Rule-oriented method for parameterized computer-aided design

A Verroust, F Schonek and D Roller*

The paper presents an implementation for computer-aided design with dimensional parameters. The approach is based on the use of an expert system to uncouple constraint equations, and to find a possible sequence for the computation of the geometric elements for given dimension values. A set of rules for the expert system is described that solves the problem for 2D designs. The method is illustrated with an example design.

dimensioning, geometric constraints, CAD systems, parameterized designs, geometric-reasoning methods

Contemporary CAD systems have proven to be effective for generating technical drawings and modeling 3D objects. However, in the conception stage of the design process, most CAD systems still do not have all of the flexibility required. One particularly important aspect is design with dimensional parameters.

Variable dimensions provide the means for the designer to create his or her initial design without defining the exact dimensions, which, in the early design stages, are unknown anyway. Moreover, in the design-to-manufacturing process, a number of iteration cycles are usually required before the design meets all its functional and manufacturing-related requirements. In many of these cycles, the design can undergo dimensional changes.

Work has been carried out to improve the designer interface as follows. In Reference 1, the authors classified the different approaches into three families:

- *primary approaches* that provide a solution in specific cases, such as those in References 2 and 3, or approaches that introduce an aid through macros, such as that in Reference 4,
- *algebraic approaches*[5–10] that transform the dimensioning problem into a numerical problem: the resolution of a system of (nonlinear) equations; these use a classic method such as the Newton–Raphson method or an improvement of it to solve the system; however, these approaches have limited capabilities with respect to the handling of incompletely specified drawings; further, nonconsistent constraining schemes are not rapidly detected (cf. Reference 11 for a preprocessing method that solves the problem),
- *artificial-intelligence-oriented approaches* that use inference to construct the drawing of a design progressively; the first approaches[12,13] use simple rules to fix the design gradually, but they do not detect the inconsistencies in the constraining scheme; more recently, Brüderlin[14], Aldefeld[15] and Sunde[16,17] have proposed rule-oriented approaches which detect and provide an explanation when a part of the drawing is overdetermined, and that are able to give all the numerical solutions corresponding to the set of constraints.

The authors' method follows Aldefeld's and Sunde's most recent work, but their goal is a little different: they focus on the main rules used to evaluate 2D models and the scope of the rule-based approaches.

The paper is structured as follows. The next section precisely states the problem that is to be handled. Then, the main rules used in the expert system, i.e. the creation, construction and verification rules, are explained.

The domain of application of the method is then studied: the class of constraint-based geometric models that can be evaluated by the system are described.

A description of the implementation follows. The characteristics of the expert-system shell used are given, together with the different facts used by the rules and a list of some rules that are used by the system.

LIENS, Ecole Normale Supérieure, CNRS URA 1327, 45 rue d'Ulm, 75230 Paris Cedex 05, France
*Universität Stuttgart, Institut für Informatik Breitweisenstrasse 20–22, 7000 Stuttgart 80, Germany

Paper received: 10 October 1991. Revised: 20 March 1992

0010–4485/92/100531–10 © 1992 Butterworth-Heinemann Ltd

Finally, the operation of the system is explained using an example.

RULE-ORIENTED 2D APPROACH

Geometric models that represent 2D computer-aided design are considered. These models consist of oriented line segments with their endpoints and geometric constraints, such as the following:

- two line segments share a common endpoint,
- two points are a given distance apart,
- two line segments make a given angle.

In fact, a mechanical design also includes circles and axes of symmetry.

- Tangency constraints between circles or between a line and a circle, as well as constraints on the radius, can be expressed as follows (see Figure 1):
  - A tangency constraint between a line and a circle is expressed by a right-angle constraint between the radius (to the circle–line tangency point) of the circle and the line.
  - A tangency constraint between two circles is expressed by a flat angle constraint between the radii (to the circle–circle tangency point) of the circles.
  - The diameter is translated in a distance constraint of the radius of the circle.
- In the presence of an axis of symmetry, a distance or an angular constraint in one side of the design induces a similar constraint on the other side. This can be done in a preprocessing stage.

Therefore, it is assumed in this paper that the translations from 'circle constraints' and 'distance constraint perpendicular to a segment' to 'distance and angle constraints', and the insertion of distance or angular constraints owing to the presence of axes of symmetry, has been already performed.

The problems that are to be solved are as follows:

- For a given model and given values for distance and angle constraints, the coordinates for all the model points have to be evaluated with respect to an original figure, to solve ambiguities during the computation.
- During the creation of a model, overconstrained situations have to be detected as soon as a redundant constraint is inserted.

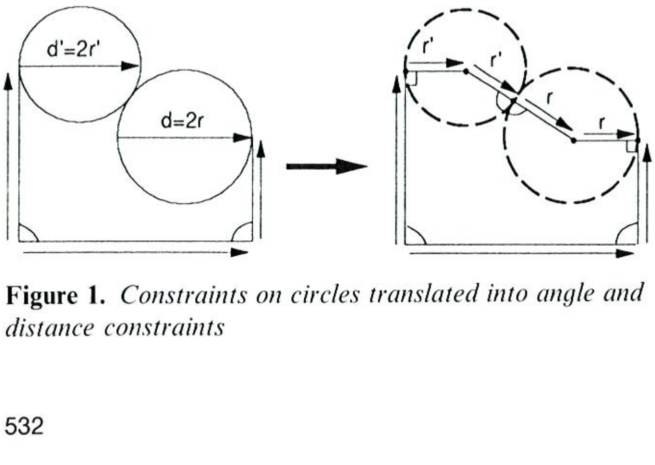

Figure 1. *Constraints on circles translated into angle and distance constraints*

The method is based on an expert-system shell, i.e. the constraints and the points are facts. Rules about these facts are used to evaluate the location of the model points and to detect inconsistent constraining schemes. In fact, to build a real interactive environment, one can follow Roller[18], Sunde[16] or Aldefeld[15] as follows:

- permanent evaluation of the model throughout the session, and maintenance of the design history, for the following reasons:
  - to give immediate explanations to the user when a part of the design is overconstrained,
  - to maintain a part of the construction when a constraint is edited,
  - to handle automatically`families of designs that have the same constraining scheme but different numerical values,
- proposal of different solutions to the user when constraints lead to multiple choices.

