Purdue University

Purdue e-Pubs

Department of Computer Science Technical Reports

Department of Computer Science

1993

Correctness Proof of a Geometric Constraint Solver

Ioannis Fudos

Christoph M. Hoffmann
Purdue University, cmh@cs.purdue.edu

Report Number:
93-076

Fudos, Ioannis and Hoffmann, Christoph M., "Correctness Proof of a Geometric Constraint Solver" (1993). *Department of Computer Science Technical Reports.* Paper 1089.
https://docs.lib.purdue.edu/cstech/1089

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

CORRECTNESS PROOF OF A GEOMETRIC CONSTRAINT SOLVER

Ioannis Fudos
Christoph M. Hoffmann

CSD-TR-93-076
December 1993

Correctness Proof of a Geometric Constraint Solver[*]

Ioannis Fudos[†] Christoph M. Hoffmann

Department of Computer Science
Purdue University
West Lafayette, IN 47907-1398

Report CSD 93-076[‡]
December 1993

Abstract

We present a correctness proof of a graph-directed variational geometric constraint solver. First, we prove that the graph reduction that establishes the sequence in which to apply the construction steps defines a terminating confluent reduction system, in the case of well-constrained graphs. For overconstrained problems there may not be a unique normal form. Underconstrained problems, on the other hand, do have a unique normal form.

Second, we prove that all geometric solutions found using simple root-selection rules must place certain triples of elements in the same topological order, no matter which graph reduction sequence they are based on. Moreover, we prove that this implies that the geometric solutions derived by different reduction sequences must be congruent. Again, this result does not apply to overconstrained problems.

[*]Work supported in part by ONR contract N00014-90-J-1599, by NSF Grant CDA 92-23502, and by NSF Grant ECD 88-03017.

[†]Supported by a David Ross Fellowship.

[‡]This and related reports are available via anonymous ftp to arthur.cs.purdue.edu under pub/cmh/Reports and subsidiaries.

1 Introduction

Geometric constraint solving has broad applications in a wide range of subjects, including in manufacturing where it arises in particular in the context of engineering drawings and CAD design interfaces. Constraints imposed on engineering drawings are intended to define a precise configuration of geometric elements in the plane. The importance and role of powerful, interactive constraint solvers in the new generation of CAD systems has been discussed in [1, 2, 3], and in the other papers collected in this issue.

In [2, 1], we have described a variational constraint solver that is based on an elegant graph reduction strategy. The solver handles geometric configurations composed from points, lines, circles with prescribed radii, arcs, segments, and rays. The constraints that can be imposed on those objects include distance, angle, tangency, incidence, concentricity, perpendicularity and parallelism.

The solver has been integrated into a CAD system presently under development that is based on an Erep, a high-level, editable design representation [5]. This CAD system addresses the problems described in [4].

As described in [2], the variational constraint problem described before can be reduced to the following problem:

> Given a set V of n points and lines and a set $E \subseteq V \times V$ that represents the existence of a constraint between a pair of geometric elements, find a sequence for placing the elements using a fixed repertoire of construction steps and such that the given constraints are satisfied.

Here, a constraint may be a distance between a point and a line, possibly zero; an angle between two lines, possibly zero; a nonzero distance between two points. We call the ordered pair $G = (V, E)$ the *constraint graph* of the geometric problem.

The constraint graph is an undirected graph whose nodes represent the geometries and whose edges represent the constraints. The edges are labeled with the type and value of the constraint (distance or angle). Note that each geometric element of the constraint problem has two degrees of freedom, and that each constraint can be translated into a single algebraic equation which is linear or quadratic.

In this paper, we are concerned with proving the correctness of the algorithm described in [1]. This is done in two parts. First, we establish that the construction sequence determination can be conceptualized as a confluent, terminating reduction system. Second, we establish that the geometric solutions associated with different reduction sequences must be the same in a topological sense. We show that this topological equality implies congruence of the solutions.

This two-step proof of correctness mirrors two conceptual phases of the algorithm:

1. A constraint graph is analyzed by a reduction process that produces a sequence in which geometric elements must be constructed.
2. The actual construction of the geometric elements is carried out, in the order determined before, by solving certain standard sets of algebraic equations.

The sequence determined in Phase 1 does not account for specific dimension values of the sketch. Moreover, more than one sequence will solve the same sketch in general. Confluence of Phase 1 means, therefore, that at the end of Phase 1 the same set of geometric elements has been accounted for, no matter which reduction sequence has been chosen.

In Phase 2, some construction steps must choose from among different possible ways to position the next geometric element. Here we show that no matter which sequence has been computed in Phase 1, Phase 2 must construct the same geometric solution. That is, if a geometric element is placed, no matter where it occurs in the construction sequence, it will be placed based on the same local information. Moreover, this consistency of placement implies congruence of the final solution.

For a thorough literature review see, e.g., [2, 1]. In the next section we recall the essential parts of the algorithm and define when a constraint graph is structurally well-, under-, or overconstrained. Then we prove the correctness of the algorithm excepting line-parallelism constraints. Finally, we explain how parallel constraints affect the details of the correctness proof.

