ELSEVIER

Artificial Intelligence 99 (1998) 73–119

Artificial Intelligence

Geometric construction by assembling solved subfigures [1]

Jean-François Dufourd *, Pascal Mathis [2], Pascal Schreck [3]

Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection (L.S.I.I.T., URA CNRS 1871), Université Louis Pasteur, 7, rue René Descartes, 67084 Strasbourg, France

Received June 1996

Abstract

Among the expected contributions of Artificial Intelligence to Computer-Aided Design is the possibility of constructing a geometric object, the description of which is given by a system of topological and dimensional constraints. This paper presents the theoretical foundations of an original approach to formal geometric construction of rigid bodies in the Euclidian plane, based on invariance under displacements and relaxation of positional constraints. This general idea allows to explain in greater detail several methods proposed in the literature. One of the advantages of this approach is its ability to efficiently generalize and join together different methods for local solving. The paper also describes the main features of a powerful and extensible operational prototype based on these ideas, which can be viewed as a simple multi-agent system with a blackboard. Finally, some significant examples solved by this prototype are presented. © 1998 Elsevier Science B.V.

Keywords: Geometric formal construction; System of geometric constraints; Computer-aided design; Local solving; Assembling of figures; Multi-agent system; Blackboard

1. Introduction

Following the seminal work of Sutherland [52], an expected contribution of Artificial Intelligence to Computer-Aided Design (CAD) is the possibility of building a 3D

* Corresponding author. Email: dufourd@dpt-info.u-strasbg.fr.

[1] This research is supported by the *GDR-PRC de Programmation* and *GDR-PRC Algorithmique, Modèles et Infographie* (French CNRS).

[2] Email: mathis@dpt-info.u-strasbg.fr.

[3] Email: schreck@dpt-info.u-strasbg.fr.

rigid object defined by a *system of geometric constraints* [45] for its topology and embedding.

The topological constraints express incidence and adjacency relationships between the components of the object, namely its vertices, edges and faces. Usually, in CAD, drawing tools hide the setting of these constraints during the so-called *functional decomposition* of the object. The embedding constraints express the form and the metrics of the object. The designer gives them as a *system of dimensions* constraining the components of a sketch. The problem is then to build components which satisfy all these constraints.

When translating them into real number equations, we come upon the problem of solving a system of polynomial or transcendental equations. Such a question has generally been approached in a purely numerical way, sometimes after preprocessing, often using graphs to split the initial system into subsystems. That is the case with the Newton–Raphson method [35,39] or the homotopy-based method [30]. Advantages and drawbacks of such an approach have often been described in the literature, e.g. in [28,44,54].

As stated in [1,2], it seems to us interesting to tackle this question in two phases. The first phase is a solving process yielding a formal construction plan, and the second one is a numerical interpretation of the construction plan. This way, the possibility of producing several numerical solutions is preserved, problems of convergence are eliminated, errors of approximation are not propagated and failures can be fairly diagnosed. Moreover, the formal expression of a geometric construction is a powerful means of rendering the corresponding object generic.

Formally solving systems of geometric constraints in the plane has many similarities with solving geometric constructions as encountered in education area and studied in Computer-Aided Instruction (CAI) [11,18,46–48]. So, dimensioning a sketch graphically sets a constraint system similar to the ones encountered in high-school mathematics. The aims, however, are quite different. In CAI, one wants to obtain all the solutions and discuss them, even in degenerate cases, while, in CAD, one expects to obtain in the general case the most plausible solution.

Formal solving appears in some CAD knowledge-based systems, e.g. [1,8,16]. Such systems have several aspects in common with geometric mechanical provers based on axiomatics [22]. Moreover, efficient methods like constraint graph decomposition [27,41] or progressive figure *rigidification* [50,51,54,55] could be reconsidered using a two-phase treatment. But these methods are restricted to specific types of constraints and cannot be applied easily to any geometric universe.

This question has also been tackled by computer algebra systems [19,20,32]. For such systems, formal solving is quite similar to automatic proving based on formal polynomial reasoning [15,56]. Restrictions on the generality or size of the solved systems and the tedious calculation involved are common criticisms of this approach. But surely, the main drawback is that both with computer algebra as well as with numerical solving, one must work with systems of equations whose variables are the coordinates of the geometric objects rather than the geometric objects themselves.

