# Automated Elementary Geometry Theorem Discovery via Inductive Diagram Manipulation

by

Lars Erik Johnson

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015



Author .....
Department of Electrical Engineering and Computer Science
May 22, 2015

Certified by .....
Gerald J. Sussman
Panasonic Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Albert Meyer
Chairman, Masters of Engineering Thesis Committee

# Automated Elementary Geometry Theorem Discovery via Inductive Diagram Manipulation

by

Lars Erik Johnson

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2015, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

In this thesis, I created and analyzed an interactive computer system capable of exploring geometry concepts through inductive investigation. My system begins with a limited set of knowledge about basic geometry and enables a user interacting with the system to "teach" the system additional geometry concepts and theorems by suggesting investigations the system should explore to see if it "notices anything interesting." The system uses random sampling and physical simulations to emulate the more human-like processes of manipulating diagrams "in the mind's eye." It then uses symbolic pattern matching and a propagator-based truth maintenance system to appropriately generalize findings and propose newly discovered theorems. These theorems can be rigorously proved using external proof assistants, but also be used by the system to assist in its explorations of new, higher-level concepts. Through a series of simple investigations similar to an introductory course in geometry, the system has been able to propose and learn a few dozen standard geometry theorems, and through more self-directed explorations, it has discovered several interesting properties and theorems not typically covered in standard mathematics courses.

Thesis Supervisor: Gerald J. Sussman
Title: Panasonic Professor of Electrical Engineering

# Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements.

# Contents

# List of Figures

# Chapter 1

# Introduction

In this thesis, I develop and analyze an interactive computer system that emulates a student learning geometry concepts through inductive investigation. Although geometry knowledge can be conveyed via a series of factual definitions, theorems, and proofs, my system focuses on a more investigative approach in which an external teacher guides the student to "discover" new definitions and theorems via explorations and self-directed inquiry.

My system emulates such a student by beginning with a fairly limited knowledge set regarding basic definitions in geometry and providing a means by which a user interacting with the system can "teach" additional geometric concepts and theorems by suggesting investigations the system should explore to see if it "notices anything interesting."

To enable such learning, my project includes the combination of four intertwined modules: an imperative geometry construction interpreter to build constructions, a declarative geometry constraint solver to solve and test specifications, an observation-based perception module to notice interesting properties, and a learning module to analyze information from the other modules and integrate it into new definition and theorem discoveries.

To evaluate its recognition of such concepts, my system provides means for a user to extract the observations and apply its findings to new scenarios. Through a series of simple investigations similar to an introductory course in geometry, the system has

been able to propose and learn a few dozen standard geometry theorems. Furthermore, through more self-directed explorations, it has discovered several interesting properties and theorems not typically covered in standard mathematics courses.

## 1.1 Document Structure

**Chapter 2** further discusses motivation of the system and presents some examples of diagram manipulation, emphasizing the technique of visualizing diagrams "in the mind's eye."

**Chapter 8** discusses some related work to automated geometry theorem discovery and proof, as well as a comparison with existing dynamic geometry systems.

**Chapter 3** further introduces the system modules and discusses how they work together in the discovery of new definitions and theorems.

**Chapter 5** describes the implementation and function of the imperative construction module that enables the system to carry out constructions.

**Chapters 7** describes the implementation and function of the propagator-based declarative geometry constraint solver that builds instances of diagrams satisfying declarative constraints.

**Chapter 6** describes the implementation and function of the perception module focused on observing interesting properties in diagrams. A key question involves determining "what is interesting".

**Chapter 4** describes the analyzer module which integrates results from the other systems to create new discoveries. Main features include filtering out obvious or known results to focus on the most interesting discoveries, the persistence and storage of definitions and theorems, and an interface to apply these findings to new situations.

**Chapter 9** discusses several of the definitions and theorems results the overall system has been able to discover and learn.

**Chapter 10** evaluates the strengths and weaknesses of the system. Future work and possible extensions are discussed.

# Chapter 2

# Motivation and Examples

Understanding elementary geometry is a fundamental reasoning skill, and encompasses a domain both constrained enough to model effectively, yet rich enough to allow for interesting insights. Although elementary geometry knowledge can be conveyed via series of factual definitions, theorems, and proofs, a particularly intriguing aspect of geometry is the ability for students to learn and develop an understanding of core concepts through visual investigation, exploration, and discovery.