2 The Basic Algorithm and Terminology

As explained in [1], the constraint solving algorithm first decomposes the constraint graph into clusters, where a cluster can be constructed by a sequential method placing all geometric elements using very simple construction steps. This decomposition is not unique. For the purposes of proving correctness it is simplest to decompose the graph into a maximum number of clusters, each cluster consisting of exactly two geometric elements between which there exists a constraint. Geometrically, such a cluster corresponds to a pair of geometric elements whose position and orientation relative to each other is known. Thus, a cluster is considered a rigid body with three degrees of freedom.[§]

Three clusters can be combined into a single cluster if they pairwise share a single geometric element. Geometrically, the combination corresponds to placing the associated geometric objects with respect to each other so that the given constraints can be satisfied.

[§]Two parallel lines when considered a cluster do not fit this description. Therefore, we exclude them from consideration in this and the next section.

Intuitively, this approach differs from Todd's [10] in that Todd constructs the geometric elements in a sequence where the next element is placed with respect to two elements already placed. By placing three rigid bodies with respect to each other, our method can cope with circularly constrained configurations. In a sense, our method is a recursive extension of Todd's. Note also that our algorithm has many similarities with Owen's [8]. In particular, Owen proceeds top-down, first studying the interaction between graph components, and then analyzes the components themselves. In contrast, we build clusters bottom-up and coalesce them on the fly.

2.1 Algebraic Equations for Distances and Angles

We denote the Euclidean distance norm of a vector $n = (n_x, n_y)$ with $\|n\| = \sqrt{n_x^2 + n_y^2}$. Let p_1 and p_2 be two points and $l_1(n_1, r_1)$ and $l_2(n_2, r_2)$ be two lines, where n_1, n_2 are the normal vectors ($\|n_1\| = \|n_2\| = 1$), and r_1, r_2 are the signed distances of the lines from the origin. We explain the meaning of the constraints.

A distance d, $d \geq 0$ between two points p_1 and p_2 means simply

$$\|p_1 - p_2\| = d$$

and has in general two solutions. A *signed distance* d ($d \epsilon \Re$), between a point p_1 and a line l_1 means

$$p_1 \cdot n_1 = r_1 + d$$

(There is in general only one solution.) A *signed angle* α between two oriented lines $l_1 = (n_1, d_1)$ and $l_2 = (n_2, d_2)$ means:

$$n_2 = (n_x \cos\alpha - n_y \sin\alpha, n_y \cos\alpha + n_x \sin\alpha)$$

where $n_1 = (n_x, n_y)$.

2.2 Construction Steps

The construction has two phases, cluster creation and cluster merging. During cluster creation, we place two geometric elements that have a constraint between them with respect to each other. The construction is obvious, since it is a direct interpretation of the equations for distances and angles. Note that the sign of the distances and angles is specified based on what the user has sketched.

In the cluster merging phase we repeat the following: three clusters with three pairwise common geometries are merged into a single cluster. This means in particular that we will place the three common geometries, having pairwise predetermined relative positions between them. Having positioned the shared geometries, we translate and rotate the three clusters so that the three geometries are in the required location.

We describe now the relative placement of the three shared geometric elements.

Three points

Assume that the shared geometries are three points A, B, and C, with the respective distances $a, b, c > 0$. We place B at the origin and C at $(a, 0)$.¶

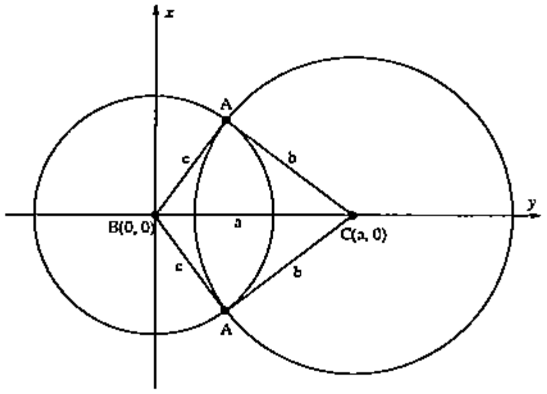

Figure 1: Placing three points

For finding the third point A we intersect two circles as illustrated in Figure 1, one circle centered at B with radius c, the other centered at C with radius b. The two circles can be disjoint (no solution), tangent (one solution) or intersecting in two points (two distinct solutions).

Algebraically, to find the point $A(x, y)$ we solve the system

$$\begin{aligned} x^2 + y^2 &= c^2 \\ (x - a)^2 + y^2 &= b^2 \end{aligned}$$

or, equivalently,

$$\begin{aligned} x &= \frac{c^2 + a^2 - b^2}{2a} \\ y^2 &= c^2 - x^2 \end{aligned}$$

With a, b, c nonnegative, we distinguish three cases:

(i) $a + b < c$ or $b + c < a$ or $a + c < b$. The circles are disjoint; there are no real solutions.

