 contains a summary of our results.

Our evaluation considers all 39 IMO geometry problems since 2000. Of these 39 problems, 36 can be expressed in our DSL. Those that we cannot encode involve variable numbers of geometric objects. For 32 of these 36 problems, we can express the goals as eval commands in the corresponding GMBL programs. The goals of the additional four problems are not expressible in our DSL, e.g., our DSL cannot express goals of the form “Find all possible values of $\angle ABC$.”

To evaluate (2) and (3), we conducted three trials in which we ran our program on each of the 36 encodings with varying sets of arguments. With default arguments, our program generated a single model for (on average) 94% of these problems. Our program ran for an average of 27.07 seconds for each problem but there is a stark difference between time to success and time to failure (14.72 vs 223.51 seconds) as failure entails completing all optimization attempts whereas
successful generation of a model terminates the program. We achieve similar success rates with more forgiving training arguments or a higher tolerance.

For use in automated theorem proving, it is essential that models generated by our tool not only satisfy the constraint problem up to tolerance but also any other truths that follow from the set of input constraints. The most immediate example of such a truth is the goal of a problem statement. Therefore, we used the goals of IMO geometry problems as a proxy for this ability by only checking the satisfaction of the goal in the final model (i.e. with an eval statement) rather than directly optimizing for it. In our experiments, we considered such a goal satisfied if it held up to $\epsilon \times 10$ as it is reasonable to expect slightly higher floating-point error without explicit optimization. Using default parameters, the goal held up to tolerance in 86% of problems for which we found a model and could express the goal. This rate was similar across all other sets of arguments.

5 Future Work

Here we discuss various opportunities for improvement of our method.

Firstly, improvements could be made to our method of numerical optimization. While Tensorflow offers a convenient way of caching terms via a static computation graph and optimizing directly over this representation, there is not explicit support for constrained optimization. Because of this, arbitrary weights have to be assigned to each loss value. Though rare, this can result in false positives and negatives for the satisfaction of a constraint. Using an explicit constrained-optimization method (e.g. SLSQP) would enable the separation of soft constraints (e.g. maximizing the distance between points) and hard constraints (e.g. those enforced by assert), removing the need for arbitrary weights.

Secondly, cognitive overhead could be reduced as users are currently required to determine degrees of freedom; it would be far easier to write problem statements using only declarations of geometric objects and constraints between them, e.g. using only assert. This could be accomplished by treating our DSL as a low-level “instruction set” to which a higher-level language could be compiled. The main challenge of such a compiler would be appropriately identifying opportunities to reduce the degrees of freedom. To achieve this, the compiler would require a decision procedure for line and circle membership.

Lastly, we could improve our current treatment of distinctness. To prevent degenerate solutions, our method optimizes for object distinctness and rejects models with duplicates. However, there is the occasional problem for which a local optimum encodes two provably distinct points as equal up to floating point tolerance. There are many techniques that could be applied to this problem (e.g. annealing) though we do not consider them here as the issue is rare.

6 Related Work

Though many techniques for mechanized geometry diagram construction have been introduced over the decades, no method, to the best of our knowledge, can
produce models for more than a negligible fraction of olympiad problems. There exist many systems, built primarily for educational purposes, for interactively generating diagrams using ruler-and-compass constructions, e.g. GCLC [13], GeoGebra [11], Geometer’s Sketchpad [19], and Cinderella [18]. There are also non-interactive methods for deriving such constructions, e.g. GeoView [2] and program synthesis [9, 12]. However, as discussed in Section 2.2, very few olympiad problems can be described in such a form. Alternatively, Penrose is an early-stage system for translating mathematical descriptions to diagrams that relies on constrained numerical optimization and therefore does not suffer from this expressivity limitation [24]. However, this system lacks support for constraints with multiple roots, e.g. intersecting circles. There are more classical methods that similarly depart from constructive geometry. MMP/Geometer [8] translates the problem to a set of algebraic equations and uses numerical optimization (e.g. BFGS) and GEOTHER [21, 22] first translates a predicate specification into polynomial equations, decomposes this system into representative triangular sets, and obtains solutions for each set numerically. Neither of these programs are available to evaluate though we did test similar approaches using modern libraries (specifically: sympy [16] and scipy [20]) and both numerical and symbolic methods would almost always timeout on relatively simple olympiad problems.