These visual reasoning skills reflect many of the cognitive activities used as one interacts with his or her surroundings. Day-to-day decisions regularly rely on visual reasoning processes such as imagining what three dimensional objects look like from other angles, or mentally simulating the effects of one's actions on objects based on a learned understanding of physics and the object's properties. Such skills and inferred rules are developed through repeated observation, followed by the formation and evaluation of conjectures.

Similar to such day-to-day three-dimensional reasoning, visualizing and manipulating 2D geometric diagrams "in the mind's eye" allows one to explore questions such as "what happens if..." or "is it always true that..." to discover new conjectures. Further investigation of examples can increase one's belief in such a conjecture, and an accompanying system of deductive reasoning from basic axioms could prove that an observation is correct.

As an example, a curious student might notice that in a certain drawing of a

triangle, the three perpendicular bisectors of the edges are concurrent, and that a circle constructed with center at the point of concurrence intersects all three vertices of the triangle. Given this "interesting observation", the student might explore other triangles to see if this behavior is just coincidence, or conjecture about whether it applies to certain classes of triangles or all triangles in general. After investigating several other examples, the student might have sufficient belief in the conjecture to explore using previously-proven theorems (in this case, correspondences in congruent triangles) to prove the conjecture. My proposed project is a software system that simulates and automates this inductive thought process.

Automating geometric reasoning is not new, and has been an active field in computing and artificial intelligence. Dynamic geometry software, automated proof assistants, deductive databases, and several reformulations into abstract algebra models have been proposed in the last few decades. Although many of these projects have focused on the end goal of obtaining rigorous proofs of geometric theorems, I am particularly interested in exploring and modeling the more creative human-like thought processes of inductively exploring and manipulating diagrams to *discover* new insights about geometry.

The interactive computer system presented in this thesis emulates the curious student described above, and is capable of exploring geometric concepts through inductive investigation. The system begins with a fairly limited set of factual knowledge regarding basic definitions in geometry and provides means by which a user interacting with the system can "teach" the system additional geometric concepts and theorems by suggesting investigations the system should explore to see if it "notices anything interesting."

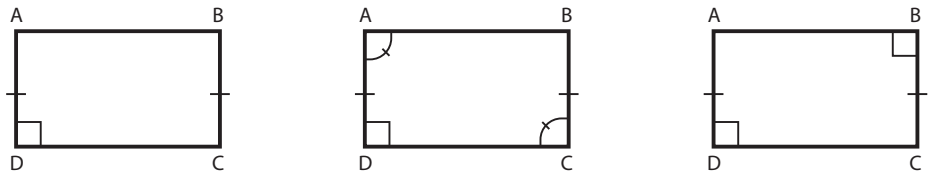
To evaluate its recognition of such concepts, the interactive system provide means for a user to extract the observations and apply such findings to new scenarios. In addition to the automated reasoning and symbolic artificial intelligence aspects of a system that can learn and reason inductively about geometry, the project also has some interesting opportunities to explore educational concepts related to experiential learning, and several extensions to integrate it with existing construction synthesis

and proof systems.

## 2.1 Manipulating Diagrams "In the Mind's Eye"

Although the field of mathematics has developed a rigorous structure of deductive proofs explaining most findings in geometry, much of human intuition and initial reasoning about geometric ideas come not from applying formal rules, but rather from visually manipulating diagrams "in the mind's eye." Consider the following example:

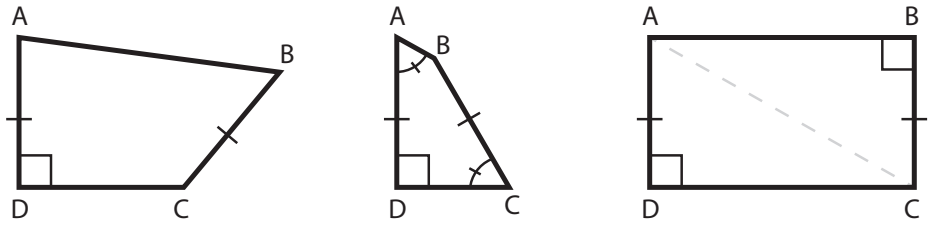
### 2.1.1 An Initial Example




**Example 1: Of the three diagrams above, determine which have constraints sufficient to restrict the quadrilateral $ABCD$ to always be a rectangle.**

An automated deductive solution to this question could attempt to use forward-chaining of known theorems to determine whether there was a logical path that led from the given constraints to the desired result that the quadrilateral shown is a rectangle. However, getting the correct results would require having a rich enough set of inference rules and a valid logic system for applying them.