The authors have focused on the geometric problem, i.e. on the rules of the expert system, rather than providing all the interactive tools for the design to the user; these tools can be easily added to the system following References 15, 16 and 18.

RULES

Creation rules

Reference 16 is followed, and the notions of constrained angle sets and constrained distance sets (in short, CA and CD sets) are used to build the rules. What these two sets are is stated in more detail:

- A CA set is a set of pairs of points whose corresponding oriented segments are mutually constrained in angle.
- A CD set is a set of points with mutually constrained distances. A frame of reference is attached to each CD set, and the location of the points belonging to the CD set are expressed in it. A model is completely constrained when all the points belong to the same CD set.

With these CD sets, the solution can be built using intermediate frames (which seems not to be the case in Aldefeld's method, when his examples are considered). This distinction is of importance when studying the scope of the method, as seen below.

Definition: In this paper, two CD sets are said to be *adjacent* if they have one point in common. A segment is said to *belong to a CA set* if its endpoints form a pair of this CA set.

Elementary CA and CD sets are created when constraints are added to the model, as shown in Figure 2:

- When a angle constraint α between two directed segments is placed, a CA set is created, and the orientations are noted.
- When a distance constraint d between two points is given, a CD and a CA set are created. The positions

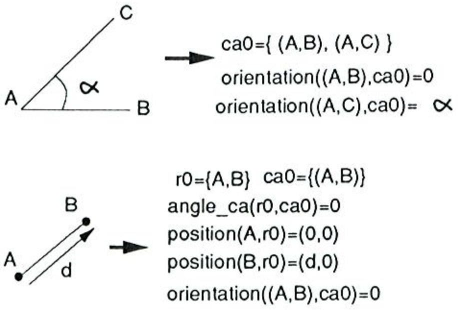

Figure 2. *Creation of CA and CD sets*

of the points in the associated frame of reference are $(0, 0)$ and $(d, 0)$. The angle of the CD set with respect to the CA set is noted as 0.

These elementary CA and CD sets are merged using the construction rules.

Construction rules

With the notion of CA and CD sets, Sunde[16] also gives two rules which result in an enlargement of the CA and CD sets, and a reduction of the number of these sets.

Rule S1: When a segment belongs to two different CA sets, these two sets are combined into one CA set.

Rule S2: If two CD sets contain a common point, and the angle between them is constrained, the two CD sets are combined into one CD set.

Apart from these rules, and what will be called the 'parallelogram' rule, the authors' main construction rules are based on the special cases of the triangle and on one case of a quadrilateral. Every time one of these rules is triggered, several CD sets are merged. This is explained in more detail below.

Triangle rules

There are three special cases for a triangle:

- *Triangle specified by three distance constraints: three CD sets defining a triangle:* The triangle rule T1 computes the intersection of two circles, and the three CD sets are merged (cf. case T1 of Figure 3)*.
- *Triangle specified by two distances and an angle not constraining the two CD sets:* The triangle rule T2 is activated, computing the intersection of a circle and a line, and the two CD sets are joined together (cf. case T2 of Figure 3).
- *Triangle specified by one distance and two angles:* A point belongs to two different lines fixed in a CD set. The triangle rule T3 computes the intersection of these lines, adding the intersection point to the CD set (cf. case T3 or Figure 3).

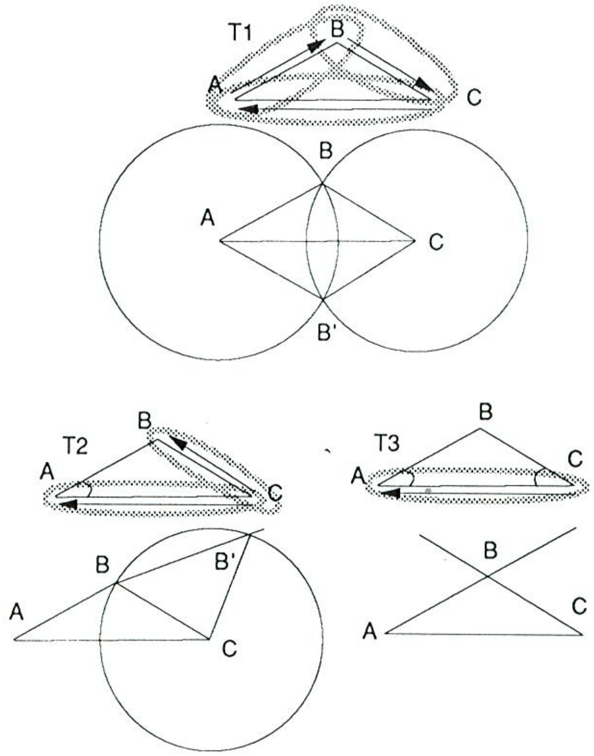

Figure 3. *Triangle rules T1, T2 and T3*

The computation of the intersection of two circles, or of a circle and a line, may lead to zero, one, two or, when the circles are identical, an infinite number of solutions. In the first and the last case, there is a numerical impossibility. This is detected during the triggering of the rules. When there are two solutions, one of them is chosen, using an angular criterion on an original figure (cf. Reference 19 for details).

Parallelogram rule

Another rule is introduced to manage angle constraints between nonadjacent segments as they may appear in technical drawings[†]. This rule assembles nonadjacent CD sets constrained by an angle. In this rule, a parallelogram is inserted in the model by the addition of a point and two segments.

In Figure 4a[‡], for example, the segments BC and DE have fixed directions, and their lengths are known. This is also the case for AB and CD. Then, the point C' is added, with

- BC' parallel to CD and equal in length,
- $C'D$ parallel to BC and equal in length.