Generating models for systems of geometric constraints is also a challenge in computer-aided design (CAD) for engineering diagram drawing. Recent efforts focus on graph-based synthetic methods, a subset of techniques concerned with ruler-and-compass constructions [3, 5, 6, 10, 14, 15, 17]. Most relevant to our method are Bettig and Shah’s “solution selectors” which, similar to root selectors in the GMBL, allow users to specify the configuration of a CAD model [4]. However, these solution selectors are purpose-built and do not generalize.

7 Conclusion

It is standard in GTP to rely on diagrams for heuristic reasoning but the scale of automatic diagram construction is limited. To enable efforts to build a solver for IMO geometry problems, we developed a method for building diagrams for olympiad-level geometry problems. Our method is based on the GMBL, a DSL for expressing geometry problems that induces (usually) tractable numerical optimization problems. The GMBL includes a set of commands that have a direct interpretation for accumulating real-valued parameters and differentiable losses. Arguments to these commands are constructed with a library of functions and predicates that includes notions of root selection, joint distributions, and reparameterizations to minimize degeneracy and the number of parameters. We implemented our approach in an open-source tool that translates GMBL programs to diagrams. Using this program, we evaluated our method on all IMO geometry problems since 2000. Our implementation reliably produces models; moreover, known truths that are not directly optimized for typically hold up to tolerance. By handling configurations of this complexity, our system clears a roadblock in GTP and provides a critical tool for undertakers of the IMO Grand Challenge.
References

1. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16), pages 265–283, 2016.
2. Y. Bertot, F. Guilhot, and L. Pottier. Visualizing geometrical statements with GeoView. Electronic Notes in Theoretical Computer Science, 103:49–65, 2004.
3. B. Bettig and C. M Hoffmann. Geometric constraint solving in parametric computer-aided design. Journal of computing and information science in engineering, 11(2), 2011.
4. B. Bettig and J. Shah. Solution selectors: a user-oriented answer to the multiple solution problem in constraint solving. J. Mech. Des., 125(3):443–451, 2003.
5. B. N. Freeman-Benson, J. Maloney, and A. Bornig. An incremental constraint solver. Communications of the ACM, 33(1):54–63, 1990.
6. I. Fudos. Constraint solving for computer aided design. PhD thesis, Verlag nicht ermittelbar, 1995.
7. W. Gan, X. Yu, T. Zhang, and M. Wang. Automatically proving plane geometry theorems stated by text and diagram. International Journal of Pattern Recognition and Artificial Intelligence, 33(07):1940003, 2019.
8. X.-S. Gao and Q. Lin. Mmp/geometer–a software package for automated geometric reasoning. In International Workshop on Automated Deduction in Geometry, pages 44–66. Springer, 2002.
9. S. Gulwani, V. A. Korthikanti, and A. Tiwari. Synthesizing geometry constructions. ACM SIGPLAN Notices, 46(6):50–61, 2011.
10. C. M. Hoffmann and R. Joan-Arinyo. Parametric modeling. In Handbook of computer aided geometric design, pages 519–541. Elsevier, 2002.
11. M. Hohenwarter and M. Hohenwarter. GeoGebra.
12. S. Itzhaky, S. Gulwani, N. Immerman, and M. Sagiv. Solving geometry problems using a combination of symbolic and numerical reasoning. In International Conference on Logic for Programming Artificial Intelligence and Reasoning, pages 457–472. Springer, 2013.
13. P. Janičić. GCLC—a tool for constructive euclidean geometry and more than that. In International Congress on Mathematical Software, pages 58–73. Springer, 2006.
14. G. A. Kramer. Solving geometric constraint systems. In AAAI, pages 708–714, 1990.
15. R. S. Latham and A. E. Middleditch. Connectivity analysis: a tool for processing geometric constraints. Computer-Aided Design, 28(11):917–928, 1996.
16. A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, et al. SymPy: symbolic computing in Python. PeerJ Computer Science, 3:e103, 2017.
17. J. C. Owen. Algebraic solution for geometry from dimensional constraints. In Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications, pages 397–407, 1991.
18. J. Richter-Gebert and U. H. Kortenkamp. The Cinderella. 2 Manual: Working with The Interactive Geometry Software. Springer Science & Business Media, 2012.
19. D. Scher. Lifting the curtain: The evolution of the Geometer’s Sketchpad. The Mathematics Educator, 10(2), 1999.
20. P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, 17(3):261–272, 2020.