A more intuitive visual-reasoning approach usually first explored by humans is to initially verify that the marked constraints hold for the instance of the diagram as drawn and then mentally manipulate or "wiggle" the diagram to see if one can find a nearby counter-example that still satisfies the given constraints, but is not a rectangle. If the viewer is unable to find a counter-example after several attempts, he or she may be sufficiently convinced the conclusion is true, and could commit to exploring a more rigorous deductive proof.




**Solution to Example 1: As the reader likely discovered, the first two diagrams can be manipulated to yield instances that are not rectangles, while the third is sufficiently constrained to always represent a rectangle. (This can be proven by adding a diagonal and using the Pythagorean theorem.)**

### 2.1.2 Diagrams, Figures, and Constraints

This example of manipulation using the "mind's eye" also introduces some terminology helpful in discussing the differences between images as drawn and the spaces of geometric objects they represent. For clarity, a *figure* will refer to an actual configuration of points, lines, and circles drawn on a page. Constraint annotations (congruence or measure) added to a figure create a *diagram*, which represents the entire space of figure *instances* that satisfy the constraints.

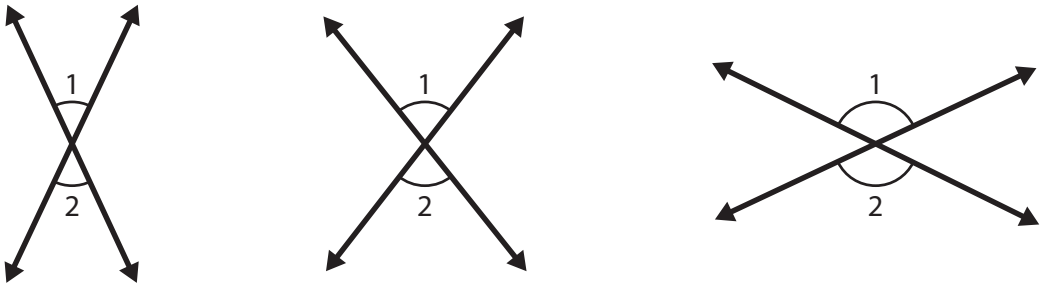
An annotated figure presented on a page is typically an instance of its corresponding diagram. However, it is certainly possible to add annotations to a figure that are not satisfied by that figure, yielding impossible diagrams. In such a case the diagram represents an empty set of satisfying figures.

In the initial example above, the three quadrilaterals figures are drawn as rectangles. It is true that all quadrilateral figures in the space represented by the third diagram are rectangles. However, the space of quadrilaterals represented by the first two diagrams include instances that are not rectangles, as shown above. At this time, the system only accepts diagrams whose constraints are satisfied in a given figure. However, detecting and explaining impossible diagrams, purely from their set of constraints could be an interesting extension.

## 2.2 Geometry Investigation

These same "mind's eye" reasoning techniques can be used to discover and learn new geometric theorems. Given some "interesting properties" in a particular figure, one can construct other instances of the diagram to examine if the properties appear to hold uniformly, or if they were just coincidences in the initial drawing. Properties that are satisfied repeatedly can be further explored and proved using deductive reasoning. The examples below provide several demonstrations of such inductive investigations.

### 2.2.1 Vertical Angles

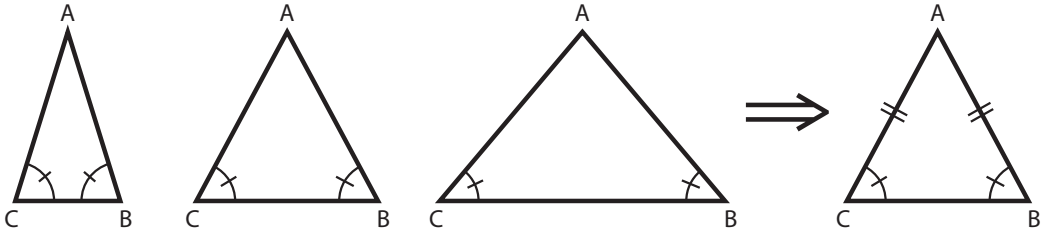



**Investigation 1: Construct a pair of vertical angles. Notice anything interesting?**

Often one of the first theorems in a geometry course, the fact that vertical angles are equal is one of the simplest examples of applying "mind's eye" visual reasoning. Given the diagram on the left, one could "wiggle" the two lines in his or her mind and imagine how the angles respond. In doing so, one would notice that the lower angle's measure increases and decreases proportionately with that of the top angle. This mental simulation, perhaps accompanied by a few drawn and measured figures, could sufficiently convince the viewer that vertical angles always have equal measure.