The CD sets containing B, C' and A, B can be merged using rule S2. The same is true for the CD sets containing D, C' and D, E. Now the problem can be solved, replacing C by C' without complicating the model. In fact, when the positions of B, C' and D are found, the CD sets

[*] In the following figures in this paper, the angle constraints between adjacent segments are shown by a curve, the distance constraints are shown by an arrow, and the CD sets are shown by a closed tinted thicker curve around the relevant points.

[†] This powerful rule is not mentioned in References 14–17.

[‡] The segments marked with the same number of strokes are fixed together in direction.

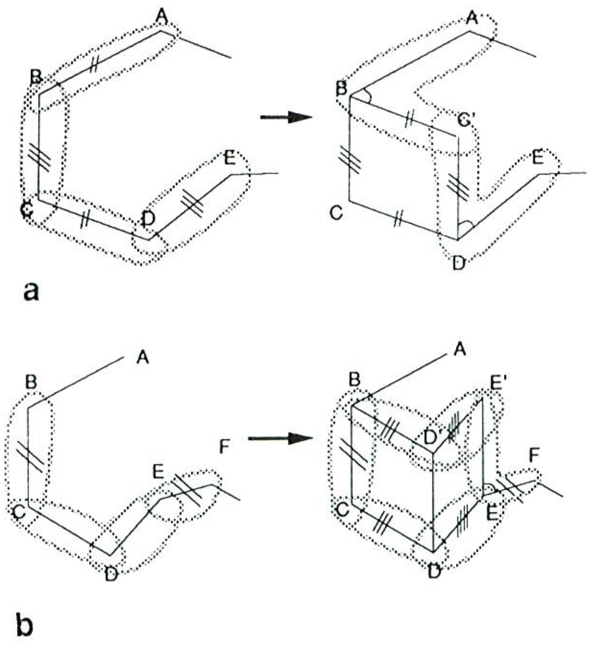

Figure 4. *Parallelogram rule; (a) one parallelogram, (b) two parallelogram*

corresponding to C, D and B, C are combined using rule S2, and the value of C is obtained.

There is a second version of the parallelogram rule that involves two parallelograms. It is shown in Figure 4b.

Quadrilateral rule

Using the triangle rules and the parallelogram rules, a quadrilateral can be constructed in most cases. One just adds rule Q for a quadrilateral in one case when a quadrilateral is fixed in angle, and when two opposite segments are constrained in distance. The problem is reduced to a computation of the intersection of two different lines, as shown in Figure 5.

The introduction of rule Q has been chosen rather than that of another parallelogram rule when two CD sets are in translation along two different lines to avoid unnecessary adjunctions of parallelograms during the computation.

The fourth section of the paper shows that the set of construction rules solves the problem for a large class of geometric models.

Verification rules

The evaluation of a model under a given set of constraints may fail for two different reasons:

- There is a numerical impossibility. If a constraint value leads to a numerical impossibility, i.e. the numerical values given for the constraint cannot lead to a solution, then the triangle rules and the quadrilateral rule detect it. The procedure associated with each rule checks whether the computation is possible before any further action is started.
- A part of the model is overdetermined. When a new constraint is introduced by the user, all the applicable rules are activated. Then, if the model is computable by the method, the CA sets at each step contain exactly the endpoints of the segments fixed in direction, and the CD sets contain the points fixed in distance. Thus, adding a redundant constraint to an already constrained part of the model leads to one of the following cases:
    - o The two segments that are constrained in angle already belong to the same CA set.
    - o The two points that are constrained in distance already belong to the same CD set.

  These conditions are easy to express as rules in the expert system. For each insertion of a constraint, these rules are activated before the construction rules: if a new constraint is redundant, the system refuses to insert it, and it sends a warning signal to the user.

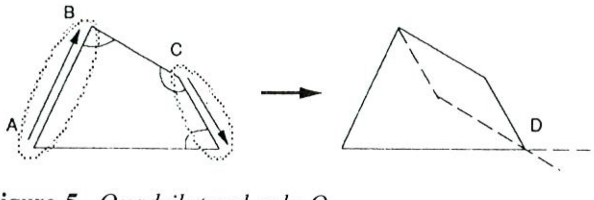

Figure 5. *Quadrilateral rule Q*

Note that, when detecting redundancy, it is presumed that all the information has been deduced from the constraints already given by the user. When the model cannot be computed by the authors' set of rules, it cannot be ensured that all the overdeterminations have been detected. The same applies in Aldefeld's and Sunde's methods. Thus, it is important to characterize the set of 2D models that the authors' method can compute.

MODEL COVERAGE

For the 2D problem, it is known that, to fix n points together, $2n - 3$ distance or angle constraints are necessary, if an origin and a rotation about the origin are given*. The problem considered here is as follows.

Can any model that is completely constrained by angle and distance constraints be computed by the authors' method, or if this is not the case, is there a describable subset of models with this property?

It is seen in the fourth section that there exist models that cannot be computed by the method. The goal is restricted to finding a description for a sufficiently large class of models that is covered by the method.

First, two classes of models consisting of n constrained points are distinguished, as follows.

Definition 1: Given a model F that includes n points and $2n - 3$ constraints, the graph $G_F = (N, E)$ is defined as follows. N is the set of points of F. E is the set of segments

* Two unknowns are associated with each point, and a distance or an angle constraint gives one equation. As an origin and a rotation are given, three unknowns are fixed. Thus, $2n - 3$ equations are required to fix the remaining unknowns

of F on which there is an angle or a distance constraint. The model is 'simple' if and only if there exists a simple cycle G' covering G_F. It is called 'nonsimple' in the other cases (see Figure 6 for examples).

Proposition 1: If the model is simple and completely constrained, and if the numerical values are consistent, the algorithm computes an evaluated version that satisfies the constraints.

The proof of this proposition is the subject of the next section.

Case of simple models

Consider the cycle G' covering the graph G_F. All the distance constraints concern adjacent points, and the angle constraints concern existing edges. G' contains exactly n edges and n points. $2n - 3$ constraints are required for the proper definition of the model. This set of constraints contains at most n distance constraints and $n - 1$ angle constraints. Thus, three cases are possible.