21. D. Wang. Geother 1.1: Handling and proving geometric theorems automatically. In *International Workshop on Automated Deduction in Geometry*, pages 194–215. Springer, 2002.

22. D. Wang. Automated generation of diagrams with Maple and Java. In *Algebra, Geometry and Software Systems*, pages 277–287. Springer, 2003.

23. K. Wang and Z. Su. Automated geometry theorem proving for human-readable proofs. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

24. K. Ye, W. Ni, M. Krieger, D. Ma’ayan, J. Wise, J. Aldrich, J. Sunshine, and K. Crane. Penrose: from mathematical notation to beautiful diagrams. *ACM Transactions on Graphics (TOG)*, 39(4):144–1, 2020.
## A Functions

Here we provide a complete list of all functions supported in the GMBL. Throughout this section, we provide examples that make use of the following declared geometric objects:

- A, B, C, D, E, F, G, H, and P are instances of point
- L1, L2, and L3 are instances of line
- C1 and C2 are instances of circle
- N1 and N2 are instances of number

| Function                  | Type   | Description                                                                 |
|---------------------------|--------|-----------------------------------------------------------------------------|
| (amidp-opp A B C)         | point  | The midpoint of the arc $\overline{AB}$ on $(ABC)$ excluding C               |
| (amidp-same A B C)        | point  | The midpoint of the arc $\overline{ACB}$ on $(ABC)$                         |
| (centroid A B C)          | point  | The centroid of $\triangle ABC$                                             |
| (circumcenter A B C)      | point  | The circumcenter of $\triangle ABC$                                         |
| (excenter A B C)          | point  | The A-excenter of $\triangle ABC$                                           |
| (foot A L1)               | point  | The perpendicular foot from A to L1                                         |
| (harmonic-conj C A B)     | point  | The harmonic conjugate of C w.r.t. $\overline{AB}$                          |
| (incenter A B C)          | point  | The incenter of $\triangle ABC$                                            |
| (inter-cc C1 C2 <root-selector>) | point | One of the intersections of C1 and C2, as specified by a root selector |
| (inter-ll L1 L2)          | point  | The intersection of 2 lines                                                |
| (inter-lc L1 C1 <root-selector>) | point | One of the intersections of L1 and C1, as specified by a root selector |
| (isogonal-conj D A B C)   | point  | The isogonal conjugate of D w.r.t. $\triangle ABC$                         |
| (isotomic-conj D A B C)   | point  | The isotomic conjugate of D w.r.t. $\triangle ABC$                         |
| (midp A B)                | point  | The midpoint of $\overline{AB}$                                            |
| (mixtilinear-incenter A B C) | point | The A-mixtilinear incenter of $\triangle ABC$                             |
| Term                        | Type    | Definition                                                                 |
|-----------------------------|---------|-----------------------------------------------------------------------------|
| (orthocenter A B C)         | point   | The orthocenter of $\triangle ABC$                                         |
| (connecting A B)            | line    | The line connecting A and B (i.e. $\overrightarrow{AB}$)                   |
| (isogonal D A B C)           | line    | The isogonal of $\overrightarrow{AD}$ w.r.t. $\triangle ABC$               |
| (isotomic D A B C)           | line    | The isotomic of $\overrightarrow{AD}$ w.r.t. $\triangle ABC$               |
| (perp-bis A B)              | line    | The perpendicular bisector of $AB$                                         |
| (perp-at A L1)              | line    | The line through A that is perpendicular to L1                             |
| (c3 A B C)                  | circle  | The circle passing through A, B, and C                                      |
| (circumcircle A B C)         | circle  | The circumcircle of $\triangle ABC$                                        |
| (excircle A B C)             | circle  | The A-excircle of $\triangle ABC$                                          |
| (incircle A B C)             | circle  | The incircle of $\triangle ABC$                                            |
| (mixtilinear-incircle A B C) | circle  | The A-mixtilinear incircle of $\triangle ABC$                              |
| (diam A B)                  | circle  | The circle with diameter $AB$                                              |
| (add N1 N2)                 | number  | $N1 + N2$                                                                   |
| (area A B C)                | number  | The area of $\triangle ABC$                                                |
| (dist A B)                  | number  | The distance between A and B                                               |
| (div N1 N2)                 | number  | $N1/N2$                                                                     |
| (mul N1 N2)                 | number  | $N1 \times N2$                                                             |
| pi                          | number  | $\pi$                                                                       |
| (pow N1 N2)                 | number  | $N1^{N2}$                                                                   |
| (neg N1)                    | number  | $-N1$                                                                       |
| (radius C1)                 | number  | The radius of C1                                                           |
| (sqrt N1)                   | number  | $\sqrt{N1}$                                                                 |
| (uangle A B C)              | number  | The undirected angle formed by A, B and C (i.e. $\angle ABC$)               |
## B Predicates

