Journal of Automated Reasoning **32:** 3–33, 2004.

GRAMY: A Geometry Theorem Prover Capable of Construction

NOBORU MATSUDA and KURT VANLEHN*
Intelligent Systems Program, University of Pittsburgh.
e-mail: mazda@pitt.edu, vanlehn@cs.pitt.edu

Abstract. This study investigates a procedure for proving arithmetic-free Euclidean geometry theorems that involve construction. "Construction" means drawing additional geometric elements in the problem figure. Some geometry theorems require construction as *a part of the proof*. The basic idea of our construction procedure is to add only elements required for applying a postulate that has a consequence that unifies with a goal to be proven. In other words, construction is made only if it supports backward application of a postulate. Our major finding is that our proof procedure is semi-complete and useful in practice. In particular, an empirical evaluation showed that our theorem prover, GRAMY, solves all arithmetic-free construction problems from a sample of school textbooks and 86% of the arithmetic-free construction problems solved by preceding studies of automated geometry theorem proving.

Key words: automated geometry theorem proving, construction, search control, constraint satisfaction problem, intelligent tutoring system.

1. Introduction

Geometry theorem proving has been a challenging problem for automated reasoning systems. Indeed, some of the earliest work in automated reasoning used geometry theorem proving as the task domain (Gelernter, 1959; Gelernter et al., 1963; Reiter, 1972; Wong, 1972), and work has continued to the present time (see, for example, Chou et al., 2000). A particularly challenging issue is to prove theorems that require constructions, namely, to find proofs with additional lines, points, or arcs constructed by a compass and a straightedge. No robust and efficient method for geometry theorem proving with construction is known so far. Finding a construction is a hard task even for human problem solvers. Since one can draw many segments and arcs at any point of a proof, the search space is enormous.

Our long-term goal is to build an intelligent tutoring system for elementary geometry. Any proofs and constructions found by our automated geometry theorem prover must be stated with the common ontology of Euclidean geometry – the axiomatized geometry system taught in schools. Thus, the algebraic techniques

* This research was supported by NSF grant number 9720359 to CIRCLE, Center for Interdisciplinary Research on Constructive Learning Environment.

used in some earlier studies on geometry theorems (e.g., the area method discussed in Chou et al., 1996) are not suitable for our purpose. The simplicity of the proof procedure and the readability of the proofs are also important design issues.

The desired intelligent tutoring system will be built with a standard technique called model tracing (Anderson et al., 1990) that requires a "complete" model of reasoning just as an instructor is supposed to be "omniscient." If the student enters a step that is acceptable to instructors but the tutoring system does not have it in its model's reasoning, then the tutoring system will tell the student that the step is unacceptable, which could have devastating consequences for learning. Thus the desired geometry theorem prover must not only be able to find a single comprehensible proof, it should also be able to find all proofs that are considered acceptable to instructors. Accordingly, ad hoc heuristics that reduce the search space are unacceptable because they may exclude proofs that the tutoring system needs.

We discovered a construction procedure that is simple and involves no ad hoc heuristics. We incorporated it into an automated geometry theorem prover called GRAMY. In addition to presenting the construction procedure and GRAMY, this paper addresses the following research questions:

- Is the construction procedure complete?
- Is the construction procedure efficient enough to run on a personal computer?

In the following sections, we first review the basic issues of geometry theorem proving and give a brief history of the study. We then introduce our proof and construction procedure in Section 3. GRAMY's performance is evaluated in Section 4, followed by discussion and future work in Section 5.

2. Overview of Geometry Theorem Proving with Construction

2.1. THE PROBLEM AND DEFINITIONS

The target domain is elementary Euclidean geometry. In this study, we deal only with proofs of equality and congruence that do not involve arithmetic operations (i.e., sums and multiplications).* Typical elements are points, segments, angles, triangles, and their quantitative properties (e.g., length of a segment, degree of an angle). Typical relations among elements are equivalence, congruence, perpendicularity, parallelism, coincident (X and Y intersect at Z), membership (X is a part of Y), and their negations.

A problem consists of a set of given propositions, a proposition to be proved, and a diagram called the problem figure. GRAMY represents propositions as formulae in restricted first-order logic. Diagrams both for problem figures and for other purposes are represented by the Cartesian coordinate system. A proposition is

* This restriction implies that the proofs of inequalities, ratios, and coincident intersections (i.e., to prove that three or more segments are intersecting at one point) are also excluded from the present study.

called *quantitatively satisfied* if the relation stated in the proposition is consistent with the measurements of the corresponding geometric elements in the problem figure. For example, a proposition $AB = CD$ is quantitatively satisfied if two segments AB and CD in the diagram are approximately the same length.

In this article, the term "postulates" refers to the true statements given to the theorem prover such as definitions, axioms, and theorems that have been shown to be true. A postulate consists of premises and a consequence that are represented as propositions. Each postulate is associated with a generic diagram that represents topological information that is not explicitly represented in the premises and the conclusions.

The output from GRAMY is proofs, each of which is a sequence of postulate applications with or without construction. Construction is represented by a set of geometric elements that have been added to the problem figure. The prover must ensure that the construction can be drawn with a compass and a straightedge. An example of invalid construction is to add lines that divide a given angle into three equal angles, as trisection of an angle is known not to be possible with a compass and a straightedge.

GRAMY finds proofs and constructions that hold within a given problem figure. In general, one must find a universal proof that, in some case, must be conditional, thus requiring different problem figures that are consistent with the given propositions. In most cases, however, classroom instruction requires students to find a proof only for a particular problem figure and does not ask them to make conditional proofs, so that is all GRAMY does.

Geometry theorem proving in general can be viewed as a state-space search. Major components of a state are a problem figure, a set of propositions either given or derived, and goals to prove. There are two kinds of operators to search through the problem space: the operators for deduction and for construction. An application of a deduction operator corresponds to an application of geometric postulates, which changes either the set of propositions by forward chaining or the set of goals by backward chaining. An application of a construction operator changes the problem figure in a way that corresponds to construction by a compass and a straightedge.

Since this study does not deal with arithmetic operators, applying a postulate introduces no new geometric elements; hence, a proof has finitely many geometric elements without construction. Furthermore, since there are finitely many relations, there are finitely many possible relationships among the elements and hence finitely many true propositions. The implication is that one can apply postulates at most only finitely many times with no repetitive deductions. Therefore, regardless of whether a proof is found or not, there is a state where no new propositions can be derived. We call this state a *quiescence state*. The propositions in a quiescence state are the *deductive closure* of the given propositions with respect to the given postulates. In this paper, forward chaining from an arbitrary state to a quiescence state is called *exhaustive forward chaining*. If a proof exists without construction,

then there must be a proposition in the quiescence state that unifies with the goal to prove.

2.2. BRIEF HISTORY OF STUDIES OF AUTOMATED THEOREM PROVING IN GEOMETRY

Although different types of geometry theorem provers have been developed so far, this section surveys only those studies that involve construction.

2.2.1. *Proof Method: Axiomatic vs. Algebraic*

The previous provers fall into one of two categories. The *axiomatic approach* uses a set of axioms and inference rules to formulate a sequence of deductions as a proof. Also, it typically uses heuristics to prune or guide the search. The *algebraic approach* translates a set of postulates as well as a theorem to prove into equations in such a way that solving the equations corresponds to finding a proof for the theorem.

The algebraic approach is known to be powerful in that it can find proofs for very hard problems. Examples of practical provers based on the algebraic approach include the characteristic set method (Wu, 2000), the elimination method (Wang, 1995), the Gröbner basis method (Kapur, 1986), and the Clifford algebra approach (Li, 2000). A drawback to these efficient approaches is that they are seldom comprehensible for students learning Euclidean geometry. As an exception, the area method (Chou et al., 1996) is axiomatized by so-called area axioms that are intuitive enough for students to understand. As a result, the theorem prover can output a "readable" proof as a sequence of transitions via axiom applications. A drawback is that the students must be taught the area axioms instead of the standard Euclidean axioms. Hence, a theorem prover based on the area method is hardly an appropriate application for geometry education in current school systems.

One interesting aspect of the area method is that for a certain class of geometry, the area method proves problems that require construction. This class is characterized as constructive geometry (Chou and Gao, 1989). However, there are problems that do not fall into the constructive geometry class that can be proved by Euclidean axioms with construction (for example, problem P108 shown in Appendix A).

Most automated theorem provers that deal with construction use the axiomatic paradigm. They all add geometric elements to the problem figure so that a desired postulate will apply. The challenge is to focus only on a limited number of constructions. Most provers utilize ad hoc heuristics to avoid unproductive constructions. One of the educational defects of those approaches is that they force students to learn many ad hoc heuristics as well. Another problem is that it is hard to show the completeness for theorem provers that use ad hoc heuristics. The next few sections review construction heuristics and other aspects of the axiomatic approach.