Simple models constrained by $n - 1$ angles and $n - 2$ distances

If the model is constrained by $n - 1$ angles, the segments are all fixed in direction.

- *One set of adjacent segments:* There is only one CD set for the whole model, containing $n - 1$ points, as shown in Figure 7a. Then, rule T3 fixes all the points.
- *Two sets of adjacent segments:* There are two CD sets containing the n points, as shown in Figure 7b. Rule Q computes their union.

Simple models constrained by $n - 2$ angles and $n - 1$ distances

As there are only $n - 2$ angle constraints, two cases are possible, as follows:

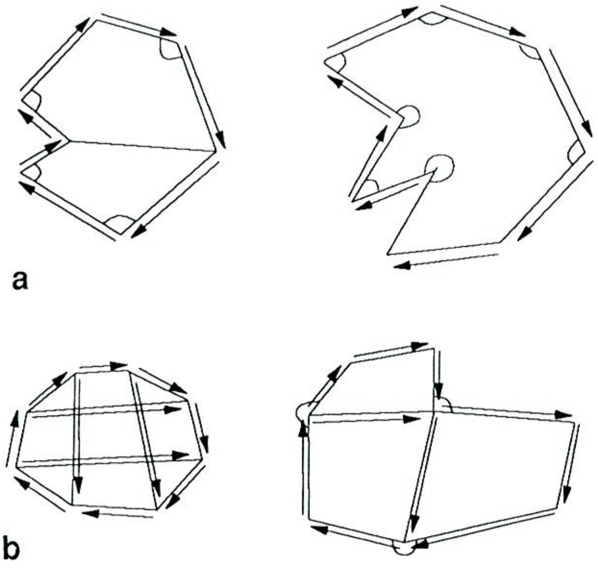

Figure 6. *Simple and nonsimple models; (a) simple models, (b) nonsimple models*

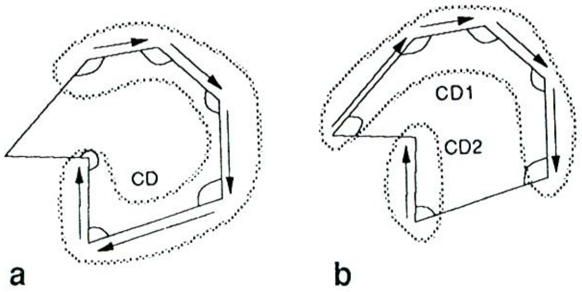

Figure 7. *$n - 1$ angles and $n - 2$ distances; (a) one CD set containing $n - 1$ points, (b) two CD sets for the n points*

- The $n - 1$ adjacent segments are fixed in their relative direction (they belong to the same CA set). Let s be the segment that is not constrained by an angle, and s' the segment that is not constrained by a distance.
  - o If $s = s'$, all the points belong to the same CD set, and the whole model can be computed (see Figure 8a).
  - o If s and s' are adjacent, the $n - 1$ points incident to the remaining segments belong to the same CD set. The last point is computed using rule T2 (see Figure 8b).
  - o If s and s' are not adjacent, the n points are partitioned into two CD sets. A parallelogram rule reduces this case to the previous case.
- The $n - 1$ segments are partitioned into two CA sets. If all the segments belonging to the same CA set are connected, there are two adjacent CD sets whose angle is fixed by a line. Rule T2 yields the result. In the other case, a sequence of applications of the 'parallelogram' rule connects the segments belonging to a common CA set (see Figure 9). There is then a model that is similar to the previous one.

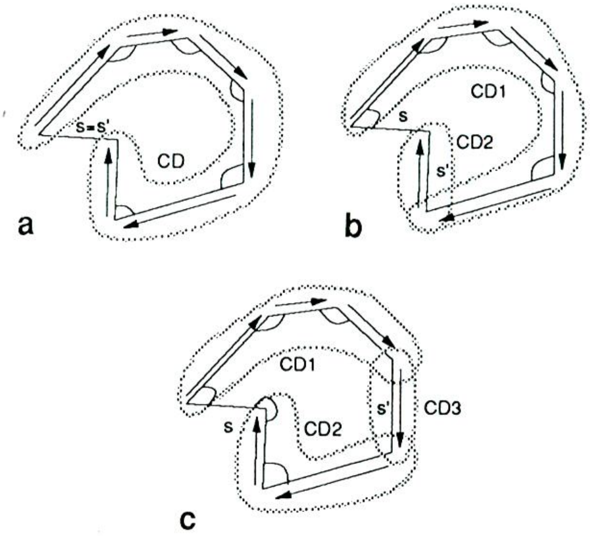

Figure 8. *$n - 1$ angles constrained together*

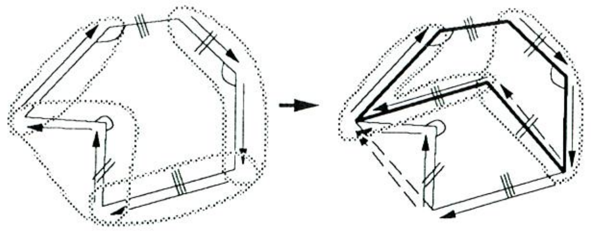

Figure 9. *Transformation from disconnected CA set into connected CA set*

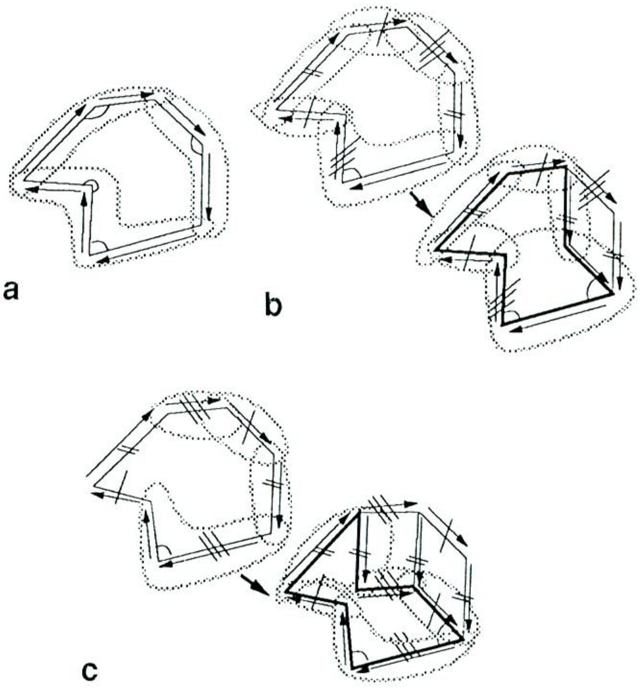

Figure 10. *$n - 3$ angles and n distances*

Simple models constrained by $n - 3$ angles and n distances

The distances of the n segments are fixed, and there are at most three CA sets. Two cases are possible, as follows:

- All the segments belonging to the same CA set are adjacent (see Figure 10a). Thus, three CD sets cover the n points, and rule T1 leads to the result.
- The CA set is disconnected, or several CA sets are mixed together. The application of the parallelogram rule transforms this case into the previous one (see, for example, Figures 10a and b).

Nearly simple models

It has been seen that the simple models can be computed by the method. This class of models is large, but the condition for the segments to belong to a cycle is still very restraining. Many practical examples that can also be solved by the method do not satisfy this condition (see Figure 11). Therefore, a larger class of models with the same properties with regard to the algorithm is now described. The elements of this class are called 'nearly simple' models. Intuitively, nearly simple models are models that can be 'decomposed' into simple models, or treated sequentially by the algorithm as a union of simple models. Definition 2 is more formal.

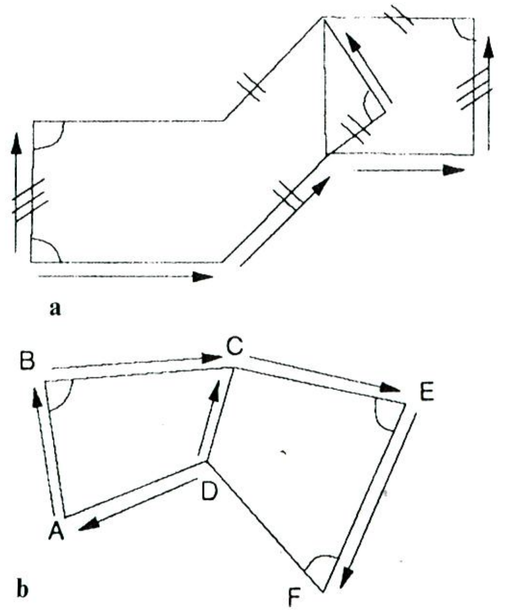

Figure 11. *Nearly simple models*

Definition 2: Let F be a model, i.e. a set of points with distance constraints on some of them, and angle constraints on segments joining some of the points. F is said to be 'nearly simple' if there exists a sequence of simple models $F_1, \ldots, F_n$ such that the following are true:

- F contains F_1.
- $\bigcup_{i \leqslant n} N_i = N$, where $G_{F_i} = (N_i, E_i)$ and $G_F = (N, E)$.
- Let $(C_i)_{i \leqslant n}$ and C be the sets of constraints of $(F_i)_{i \leqslant n}$ and F, respectively. Then,
  - $C = \bigcup_{i \leqslant n} (C_i \cap C')$,
  - if $i > 1$, C_i can be decomposed in $C_i = (C \cap C_i) \cup D_i \cup A_i$, where
    - D_i is composed of distance constraints on a couple of points such that there exist N_j, with $j \leqslant i$, containing them,
    - A_i is composed of angle constraints on a couple of segments joining points such that there exist N_j, with $j \leqslant i$, containing them.

Then, using the decomposition in simple models and Proposition 1, Proposition 2 can be stated.

Proposition 2: If the model is nearly simple and completely constrained, and if the numerical values are consistent, the algorithm computes an evaluated version that satisfies the constraints.

Note that methods using only one frame of reference in the computation, as in Brüderlin's[14], Aldefeld's[15] and Sunde's[16] first methods, cannot solve all the simple and nearly simple models. For example, to solve the case in Figure 11b, two frames are needed, one where A, B and

C are joined, and another to join C, E and F. Hence, using several frames during the computation enlarges the set of models that are resolvable using a rule-based approach. Moreover, as the parallelogram rule is not mentioned in the previous models, it cannot be known whether Sunde's last method[17] computes nearly simple models.

Models that are not nearly simple

When a model is not nearly simple, it contains a subset composed of n points and segments, with $2n - 3$ distance and/or angle constraints on the segments that cannot be decomposed into simple models. Typically, such types of subset are special cases, where the n unknowns are involved simultaneously. The value of n is not bounded. In fact, Proposition 3 can be stated.

Proposition 3: For any value of n, there exist an m, with $m > n$, and a model F constraining m points with $2m - 3$ angle and distance constraints such that F does not contain a simple model.

This proposition is proved by a way being shown to build a family of models $F_1, F_2, \ldots$ such that, for all i, F_i constrains $3 + i$ points, and F_i does not contain a simple model.

F_1 is described, and the way is given to build F_{i+1} from F_i when $i > 1$ (F_1, F_2 and F_3 are shown in Figure 12), as follows:

- F_1 has a distance constraint on p_3p_4, and it has the constraint angles (p_1p_3, p_1p_4), (p_1p_3, p_1p_2), (p_1p_2, p_2p_3) and (p_2p_3, p_2p_4).
- For $i > 1$, F_{i+1} contains $j = 3 + i$ points, and it has the constraints of F_i minus one of the following:
  - ○ the angle constraint (p_2p_{j-1}, p_2p_j) if j is even, or
  - ○ the angle constraint (p_1p_{j-1}, p_1p_j) if j is odd,

  plus the three constraints involving the new point, i.e.

  - ○ a distance constraint on $p_{j-1}\,p_j$,
  - ○ two angle constraints (p_1p_{j-1}, p_1p_j) and (p_2p_{j-1}, p_2p_j).