Here we provide a complete list of all predicates supported in the GMBL. The same variables are used as in Appendix A:

| Predicate                  | Description                                                                 |
|----------------------------|-----------------------------------------------------------------------------|
| (centroid P A B C)         | **True** i.f.f. P is the centroid of \( \Delta ABC \)                      |
| (concur L1 L2 L3)          | **True** i.f.f. L1, L2, and L3 intersect at a single point                   |
| (circumcenter P A B C)     | **True** i.f.f. P is the circumcenter of \( \Delta ABC \)                    |
| (cong A B C D)             | **True** i.f.f. \(|AB| = |CD|\)                                           |
| (contri A B C D E F)        | **True** i.f.f. \( \Delta ABC \cong \Delta DEF \)                         |
| (coll A B C)               | **True** i.f.f. A, B, and C are collinear                                   |
| (cycl P1 \ldots PN)        | **True** i.f.f. \( P_1 \ldots P_N \) lie on a common circle, where \( P_1 \ldots P_N \) are points and \( N \geq 4 \) |
| (= A B)                    | **True** i.f.f. \( A = B \)                                               |
| (= N1 N2)                  | **True** i.f.f. \( N_1 = N_2 \)                                           |
| (eq-ratio A B C D E F G H) | **True** i.f.f. \( \frac{|AB|}{|CD|} = \frac{|EF|}{|GH|} \)                   |
| (foot P A L1)              | **True** i.f.f. P is the perpendicular foot from A to L1                    |
| (> N1 N2)                  | **True** i.f.f. \( N_1 > N_2 \)                                           |
| (>= N1 N2)                 | **True** i.f.f. \( N_1 \geq N_2 \)                                         |
| (incenter P A B C)         | **True** i.f.f. P is the incenter of \( \Delta ABC \)                       |
| (inter-l1 P L1 L2)         | **True** i.f.f. P is the intersection of L1 and L2                          |
| (< N1 N2)                  | **True** i.f.f. \( N_1 < N_2 \)                                           |
| (<= N1 N2)                 | **True** i.f.f. \( N_1 \leq N_2 \)                                         |
| (midp P A B)               | **True** i.f.f. P is the midpoint of \( AB \)                              |
| (on-circ P C1)             | **True** i.f.f. P is on \( C_1 \)                                          |
| (on-line P L1)             | **True** i.f.f. P is on \( L_1 \)                                          |
| (on-ray P A B)             | **True** i.f.f. P is on \( \overline{AB} \)                               |
| (on-seg P A B)             | **True** i.f.f. P is on \( \overline{AB} \)                               |
| (opp-sides A B L1)         | **True** i.f.f. A and B are on opposite sides of \( L_1 \)                 |
| (orthocenter P A B C)      | **True** i.f.f. P is the orthocenter of \( \Delta ABC \)                    |
| Condition                          | Description                                                                 |
|-----------------------------------|-----------------------------------------------------------------------------|
| (perp L1 L2)                      | True if and only if $L_1 \perp L_2$                                          |
| (para L1 L2)                      | True if and only if $L_1 \parallel L_2$                                     |
| (same-side A B L1)                | True if and only if A and B are on the same side of $L_1$                    |
| (sim-tri A B C D E F)             | True if and only if $\Delta ABC \sim \Delta DEF$                           |
| (tangent-cc C1 C2)                | True if and only if C1 is tangent to C2                                      |
| (tangent-lc L1 C1)                | True if and only if L1 is tangent to C1                                      |
| (tangent-at-cc A C1 C2)           | True if and only if C1 is tangent to C2 at A                                 |
| (tangent-at-lc A L1 C1)           | True if and only if L1 is tangent to C1 at A                                 |
C Parameterizations and Root Selectors