Of course, this fact can also be proved deductively by adding up pairs of angles that sum to 180 degrees, or by using a symmetry arguments. However, the inductive manipulations are more reflective of the initial, intuitive process one typically takes when first presented with understanding a problem.

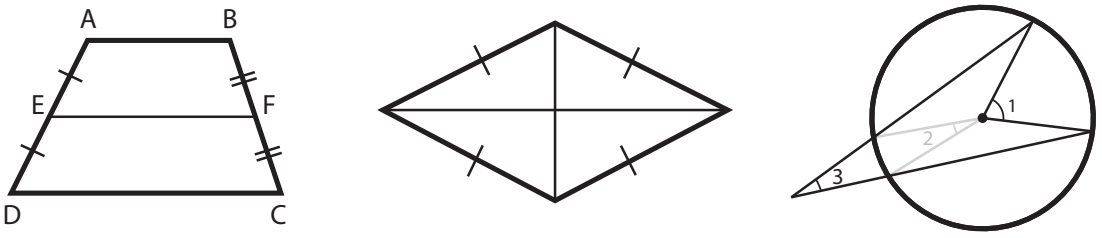
### 2.2.2 Elementary Results




**Investigation 2: Construct a triangle $ABC$ with $\angle B = \angle C$. Notice anything interesting?**

A slightly more involved example includes discovering that if a triangle has two congruent angles, it is isoceles. As above, this fact has a more rigorous proof that involves dropping an altitude from point $A$ and using corresponding parts of congruent triangles to demonstrate the equality of $AB$ and $AC$. However, the inductive investigation of figures that satisfy the constraints can yield the same conjecture, give students better intuition for what is happening, and help guide the discovery and assembly of known rules to be applied in future situations.

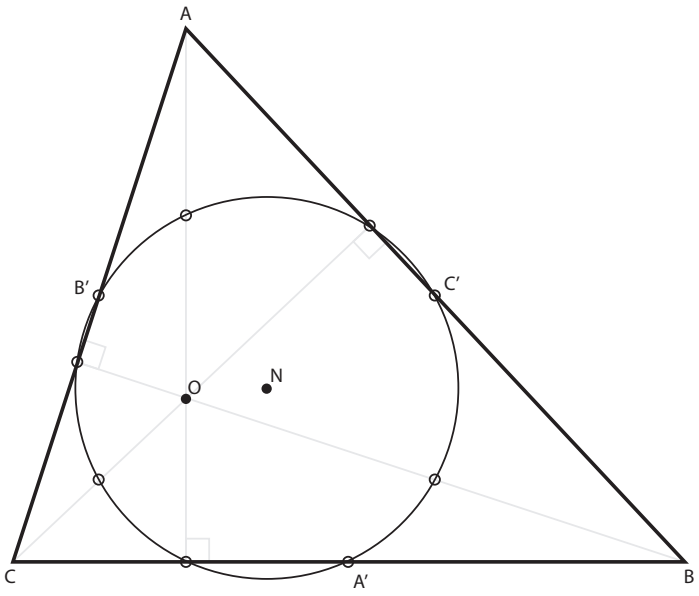
In this and further examples, an important question becomes what properties are considered "interesting" and worth investigating in further instances of the diagram, as discussed in section 3.6.3. As suggested by the examples in Investigation 3, this can include relations between segment and angle lengths, concurrent lines, collinear points, or parallel and perpendicular lines.