(ii) $a + b = c$ or $b + c = a$ or $a + c = b$. The circles are tangent; there is one solution, $x = \pm c, y = 0$

¶We could have placed C at $(-a, 0)$. This is equivalent up to rigid motion.

(iii) $|c - a| < b < c + a$. The circles intersect in two distinct points; the two solutions are

$$\begin{aligned} x &= \frac{c^2 + a^2 - b^2}{2a}, & y &= \sqrt{c^2 - x^2} \\ x &= \frac{c^2 + a^2 - b^2}{2a}, & y &= -\sqrt{c^2 - x^2} \end{aligned}$$

where the first solution corresponds to a counterclockwise arrangement of the three points (A, B, C) and the second solution corresponds to a clockwise arrangement of the three points (A, B, C).

Two Lines and One Point

Let $l_1 = (n_1, r_1)$ and $l_2 = (n_2, r_2)$ be two lines that intersect in the prescribed angle α. Also let $p = (x, y)$ be a point with prescribed signed distances d_1 and d_2 from lines l_1 and l_2 respectively. We place l_1 on the x-axis; i.e., $l_1 = ((0, 1), 0)$ and the point p at $(0, d_1)$ as shown in Figure 2. Then we have for l_2

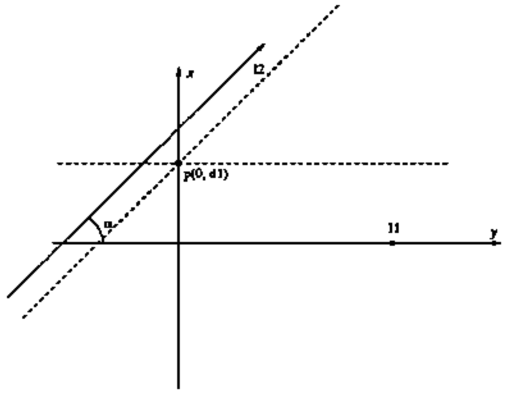

Figure 2: Placing two lines and one point

$$\begin{aligned} n_2 &= (-\sin\alpha, \cos\alpha) \\ r_2 &= d_1 \cos\alpha - d_2 \end{aligned}$$

This gives a unique solution for l_2.

Two Points and One Line

Let p_1 and p_2 be two points, and let l be a line. The distance between p_1 and p_2 be $d > 0$, and the distance between l and the points p_1, p_2 be d_1 and d_2, respectively.

We place l on the x-axis; i.e., $l = ((0,1),0)$, and we place the point p_1 on $(0, d_1)$, as illustrated in Figure 3. To find $p_2 = (x, y)$ we must intersect the line $\varepsilon : y = d_2$ with a circle with center p_1 and radius d. Algebraically, we have

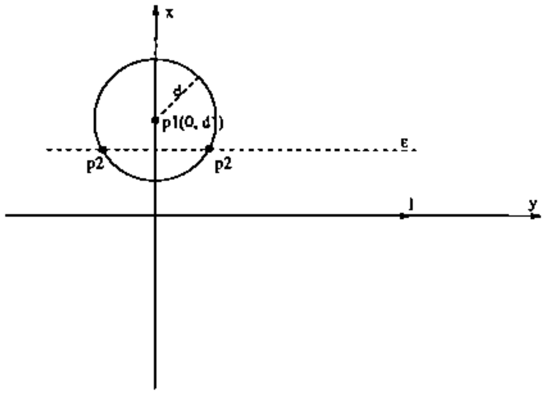

Figure 3: Placing two points and one line

$$\begin{aligned} x^2 + (y - d_1)^2 &= d^2 \\ y &= d_2 \end{aligned}$$

We distinguish three cases:

(i) $d < |d_2 - d_1|$. The line and the circle are disjoint; there are no real solutions.

(ii) $d = |d_2 - d_1|$. The line and the circle are tangent; there is one solution, $p_2 = (0, d_2)$.

(iii) $d > |d_2 - d_1|$. The line and the circle intersect; there are two distinct solutions

$$\begin{aligned} p_2 &= (\sqrt{d^2 - (d_2 - d_1)^2}, d_2) \\ p_2 &= (-\sqrt{d^2 - (d_2 - d_1)^2}, d_2) \end{aligned}$$

Three Lines

If there are three lines with angle constraints between each pair then one of the constraints must be redundant or contradictory. Technically, we consider this case overconstrained because then the constraint values cannot be independently changed.

2.3 Multiple Solutions and Root Identification

It is well known that a well-constrained geometric problem can have many incongruent solutions. Recall that at each construction step we may have to choose one of several solutions. Different choices may lead to incongruent solutions, each mathematically satisfying the given constraints.

In order to select a solution at each step, a number of heuristics are applied that make sense if the sketch with which the geometric problem has been specified is more or less like the intended solution. This is an application-specific issue that has been considered by us in [2, 1, 4].