Here we provide an enumeration of all parameterizations and root selectors supported in our DSL. The same variables are used as in Appendix A.

The following table provides a complete list of all optional parameterizations for standard usage of param. The value in the Type column denotes the type for which a parameterization is used:

| Parameterization         | Type  | Description                                                                 |
|--------------------------|-------|-----------------------------------------------------------------------------|
| (on-circ C1)             | point | Parameterizes a point on the circle C1                                     |
| (on-line L1)             | point | Parameterizes a point on the line L1                                       |
| (on-major-arc C1 A B)    | point | Parameterizes a point on the major arc of C1 connecting A and B.            |
| (on-minor-arc C1 A B)    | point | Parameterizes a point on the minor arc of C1 connecting A and B.            |
| (in-poly P_1 ... P_N)    | point | Parameterizes a point inside the polygon with vertices P_1 ... P_N, where P_1 ... P_N are points and N ≥ 3 |
| (on-ray A B)             | point | Parameterizes a point on the ray $\overrightarrow{AB}$                     |
| (on-ray-opp A B)         | point | Parameterizes a point beyond A on the ray $\overrightarrow{BA}$             |
| (on-seg A B)             | point | Parameterizes a point on the segment connecting A and B                      |
| (tangent-lc C1)          | line  | Parameterizes a line tangent to C1                                          |
| (through A)              | line  | Parameterizes a line passing through A                                      |
| (tangent-cc C1)          | circle| Parameterizes a circle tangent to C1                                        |
| (tangent-cl L1)          | circle| Parameterizes a circle tangent to L1                                        |
| (through A)              | circle| Parameterizes a circle passing through A                                    |
| (origin A)               | circle| Parameterizes a circle centered at A                                        |
| (radius N1)              | circle| Parameterizes a circle with radius N1                                       |

Alternatively, the following table provides a complete list of all supported parameterizations for introducing sets of points as triangles and polygons with param. Note that type information is omitted, as these parameterizations are always used to introduce lists of type (point, ..., point). Examples are provided as usages are subtle:
Parameterization | Example | Meaning
--- | --- | ---
acute-tri | param (P Q R) acute-tri | P, Q, and R are parameterized as an acute triangle
(acute-iso-tri <name>) | param (P Q R) (acute-iso-tri Q) | P, Q, and R are parameterized as an acute isosceles triangle with \(QP = QR\)
(iso-tri <name>) | param (P Q R) (iso-tri Q) | P, Q, and R are parameterized as an isosceles triangle with \(QP = QR\)
(right-tri <name>) | param (P Q R) (right-tri Q) | P, Q, and R are parameterized as a right triangle with a right angle at \(\angle PQR\)
triangle | param (P Q R) triangle | P, Q, and R are parameterized as a triangle
polygon | param (P Q R S) polygon | P, Q, R, and S are parameterized to form a convex quadrilateral

Lastly, the following table provides a complete list of all root selectors:

| Root Selector | Description |
| --- | --- |
| rs-arbitrary | An arbitrary root |
| (rs-neq A) | A root that is not equal A |
| (rs-opp-sides A L1) | A root on the opposite side of L1 as A |
| (rs-same-side A L1) | A root on the same side of L1 as A |
| (rs-closer-to-p A) | The root that is closer to A |
| (rs-closer-to-l L1) | The root that is closer to L1 |
D  Challenge: Local vs. Global Optimization