**Investigation 3: What is interesting about the relationship between $AB$, $CD$, and $EF$ in the trapezoid? What is interesting about the diagonals of a rhombus? What is interesting about $\angle 1$, $\angle 2$, and $\angle 3$?**

### 2.2.3 Nine Point Circle and Euler Segment

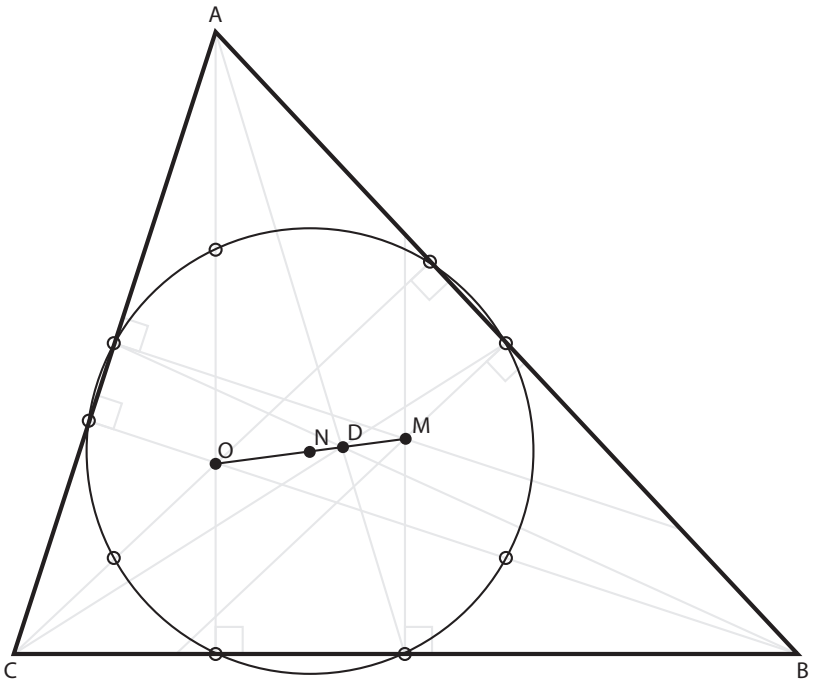
Finally, this technique can be used to explore and discover conjectures well beyond the scope of what one can visualize in his or her head:




**Investigation 4a: In triangle $ABC$, construct the side midpoints $A'$, $B'$, $C'$, and orthocenter $O$ (from altitudes). Then, construct the midpoints of the segments connecting the orthocenter with each triangle vertex. Notice anything interesting?**

As a more complicated example, consider the extended investigation of the Nine Point Circle and Euler Segment. As shown in Investigation 4a, the nine points created (feet of the altitudes, midpoints of sides, and midpoints of segments from orthocenter to vertices) are all concentric, lying on a circle with center labeled $N$.

Upon first constructing this figure, this fact seems almost beyond chance. However, as shown in Investigation 4b (below), further "interesting properties" continue to appear as one constructs the centroid and circumcenter: All four of these special points ($O$, $N$, $D$, and $M$) are collinear on what is called the *Euler Segment*, and the ratios $ON : ND : DM$ of $3 : 1 : 2$ hold for any triangle.




**Investigation 4b: Continue the investigation from 4a by also constructing the centroid $D$ (from medians) and circumcenter $M$ (from perpendicular bisectors). Notice anything interesting?**

# Chapter 3

# System Overview

Given this idea of manipulating diagrams "in the mind's eye" to explore and discover geometry theorems, I introduce the system components.

## 3.1 Goals

Theorems/Conjectures: Definitions, properties

Definitions "What is a rhombus?"

Properties "The diagonals of a parallelogram bisect one another" or

## 3.2 Representations

Construction Steps

Figure

Constraints

## 3.3 Modules

## 3.4 Example Interaction

## 3.5 Software Implementation

My MEng project involved building and analyzing a software implementation that can make inductive observations and discoveries in the domain of Elementary Geometry. Although there are many aspects involved in replicating the human-like reasoning discussed, my system focuses on a core set of four modules that is able to model the inductive exploratory behavior. Further abilities can be added as extensions or applications of the core system as described in section **??**.

### 3.5.1 Modules

These four modules include an imperative geometry construction interpreter used to build diagrams, a declarative geometry constraint solver to solve and test specifications, an observation-based perception module to notice interesting properties, and a learning module to analyze information from the other modules and integrate it into new definition and theorem discoveries.

## 3.6 Sample Interaction

This core system will provide an interpreter to accept input of construction instructions, an analytic geometry system that can create instances of such constructions, a pattern-finding process to discover "interesting properties", and an interface for reporting findings.