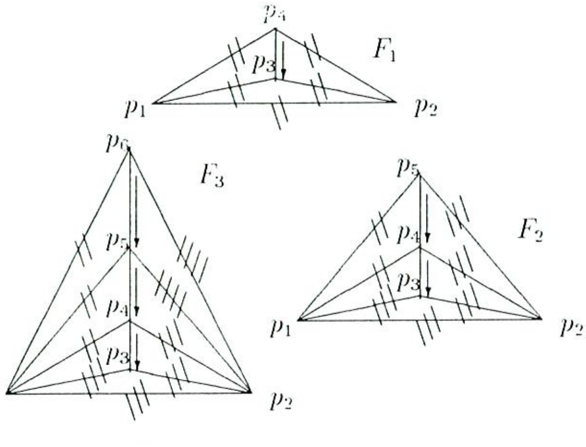

Figure 12. *F_1, F_2 and F_3*

The fact that F_i does not contain a simple model is obvious by induction using this construction.

To compute these models numerically, one can use the fact that the area of the upper triangle is equal to the sum of the areas of the small ones. This equation involves all the unknowns of the model.

Thus, the rule-based methods have their intrinsic limitations: a finite set of rules cannot solve all the 2D models constrained by angles and distances. In fact, if the constraints are seen as equations, the rule-based methods are a way of decomposing the computation of the set of nonlinear equations into a sequence of computations of subsystems of bounded size. This decomposition is impossible when the set must be solved globally. Nevertheless, this method solves the typical models of mechanical drawings, and it can be used in this context.

IMPLEMENTATION

Characteristics of expert system

An existing expert-system shell that was developed by Benoît Faller[20] was used. This was written in C, and it has the following characteristics:

- If several rules are applicable at the same time, it manages the triggering of the rules with respect to the priority order given by the user.
- It allows actions in the rules that are calls to procedures written in C.
- It allows variables in the conditions.

Each rule is structured as follows:

- The name of the rule, i.e. $R\langle$word or number$\rangle$ and a priority order for the triggering of the rule.
- A list of conditions for the facts of the system's base of facts. These conditions may contain variables (?x means that x is a variable), and they can be negated. In this case, it means that the rules can be triggered when no instantiation of the fact appears in the base of facts. Comparisons of the values of variables are allowed conditions.
- Action in the rule, i.e. the call of a procedure which will give a list of new facts as a result. These procedures are used here to compute the positions of the points, the angles between two CA sets etc. They return facts that are inserted in the base of facts.
- The insertion or deletion of facts.

Example:

```
RuleS1 priorite 0 si      (orientation ?seg1 ?ca1 ?alpha)
                          (orientation ?seg1 ?ca2 ?beta)
                          -(equivca ?ca1 ?ca2 ?delta)
                 et si (<> ?ca1 ?ca2)
                 essayer (directe ?ca1 ?ca2 ?alpha ?beta)
                 alors -(orientation ?seg1 ?ca2 ?beta)*
```

If the base of facts contains

- (orientation *s*1 ca 2 0.)
- (orientation *s*1 ca1 90.)
- (orientation *s*2 ca2 180.)

as ca1 and ca2 are different, the RuleS1 is applicable. Then, the procedure directe(ca1, ca2, 90., 0.) is called, and the fact (orientation *s*1 *s*2 0.) is deleted.

The relative order for the triggering of the rules is as follows:

- the verification rules, to detect overdeterminations as soon as a new constraint is inserted,
- the 'cleaning' rules, to delete facts after the union of CA or CD sets,
- the 'intermediate' rules, such as rules S1 and S2,
- the triangle rules T1, T2, T3, and the quadrilateral rule Q,
- the parallelogram rule.

Facts used in rules

The facts used in the rules can be classified into three families:

- entrance facts,
- facts involving CA sets,
- facts involving CD sets.

These facts are described further below.

Entrance facts

Entrance facts are generated when the user inserts a new constraint or enters the geometry of the model. Thus, there is a type of fact that corresponds to each type of constraint. As segments and points are dealt with, another type of fact denotes the topology of the model. Thus, there are the following facts:

- (*adjacent p seg*): The point p is one of the extremities of the segment seg.
- (*original p x y*): x and y are the coordinates of the point p in the model originally entered.
- (*distance p1 p2 d*): the distance between p_1 and p_2 is equal to d.
- (*angle seg1 seg2 alpha*): the angle taken in the counterclockwise direction between the directed segments seg1 and seg2 is equal to alpha.

Facts involving CA sets

All the facts involving CA sets denote the angular position of objects with respect to a CA set:

- (*orientation seg ca alpha*): for a segment seg,
- (*angle_ca r1 ca alpha*): for a CD set r_1,
- (*equivca ca1 ca2 alpha*): for two CA sets ca1 and ca2.

Facts involving CD sets

A frame of reference is associated with each CD set. In these facts, the same name is used for the CD set and for its frame of reference.

- (*position p r1 a b*): The point p, element of the CD set r_1, has the coordinates (a, b) in r_1.
- (*on_line p1 r1 A B C p2*): The point p_1 belongs to the line that satisfies the equation $Ax + By + C = 0$ in the CD set r_1, and passes through the point p_2 whose position is known in r_1.
- (*rotation r1 r2 a1 b1 a2 b2 p*): The CD sets r_1 and r_2 have the point p in common. Its coordinates are (a_1, b_1) in r_1 and (a_2, b_2) in r_2.
- (*transl_R r1 r2 a1 b1 A B C alpha p1 p2*): The CD sets r_1 and r_2 are fixed in direction by the angle α, which is the angle between r_1 and r_2. r_2 is free to move on a line that satisfies the equation $Ax + By + C = 0$ in r_1, and passes through the point p_2, which is known in r_1. More precisely, the point p_1 belongs to this line, and (a_1, b_1) are its coordinates in r_2.
- (*transl_C r1 r2 r3 a1 b1 a2 b2 a3 b3 a4 b4 alpha p1 p2*): The CD sets r_1 and r_2 are fixed in direction by the angle α, which is the angle between r_1 and r_2. r_2 is free to move on a circle whose center is p_1, which is known in r_1, and whose radius is the distance between p_1 and p_2. More precisely, p_1 has the coordinates (a_1, b_1) in r_1 and (a_2, b_2) in r_3, and p_2 has the coordinates (a_3, b_3) in r_2 and (a_4, b_4) in r_3.
- (*equiv r1 r2 m11 m12 m13 m23*): The two CD sets are fixed in translation and in rotation, and the transformation from r_2 to r_1 has the matrix

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ -m_{12} & m_{11} & m_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

EXAMPLE OF WORKING OF SYSTEM

The following example is shown in Figure 13. The user's insertion of facts is preceded by '←', and the insertions caused by the rules are preceded by '⇒'. A '–' preceding the face denotes the deletion of the fact from the base of facts.