Here we present a distilled example where global optimization is required to reliably produce models. Optimization is tasked with assigning \( A \) and \( B \) to the two distinct intersection points of a line and a circle. To do so, the distance from each point from both the line and circle will be minimized while maintaining a minimum distance between points. \( A \) will be arbitrarily assigned to one of the two choices. However, without global knowledge of the root occupied by \( A \), an initialization of \( B \) too close to \( A \) will be trapped in a local minimum. Blue indicates a region of low loss and red a region of high loss. Points are represented as green circles.
E Examples

Here we provide a gallery of example diagrams generated by our tool and their associated problem statements (both in text and as GMBL programs).

IMO 2011, Problem 6

**IMO Problem Statement**

Let $ABC$ be an acute triangle with circumcircle $\Gamma$. Let $\ell$ be a tangent line to $\Gamma$, and let $\ell_a$, $\ell_b$, and $\ell_c$ be the lines obtained by reflecting $\ell$ in the lines $BC$, $CA$, and $AB$, respectively. Show that the circumcircle of the triangle determined by the lines $\ell_a$, $\ell_b$, and $\ell_c$ is tangent to the circle $\Gamma$.

**GMBL Problem Statement**

```gml
(param (A B C) acute-tri)
(define Gamma circle (circumcircle A B C))
(param l line (tangent-lc Gamma))
(define la line (reflect-ll l (line B C)))
(define lb line (reflect-ll l (line C A)))
(define lc line (reflect-ll l (line A B)))
(eval (tangent-cc Gamma (circumcircle (inter-ll la lb)
                                 (inter-ll la lc) (inter-ll lb lc))))
```

**Diagram**

![Diagram](image)
IMO 2008, Problem 1

**IMO Problem Statement**

An acute-angled triangle $ABC$ has orthocentre $H$. The circle passing through $H$ with centre the midpoint of $BC$ intersects the line $BC$ at $A_1$ and $A_2$. Similarly, the circle passing through $H$ with centre the midpoint of $CA$ intersects the line $CA$ at $B_1$ and $B_2$, and the circle passing through $H$ with centre the midpoint of $AB$ intersects the line $AB$ at $C_1$ and $C_2$. Show that $A_1, A_2, B_1, B_2, C_1, C_2$ lie on a circle.

**GMBL Problem Statement**

```gmbl
(param (A B C) acute-tri)
(define H point (orthocenter A B C))
(define A1 point (inter-lc (line B C) (coa (midp B C) H) rs-arbitrary))
(define A2 point (inter-lc (line B C) (coa (midp B C) H) (rs-neq A1)))
(define B1 point (inter-lc (line C A) (coa (midp C A) H) rs-arbitrary))
(define B2 point (inter-lc (line C A) (coa (midp C A) H) (rs-neq B1)))
(define C1 point (inter-lc (line A B) (coa (midp A B) H) rs-arbitrary))
(define C2 point (inter-lc (line A B) (coa (midp A B) H) (rs-neq C1)))
(eval (cycl A1 A2 B1 B2 C1 C2))
```

**Diagram**

![Diagram of the problem statement with points A, B, and C, and additional points A1, A2, B1, B2, C1, C2, illustrating the circle properties as described in the IMO Problem Statement.](image-url)
IMO 2009, Problem 2

IMO Problem Statement

Let $ABC$ be a triangle with circumcentre $O$. The points $P$ and $Q$ are interior points of the sides $CA$ and $AB$, respectively. Let $K$, $L$, and $M$ be the midpoints of the segments $BP$, $CQ$, and $PQ$, respectively, and let $\Gamma$ be the circle passing through $K$, $L$, and $M$. Suppose that the line $PQ$ is tangent to the circle $\Gamma$. Prove that $OP = OQ$.

GMBL Problem Statement

(param (A B C) triangle)
(define O point (circumcenter A B C))
(param P point (on-seg C A))
(param Q point (on-seg A B))
(define K point (midp B P))
(define L point (midp C Q))
(define M point (midp P Q))
(define Gamma circle (circ K L M))
(assert (tangent-lc (line P Q) Gamma))
(eval (cong O P O Q))