We assume that the geometric problem has been specified by a user-prepared sketch. The point-line distances, and the angles between oriented lines are assumed to be signed quantities. As explained before, all placements have a unique solution except in two cases, which are solved as follows:

1. The relative placement of three points in a construction step has the same cyclic ordering in the plane as the ordering of the points in the original drawing.

2. The relative placement of two points and an oriented line is such that the inner product of the direction vector of the points and the line is sign invariant between the original sketch and in the chosen solution.

The geometric construction first places three geometries in this manner, and then applies a rigid-body transformation to align the three clusters accordingly. In particular, placing clusters by the shared geometric elements does not involve a reflection. We will prove later that no matter in which order the clusters are combined, the same set of triples is used to select the geometric solution, and that this implies congruence.

2.4 Well-constrained, Overconstrained and Underconstrained Sketches

Each line or point has two degrees of freedom. Each distance or angle corresponds to one equation. If we have no fixed geometries then we expect that

$$|E| = 2|V| - 3, \text{ where } |V| = n$$

Note that the solution will be a rigid body with three remaining degrees of freedom, because the constraints determine only the relative position of the geometric elements. We use this argument to define a technical notion of well-constrained sketches in which no attempt is made to account for the possibility that for special dimension values an otherwise well-constrained sketch may happen to be underconstrained. An example is shown in Figure 4. In the figure, the vertex P of the quadrilateral has a well-defined position when $\alpha + \beta \neq 90°$.

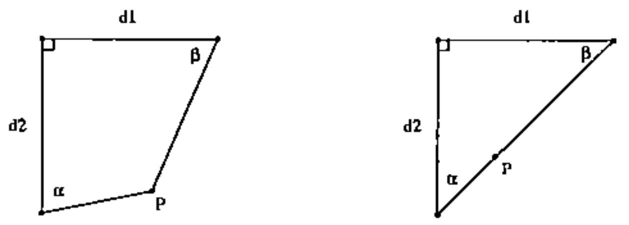

Figure 4: Degenerate Configuration (right) for $\alpha + \beta = 90°$.

But for $\alpha + \beta = 90°$ the position of P is not determined. This "semantic" notion of well-constrained problems is too specific for the constraint graph analysis because there the generic problem of constructing a solution is considered independent of dimension values.

Intuitively a dimensioned sketch is considered to be well constrained if it has a finite number of solutions for nondegenerate configurations. Similarly a dimensioned sketch is considered to be underconstrained if it has an infinite number of solutions for nondegenerate configurations. Finally a dimensioned sketch is considered to be overconstrained if it has no solutions for nondegenerate configurations.

The intuitive notions above can be made technically precise as follows:

Definition 2.1. A graph with n nodes is *structurally overconstrained* if there is an induced subgraph with $m \leq n$ nodes and more than $2m - 3$ edges.

Definition 2.2. A graph is *structurally underconstrained* if it is not overconstrained, and the number of edges is less than $2n - 3$.

Definition 2.3. A graph is *structurally well-constrained* if it is not overconstrained, and the number of edges is equal to $2n - 3$.

For an algorithm to test whether a graph is structurally well-constrained see, e.g., [6]. Note also that a structurally well-constrained graph can be overconstrained in a geometric sense, for example if there are three lines with pairwise angle constraints.

3 Correctness and Uniqueness

3.1 Problem Formulation

We are given a constraint graph $G = (V, E)$ whose nodes V are the geometric elements of the sketch, and whose edges E are the constraints between geometries. Note that the geometric elements are reduced to points and lines, and the constraints to those of distance and angle. As explained before, this is no loss of generality. For now, we restrict to geometric constraint problems in which no two lines are constrained to be parallel. As explained later, the restriction simplifies the correctness proof but is not essential to it.

We consider sets $\mathbf{C}$ whose elements are sets S that in turn have as elements nodes of G. Each set S is called a *cluster*. Intuitively, a cluster S consists of geometric elements whose position relative to each other has already been determined. A cluster thus can be considered a rigid geometric structure that has three degrees of freedom, two translational and one rotational.

Initially, we form a set $\mathbf{C}_G$ from G. For each edge $e = (u, v)$ in G, there is a cluster $S_e = \{u, v\}$. The construction steps that solve the constraint problem correspond to a *reduction step* that merges three clusters whose pairwise intersection is a singleton. The reduction is denoted by $\rightarrow$, where we sometimes indicate with subscripts which clusters are to be merged.

In this section, we consider clusters as sets and study their structure under reduction. We prove first a weak notion of correctness:

> If the constraint graph is well-constrained, then our algorithm reduces the initial set $\mathbf{C}_G$ to a singleton, no matter in which order the reduction steps are applied. That is, the set $\mathbf{C}_G$ and the reduction $\rightarrow$ have the Church-Rosser property and are Noetherian; e.g., [9].

Notice, however, that a well-constrained geometric problem has in general several incongruent solution [1]. We prove therefore a stronger uniqueness theorem in the next section:

> If the constraint graph is well-constrained and our algorithm reduces the initial set $\mathbf{C}_G$ to a single cluster using, in the construction phase, the placement rules given before, then the solutions derived by different reduction sequences place a fixed set of triples of geometric elements in the same relative position.