2.2.2. *Heuristics of Construction*

Heuristics of construction can be divided into three types: (1) constructing new segments by only connecting existing points, (2) drawing segments and points that are sufficient to make desired postulates applicable, and (3) applying ad hoc construction heuristics.

Connecting Existing Points. The simplest heuristic is to draw a new line by connecting any existing two points (Anzai et al., 1979; Elcock, 1977; Gelernter, 1959; Greeno et al., 1979). This heuristic does not introduce new points except the intersections of new segments and existing ones. Hence a construction procedure that applies only this heuristic is not complete.

Making a Desired Postulate Applicable. In many construction problems, a particular postulate has a consequence that unifies with the goal to prove but its entire configuration does not match with the problem figure.* Hence, one heuristic is to add elements to the figure that will enable the postulate to apply. Most of the construction procedures in the literature fall into this category. Two major issues here are (1) selecting a desired postulate to apply and (2) ensuring that the constructed elements can be drawn with a compass and a straightedge. Most previous work resolves both issues by considering construction only for a particular postulate. Nevins (1975) implemented construction only for the right-angle triangle axiom. Elcock (1977) designed a procedure to construct a new point as an intersection of two existing segments. Greeno et al. (1979) implemented a construction for congruent triangles sharing a segment. Coelho and Pereira (1986) implemented construction only for the quadrilateral axiom.

Ad Hoc Heuristics. Performance of a theorem prover could be improved by adding ad hoc heuristics for construction. For example, Wong (1972) proposed a heuristic, called midpoint reflection, that says, "If there exists a segment AB such that one of its end points B is a midpoint of segment XY, then construct a new point by reflecting A around XY." This heuristic leads Wong's prover to the construction for Problem P103 shown in Appendix A. Our proof procedure identified that this problem requires three constructions for three different postulates, but because of a technical issue in its implementation, GRAMY could not solve this problem. Although Wong's heuristic makes much sense, because of the excessive generality we doubt it would work in a practical situation without a search explosion (indeed, Wong's heuristic has never been implemented). Another example of ad hoc heuristics can be seen in AUXIL (Suwa and Motoda, 1989). AUXIL requires a human problem solver to input proofs with construction. It then learns how to make the construction by generalizing the operations used in the proofs. The acquired construction operators and heuristics can be so complicated that they are hardly

* A counterexample of this observation is the construction with a transitive substitution, described in Section 3.2.5.

teachable. Hence we doubt that this type of prover can be a building block of a tutoring system, even though it is computationally tractable (see Suwa and Motoda (1989) for more about their so-called frustration-based search control technique). An interesting study was conducted by Chou et al. (2000) in which they built a powerful theorem prover capable of construction that can solve very hard problems. The construction procedure is implemented as a set of rules that specify how to construct a new point in a certain situation. Unfortunately, because their construction rules are not goal oriented, and the prover has the potential to make many unsuccessful constructions. Furthermore, their proof procedure works only for the class of constructive geometry, and therefore it has the same limitation as the area method mentioned in the previous section.

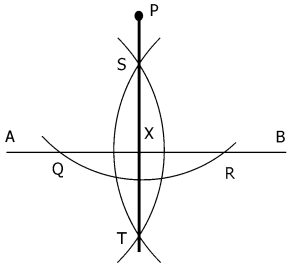
2.3. WHY IS CONSTRUCTION SO HARD?

To understand why theorem proving with construction is so hard, consider the following brute-force proof procedure:

1. Apply exhaustive forward chaining without construction.
2. If a proof is found, namely, if there exists a derived proposition that unifies with the goal, then output the proof and quit.
3. Apply every applicable *primitive construction operator* once to the problem figure, and make a new state with the modified figure. A primitive construction operator is either (1) drawing a line between two existing points or (2) drawing an arc of a specified radius about an existing point.
4. Go to step 1.

If a proof with construction does exist, then the necessary construction should be made with a finite number of applications of the primitive construction operators, which takes place during the third step mentioned above. Thus, the above procedure will eventually find the proof. However, there are usually a large number of applicable primitive construction operators at step 3, which greatly expands the problem figure. This means that exhaustive forward chaining may produce a very large number of propositions when it next applies at step 1. In short, this proof procedure is complete but not practical.

We need to restrict the application of construction operators so that useless construction will never be made. One approach to making construction more efficient is to combine several primitive operators into a macro operator that does only "meaningful" construction and uses only such macro operators. To drop a perpendicular line from a given point to a given line is an example of a macro operator. Furthermore, the search can be made more efficient if macro operators contain more information than just a sequence of primitive operators. For example, consider again the construction of a perpendicular line from a given point to a given segment. As shown in Figure 1, this construction is done by a macro operator that comprises a sequence of seven primitive operators. This sequence does not hold



1. Draw arc with P as the center.
2. Plot two intersections, Q and R, of the arc and AB.
3. Draw an arc with Q as the center.
4. Draw an arc with R as the center.
5. Plot the intersection points, S and T, of those arcs.
6. Draw a segment connecting S and T.
7. Plot intersection of AB and ST as X.

Figure 1. Construction for a perpendicular by primitive operators.

enough information to prove that $\angle PXB$ is a right angle, which requires drawing auxiliary segments *SQ*, *QT*, *TR*, and *RS*. Instead of forcing the theorem prover to find this additional construction, we can augment the macro operator to assert the right angles as true propositions. This augmentation not only improves the efficiency but also provides a rationale for the construction.

However, replacing primitive construction operators with macro operators does not settle the explosion of the search. We tested the brute-force procedure described above with five macro operators: copying a distance onto a line, dropping a perpendicular from a point to a line, drawing a perpendicular to a point on a line, drawing a parallel line, and plotting a midpoint. On Problem P123 in Appendix A, for instance, the average number of macro operators applicable at each state was 78. These macro operators did not constrain the search enough.

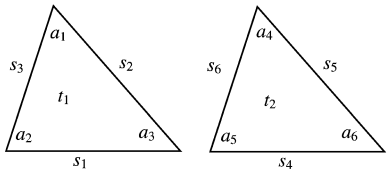
In sum, a brute-force search for construction would not be computationally tractable even with macro operators. We need to add more conditions that tell us when to apply the operators. The next section describes the approach used in GRAMY.

3. Proof and Construction Procedures in GRAMY

This section describes the whole proof procedure with a construction technique implemented in GRAMY. A discussion on its completeness follows.

3.1. KNOWLEDGE REPRESENTATION

In GRAMY, postulates that share the same topological configuration of points and lines are composed into a single knowledge piece, called a *diagrammatic schema* (DS). A DS consists of a diagram representing the topological configuration of the geometric elements involved in a DS, a set of propositions representing geometric relations (equal, parallel, perpendicular, etc.) that refer to the elements in the diagram, and deductive statements in the form "if a set of (possibly zero) premises holds, then a set of consequences holds" where the premises and the consequences refer to the geometric propositions. Figure 2 shows an example of a DS that rep-



Proposition:	**Deduction**:
1. $t_1 \equiv t_2$	5, 6, 7 → 1
2. $a_1 = a_4$	2, 6, 7 → 1
3. $a_2 = a_5$	2, 3, 7 → 1
4. $a_3 = a_6$	1 → 2, 3, 4, 5, 6, 7
5. $s_1 = s_4$	
6. $s_2 = s_5$	
7. $s_3 = s_6$	

Figure 2. Example of a diagrammatic schema (DS).

resents the postulates related to triangle congruence. Seven relations are shown under "Proposition" and four deductive statements under "Deduction." Of those four deductive statements, the first three statements show the conditions for two triangles to be congruent. The last statement represents a postulate that says, "If two triangles are congruent, then corresponding segments and angles are equal."

The topological configuration associated with each DS is called a *DS diagram*. We use the knowledge representation technique developed in Perdix (Greeno et al., 1979) to represent the DS diagram as a semantic network. The nodes correspond to geometric elements in the DS diagram (points, segments, rays, lines, angles, triangles, quadrangles, etc.) and the links correspond to relations between two elements (e.g., a point is an end point of a segment). For example, associated with the DS for the triangle-congruent postulate are 6 points, 6 segments, 12 rays, 6 angles, and 2 triangles. The problem figure is represented the same way.

Geometric propositions are represented with first-order predicates. The arguments of a predicate are geometric elements in a problem figure or a DS diagram. For example, if two segments *AB* and *CD* are represented as the nodes s_1 and s_2 in the semantic network, then the proposition $AB = CD$ is represented as $\mathtt{eq}(s_1, s_2)$. Although many provers use only points as arguments, GRAMY's representation reduces the combinatorics that would otherwise be caused by unifying coreferring expressions such that $AB = CD$ and $BA = DC$.

3.2. CONSTRUCTION PROCEDURE GUIDED BY DIAGRAMS

A proof is a sequence of postulate applications. Hence, if construction is made for a proof, it is made so that some postulate, which otherwise is not applicable, can be applied. In other words, the construction involves adding segments that are not part of the original problem figure but are necessarily involved in postulate applications in the proof.