In this paper, we present a general formal framework for systems of geometric constraints as encountered in CAD to specify rigid body. We propose an original approach for a solving process based on invariance under displacements and local solving of parts. This framework specifies the methodological foundation proposed in the literature [7,30,41,50,51], in particular by formalizing the *assembling* of subsystems and computed metric constraints, using what we call a *border*. The approach used in [7], in its use of two levels of construction, local and global, looks similar to ours. There are however considerable differences in its underlying concepts, the content of the two levels, and the numerical rather than formal character. Moreover, besides its efficiency for common problems, one of the advantages of our approach is its ability to encompass different methods, including ones based on numerical iterations and computer algebra. It has great solving power and wide application.

Next, we present the current implementation of our framework, employing different strategies and tactics. We describe a prototype which works in the plane using two phases, one formal and one interpretative, by using assembling and local solving methods. These methods are two knowledge-based systems and the Newton–Raphson method [35,39]. We focus here on assembling, on the general characteristics and use of methods, rather than on technical details. We show that our CAD prototype is closely related to Artificial Intelligence multi-agent systems with a blackboard [13]. Finally, we give several examples of significant problems which have been solved by the prototype.

The structure of the paper is as follows. Section 2 gives an example of an easy to understand problem solved using our method. Section 3 outlines the formal framework for the geometric universe and systems of constraints. Section 4 defines the formal solving method and the assembling, together. Section 5 presents some indications of workable strategies with local solving methods to make this framework effective. Section 6 briefly presents the prototype. Section 7 shows on three examples how they can be used. Section 8 compares our propositions with other works, and Section 9 concludes our discussion.

2. An example of geometric construction with assembling

Our method allows us to build step by step geometric constructions in the Euclidian plane by locally solving parts of the system of constraints and assembling them by displacements. We examine the principles of this approach in the example illustrated by the constraint system of Fig. 1.

Fig. 1 represents a dimensional part where characteristic points, curves and numerical values are respectively named $a, \ldots, g$, Γ and $k1, \ldots, k5, \alpha 1, \ldots \alpha 3$, in order to specify constraints. In the technical drawing area, the meaning of a dimensional double-arrow differs according to the positions of the joined lines. In the example, $k1$ corresponds to a constraint of distance between points a and b, and $k5$ to a constraint of distance between point e and oriented line fg.

Fig. 1 specifies all the constraints imposed on our sketch, the hexagon $cdefga$ with point b. Some constraints are drawn in the figure by dimensions, and others, like constraints of tangency, are implicit. The question of transforming them into explicit

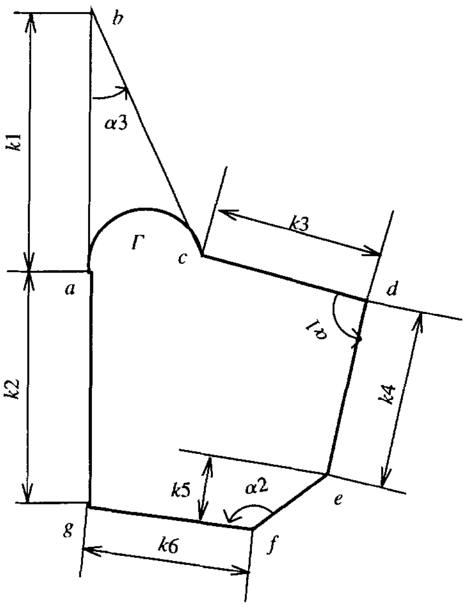

Fig. 1. A constraint system.

constraints is beyond the scope of this paper. We thus consider in the following that we have a problem textually set by:

distance from point a to point $b = k1$
distance from point a to point $g = k2$
distance from point c to point $d = k3$
distance from point d to point $e = k4$
distance from point e to oriented line $fg = k5$
distance from point f to point $g = k6$
angle from oriented line dc to oriented line $de = \alpha 1$
angle from oriented line fe to oriented line $fg = \alpha 2$
angle from oriented line ba to oriented line $bc = \alpha 3$
points a, b, g are collinear
curve Γ is a circular arc
oriented line ab is tangent to Γ at a
oriented line bc is tangent to Γ at c

It must be completed by topological constraints coming from the drawing, e.g. points-lines and points-circles incidence. Dimensions being implicitly given, it is impossible to produce neither a drawing nor numerical values to solve the system of constraints. For us, a solution must have the form of a simple program of construction, that we call a *plan of construction*. It can later be interpreted with particular numerical data to give one or several numerical and graphical solutions, that is true figures [16].