This result implies that different reduction sequences must produce congruent solutions.

3.2 Termination and Unique Normal Forms

Given the constraint graph $G = (V, E)$, we consider the set of clusters

$$\mathbf{C}_G = \{\{u, v\} : (u, v) \in E\}$$

Cluster sets are rewritten using a reduction $\rightarrow$. The reduction $\rightarrow$ is formally defined as follows:

Definition 3.0. Let $\mathbf{C}$ be a set of clusters $\mathbf{C}$ in which there are three clusters S_1, S_2, S_3 such that

$$\begin{aligned} S_1 \cap S_2 &= \{g_1\} \\ S_2 \cap S_3 &= \{g_2\} \\ S_3 \cap S_1 &= \{g_3\} \end{aligned}$$

Then

$$\mathbf{C} \rightarrow \mathbf{C}'$$

where

$$\mathbf{C}' = (\mathbf{C} \cup \{S_1 \cup S_2 \cup S_3\}) - \{S_1, S_2, S_3\}$$

We will show that the reduction system $(\mathbf{C}_G, \rightarrow)$ has a unique normal form that is obtained after finitely many reductions.

Definition 3.1. The set $\mathbf{C}'$ of clusters is *derived* form $\mathbf{C}$, if $\mathbf{C}'$ can be obtained by applying a sequence τ of reductions to $\mathbf{C}$. We denote this by

$$\mathbf{C} \rightarrow^* \mathbf{C}'$$

Definition 3.2. A set of clusters $\mathbf{C}$ to which $\rightarrow$ is not applicable is called *irreducible.* If $\mathbf{C}$ can be derived from $\mathbf{C}'$, then $\mathbf{C}$ is called a *normal form* of $\mathbf{C}'$.

We will show that the initial cluster set $\mathbf{C}_G$ obtained from a constraint graph that is not structurally overconstrained has a unique normal form, and that this normal form is derived by a finite sequence of reduction steps.

Lemma 3.3. If $\mathbf{C}_G$ is obtained from a constraint graph with n nodes and e edges, then every reduction sequence τ has length less than $(e+1)/2$.

Proof: $\mathbf{C}_G$ has initially e clusters. Each reduction step reduces the number of clusters by 2. □

Lemma 3.4. Let G be a constraint graph. If $\mathbf{C}_G \rightarrow^* \mathbf{C}$, then the subgraph that corresponds to a cluster in $\mathbf{C}$ is structurally well-constrained.

Proof: The proof is by induction on the length of the reduction sequence deriving $\mathbf{C}$. The induction basis is $\mathbf{C} = \mathbf{C}_G$ and is trivial. For the induction step, consider the last reduction step which merges three clusters S_1, S_2 and S_3 into a new cluster S. Let G_S be the subgraph of G corresponding to S, G_k the subgraph of G corresponding to S_k, $k = 1, 2, 3$. Note that these graphs are not necessarily induced subgraphs of G, and that the edge sets of the G_k are disjoint.

Let n_i be the number of nodes, e_i the number of edges in the subgraph of G induced by S_i. From the induction hypothesis, $e_i = 2n_i - 3$. Since $|S_1 \cap S_2| = |S_2 \cap S_3| = |S_3 \cap S_1| = 1$, S has $n = n_1 + n_2 + n_3 - 3$ vertices. Then $e_1 + e_2 + e_3 = 2n - 3$. □

Lemma 3.5. Let $\mathbf{C}_G \rightarrow^* \mathbf{C}$. If the clusters $S_1, S_2 \in \mathbf{C}$ intersect in more than one node then G is structurally overconstrained.

Proof: Assume $|S_1 \cap S_2| = m > 1$ and that G_1 and G_2 are the corresponding subgraphs. By Lemma 3.4, $e_1 = 2n_1 - 3$ and $e_2 = 2n_2 - 3$. Consider the subgraph G_0 induced by $S_1 \cup S_2$. Then G_0 has $n_0 = n_1 + n_2 - m$ vertices and $e_0 \geq e_1 + e_2$ edges. But $e_0 \geq e_1 + e_2 = 2(n_1 + n_2) - 6 > 2n_0 - 3$. Therefore, G is structurally overconstrained. □

Theorem 3.6. If the original constraint graph is not structurally overconstrained, then the reduction system $(\mathbf{C}, \rightarrow)$ is Church-Rosser, where $\mathbf{C}_G \rightarrow^* \mathbf{C}$.

Proof: Let $\mathbf{C}$ be a set of clusters, $\mathbf{C}_G \rightarrow^* \mathbf{C}$. Assume there are two different reductions possible, $\mathbf{C} \rightarrow_1 \mathbf{C}_1$ and $\mathbf{C} \rightarrow_2 \mathbf{C}_2$. It suffices to show that then $\mathbf{C}_1 \rightarrow_2 \mathbf{C}_3$ and $\mathbf{C}_2 \rightarrow_1 \mathbf{C}_3$. The reduction $\rightarrow_1$ involves the clusters S_{11}, S_{12}, S_{13}, and the reduction $\rightarrow_2$ involves the clusters S_{21}, S_{22}, S_{23}. Several cases must be distinguished based on how many distinct clusters there are among the clusters S_{ik}.