The user first enters the topology of the model and the coordinates of the points in the original model (see Figure 13a):

← (original *p*2 306 –259)
← (original *p*1 143 –255)
← (original *p*4 119 –209)
← (original *p*3 265 –141)
← (adjacent *p*4 *p*3*p*4) . . .

The user inserts a distance constraint. A CD set *r*0 associated with the CA set ca0 is created, as shown in Figure 13b:

← (distance *p*3 *p*4 161.000)
⇒ (position *p*3 *r*0 0.00 0.00)
⇒ (position *p*4 *r*0 161.000 0.00)
⇒ (angle_ca *r*0 ca0 0.00)
⇒ – (distance *p*3 *p*4 161.000)
⇒ (orientation *p*3*p*4 ca0 0.000)

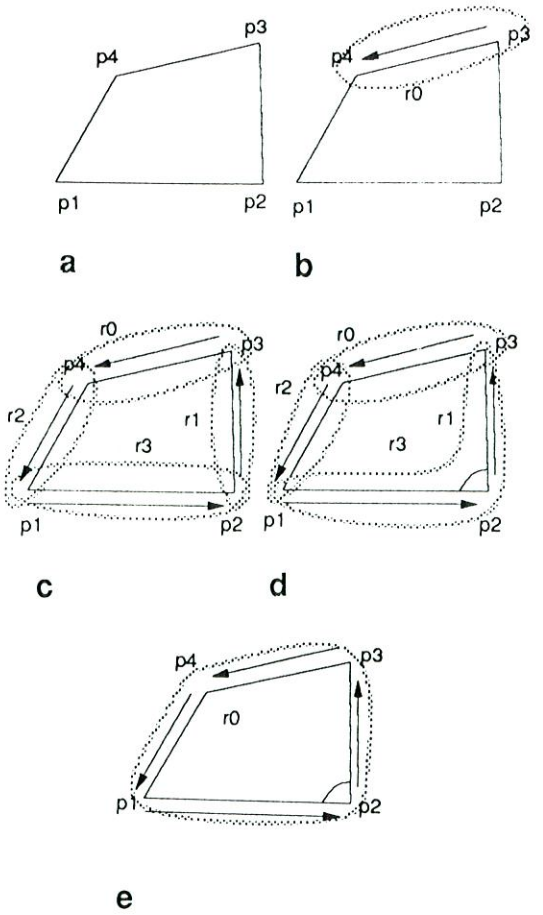

Figure 13. *Example; (a) entry of topology and coordinates, (b) insertion of distance constraint, (c) insertion of three distance constraints, (d) computation of union, (e) all points belong to CD set r0*

The insertion of three distance constraints follows (see Figure 13c):

← (distance $p2$ $p3$ 106.000)
⇒ (position $p2$ $r1$ 0.00 0.00)
⇒ (position $p3$ $r1$ 106.000 0.00)
⇒(angle_ca $r1$ ca1 0.00)
⇒ (orientation $p2p3$ ca1 0.000)
⇒ −(distance $p2$ $p3$ 106.000)
⇒ (rotation $r0$ $r1$ 0.00 0.00 106.00 0.00 $p3$)
← (distance $p1$ $p4$ 51.000)
⇒ (position $p1$ $r2$ 0.00 0.00)
⇒ (position $p4$ $r2$ 51.000 0.00)
⇒(angle_ca $r2$ ca2 0.00)
⇒ (orientation $p1p4$ ca2 0.000)
⇒ −(distance $p1$ $p4$ 51.000)
⇒ (rotation $r0$ $r2$ 161.00 0.00 51.00 0.00 $p4$)
← (distance $p1$ $p2$ 163.000)
⇒ (position $p1$ $r3$ 0.00 0.00)
⇒ (position $p2$ $r3$ 163.000 0.00)
⇒ (angle_ca $r3$ ca3 0.00)
⇒ (orientation $p1p2$ ca3 0.000)
⇒ −(distance $p1$ $p2$ 163.000)
⇒ (rotation $r2$ $r3$ 0.00 0.00 0.00 0.00 $p1$)
⇒ (rotation $r1$ $r3$ 0.00 0.00 163.00 0.00 $p2$)

Then, the user inserts an angle constraint which induces the triggering of rule S1:

← (angle $p1p2$ $p2p3$ −249)
⇒ (orientation $p1p2$ ca1 111.000)
⇒ −(angle $p1p2$ $p2p3$ −249)
⇒ (equivca ca1 ca3 −111.000)
⇒ −(orientation $p1p2$ ca3 0.000)

The two CD sets r_2 and r_3 have the point p_2 in common, and they are fixed in direction. Rule S2 computes their union (see Figure 13d):

⇒ (equiv $r1$ $r3$ −0.358368 0.933580 58.414 152.174)
⇒ −(position $p2$ $r3$ 163.00 0.00)
⇒ (position $p1$ $r1$ 58.410 152.170)
⇒ −(position $p1$ $r3$ 0.00 0.00)
⇒ −(rotation $r1$ $r3$ 0.00 0.00 163.00 0.00 $p2$)
⇒ (rotation $r1$ $r2$ 58.410 152.170 0.00 0.00 $p1$)
⇒ −(rotation $r2$ $r3$ 0.00 0.00 0.00 0.00 $p1$)

The CD sets $r0$, $r1$ and $r3$ are in rotation, and rule T1 is actioned:

⇒ (equiv $r0$ $r2$ 0.188861 0.982004 151.36807 50.08219)
⇒ −(position $p3$ $r1$ 106.00 0.00)
⇒ −(position $p2$ $r1$ 0.00 0.00)
⇒ (position $p2$ $r0$ −1.740 105.990)
⇒ −(position $p1$ $r1$ 58.410 152.170)
⇒ (position $p1$ $r0$ 151.598 50.623)

All the points belong to the CD set $r0$ (see Figure 13e). The model satisfying the constraints is displayed, and the session is finished.