The above observation implies that knowing which postulates are used in a proof is sufficient to determine the target elements of construction. Three issues arise: (1) how to find a set of postulate applications that constitutes a proof, (2) how to identify the segments required making those postulates applicable, and (3) how to construct (i.e., how to calculate coordinates of the end points of) those missing segments. These issues are discussed in the next three sections.

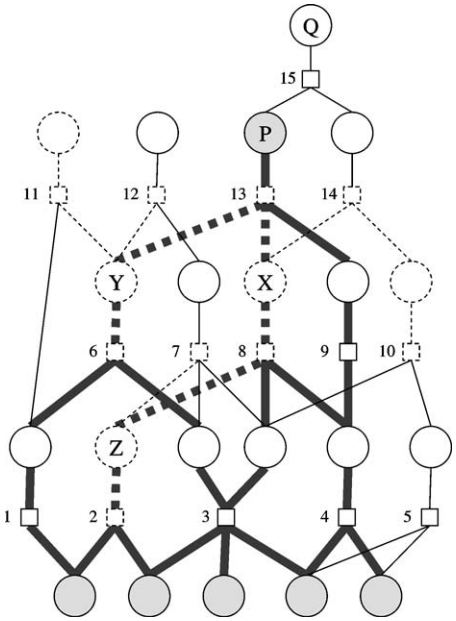


Figure 3. A dependency network representing a proof with construction.

3.2.1. *Identifying Postulates in a Proof*

A postulate is called *useful* if its consequence unifies with the goal to prove and all premises that match with the problem figure are quantitatively satisfied, whereas some premises might not match with the problem figure. The basic idea on controlling a search for construction is that *GRAMY attempts construction only when required for the backward application of a "useful" postulate*. This idea is best explained with an example, depicted in Figure 3. The figure shows a dependency network produced by exhaustive forward chaining on a geometry problem that has five given propositions shown as the gray circles at the bottom row. A small square shows a postulate application. The links coming into a square from the bottom correspond to premises of a postulate application, and the links going out from the top correspond to its consequences. White circles are derived propositions. If the problem is solvable, then one of the derived propositions must unify with the goal to be proven. Let P be a proposition that unifies with the goal. Thus, the heavier links, both solid and dotted, show a proof for the proposition P. Hence exhaustive forward chaining would eventually find a proof for the problem. The lighter links indicate the extra work done by exhaustive forward chaining, namely, postulate applications that do not appear in the proof.

Now we extend the above model by introducing construction. Let F be the problem figure required to prove P, namely, all geometric elements appearing in the proof of P are in F. Suppose that we remove a few segments from F, taking care that they can be added back by construction. Some of the propositions can no

longer hold in the resulting figure (e.g., if segment AB is removed, then $AB = CD$ does not hold). Assume that those propositions are indicated by the dotted circles in Figure 3. Consequently, postulate applications that involve dotted circles, indicated by the dotted squares, cannot apply either.

To find a proof, that is, to find postulate applications in the heavier links both solid and dotted, one must construct the segments that have been removed. To avoid search explosion, we would like the theorem prover to perform construction only for the postulate applications specified by heavier dotted links. More precisely, the theorem prover must first identify that postulate applications 13, 8, 6, and 2 are part of the proof and construct only those segments such that propositions X, Y, and Z hold in the resulting figure.

One approach is to first apply exhaustive forward chaining without construction, then backward chain once with a useful postulate. We then apply construction operators so that the diagram associated with the useful postulate matches the problem figure completely. Now we have a new state with a new problem figure and new goals. Since the problem figure has changed, forward chaining may assert new propositions. So, we apply exhaustive forward chaining followed by backward chaining with construction again, and we keep repeating this cycle until a proof is found.

In Figure 3, postulate applications 1, 3, 4, 5, and 9 occur during initial forward chaining. Assume there is no postulate applicable at this point. The prover then makes a construction so that backward chaining for a useful postulate application 13 would succeed. If this construction eventually makes propositions X, Y, and Z appear in the problem figure, then succeeding exhaustive forward chaining causes postulate applications 2, 6, and 8, which completes a proof.

It is not necessarily the case that a desired construction can be made by a useful postulate relative to the top-level goal, namely, postulate application 13 for P in the above example. Consider a problem that has the same five givens as shown at the bottom of Figure 3 but has Q as a goal to prove instead of P. In this case, the prover must do postulate application 15 backwards to create P as a subgoal of Q. In general, a theorem prover might need to apply backward chaining multiple times before carrying out construction.

Once a useful postulate is determined, there arise two technical issues to make use of the above idea: (1) matching the DS diagram to the problem figure partially, which in turn identifies DS segments that need to be constructed, and (2) discovering a sequence of construction operators that constructs these segments.

3.2.2. *Identifying Missing Segments to Be Constructed*

Intuitively, overlapping a DS diagram associated with a selected useful postulate with the problem figure reveals missing segments necessary to apply the useful postulate. This overlapping must be constrained so that the consequence of the useful postulate overlaps the goal to be proven. For example, if the goal is to prove $AB = DE$, then the triangle-congruent postulate $\triangle uvw \equiv \triangle xyz \rightarrow uv = xy$ is

useful. In this case, one may wish to overlap two triangles, $\triangle uvw$ and $\triangle xyz$, onto the problem figure so that the segment uv overlaps AB and xy overlaps DE. The segments in the DS diagram that cannot be overlapped indicate which segments must be added to the problem figure.

As mentioned in Section 3.1, DS diagrams are represented as semantic networks in GRAMY. Overlapping is equivalent to finding a *partial match* between a semantic network representing a problem figure and a semantic network representing the DS diagram of a useful postulate. A pair of an element in the DS diagram and an element of the problem figure is called a *binding*. A partial match is a set of bindings, which we hereafter call a *binding list*. An element in a DS diagram involved in a binding list is said to be *bound*. For the sake of explanation, we need to discriminate between the elements in the problem figure and the ones in the DS diagram; for example, a *problem point* refers to a point in the problem figure, whereas a *DS point* refers to a point in the DS diagram.

Although finding a partial match for two directed graphs is NP-hard, we have achieved adequate performance by formalizing it as a constraint satisfaction problem and solving it with a forward-checking backtracking algorithm (Haralick and Elliott, 1980) as described below.

Assume that semantic network SN_{DS} represents the DS diagram associated with a useful postulate and that semantic network SN_P represents a problem figure. A partial match binds each node in SN_{DS} with a node in SN_P so that the links in SN_{DS} and SN_P are consistent with the bindings. We let the variables v_i $(i = 1, \ldots, n)$ of the constraint satisfaction problem represent the n nodes of SN_{DS}. The domain of a variable v_i consists of a subset of nodes in SN_P that are the same type as v_i. For example, the domain of a variable that represents a DS point is a set of all problem points. A partial match occurs when some DS elements are not bound to any problem elements. This is implemented by binding the value NIL to the variables representing such DS elements. The value NIL is explicitly added into the domain of each variable so that the constraint-satisfaction algorithm need not be modified.

The constraints model relations among geometric elements. There are two classes of constraint:

(1) *Constraints on bound variables*: These constraints check the consistency among the relations between the bound geometric elements. For example, if v_1 and v_2 are parallel lines in SN_{DS} and they are bound to lines x_1 and x_2 in SN_P, then $x_1 \parallel x_2$ must be quantitatively satisfied. The initial domain values that are inconsistent with the variable to be bound are also filtered out, prior to the search.

(2) *Constraints for partial matching*: These constraints check whether a variable can be bound to NIL by testing topological feasibility with the bound variables. For example, assume that v_1 and v_2 are two rays intersecting at point v_3. If both v_1 and v_2 are bound to non-NIL values x_1 and x_2, then v_3 cannot be bound to NIL (indeed, it must be bound to the problem point that is an

intersection of problem rays x_1 and x_2). As another example, if all three segments of a triangle are bound to non-NIL values, then the triangle itself must be bound to a non-NIL value as well.

3.2.3. *Construction Operators to Construct a Missing End Point*

Given a partial match, we now describe a technique for finding a sequence of construction operators to create missing segments in the problem figure.

There exist only three types of partial match for a DS segment with respect to bindings for its end points: both end points are bound, only one of the end points is bound, or none of the end points is bound. One of these cases can be eliminated by noticing that no postulate has an isolated DS segment. That is, all the segments in a postulate have at least one end point shared with other segments. Thus, repeatedly constructing a problem segment for a DS segment with a single end point bound would eventually make all DS segments have both of their end points bound. For example, consider a useful postulate with a rectangle *abcd* that gets bindings $\{a/\text{NIL}, b/X, c/Y, d/\text{NIL}\}$.* In this case, we have a DS segment (*bc*) with both end points bound, two DS segments (*ab* and *cd*) with a single end point bound, and a DS segment (*ad*) with no end points bound. If we successfully construct a problem segment that corresponds with DS segment *ab*, then the DS point *a* is bound to a new problem point. Now, the DS segment *ad* has a single end point bound. Furthermore, constructing one of the remaining two DS segments makes the last DS segment have both end points bound.