Case (3 clusters): Since there are only three distinct clusters, and since the reductions are different, there must be two clusters that have more than one element in common. Thus the constraint graph is overconstrained.

Case (4 clusters): Since each reduction requires three distinct clusters, the cluster not used in the reduction $\rightarrow_1$ must connect two of the clusters S_{11}, S_{12}, S_{13}. But then $\mathbf{C}_1$ contains two distinct clusters whose intersection is larger than one element. Hence the original graph is overconstrained. By symmetry, using $\rightarrow_2$ similarly certifies an overconstrained.

Case (5 clusters): Let $S_{11} = S_{21}$, and assume that $S_{11} \cap S_{12} = \{g_1\}$ and $S_{11} \cap S_{13} = \{g_2\}$. Clearly S_{22} and S_{23} intersect S_{11} in singletons. If S_{22} and S_{23} are both disjoint from $S_{12} \cup S_{13} - \{g_1, g_2\}$, then it is clear that $\mathbf{C}_1 \rightarrow_2 \mathbf{C}_3$ and $\mathbf{C}_2 \rightarrow_1 \mathbf{C}_3$. But if $(S_{22} \cup S_{23}) \cap (S_{12} \cup S_{13} - \{g_1, g_2\}) \neq \emptyset$, then $\mathbf{C}_1$ and $\mathbf{C}_2$ must contain two clusters that share more than one common element, contradicting that the original is not overconstrained.

Case (6 clusters): Since the clusters are all distinct, the reductions $\rightarrow_1$ and $\rightarrow_2$ commute. $\square$

Corollary 3.7. (Normal Form Theorem)
If the constraint graph G is not overconstrained, then the reduction system $\mathbf{C}_G$ has a unique normal form that is obtained by finitely many reduction steps.

Proof: Immediate from Lemma 3.3 and Theorem 3.5. $\square$

4 Geometric Uniqueness

We have shown that the reduction sequence, and hence the cluster formation, cannot interfere with termination or uniqueness of the normal form. But each reduction implies a geometric construction that places three clusters, by placing the three associated geometries with respect to each other. Since these constructions have more than one solution, it is not at all evident whether different reduction sequences will produce congruent solutions. However, we will show that this is the case for all graphs that are not structurally overconstrained.

In Section 2.3 we explained that heuristic rules are used to select one among several possible solutions in each construction step. The three elements whose relative orientation is preserved correspond to the graph nodes that are the pairwise intersections of a triple of clusters merged in a reduction step. It turns out that two different reduction sequences make use of the same set of triples. By Lemma 3.5, it is clear that in a set of clusters $\mathbf{C}$ the same triple of nodes cannot be used for two different reductions.

Definition 4.1. Let g_1, g_2, and g_3 be the three geometric elements corresponding to the shared nodes when merging the clusters S_1, S_2, and S_3. Denoting the reduction with ρ, we call the triadic set

$$\mathrm{g}(\rho) = \{g_1, g_2, g_3\}$$

the *geometry triple* of ρ.

Definition 4.2. Let $\mathbf{C}$ be a set of clusters and τ be a sequence of reductions applied to $\mathbf{C}$. Then the set

$$\mathbf{T_C}(\tau) = \{\mathrm{g}(\rho) : \rho \epsilon \tau\}$$

is the *set of geometry triples*, of $\mathbf{C}$ under τ.

Theorem 4.3. Let G be a constraint graph that is not structurally overconstrained, and assume that $\mathbf{C}_G \rightarrow^* \mathbf{C}$. Let τ_1 and τ_2 be two reduction sequences that reduce $\mathbf{C}$ to normal form $\mathbf{C}_f$. Then $\mathbf{T_C}(\tau_1) = \mathbf{T_C}(\tau_2)$.

Proof: By Corollary 3.7, $\mathbf{C}$ has a unique normal form $\mathbf{C}_f$. Moreover, it is clear from Theorem 3.6 that the reduction sequences τ_1 and τ_2 must have the same length. Consequently, the sets $\mathbf{T_C}(\tau_1)$ and $\mathbf{T_C}(\tau_2)$ have equal cardinality. We proceed by induction on the length of τ_1.

Basis: τ_1 is a single reduction. By Theorem 3.6, $\tau_1 = \tau_2$, so $\mathbf{T_C}(\tau_1) = \mathbf{T_C}(\tau_2)$.

Induction Step: Assume that the theorem is true for reduction sequences of length n, and let $|\tau_1| = |\tau_2| = n+1$. Let $\mathbf{C}_G \rightarrow^* \mathbf{C}$, $\mathbf{C} \rightarrow_{\tau_1} \mathbf{C}_f$, and $\mathbf{C} \rightarrow_{\tau_2} \mathbf{C}_f$.