This model is computed in 0.9 s by the authors' expert system on a SparcStation 1+. Note the following:

- The computing time depends on the size of the model, and also on the number of CD or CA sets used in the resolution. For example, the CPU time needed for models involving eight points can vary over 1.5–2.2 s.
- The computation time can differ for the same model when the insertion order of the constraints is changed, but the resulting figure is the same.
- When the model is underconstrained, the figure is partially fixed. The user can only have the list of the CD and CA sets present in the base of facts, and the position of the points and the angles of the segments with respect to these sets. Numerical methods could be used to compute a solution using these sets instead of the set of equations induced by the constraints.

CONCLUSIONS

It has been shown that using rules to compute parameterized mechanical designs is feasible. The set of models that the proposed system can solve is sufficiently general, and it essentially comprises the models that are of relevance to 2D mechanical designs.

Experiments with an implementation of this approach have indicated that the computation of constrained models is fast.

The amount of memory used during the session is not negligible. As the rules contain variables, the system maintains the possible instantiations of the premises of the rules during all the session to accelerate the deduction process. An improvement would be to structure the drawings to reduce the number of variables during the execution process.

The scope of the method has been studied, and the intrinsic limitations of rule-oriented approaches have been shown. An idea for improving the resolution would be to mix numerical and rule-oriented methods.

A detailed description of this approach and an extension to 3D models can be found in Reference 19.

ACKNOWLEDGEMENTS

The work described in this paper was sponsored by Hewlett–Packard GmbH, Germany. The authors would like to thank Professor Claude Puech, the head of the Graphics Group at LIENS, France, and Dr Steve Hull from Hewlett–Packard GmbH for valuable discussions about various aspects of this research project.

REFERENCES

1 **Roller, D, Schonek, F and Verroust, A** 'Dimension-driven geometry in CAD: a survey' *in Theory and Practice of Geometric Modeling* Springer-Verlag (1989) pp 509–523

2 **Fitzgerald, F** 'Using axial dimensions to determine the proportions of line drawings in computer graphics' *Comput-Aided Des.* Vol 13 No 6 (1981) pp 377–382

3 **Gossard, D, Zuffante, R and Sakurai, H** 'Representing dimensions, tolerances and features in MCAE systems' *IEEE Comput. Graph. & Applic.* Vol 8 (Mar 1988) pp 51–59

4 **Cugini, U, Devoti, C and Galli, P** 'System for parametric definition of engineering drawings' *Proc. MICAD '85* (1985)

5 **Hillyard, R and Braid, I** 'Analysis of dimensions and tolerances in computer-aided mechanical design' *Comput.-Aided Des.* Vol 10 No 3 (1978) pp 161–166

6 **Hillyard, R and Braid, I** 'Characterizing non ideal shapes in terms of dimensions and tolerances' *Comput. Graph.* Vol 12 No 3 (1978) pp 234–238 (*Proc. SIGGRAPH '78*)

7 **Light, R** 'Symbolic dimensioning in computer-aided design' *Master's Thesis* Massachusetts Institute of Technology, USA (May 1979)

8 **Light, R, Lin, V and Gossard, D** 'Variational geometry in CAD' *Comput. Graph.* Vol 15 No 3 (1981) pp 171–177 (*Proc. SIGGRAPH '81*)

9 **Light, R and Gossard, D** 'Modification of geometric models through variational geometry' *Comput.-Aided Des.* Vol 14 No 4 (1982) pp 209–214

10 **Lee, L and Andrews, G** 'Inference of the positions of components in an assembly: Part 2' *Comput-Aided Des.* Vol 17 No 1 (1985) pp 20–24

11 **Chyz, W** 'Constraint management for CSG' *Master's Thesis* Massachusetts Institute of Technology, USA (Jun 1985)

12 **Brüderlin, B** 'Using Prolog for constructing geometric objects defined by constraints' *Proc. Europ. Conf. Computer Algebra* (1985) pp 448–459

13 **Sunde, G** 'Specification of shape by dimensions and other geometric constraints' *Proc. IFIP WG 5.2 Geometric Modeling* Rensselaerville, NY, USA (May 1986)

14 **Brüderlin, B** 'Constructing three-dimensional geometric objects defined by constraints' *Proc. 1986 Wkshp. Interactive 3D Graphics* Chapell Hill, NC, USA (Oct 1986) pp 111–129

15 **Aldefeld, B** 'Variation of geometries based on a geometric-reasoning method' *Comput-Aided Des.* Vol 20 No 3 (1988) pp 117–126

16 **Sunde, G** 'A CAD system with declarative specification of shape' *Eurographics Wkshp. Intelligent CAD Systems* Noordwijkerhout, Netherlands (Apr 1987) pp 90–105

17 **Sunde, G and Kallevik, V** 'A dimension-driven CAD system. Utilizing AI techniques in CAD' *Report 860216-1* Senter for Industriforskning (Dec 1987)

18 **Roller, D** 'A system for interactive variation design' *1988 IFIP/NSF Wkshp. Geometric Modeling: Geometric Modeling for Product Engineering* North Holland (1989)

19 **Verroust, A** 'Etude de problèmes liés à la définition, la visualisation et l'animation d'objets complexes en informatique graphique' *Thèse d'Etat* Université Paris-Sud, France (Dec 1990)

20 **Faller, B** 'Le système METRO: rapport préliminaire' *Rapport Interne* Faculté d'Orsay, LRI, France (Apr 1988)