The above observation allows us to safely focus only on the construction of segments with one or two bound end points. The basic idea follows:

1. For each unbound DS segment with both end points bound, construct a corresponding problem segment by simply connecting the two problem points corresponding to the two bound DS end points. Update the partial match. If a perfect match is found, then halt (i.e., construction is completed).
2. Nondeterministically choose an unbound DS segment that has exactly one of its end points, say p, unbound.
3. Try to construct a problem point for the DS point p using the techniques described below. If the construction fails (i.e., there is no construction with a compass and a straightedge), back up to step 2 and choose another unbound DS segment. Otherwise update the partial match.
4. Go to step 1. Notice that the unbound DS segment chosen in step 2 now has both end points bound, so step 1 will construct its problem segment at this time.

Let us call the unbound DS segment chosen in step 2 the *target DS segment*. Also, let us call the unbound DS end point of the target DS segment the *target DS end point*. Let us call the problem segment that we will construct for the target

* In this article hereafter, the small letters are used to represent the elements in a DS diagram and the capital letters to represent the elements in a problem figure.

segment the *target problem segment*, and also call the to-be-constructed end point the *target problem end point*.

For the target problem segment to be constructed, it is sufficient to construct the target problem end point. Determining the coordinate of the target problem end point is equivalent to determining the direction and length of the target problem segment relative to its end point that is bound. This motivates the following definitions regarding the measurement of those quantities:

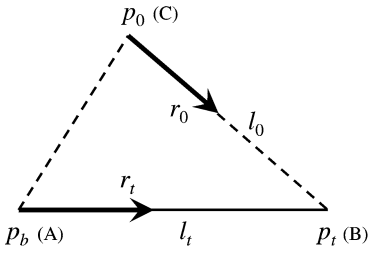
- The length of the target problem segment is *determined* if the useful postulate states that the target DS segment is congruent to some DS segment that is bound. For instance, suppose the target DS segment is xy. If a premise of a useful postulate requires $xy = uv$ and if uv is bound to problem segment AB, then the length of the target problem segment must be equal to the length of AB.
- The direction of the target problem segment is *determined* if one of the following four conditions holds. Suppose r_t is a DS ray on the target DS segment where the end point of r_t is the same as the bound end point of the target DS segment (see Figure 4 where l_t is the target DS segment):

  1. Ray r_t is bound; hence the problem ray bound to r_t gives the direction.
  2. Ray r_t contains a bound DS point p; hence the problem points bound to p and to the end point of r_t give the direction.
  3. A premise of the useful postulate specifies that r_t is parallel to a bound DS ray r'_t.
  4. A premise of the useful postulate specifies that r_t is perpendicular to a bound DS ray r'_t.

  For each of these four cases, there exist a compass procedure and a straightedge procedure for drawing a problem ray from A in Figure 4 (e.g., for the last case, we extend r'_t and then draw a perpendicular to r'_t through A, as shown in Figure 1).

Given these definitions, we now examine different cases to construct a target problem end point depending on whether the length and/or direction are determined. As shown in Figure 4, assume that p_t is the target DS end point and p_b is the bound end point of the target DS segment. Also assume that A is the problem point to which p_b is bound, and B is the target problem end point that we wish to construct.

We consider the easiest case first. If the length and the direction of the target problem segment are both determined, then GRAMY constructs point B as follows. It extends a ray from A in the determined direction. GRAMY then plots B on the ray such that $|AB|$ is equal to the length of the problem segment that determines the length of the target problem segment. The second operation corresponds to drawing an arc about A with the radius copied by a compass. This construction procedure must always succeed because the ray intersects the arc exactly once.



l_t : Target DS segment
p_t: Target DS end point
p_b: Bound DS point that is an end point of l_t
p_0: Bound DS point
r_0: DS ray on l_0 with p_0 as the end point
r_t : DS ray on l_t with p_b as the end point

Figure 4. Construction of a target problem end point with the third point.

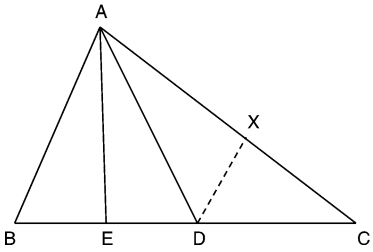
Next, we consider the hard case where either the length or the direction of the target problem segment is determined, but not both. GRAMY nondeterministically chooses a bound DS point p_0 not equal to p_b or p_t (see Figure 4). Suppose the problem point that is bound to p_0 is C. There are several cases to consider. First, if the length of the problem segment BC is determined and the length of the target problem segment AB is determined, then GRAMY can draw arcs of determined radii about A and C. Second, if the direction of BC is determined and the length of AB is determined, then we can draw a ray from C in the determined direction and an arc of determined radius around A.⋆ Third, if the directions of both AB and BC are determined, then GRAMY can draw rays from A and C in the determined directions. There is a fourth case, wherein the length of BC is determined and the direction of AB is determined, but this is already covered by the second case, given that one switches p_0 and p_b. To avoid duplication, GRAMY does not generate a construction for this case.

It turns out that there exists a certain kind of problem that cannot be solved by the above construction technique. This kind of problem requires a constructed segment to hold a certain relation with a problem segment that is not involved in a partial match for the useful postulate. In order to handle such cases, GRAMY uses a novel technique described in the next section. The technique also solves the problems that require construction where the existence of constructed points is dubious (e.g., ray grazing a segment; see the footnote).

3.2.4. *Construction with a Reference to a Free Segment*

The problem shown in Figure 5 cannot be solved with the construction techniques described in the preceding section. The useful postulate is the triangle-congruent axiom with a partial match involving $\{p_b/D,\ p_0/A,\ r_0/rayAC\}$, using the same

⋆ The arcs about A and the ray from C may have either one or two intersections. If the construction apparently has only one intersection (i.e., the arc just grazes the ray), a teacher would either reject a student's proof that uses such a construction or require a proof of the existence of such an intersection point. GRAMY simply does not generate such a construction and picks up other postulates or bindings. A future extension could perhaps generate a construction and add a goal to the search state to prove that the point or points exist.



Goal: $\angle EAD = \angle DAC$
Given: $AB = BD$
D midpoint of BC
E midpoint of BD

Figure 5. An example of construction referring to a free segment.

configuration elements in Figure 4. Since the useful postulate states that the target problem segment is equal to *ED*, GRAMY can draw the target problem point (X) as an intersection of ray *AC* and an arc about D with $|DE|$ as the radius, and assert $DX = DE$ as a true proposition. However, such construction does not lead to a proof. Instead, the segment DX must be constructed to be parallel to *AB*.

To deal with this kind of problem, GRAMY tries to convert the first two hard cases mentioned in the previous section into construction without drawing an arc. Specifically, if drawing an arc successfully finds construction, then GRAMY also tries to construct the same point via a ray instead of an arc. To do so, GRAMY searches for a problem segment, even though it is not involved in the partial match, that is approximately parallel or perpendicular to the target problem segment found. It then treats the target segment as having a direction determined by the parallel or perpendicular segment.

In the example shown in Figure 5, once the coordinate of problem point X is determined as an intersection of ray *AC* and the arc from D, it can be seen that *AB* is approximately parallel to *DX*. This observation is possible by calculating slopes of *AB* and *DX* based on the *xy*-coordinates of A, B, D, and X. As a result, GRAMY constructs X as an intersection of *AC* and a line from C parallel to *AB*, and asserts $DX \parallel AB$ as a true proposition. This technique is also effective even when the circle about D just grazes the line *AC*.

We call this technique *construction with a reference to a free segment*. This construction procedure can produce different constructions by using different reference segments in the problem figure that are parallel or perpendicular to the target segment. GRAMY finds all reference segments, and it outputs a different construction for each.

3.2.5. *Construction with Transitive Substitution*

Like many theorem provers, GRAMY treats equality specially. It maintains a set of equivalence classes. When it derives the equality of two elements (e.g., two segments are equal, or two triangles are congruent), it puts them into the same equivalence class and adds this class to the set of equivalence classes unless the class is there already. GRAMY's pattern matcher for propositions is modified so

that when it tests whether a proposition holds, it succeeds not only if the two constants are known to be equal but also if they are in the same equivalence class. Theorem provers that do not treat equality in this way must have transitivity axioms of the form "If $X = Y$ and $Y = Z$, then $X = Z$," which not only increase the combinatorics but clutter the proof with "trivial" inferences.