Case 1: If $\tau_1 = \sigma\tau_{1*}$ and $\tau_2 = \sigma\tau_{2*}$, where σ is a single reduction, then the theorem follows from the induction hypothesis applied to $\mathbf{C}'$ where $\mathbf{C} \rightarrow_\sigma \mathbf{C}'$ and the sequences τ_{1*} and τ_{2*}. See also Figure 5.

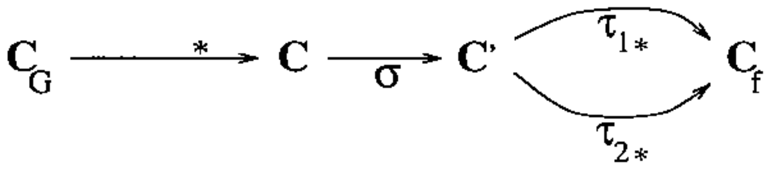
$$\begin{tikzcd} C_G \arrow[r, "*"] & C \arrow[r, "\sigma"'] & C' \arrow[r, bend left, "\tau_{1*}"] \arrow[r, bend right, "\tau_{2*}"'] & C_f \end{tikzcd}$$

Figure 5: Only one reduction can be applied to $\mathbf{C}$

Case 2: Let $\tau_1 = \sigma\tau_{1*}$ and $\tau_2 = \rho\tau_{2*}$, where σ and ρ are single reductions. Let $\mathbf{C} \rightarrow_\sigma \mathbf{C}_\sigma$ and $\mathbf{C} \rightarrow_\rho \mathbf{C}_\rho$. By Theorem 3.6, σ and ρ commute, so there is a cluster set $\mathbf{C}'$ such that

$$\mathbf{C} \rightarrow_\sigma \mathbf{C}_\sigma \rightarrow_\rho \mathbf{C}'$$
$$\mathbf{C} \rightarrow_\rho \mathbf{C}_\rho \rightarrow_\sigma \mathbf{C}'$$

$$\begin{tikzcd}
 & & \mathbf{C}_\sigma \arrow[dr, "\rho"] \arrow[drr, bend left, "\tau_{1*}"] & & \\
\mathbf{C} \arrow[r, "*"] & \mathbf{C} \arrow[ur, "\sigma"] \arrow[dr, "\rho"'] & & \mathbf{C}' \arrow[r, "\tau_3"] & \mathbf{C}_f \\
 & & \mathbf{C}_\rho \arrow[ur, "\sigma"] \arrow[urr, bend right, "\tau_{2*}"'] & &
\end{tikzcd}$$

Figure 6: Two or more reductions applied to $\mathbf{C}$

Let τ_3 be any sequence that reduces $\mathbf{C}'$ to normal form. See also Figure 6.

Applying the induction hypothesis to $\mathbf{C}_\sigma$ and the sequences τ_{1*} and $\rho\tau_3$, we have

$$\mathbf{T}_{\mathbf{C}_\sigma}(\tau_{1*}) = \mathbf{T}_{\mathbf{C}_\sigma}(\rho\tau_3) = \mathbf{T}_{\mathbf{C}'}(\tau_3) \cup \{\mathrm{g}(\rho)\}$$

And applying the induction hypothesis to $\mathbf{C}_\rho$ and the sequences τ_{2*} and $\sigma\tau_3$, we have

$$\mathbf{T}_{\mathbf{C}_\rho}(\tau_{2*}) = \mathbf{T}_{\mathbf{C}_\rho}(\sigma\tau_3) = \mathbf{T}_{\mathbf{C}'}(\tau_3) \cup \{\mathrm{g}(\sigma)\}$$

But then

$$\begin{aligned}
\mathbf{T}_{\mathbf{C}}(\tau_1) &= \mathbf{T}_{\mathbf{C}_\sigma}(\tau_{1*}) \cup \{\mathrm{g}(\sigma)\} \\
&= \mathbf{T}_{\mathbf{C}'}(\tau_3) \cup \{\mathrm{g}(\sigma)\} \cup \{\mathrm{g}(\rho)\} \\
&= \mathbf{T}_{\mathbf{C}_\rho}(\tau_{2*}) \cup \{\mathrm{g}(\rho)\} \\
&= \mathbf{T}_{\mathbf{C}}(\tau_2)
\end{aligned}$$

□

Lemma 4.4. Let G be structurally well-constrained. Assume that there are two reduction sequences $\tau_1\sigma\rho\tau_2$ and $\tau_1\rho\sigma\tau_2$ of $\mathbf{C}_G$ to normal form. Then the solutions constructed by the two sequences are congruent, provided that cluster placement does not involve reflections.