### 3.6.1 Interpreting Construction Steps

The first step in such explorations is interpreting an input of the diagram to be explored. To avoid the problems involved with solving constraint systems and the

possibility of impossible diagrams, the core system will take as input an explicit list of construction steps that results in an instance of the desired diagram. These instructions can still include arbitrary selections (let $P$ be some point on the line, or let $A$ be some acute angle), but otherwise will be restricted to basic construction operations using a compass and straight edge.

To simplify the input of more complicated diagrams, some of these steps can be abstracted into a library of known construction procedures. For example, although the underlying figures will be limited to very simple objects of points, lines, and angles, the steps of constructing a triangle (three points and three segments) or bisecting a line or angle can be encapsulated into single steps.

### 3.6.2 Creating Figures

Given a language for expressing the constructions, the second phase of the system will be to perform such constructions to yield an instance of the diagram. This process will mimic "imagining" manipulations and will result in an analytic representation of the figure with coordinates for each point. Arbitrary choices in the construction ("Let $Q$ be some point not on the line.") will be chosen via an random process, but with an attempt to keep the figures within a reasonable scale to ease human inspection.

### 3.6.3 Noticing Interesting Properties

Having constructed a particular figure, the system will need to be able to examine it to find interesting properties. These properties involve facts that appear to be "beyond coincidence". As mentioned in section 2.2.2, this generally involves relationships between measured values, but can also include "unexpected" configurations of points, lines, and circles. As the system discovers interesting properties, it will reconstruct the diagram using difference choices and observe if the observed properties hold true across many instances of a diagram.

### 3.6.4 Reporting Findings

Finally, once the system has discovered some interesting properties that appear repeatedly in instances of a given diagram, it will need a means of reporting its results to the user. Although this could easily be a simple list of all simple relationships, some effort will be taken to avoid repeating observations that obvious in the construction. For example, if a perpendicular bisector of segment $AB$ is requested, the fact that it bisects that segment in every instance is not informative. To do so, the construction process will also maintain a list of facts that can be reasoned from construction assumptions so that these can be omitted in the final reporting.

# Chapter 4

# Learning Module

## 4.1 Overview

The Learning and learning module is one the core elements integrating information from the other components of the system. It maintains

## 4.2 Interactions with Learning Module

### 4.2.1 Querying

```
(what-is 'parallelogram')
```

### 4.2.2 Learning Definitions

```
(learn-term 'parallelogram random-parallelogram)
```

### 4.2.3 Applying Learned Properties

```
(analyze-figure figure-proc)
```

## 4.3 Representing Discoveries

For example, assertions about equality of segments |AB| = |CD| are independent of the ordering of points within the elements.

### 4.3.1 Pattern Matching against existing conjectures

Based on dependencies, we replace the lowest-level random dependencies with arbitrary pattern elements `(? s1 ,segment)` for instance. Then, when new conjectures are being considered, we attempt to pattern match based on existing elements to see if there is a redundant observation.

# Chapter 5

# Imperative Construction System

## 5.1 Overview

The first module is an imperative system for performing geometry constructions. This is the typical input method for generating coordinate-backed instances of figures and thus declares.

We will first discuss the basic underlying structures that comprise figures, then describe the higher-order language used to specify construction steps.

## 5.2 Basic Structures

### 5.2.1 Points

Points form the basis of most elements. Throughout the system, points are

## 5.3 Linear Elements

The linear elements of Segments, Lines, and Rays are built upon points. Initially the internal representation of

To better specify angles (see below), all linear elements, including segments and lines are directioned.

### 5.3.1 Angles

Initially angles were represented as three points, now vertex + two directions. CCW orientation. Methods exist to determine them from various pairs of linear elements, uses directionality of linear elements to determine which "quadrant" of the angle is desired.

[Future]: Extract angles from the diagrams in analysis rather than specifying each angle of interest while creating the diagram.

### 5.3.2 Math Support

Some "core" math structures to help these calculations: Direction represents a direction in [0, 2pi], fixes principal value [0, 2pi], and support various operations for direction intervals (basic intersection, adding, shifting, etc). Currently all represented by single theta value, could generalize via generics to dx, dy, or theta depending on computation source.

## 5.4 Higher-level structures

In addition to the basic geometry structures, the system uses several grouping structures to combine and abstract the basic figure elements into higher-level figures elements.

### 5.4.1 Polygons

Polygons are represented as groups of points.

### 5.4.2 Figures

Figures are currently groups of elements. In the creation of figures we extract additional information and build a graph out of adjacent components for use in the analysis stages.