Since GRAMY treats equality specially when doing its regular reasoning, it must also treat equality specially when doing construction. For example, to prove a goal $AB = CD$, it can consider finding a third segment, say, PQ, that is equal to AB, and then can try to prove $PQ = CD$. Assuming that segment PQ does not appear in the original problem figure, the triangle-congruent postulate involving $\triangle uvw \equiv \triangle xyz$ might be useful with a partial match that involves $\{uv/AB, xy/\text{NIL}\}$. Once a partial match is found, we can apply the same procedure as described in the previous sections. If construction eventually draws PQ, and there exists a proof for $\triangle uvw \equiv \triangle xyz$ (hence $AB = PQ$), then one may find a successful proof for $AB = CD$ by a transitive substitution provided that $PQ = CD$ can also be proven. GRAMY considers doing such construction not only for segment congruence but also for all types of relationships that are transitive. Six problems in our problem corpus require this construction technique (see Section 4.1).

3.3. PROOF PROCEDURE WITH CONSTRUCTION

This section summarizes the procedure to search for a proof with construction. The states in the search space consist of a problem figure, a set of true propositions (i.e., given and derived facts), and an ordered list of goals called the goal queue. The search maintains a state queue for breadth-first search. The following is the main search procedure:

1. Let S be the initial state, and let its goal be the top-level goal. Initialize the state queue to hold just S.
2. (Exhaustive forward chaining) If the state queue is empty, then quit. Otherwise remove state S from the front of the state queue. Apply exhaustive forward chaining to S. S now holds the deductive closure of the given propositions and the problem figure under the given postulates.
3. (Goal test) Remove all the goals in the goal queue of S that unify with the propositions in S. If S has no goals left in its goal queue, then one proof has been found. Put S aside, and go to step 2.
4. (Backward chaining) For each postulate p such that its DS diagram perfectly matches with the problem figure in S, all p's premises are quantitatively satisfied, and p's consequence unifies with the first goal in the goal queue of S, make a state S_p that is a copy of S with all the premises of p put at the front of the goal queue. Put S_p onto the end of the state queue. Notice that S_p has the same problem figure as S.
5. (Construction) For each useful postulate for the first goal in the goal queue of S, find all constructions for each partial match with the construction procedure

below. For each successful construction, create a new state with a new problem figure. Put these states on the end of the state queue. Notice that the new state has the same goal queue as in S.

6. Go to step 2.

The construction procedure takes a useful postulate and a partial match as input, and outputs all possible constructions. This is also done by state-space search. Major components of a state include a problem figure, a list of missing segments, and a binding list. A single application of a construction operator adds one missing segment into the problem figure. Thus, each state has exactly one missing segment less than its ancestor state. Since there are only a finite number of missing segments to be constructed, depth-first search works fine. The following is the construction procedure:

1. Make an initial state with the original problem figure and the partial match. Initialize state queue with it.
2. (Goal test) If the state queue is empty, then halt. Otherwise remove a state C from the front of the state queue. If C has no missing segments left, then construction is done. So put C aside and repeat this step.
3. (Construction by connecting bound end points) For each unbound DS segment with both end points bound, add the corresponding problem segment to the problem figure by simply connecting the two problem points corresponding to the bound DS end points. Update the binding list with the newly constructed segments.
4. (Construction operator for missing segment with length and direction determined) Nondeterministically choose an unbound DS segment that has exactly one of its end points unbound and has both its length and direction determined. Calculate the coordinates of the target problem end point as an intersection of an arc with the determined length as its radius and a ray with the determined direction. Create a new state with the new problem figure. Assert appropriate facts to the new state (e.g., that the target segment is congruent to the segment that determined its length). Update the binding list in the new state so that the target DS end point is bound to the newly added intersection.
5. (Construction operator for a missing segment with either length or direction determined) Nondeterministically choose an unbound DS segment that has exactly one of its end points, p_t, unbound and has either its length or direction (not both) determined (see Figure 4 where l_t is the unbound DS segment chosen).

    5.1. Nondeterministically choose a bound DS point p_0 such that the segment between p_0 and p_t has either its length or direction determined.

    5.2. Calculate the coordinate of an intersection(s) of the appropriate arcs and rays (depending on whether lengths or directions are determined) for the problem point that is supposed to be bound to p_t.

5.3. For each intersection, create a new state with the new problem figure by adding the intersection. Assert appropriate facts (those made true by construction) to the new state. Update the binding list in the new sate so that the target DS end point is bound to the newly added intersection.

6. (Construction operator for a construction with a reference to a free segment) Perform the same step specified in step 5 but not step 5.3. Nondeterministically choose a reference segment in the problem figure that is parallel or perpendicular to the target problem segment determined by its bound end point and the calculated intersection. For each reference segment, create a new state with the new problem figure by adding the intersection. Assert appropriate facts including that the new segment is parallel or perpendicular to the reference segment. Update the binding list in the new state. Go to step 2.

At several steps in this procedure, failure can occur. For instance, after drawing the relevant arcs and rays, there may be no intersection points. In this case, no construction is produced, and that nondeterministic choice dies. If all such choices fail, then this branch of the search also fails, which in turn abandons the whole attempt to apply the selected useful postulate with the selected bindings at step 5 in the main search procedure.

3.4. SEMICOMPLETENESS OF THE PROOF PROCEDURE

Now we examine the completeness of the proof procedure. Since the ability of our construction procedure to produce a particular construction thoroughly depends on a set of postulates available, we emphasize that we discuss the completeness of the *search procedure*, not the completeness of the set of postulates given to GRAMY. In other words, we claim that GRAMY can find all proofs that consist only of the postulates given.⋆

In the proof procedure above, the first list of steps defines the main procedure, and the second set defines the subprocedure for generating constructions that provide a perfect match for the useful postulate. We have not yet succeeded in proving that the subprocedure is complete. In particular, the restriction that step 5.1 choose a bound DS point p_0 is necessary to prevent combinatorial explosion. Arcs or rays that are drawn via p_0 are actually drawn from the problem point that p_0 is bound to. But it may prevent the subprocedure from generating constructions that both make the useful postulate match and lead ultimately to a successful proof. In principle, one could draw arcs or rays from any problem points that are not involved in the partial match. If the resulting construction makes the figure match the useful postulate, then this branch of the search might survive to become a successful proof. For example, assume that a goal $AB = CD$ is proved with the triangle congruent postulate $\triangle ABP \equiv \triangle CDQ$ where the points P and Q are not

⋆ We also claim that GRAMY's procedure is sound, which means that if no proof exists, it will not find one. This claim follows from the soundness of both forward and backward chaining.

in the original problem configuration and hence needed to be constructed. Those points could have been constructed as an intersection of two arcs about some points in the problem figure that are not involved in the partial match for this postulate. We have yet to see a problem that requires such a branch in the search, but we are also unable to show that such branches always fail.

Assuming that the subprocedure is complete, it can be shown that the main procedure is complete. Thus, we claim that GRAMY is "semi-complete." The proof follows. Let P be a problem to be proven. Let F_0 be an original problem figure for P, and F_i be the problem figure after the i-th construction has been made. Let $\Gamma = \{a_1, \ldots, a_n\}$ be a set of postulates given to the theorem prover sufficient to prove P. Let $PA(\Gamma, F_i)$ be a set of all possible ground instantiations of postulate applications for all elements in Γ with respect to F_i. Since there are finitely many elements in a problem figure, there are only finitely many instantiations for each postulate; hence $PA(\Gamma, F_i)$ is finite. If P has a proof $Pr(P)$ with k constructions, then it must be a sequence of some ground instantiations in $PA(\Gamma, F_k)$. Let $Pr(P) = p_1, \ldots, p_n$ $(p_i \in PA(\Gamma, F_k))$ where the consequence of p_1 unifies with the to-be-proven goal specified in P, and the sequence is made by traversing a proof tree depth-first.

In the case that P does not require construction at all, $Pr(P)$ must be found because exhaustive forward chaining will eventually produce all the ground instantiations for the postulates in $PA(\Gamma, F_0)$.

Assume that the proof procedure can find the proofs with $k - 1$ constructions $(k \geq 1)$. We now prove that the proof procedure can also find the proofs with k constructions. Let $Pr(P) = p_1, \ldots, p_n$ $(p_i \in PA(\Gamma, F_k))$ be a proof with k constructions. There must be at least one ground instantiation of a postulate application in the proof that does not have a perfect match with F_0. Let p_j $(1 \leq j \leq n)$ be the first postulate application that satisfies this property, namely, p_m has a perfect match with F_0 for $\forall m$ such that $1 \leq m < j$. Since the top-level goal of the problem P refers to the geometric elements in F_0 and backward chaining is always done for the elements in the problem figure, it is easy to see that the consequence of p_j also refers to the elements in F_0. Hence, a partial match for a postulate application p_j, which involves bindings for geometric elements referred in the consequence of p_j, must be found by the search step 5 (construction) of the main procedure. If the subprocedure is complete, it must produce constructions that generates F_1. Now the problem becomes one that requires $k - 1$ construction, which can be solved, according to our inductive hypothesis. Hence the proof procedure must find a proof of P with k constructions. This completes the proof for the semi-completeness of the proof procedure.