Proof Assume that

$$\mathbf{C}_G \rightarrow_{\tau_1} \mathbf{C} \rightarrow_{\rho\sigma} \mathbf{C}_1$$
$$\mathbf{C}_G \rightarrow_{\tau_1} \mathbf{C} \rightarrow_{\sigma\rho} \mathbf{C}_2$$

We know that $\mathbf{C}_1$ and $\mathbf{C}_2$ are the same set of geometric elements. We prove that the corresponding geometric clusters in $\mathbf{C}_1$ and $\mathbf{C}_2$, constructed by $\rho\sigma$ and by $\sigma\rho$, are congruent. From the proof of Theorem 3.6, we only consider the cases where ρ and σ involve five or six clusters. It is clear that in the case of six clusters corresponding geometric clusters of $\mathbf{C}_1$ and $\mathbf{C}_2$ are congruent.

Assume, next, that σ merges S_{11}, S_{12}, and S_{13}, and that ρ merges S_{11}, S_{22}, and S_{23}. We may consider S_{11} fixed, so that σ and ρ determine only the positions of $g_1 = S_{12} \cap S_{13}$ and $g_2 = S_{22} \cap S_{23}$, followed by translations and rotations of the geometric clusters of S_{12}, S_{13}, S_{22}, and S_{23}. Clearly, g_1 and g_2 are different and are therefore placed independently. By Theorem 4.3, the placement of g_i is unique. Moreover, the placement of the geometric clusters of S_{12} and S_{13} is independent from the placement of the geometric clusters of S_{22} and S_{23}, since S_{11} is fixed and G is not structurally overconstrained. Hence the geometric clusters of $\mathbf{C}_1$ must be congruent to the corresponding geometric clusters of $\mathbf{C}_2$. □

Theorem 4.5. Let G be well-constrained; then the solutions constructed by two different reduction sequences leading to normal form are congruent, provided that cluster placement does not involve reflection.

Proof Let the two sequences be τ_1 and τ_2. By Theorem 3.6 we know that the two sequences are permutations of each other. The theorem now follows from Lemma 4.5 by an induction on the number of transpositions of commuting reductions needed to change τ_2 into τ_1. □

5 Parallel Lines and Implementation Aspects

Two parallel lines form a cluster in $\mathbf{C}_G$, but geometrically the lines do not constitute a rigid body unless the distance between the lines is also known, directly by a constraint, or indirectly, by the distances of the lines from a common point. When two parallel lines are distance dimensioned, then the two constraints (parallelism and distance) should be counted as a single constraint. In this case, the two lines behave like the structures considered earlier, and it is not difficult to see that the results of Sections 3 and Theorem 4.3 apply unchanged.

For clusters containing lines that are only known to be parallel, the results of Section 3 apply unchanged since the geometric properties are nowhere used to prove the properties of reduction sequences. Furthermore, Theorem 4.3 remains valid, but the generalization of Theorem 4.5 necessitates special treatment of the case of parallel lines.

In our implementation [1, 2], sets of parallel lines at unknown distance to each other are specially marked in each cluster. The actual construction fixing the three shared geometric elements has to account for such lines, and can only fix two of them at each merge step where the third element is a point. The implementation also allows overconstrained line sets as long as the angle sum of each triple of lines is 180°.

6 Conclusions

The results presented in this paper apply to the basic algorithm explained in more detail in [1]. Most likely, they can be extended to the algorithm extensions presented there. Moreover, the techniques we used in our proofs should have wider applicability to other algorithms in the literature, especially those based on graph reductions. See Leler [7] for some example candidates.

Acknowledgements

The fruitful collaboration with B. Bouma, J. Cai and R, Paige on the development of the basic solver algorithm is gratefully acknowledged. Moreover, our discussions with V. Capoyleas have been very helpful.

References

[1] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. Technical Report TR-93-054, Purdue University, Computer Science, 1993.

[2] I. Fudos. Editable representations for 2d geometric design. Master's thesis, Purdue University, Dept. of Comp. Sci., 1993.

[3] C. M. Hoffmann. Modeling the DARPA engine in pro/engineer. In *DARPA Workshop Manufacturing, Engineering, Design, Automation*, pages 631–639, Stanford, 1992.

[4] C. M. Hoffmann. On the semantics of generative geometry representations. In *19th ASME Design Automation Conference*, 1993.

[5] C. M. Hoffmann and R. Juan. Erep, a editable, high-level representation for geometric design and analysis. In P. Wilson, M. Wozny, and M. Pratt, editors, *Geometric and Product Modeling*, pages 129–164. North Holland, 1993.

[6] H. Imai. On combinatorial structures of line drawings of polyhedra. *Discrete and Applied Mathematics*, 10:79, 1985.

[7] Wm. Leler. *Constraint Programming Languages*. Addison Wesley, 1988.

[8] J. Owen. Algebraic solution for geometry from dimensional constraints. In *ACM Symp. Found. of Solid Modeling*, pages 397–407, Austin, Tex, 1991.

[9] B. Rosen. Tree-manipulating systems and Church-Rosser theorems. *J. ACM*, 20:160–187, 1973.

[10] P. Todd. A k-tree generalization that characterizes consistency of dimensioned engineering drawings. *SIAM J. Discrete Math.*, 2:255–261, 1989.