## 5.5 Construction Operations

### 5.5.1 Traditional constructions

Midpoint, perpendicular line, bisectors

### 5.5.2 Intersections

Generic intersections, mathematically based at line/line or line/circle at the core. Other intersections also add the check that the resulting point(s) are on the elements.

### 5.5.3 Measurement-based operations

A "Ruler + Protractor" is generally not permitted in traditional construction problems. However, sometimes its nice to be able to use measurements to more quickly compute a result (e.g. angle bisector by halving angle) vs. going through the whole ray/circle based construction process.

### 5.5.4 Transformations

Currently, rotate about a point or translate by a vector. Also interfaces for by *random* point or vector.

## 5.6 Randomness

### 5.6.1 Random Choices

At the basis of all random

### 5.6.2 Remembering choices

### 5.6.3 Bakktracking

Currently, the system does not backtrack based on random choices. However, there are plans to perform checks on randomly-generated elements that are too close to one another and to retry the random choice to avoid degenerate choices.

### 5.6.4 Avoiding almost-degenerate points

As discussed above, randomly making choices in

### 5.6.5 Animating choices

I animate over a small range within the specified random range. Top-level infrastructure determinies frames, sleeping, etc. Constructions can request to animate functions of one arg [0, 1]. As the figure and animation is run, each call to randomize gets a call to random whenever their value is non-false.

## 5.7 Dependencies

### 5.7.1 Implementation

Eq-properties, etc.

### 5.7.2 Naming

Sometimes derived if unknown, figure out how name metadata relates to the dependencies.

### 5.7.3 Forcing higher-level random dependenceis

"Inverts" the dependency tree that would otherwise usually go down to points. set-dependency! as random-square. When given an element by the teacher, generally we

don't know how the construction was performed.

### 5.7.4 Dependency-less diagrams

In some cases, the dependency structure of a figure can be wiped.

## 5.8 Construction Language

Constructions and instruction-based investigations are specified by scheme procedures that return the desired figures.

### 5.8.1 Macros

I created a let-geo* special form that is similar to Scheme's (let ...) form, but sets the element names as specified so they can be more easily referred to later.

### 5.8.2 Multiple Assignment

In let-geo*, I also permit some constructions to optionally map to multiple assignments of names, such as the case in which you create a triangle and simulatneously want to store and name the triangle's vertex points.

## 5.9 Graphics

### 5.9.1 MIT Scheme X Graphics

### 5.9.2 Bounds

# Chapter 6

# Observation Module

## 6.1 Overview

The observation module is primarily concerned with the task of observing interesting properties in diagrams.

### 6.1.1 Extracting more information

The observation module also builds and traverses a graph-representation of the object of connectedness and adjacencies to extract more segments and angles, or include intersections of elements in its investigation.

#### Auxillary Segments

In some circumstances, the system can insert and consider segments between all pairs of points. Although this can sometimes produce interesting results, it can often lead to too many elements being considered. This option is off by default but can be enabled in a self-exploration mode.

### 6.1.2 What is Interesting?

Concurrent points, collinear points, equal angles, supplementary/complementary angles, parallel, perpendicular elements, concentric points, (future:) ratios between mea-

surements, etc.

### 6.1.3 Removing Obvious Properties

This module makes use of available dependency information to eliminate some obvious properties. At this phase, the eliminations arise only from basic geometry knowledge "hard-coded" into the system, and not upon any specific prior-learned formula.

#### Trivial relations

Points being on lines, segments, circles directly dependent on that point.

#### Branch Relations

Other examples include "branch" relations. [TODO REF: Chen, Song, etc.]. ABCD on a line with AB = CD also means that AC = BD, for instance.

# Chapter 7

# Declarative Geometry Constraint Solver

## 7.1 Overview

The final module is a declarative geometric constraint solver. Given a user-specified topology of the diagram and various constraints, this system is able to solve those constraints and instantiate a diagram that satisfies them if possible.

This system is implemented using propagators, involved the creation of new partial information about point regions and direction intervals, and focuses on a

Future efforts involve a backtrack-search mechanism if constraints fail, and a system of initializing the diagram with content from an existing figure, kicking out and wiggling arbitrary premises, and seeing how the resulting diagram properties respond.