4. Evaluation of GRAMY

This section shows the performance of GRAMY on the proofs with construction. To compare GRAMY with other theorem provers, we collected 23 construction

problems from the ten literature studies cited earlier. Few construction problems have been addressed in the literature, possibly because of the difficult nature of automated construction. Construction is also difficult for humans, so we found only 17 construction problems in two American and three Japanese textbooks (Aref and Wernick, 1968; Coxford and Usiskin, 1971; Kyogaku-Kenkyusha, undated; Matsushita, 1993; Shinshindo, 1993). Of those 17 problems from the textbooks, 8 were also used in the preceding studies. As a result, the problem corpus contains 32 construction problems.

Table I shows the performance of the theorem provers on our problem corpus. In the table, a row corresponds to a problem, and a column corresponds to a textbook or a study (including the current one denoted as GRAMY04). A label in a cell shows a problem number specified in the textbook or the literature, or it shows the page number where the problem is first mentioned. An italicized, label means that the theorem prover failed to find a proof. A nonitalicized, bold label means the prover did find a proof. The sixth column, GRAMY04, lists all 32 construction problems used in the current study.

In Wong (1972), 50 problems are used. However, we have picked only 18 problems from his study because those are the only problems that do not deal with sums, inequality, ratio, and coincident intersections. The same filtering policy was applied to selection from the textbooks. All the other literature has the same number of problems shown in Table I.

The problems are classified into three categories according to when constructions should be made. (1) Twenty-six out of 32 problems can be solved by applying a single construction in the first quiescence state, which is produced by exhaustive forward chaining from the initial state. That is, the construction was made for the top-level goal. (2) Five out of 32 problems, which are all italicized in Table I, required multiple constructions applied for different subgoals. (3) Only one problem required backward chaining to replace the top-level goal with its immediate subgoal before construction. The next few sections discuss each type of problem.

4.1. SINGLE CONSTRUCTION PROBLEMS

Table II shows results of the search for proofs on the problems that could be solved by applying single construction to the top-level goal without transitive substitution. As mentioned earlier, if GRAMY is to be used for tutoring, it must be able to generate all correct proofs. Hence the table consists of two parts: the search complexity to find the first proof, and the search complexity to find all proofs. For the search for the first proof, the table shows the depth of search (i.e., the number of cycles of the main search procedure mentioned in Section 3.3 taken before the first proof is found), the length of the proof (i.e., the minimum number of postulate applications for a proof for the problem figure modified by the first successful construction), the number of states expanded before the proof is found, the number of true propositions asserted by the time the proof is found (i.e., the number of statements in the

Table I. Problems used for the evaluation

Textbooks					Theorem Provers										
Aref68	Coxford71	Shinshindo93	Matsushita93	Kyogaku	GRAMY04	Chou00	Suwa89	Elcock77	Ullman75	Nevins75	Goldstein73	Reiter72	Wong72	Gelernter59	Gelernter63
					P132									**Fig. 5**	
					P111	**Ex 6.4**		**p. 26**	**Ex 2**		**p. 7**	**Ex 7**			**Appendix 2**
					P123						**p. 34**				
		5			**P116**		**Problem 1**								
					P124		**Problem 2**								
					P105								**Ex 1**		
1.11					**P101**								**Ex 2**		
1.9					**P102**								**Ex 3**		
					P103								**Ex 4**		
					P108								**Ex 8**		
					P109								**Ex 9**		
					P112								**Ex 13**		
					P110								**Ex 14**		
1.1					**P115**								**Ex 22**		
					P125								**Ex 30**		
					P142								**Ex 31**		
1.22					**P143**								**Ex 34**		
	p. 265				**P144**								**Ex 35**		
1.7					**P135**								**Ex 40**		
					P117					**Ex 1**			**Ex 41**		
1.18					**P139**								**Ex 45**		
					P152								*Ex 48*		
					P154								*Ex 50*		
	12.4.2			p. 14	**P126**										
				p. 15	**P127**										
		28			**P128**										
			Pat 6		**P129**										
			Pat 15		**P130**										
			5		**P131**										
1.3					**P156**										
				p. 26	**P158**										
	12.3.1				**P159**										

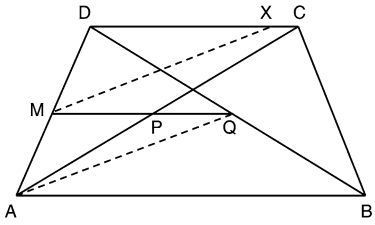
Table II. Search complexity for proofs with single construction to the top-level goal

Problem	First Proof					All Proofs				
	Depth	Length	State	Prop.	Time	All Const	Suc. Const	State	Prop.	Time
P132	4	3	4	12	11	40	40	40	1342	76
P123	5	4	15	38	35	5	1	9	200	35
P109	5	5	198	91	322	76	3	149	2414	512
P127	5	3	296	70	1529	101	7	189	12252	2381
P101	5	5	313	72	1764	109	8	205	13075	2573
P117	6	9	48	178	54	17	1	33	1642	205
P116	6	6	130	79	109	76	1	151	3303	392
P112	6	8	23	79	151	181	14	348	21584	3908
P108	6	14	112	92	185	49	1	97	2137	432
P115	7	8	26	65	61	55	1	109	2785	329
P129	7	10	13	349	278	36	1	71	15095	5268
P128	7	11	93	103	770	146	2	290	20255	2951
P111	8	7	80	146	544	135	4	256	18229	3967
P131	8	6	492	63	2156	122	16	228	12880	2027
P142	9	10	13	127	84	95	7	183	16251	3356
P144	9	19	85	152	146	61	6	112	2691	472

deductive closure), and the CPU time in seconds to find the proof. For the search for all proofs, the table shows the number of constructions found regardless of whether they led to a proof, the number of constructions that led to a correct proof, the total number of states expanded, the total number of true propositions asserted, and the total CPU time in seconds.* The last three numbers aggregated searches for all constructions found regardless of their correctness. As an example, Appendix B shows successful construction for the problem P111 (an example of unsuccessful, but reasonable, construction for P111 is shown in Figure 6).

As mentioned in Section 3.2.5, some problems require construction with transitive substitution. The current version of GRAMY first applies the construction procedure without transitive substitution; only when no proof is found does it apply construction with transitive substitution. Since the number of possible constructions with transitive substitution could be large, this trial-and-error approach could be very costly. Table III shows search complexities on those problems. The number in parentheses shows the number of successful constructions. In general, a partial match for construction with transitive substitution has more variables assigned to NIL. Hence the number of possible constructions made by transitive substitution is bigger than the one by normal construction. As shown in the table, the ratio of suc-

* The current version of GRAMY is written with Common LISP running on an Intel Pentium-III 600 MHz processor.



Construction:

1. Connect A and Q
2. Plot X on CD such that $MQ = DX$
3. Connect M and X

Useful postulate:

Triangle Congruent Postulate ($\triangle AQM \equiv \triangle MXD$)

Figure 6. A "reasonable" but unsuccessful construction for P111.

Table III. Search complexity for proofs with single construction by transitive substitution

	Normal Construction					Construction with Transitive Substitution				
Problem	Length	#Const	#State	#Prop.	Time	Length	#Const	#State	#Prop.	Time
P126	0	2	5	24	9.39	3	285 (5)	5	79	270.56
P139	0	0	3	55	1.37	4	510 (1)	36	175	800.65
P130	0	1	6	89	28.56	5	389 (2)	8	91	552.22
P110	0	2	7	42	15.98	6	293 (4)	9	94	349.77
P102	0	0	3	467	1.48	7	23 (6)	9	1087	314.01
P105	0	2	5	12	2.97	13	184 (184)	52	49	78.27

cessful to unsuccessful constructions is generally quite low. Nonetheless, GRAMY found proofs for those problems in reasonable time. To find *all* proofs, however, GRAMY should try transitive substitution on all problems, even those where other proofs are found without transitive substitution. We have not yet experimented with this extension.

4.2. MULTIPLE CONSTRUCTION PROBLEMS

Five problems in the corpus require multiple constructions. GRAMY could solve those problems given unlimited resources (i.e., memory space and time). However, because of inefficient memory management in the current implementation, GRAMY could not find a proof for these and only these problems. For example, problem P103 needs three constructions for three consecutive subgoals. For the first goal, 153 constructions were found. For those constructions, GRAMY expanded a total of 3,687 immediate states. GRAMY crashed when searching for constructions for the third subgoal from the top-level goal. We believe that this is a fixable technical problem, not a fatal theoretical problem.

Table IV. Search complexity for proofs with single construction by transitive substitution

Problem	Length	#Count	#Prop.	Time
P143	30	78	220	1320

4.3. CONSTRUCTION FOLLOWING BACKWARD CHAINING

Problem P143 requires backward chaining prior to a construction. Table IV shows the search complexity on this problem. Construction was carried out immediately following one backward deduction applied at the first quiescence state. There are four disjunctive subgoals for this backward deduction, and 20, 20, 38, and 0 constructions were found for each of the subgoals (hence, 78 total constructions as in the table).

Since P143 is the only problem in our corpus that requires backward chaining prior to construction, it is hard to evaluate the effectiveness of GRAMY on this type of problem. One would ask whether, since backward chaining may be applied multiple times, the search might explode. In such cases, GRAMY can iteratively increase the depth at which construction is carried out. For each iteration GRAMY can apply depth-first iterative-deepening search to find a proof. Thus, we believe that GRAMY is effective for constructions that follow multiple backward chaining.

4.4. COMPLEXITY OF DIAGRAM MATCHING

Matching a DS diagram to a problem figure is complex. Suppose a problem figure has P points, S segments, A angles, R rays, T triangles, and L lines. For a postulate that has a DS diagram with p points, s segments, a angles, r rays, t triangles, and l lines, there are $P^p \times S^s \times A^a \times R^r \times T^t \times L^l$ possible assignments. This number tends to run into astronomical figures. For example, the triangle-congruent postulate has 6 points, 6 segments, 6 angles, 2 triangles, and 12 rays. For P111, which has 8 points, 21 segments, 30 angles, 11 triangles, and 26 rays, the number of possible assignments with the triangle-congruent postulate is 10^{41}.

As mentioned earlier, GRAMY uses a constraint satisfaction algorithm to search such huge spaces effectively. Table V shows the number of partial matches found for each DS diagram with problems regarding the equality of segments. The first six columns are the numbers of elements in the problem figure (Points, Segments, Angles, Triangles, Rays, and Lines). The remaining columns show the number of matches per DS diagram. As can be seen in the table, the diagram-matching module works effectively.

Table V. The number of matches for problems regarding equality of segments

No.	P	S	A	T	R	L	TRI-CONG	RIGHT-TRI-CONG	SEGMENT-SUM	RECTANGLE-DIAGONAL	RIGHT-TRI-MEDIAN	RHOMBUS	TRI-MIDPOINT-LAW
P132	4	4	4	0	8	0	252	34	0	0	0	0	0
P144	4	5	8	2	10	0	276	56	12	0	0	0	0
P109	4	6	8	3	10	1	252	87	12	1	2	6	0
P131	6	11	19	4	18	2	582	106	12	2	3	4	0
P108	6	12	15	4	16	4	228	46	34	0	0	0	0
P112	7	16	20	8	20	5	552	68	34	70	6	18	36
P101	7	18	30	12	24	4	678	68	12	2	3	4	0
P127	7	18	30	12	24	4	654	68	12	2	3	4	0
P111	8	21	30	11	26	6	546	56	34	46	5	18	40

5. Conclusion

We have implemented an efficient and semi-complete procedure for proofs with construction. Goldstein (1973) and Anzai et al. (1979) appear to have used the same main procedure as GRAMY, namely, alternating forward chaining with a selection of a useful postulate as the subject of construction. Their work may have been based on Wong's work (1972) that proposed a search for a desired postulate in both forward and backward chaining. These authors appear not to have noticed that the main procedure is complete, assuming the completeness of the subprocedure. No previous provers have been proven to be complete for theorem proving with construction. GRAMY is not complete either, but we have succeeded in proving its semi-completeness, and Section 3.4 localizes the lack of completeness to a single subprocedure, thus setting the stage for further work. Moreover, the results from an empirical evaluation show that GRAMY works on most of the problems gathered from textbooks and the literature. This section summarizes the lessons learned from GRAMY and discusses future issues.

5.1. WHY GRAMY WORKS

We attribute the success of GRAMY to several features, discussed below.

5.1.1. *Use of Exhaustive Forward Chaining*

Because it does not cost much to apply exhaustive forward chaining to calculate the deductive closure, finding a proof within a fixed problem figure is not costly either. Thus, most of the computation is spent finding possible constructions by depth-first search. We note, however, that the proof procedure guarantees exhaustive forward chaining to stop only because arithmetic operations are excluded. Introducing arith-

metic operations no longer guarantees reaching a quiescence state (e.g., one can divide a segment infinitely many times).

5.1.2. *Heuristics for Construction*

GRAMY differs from other provers in the way missing segments are constructed without relying on ad hoc heuristics. Thus, GRAMY has the power to find more constructions. For example, Goldstein (1973) claims that only two constructions for P111 are based on the heuristics applied. GRAMY found 135 successful and unsuccessful constructions, including the one shown in Figure 6. Although this construction does not lead to a proof, it is just as reasonable as the other four successful constructions shown in Appendix B at the time it was made. This kind of deficit in completeness of a construction procedure affects the completeness of the whole theorem prover as well.

5.1.3. *Use of Diagrams for Geometry Theorem Proving*

Beside the use of diagrams as search control for the backward inference, as done by Gelernter (1959) and many other provers, another advantage of using diagrams is that it *localizes* necessary information into a single knowledge piece. Larkin and Simon (1987) argue that due to the dense information embedded in a diagram, using diagrammatic representation requires less effort to draw information necessary to make inference.

The diagram configuration model developed by Koedinger and Anderson (1990) is based on this principle. In this model, a set of geometric postulates that shares the same geometric configuration is represented as a single piece of knowledge called a diagram configuration schema. A diagram configuration schema consists of a diagram and a set of propositions to be held in the diagram. A proposition that is a sufficient condition of all other propositions is called a whole-statement. The propositions that are not whole-statements are called part-statements. A diagram configuration schema also has all possible combinations of the part-statements representing the conditions of the whole-statement to be a true proposition. When any of the conditions is satisfied, then the whole-statement and all other part-statements in the schema are asserted as the true propositions. This collective deduction plays a central role to improve the efficiency of search.

GRAMY uses the same kind of inference procedure and knowledge representation implemented as the diagrammatic schema (DS). A difference is that GRAMY's knowledge representation schema has more accumulative power than the one developed by Koedinger and Anderson. This is because GRAMY's diagrammatic schema does not discriminate between the whole-statement and the part-statements; that is, GRAMY's knowledge representation does not require any single proposition to be the sufficient condition of all other propositions. Instead, all possible deductions relating to a single diagram are combined into a single schema

whether they share a single condition or not. Consider, for example, the midpoint law that does not have a single proposition corresponding to a whole-statement.*

Besides the expressive power of the diagrammatic schema used in GRAMY, a major difference is in the way they are used; GRAMY uses a partial match of the diagram to carry out construction. Although we agree with Koedinger and Anderson's opinion that the diagram configuration model is "particularly well-suited for adding a construction capability" (1990, p. 532), we further claim that finding plausible construction requires an intensive search, and hence implementing an effective search control is essential for successful construction.

5.1.4. *Construction with Reference to a Free Segment*

We realized that construction with a reference to a free segment was neglected by preceding studies. All of them discuss only how to construct a missing element by considering the geometric structure of the desired postulate per se. The empirical evaluation revealed that construction with reference to a free segment is essential for some problems because there does not seem to be a proof other than the ones produced by GRAMY with a construction with reference to a free segment.

5.2. GRAIN SIZE OF THE POSTULATES

Discussions in the preceding sections depend on a set of postulates given to the theorem prover. For example, suppose a theorem prover eventually found a proof with multiple constructions. By adding the problem's premises, the goal, and the final problem figure to the prover as a postulate, the prover can find the same proof with single construction applied at the first quiescent state. For example, GRAMY could solve P125 by single construction if it were given the incenter theorem ("all three bisectors of angles in a triangle meet at a single point") as a DS where the three perpendicular segments from the incenter appear in the DS diagram.

Although adding proven problems as the postulates is quite natural in axiomatized systems like Euclidean geometry, we must make a good balance among the grain size of postulates and a load for students to learn those postulates. The experiments showed that the postulates currently implemented in GRAMY are sufficient to prove many problems used in textbooks. Hence we apparently need to add more specific postulates only to prove those problems that require multiple constructions.

5.3. IMPLICATIONS FOR GEOMETRY EDUCATION

We conclude that GRAMY is adequate as a building block for a tutoring system on theorem proving for several reasons:

* The midpoint law could be implemented as a single diagrammatic configuration schema with a geometric statement, say, the-midpoint-law-is-held-in-$\triangle ABC$ as a whole-statement. However, that kind of statement would never appear in a proof.

(1) GRAMY is skillful enough to solve all arithmetic-free construction problems that appear in some school textbooks.
(2) The deductions used in its proofs employ the Euclidean axiom system that is widely used in current school curricula.
(3) As a result, the proofs output by GRAMY are natural for students who are learning Euclidean geometry.

Since GRAMY can generate most of the solutions for a given problem, one can build a model tracing tutor that can recognize a student's proof steps and provide appropriate feedback. For each problem, a model tracing tutor has ideal solutions required to find a solution. It compares the student's problem-solving steps to the steps taken in ideal solutions. For geometry theorem proving, the ideal solutions contain all proofs that instructors would accept, which GRAMY can create in a reasonable amount of time.