## 7.2 Mechanical Analogies

The geometry constraint solver - physical manipulation, simulation, and "wiggling"

### 7.2.1 Bar and Joint Linkages

Bars have endpoints, directions and length. Joints have a vertex point and two directions. Currently, most joints are directioned and have max value of 180 degrees.

### 7.2.2 Mechanism

The Mechanism in our declarative system is analogous to Figure, grouping elements. Also computes various caching and lookup tables to more easily access elements.

## 7.3 Partial Information

### 7.3.1 Regions

Propagating partial information across bars and joints yields a new region system: Regions include point sets of one or more possible points, an entire ray, or an entire arc. These rays and arcs are from an anchored bar with only one of direction or length specified, for instance.

### 7.3.2 Direction Intervals

Ranges of intervals. Full circle + invalid intervals. Adding and subtracting intervals of direction and thetas gets complicated at times.

Challenges with intersection, multiple segments. Eventually just return nothing is okay.

## 7.4 Propagator Constraints

### 7.4.1 Basic Linkage Constraints

Direction, dx, dy, length, thetas.

### 7.4.2 Higher Order Constraints

Angle sum of polygon, or scan through polygon and ensure that the angles don't not match. Example is equilateral triangle, for instance... Could also observe always "60 degrees" as an interesting fact and put that in as a constraint. They're alebgraically quite similar, but my propagators currently don't perform symbolic algebra.

## 7.5 Solving: Specification Ordering

Given a wired diagram, process is repeatedly specifying values for elements

### 7.5.1 Anchored vs. Specified vs. Initialized

## 7.6 Backtracking

TODO in future, sounds fun.

## 7.7 Interfacing with existing diagrams

TODO in future, sounds fun.

## 7.8 Specification Interface

**Establish Polygon Topology**

Nice techniques for establishing polygon topology.

# Chapter 8

# Related Work

The topics of automating geometric proofs and working with diagrams are areas of active research. Several examples of related work can be found in the proceedings of annual conferences such as *Automated Deduction in Geometry* [**?**] and *Diagrammatic Representation and Inference* [**?**]. In addition, two papers from the past year combine these concepts with a layer of computer vision interpretation of diagrams. Chen, Song, and Wang present a system that infers what theorems are being illustrated from images of diagrams [**?**], and a paper by Seo and Hajishirzi describes using textual descriptions of problems to improve recognition of their accompanying figures [**?**].

Further related work includes descriptions of the educational impacts of dynamic geometry approaches and some software to explore geometric diagrams and proofs. However, such software typically uses alternate approaches to automate such processes, and few focus on inductive reasoning.

## 8.1 Dynamic Geometry

From an education perspective, there are several texts that emphasize an investigative, conjecture-based approach to teaching such as *Discovering Geometry* by Michael Serra [**?**], the text I used to learn geometry. Some researchers praise these investigative methods [**?**] while others question whether it appropriately encourages deductive reasoning skills [**?**].

## 8.2 Software

Some of these teaching methods include accompanying software such as Cabri Geometry [?] and the Geometer's Sketchpad [?] designed to enable students to explore constructions interactively. These programs occasionally provide scripting features, but have no proof-related automation.

A few more academic analogs of these programs introduce some proof features. For instance, GeoProof [?] integrates diagram construction with verified proofs using a number of symbolic methods carried out by the Coq Proof Assistant, and Geometry Explorer [?] uses a full-angle method of chasing angle relations to check assertions requested by the user. However, none of the software described simulates the exploratory, inductive investigation process used by students first discovering new conjectures.

## 8.3 Automated Proof and Discovery

Although there are several papers that describe automated discovery or proof in geometry, most of these use alternate, more algebraic methods to prove theorems. These approaches include an area method [?], Wu's Method involving systems of polynomial equations [?], and a system based on Gröbner Bases [?]. Some papers discuss reasoning systems including the construction and application of a deductive database of geometric theorems [?]. However, all of these methods focused either on deductive reasoning or complex algebraic reformulations.

# Chapter 9

# Results

## 9.1 Overview

Still working on getting some good results to talk about... Isoceles triangle angles vs. theorems, bisectors of kite, lots of cool collinear / concurrent points in Triangles, for instance.

# Chapter 10

# Conclusion

## 10.1 Overview

To be concluded ...

## 10.2 Extensions

Possible extensions include integrating with existing automated proof systems (Coq, etc.)

Also: learning construction procedures from the declarative constraint solver.

# Bibliography