Even though GRAMY draws inferences that are familiar to students, the proof procedure of GRAMY may be too complex for students to carry out despite the fact that it can be simply described (cf. the description of Section 3.3). For instance, students can hardly be expected to keep a queue of search states. Nonetheless, students could explore the search space generated by GRAMY and receive feedback whenever they start to head down a dead-end path. This pedagogical tactic is used by Andes (VanLehn et al., 2002), a successful intelligent tutoring system for physics based on a semi-complete physics theorem prover.

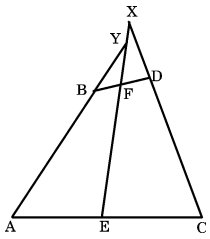
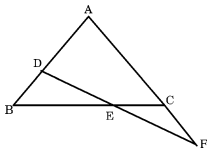
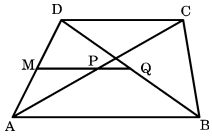
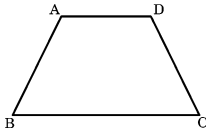
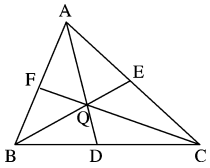
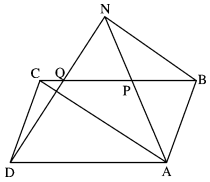
5.4. FUTURE WORK

One major task is to determine the completeness of the subprocedure that generates constructions for a useful postulate. Another task is to find some kind of search control that will allow GRAMY to solve the multiple construction problems efficiently without harming its completeness. It would also be an interesting issue to make GRAMY capable of proving that existence of the points constructed. We would like to extend GRAMY to solve geometry problems that involve arithmetic operations. Lastly, an intelligent tutoring system for geometry theorem proving must be built with GRAMY, and its educational effectiveness must be examined with appropriate students.

Appendix A. Construction Problems Used in the Evaluation

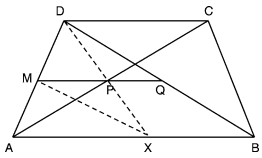
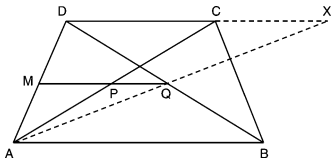
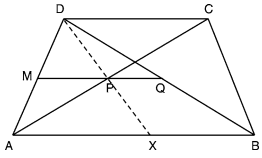
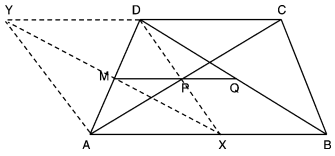
This appendix shows only the selected construction problems that are mentioned in this article.

Table VI. Selected construction problems

P103		Given: Goal:	$AB = CD$ $AE = CE$ $BF = DF$ $\angle AYE = \angle CXE$
P108		Given: Goal:	$AB = AC$ $DE = EF$ $BD = CF$
P111		Given: Goal:	$DQ = BQ$ $AP = CP$ $CD \parallel AB$ $AM = DM$
P123		Given: Goal:	$AB = CD$ $\angle ABC = \angle DCB$ $\angle BAD = \angle CDA$
P125		Given: Goal:	$\angle ABE = \angle CBE$ $\angle BCF = \angle ACF$ $\angle BAQ = \angle CAQ$
P143		Given: Goal:	Parallelogram $ABCD$ $BN \parallel AC$ $DN \perp AC$ $PQ = BP$

Appendix B. Example of Construction

Table VII. Successful construction for Problem P111

	**Construction**: 1. Connect D and P 2. Extend DP to AB and plot X 3. Connect M and X **Useful Postulate**: Triangle congruent theorem ($\triangle$AXM $\equiv$ $\triangle$DXM)
	**Construction**: 1. Connect A and Q 2. Extend DC 3. Extend AQ and plot X **Useful Postulate**: The midpoint-connector theorem ($\triangle$AXC)
	**Construction**: 1. Connect D and P 2. Extend DP to AB and plot X **Useful Postulate**: The midpoint-connector theorem ($\triangle$DAX)
	**Construction**: 1. Connect D and P 2. Extend DP to AB and plot X 3. Connect X and M 4. Extend CD 5. Extend XM to the extension of CD and plot Y 6. Connect A and Y **Useful Postulate**: The rhombus theorem (Rhombus YAXD)

References

Anderson, J. R., Boyle, C. F., Corbett, A. T. and Lewis, M. W. (1990) Cognitive modeling and intelligent tutoring, *Artificial Intelligence* **42**(1), 7–49.

Anzai, Y., Ishibashi, N., Mitsuya, Y. and Ura, S. (1979) Knowledge-based problem solving by a labelled production system, *IJCAI*, pp. 22–24.

Aref, M. N. and Wernick, W. (1968) *Problems & Solutions in Euclidean Geometry*, Dover Publications, New York.

Chou, S.-C. and Gao, X.-S. (1989) A class of geometry statements of constructive type and geometry theorem proving, Technical Report TR-89-37, Department of Computer Science, University of Texas at Austin.

Chou, S.-C., Gao, X.-S. and Zhang, J.-Z. (1996) Automated generation of readable proofs with geometric invariants, I. Multiple and shortest proof generation, *J. Automated Reasoning* **17**, 325–347.

Chou, S.-C., Gao, X.-S. and Zhang, J.-Z. (2000) A deductive database approach to automated geometry theorem proving and discovering, *J. Automated Reasoning* **25**(3), 219–246.

Coelho, H. and Pereira, L. M. (1986) Automated reasoning in geometry theorem proving with Prolog, *J. Automated Reasoning* **2**(4), 329–390.

Coxford, A. F. and Usiskin, Z. P. (1971) *Geometry: A Transformation Approach*, Laidlay Brothers, Palo Alto, CA.

Elcock, E. W. (1977) Representation of knowledge in geometry machine, in E. W. Elcock and D. Michie (eds.), *Machine Intelligence*, Vol. 8, Wiley, pp. 11–29.

Gelernter, H. (1959) Realization of a geometry-theorem proving machine, in *Proceedings of the International Conference on Information Processing*, pp. 273–282.

Gelernter, H., Hansen, J. R. and Loveland, D. W. (1963) Empirical explorations of the geometry-theorem proving machine, in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, pp. 153–163.

Goldstein, I. (1973) Elementary geometry theorem proving, Technical Report AI Memo No. 280, Massachusetts Institute of Technology.

Greeno, J. G., Magone, M. E. and Chaiklin, S. (1979) Theory of constructions and set in problem solving, *Memory and Cognition* **7**(6), 445–461.

Haralick, R. M. and Elliott, G. L. (1980) Increasing tree search efficiency for constraint satisfaction problems, *Artificial Intelligence* **14**(3), 263–313.

Kapur, D. (1986) Using Gröbner bases to reason about geometry problems, *J. Symbolic Comput.* **2**(4), 399–408.

Koedinger, K. R. and Anderson, J. R. (1990) Abstract planning and perceptual chunks: Elements of expertise in geometry, *Cognitive Science* **14**(4), 511–550.

Kyogaku-Kenkyusha (n.d.) *Congruence and Similarity*, Kyogaku Kenkyusha, Osaka (in Japanese).

Larkin, J. H. and Simon, H. A. (1987) Why a diagram is (sometimes) worth ten thousand words, *Cognitive Science* **11**(1), 65–100.

Li, H. (2000) Clifford algebra approaches to mechanical geometry theorem proving, in X.-S. Gao and D. Wang (eds.), *Mathematics Mechanization and Applications*, Academic Press, San Diego, CA, pp. 205–299.

Matsushita, I. (1993) *APT: A Textbook for Advanced High School Entrance Examinations*, Shin-shin-do, Tokyo (in Japanese).

Nevins, A. J. (1975) Plane geometry theorem proving using forward chaining, *Artificial Intelligence* **6**(1), 1–23.

Reiter, R. (1972) The use of models in automatic theorem-proving, Technical Report Computer Science 72-09, University of British Columbia.

Shinshindo (1993) *High-Level Geometry Problems*, Shin-shin-do, Tokyo (in Japanese).

Suwa, M. and Motoda, H. (1989) Acquisition of associative knowledge by the frustration-based learning method in an auxiliary-line problem, *Knowledge Acquisition* **1**(1), 113–137.

VanLehn, K., Lynch, C., Taylor, L., Weinstein, A., Shelby, R., Schulze, K., Treacy, D. and Wintersgill, M. (2002) Minimally invasive tutoring of complex physics problem solving, in S. A. Cerri, G. Gouarderes and F. Paraguacu (eds.), *Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, pp. 367–376.

Wang, D. (1995) Reasoning about geometric problems using an elimination method, in J. Pfalzgraf and D. Wang (eds.), *Automated Practical Reasoning*, Springer, New York, pp. 147–185.

Wong, R. (1972) Construction heuristics for geometry and a vector algebra representation of geometry, Technical Report Project MAC 28, Massachusetts Institute of Technology.

Wu, W.-T. (2000) The characteristic set method and its application, in X.-S. Gao and D. Wang (eds.), *Mathematics Mechanization and Applications*, Academic Press, San Diego, CA, pp. 3–41.