In order to solve this system *modulo a displacement*, we can arbitrarily fix a point as origin and fix a direction as x-axis, by choosing another point. When constructing a particular solution for the system, adjoining a suffix $_1$ to any identifier will indicate that we are working in this first coordinate system. For instance, point d is fixed at d_1 for origin and then by choosing c at c_1, the x-axis is fixed. In order to better define the elements of construction, we can transform the constraints into incidence relationships, using the *method of the loci* [12,32,42], as in [11,18,28,29,47,48]. Thus, we easily construct points c_1 and e_1 in the following way:

```
fix point $d_1$;
fix direction $d_1c_1$;
construct the oriented line $l1_1$ of direction $d_1c_1$ passing through $d_1$;
construct the point $c_1$ intersection of $l1_1$ and of the circle with centre $d_1$
  and radius $k3$;
construct the oriented line $l2_1$ passing through $d_1$ and forming an oriented
  angle $\alpha 1$ with $l1_1$;
construct the point $e_1$ intersection of $l2_1$ and of the circle with centre $d_1$
  and radius $k4$;
```

Continuing the construction is not as easy: the problem is how to draw the circular arc Γ or to locate point f_1. In fact, we have successfully constructed with the method of loci the auxiliary subfigure (c_1, d_1, e_1), but without possible continuation by the same method. Note that points c_1 and e_1 are not uniquely determined as intersections of a line and a circle. This is well known in geometry where constraints can be translated into polynomial equations of degree greater or equal to 2. This question is discussed in [47], regarding degeneracy. Let us simply indicate that it is treated during the numerical interpretation, by taking into account the orientation and the *proximity* of the solutions with respect to the sketch.

We can try to do the construction from a second coordinate system, determined by fixing point f at f_2 and direction fg at f_2g_2. We construct the auxiliary subfigure (f_2, g_2, e_2) in the following way:

```
fix point $f_2$;
fix direction $f_2g_2$;
construct the oriented line $l3_2$ with direction $f_2g_2$ passing through $f_2$;
construct the point $g_2$ intersection of $l3_2$ and of the circle with centre $f_2$
  and radius $k6$;
construct the oriented line $l4_2$ passing through $f_2$ and forming an oriented
  angle $-\alpha 2$ with $l3_2$;
construct the oriented line $l5_2$ parallel to $l3_2$, of same sense, and distant
  from it by $k5$;
construct $e_2$ intersection of $l4_2$ and $l5_2$;
```

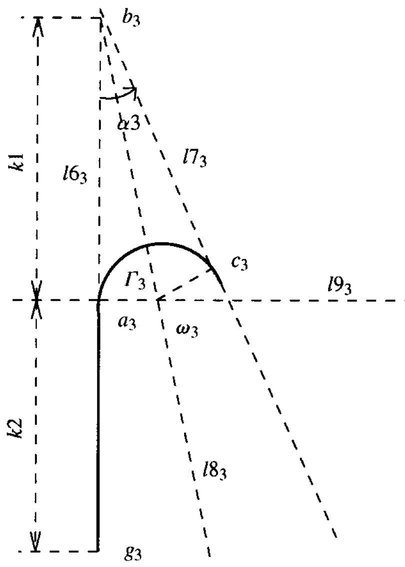

Fig. 2. Auxiliary subfigure with subscript 3.

but without any possibility of continuation. Once again, we try to go on with the construction in a third coordinate system, by fixing point a_3 and the direction a_3b_3. We then easily construct a_3, b_3, g_3 and c_3 (Fig. 2):

fix point a_3;
fix direction a_3b_3;
construct the oriented line $l6_3$ of direction b_3a_3 passing through a_3;
construct the point b_3 intersection of $l6_3$ and of the circle with centre a_3 and radius $k1$;
construct the point g_3 intersection of $l6_3$ and of the circle with centre a_3 and radius $k2$;
construct the oriented line $l7_3$ passing through b_3 and forming an oriented angle $\alpha 3$ with $l6_3$;
construct the oriented line $l8_3$ bisector of $l6_3$ and $l7_3$;
construct the oriented line $l9_3$ passing through a_3 and perpendicular to $l6_3$;
construct the point ω_3 intersection of $l8_3$ and $l9_3$;
construct the point c_3 orthogonal projection of ω_3 on $l7_3$;
construct the circular arc $\Gamma_3 = a_3c_3$ of centre ω_3;

One may continue the construction by using the metric properties of the already solved subfigures. Thus, by auxiliary subfigure (c_1, d_1, e_1), the distance c_3e_3 from c_3 to e_3 is equal to c_1e_1. In the same way, by auxiliary subfigure (f_2, g_2, e_2), we have $g_3e_3 = g_2e_2$. The point e_3 is thus determined as the intersection of the circles with the respective centres c_3 and g_3, and respective radii c_1e_1 and g_2e_2. We can add to our construction plan of the subfigure indexed by 3 the line:

> construct the point e_3 intersection of the circles with the respective centres c_3 and g_3, and respective radii c_1e_1 and g_2e_2;

It would be easy to construct d_3 and f_3 in order to achieve the whole construction. But that is useless because subfigure (c_3, d_3, e_3) has already been constructed in the first coordinate system under the designation (c_1, d_1, e_1). More precisely, each of these two subfigures can be deduced from the other by a displacement. Now, since points c and e were determined in the first coordinate system as c_1 and e_1 and in the third as c_3 and e_3, the unique displacement $\varphi_{1\to 3}$ which transforms c_1 into c_3 and e_1 into e_3 can be computed. Thus, $\varphi_{1\to 3}$ is such that $\varphi_{1\to 3}(c_1, d_1, e_1) = (c_3, d_3, e_3)$. In the same way, we can compute the unique displacement $\varphi_{2\to 3}$ transforming e_2 into e_3 and g_2 into g_3 such that $\varphi_{2\to 3}(e_2, f_2, g_2) = (e_3, f_3, g_3)$. Thus, we end the construction plan by assembling in the subfigure indexed by 3 the two other subfigures by displacement:

> compute $\varphi_{1\to 3}$ which transforms (c_1, e_1) into (c_3, e_3);
> determine $d_3 = \varphi_{1\to 3}(d_1)$;
> compute $\varphi_{2\to 3}$ which transforms (e_2, g_2) into (e_3, g_3);
> determine $f_3 = \varphi_{2\to 3}(f_2)$;

The concatenation of the construction plans of subfigures indexed by 1, 2, and 3, forms a general plan which is brought back in subfigure indexed by 3. This plan is a particular formal solution for the initial problem in the last fixed coordinate system, other solutions being obtained through displacements. Notice that instead of the third coordinate system, we could have chosen any one of the two others and obtained two other particular solutions.

3. Geometric constraint systems

Our approach of the formal construction of figures leads us to distinguish between a syntactical—or formal—level and a semantical—or interpretative—level. So, the notion of figure concerns the semantical level while the specification of a figure is accounted for by the syntactical level. A formal solver acts essentially at the syntactical level: it turns a declarative specification into an imperative one equivalent to it.

Despite this, an accurate and rigorous description of a geometric universe as the one given in [24] is a little bit tedious and makes the basic ideas of our method less natural. For that reason, the syntactical level will be much less detailed than the semantical one.

3.1. Geometric universes and figures

Our formal framework can be seen as a geometric universe whose classical interpretation is the Euclidian plane with a coordinate system $(O, \vec{\imath}, \vec{\jmath})$ fixed once and for all.

The syntactical description of the geometric universe contains classically a *heterogeneous functional signature* Σ consisting in a set Θ of symbols for *atomics types*, or *sorts*, and *type constructors* $\rightarrow$ and $\times$. Each symbol α from Θ is interpreted as a set E_α of objects with type α. With two types α and β from Σ, $\alpha \rightarrow \beta$ is interpreted as the set of the functions from E_α to E_β, $\alpha \times \beta$ as the Cartesian product of E_α with E_β and $\alpha \rightarrow$ as the set of predicates on E_α. Among these sorts, we distinguish a set Θ_g of *geometric sorts*. So, a *geometric type* is either a geometric sort or a Cartesian product of geometric sorts.

Example 3.1. Our geometric universe contains the natural and real numbers corresponding to the sorts *Nat* and *Real*. It has also some geometric objects as points, oriented (straight) lines, circles, directions, lengths, angles and displacements. They correspond respectively to the sorts *Point*, *Line*, *Circle*, *Direction*, *Length*, *Angle* and *Displacement*. The use of oriented lines to define oriented angles leads to more precise statements. A direction is an equivalence class of oriented lines. Later, we will measure a direction by the angle between a representative line and the x-axis. Thus, $\Theta_g = \{Point, Line, Circle, Direction, Length, Angle, Displacement\}$, and $\Theta = \Theta_g \cup \{Nat, Real\}$. For compound types, we have for instance $Point \times Point$, $Point \times Direction$ and $Point \times Point \rightarrow Point$ which we use later.

The signature Σ contains as well functional and predicative symbols. Indeed they are interpreted respectively by functions and predicates on the sets E_α. Since there is no confusion, we note every function or predicate using its corresponding symbol.

Example 3.2. The functional symbols with the following profiles

$$midp : Point \times Point \rightarrow Point$$
$$centre : Circle \rightarrow Point$$
$$distpp : Point \times Point \rightarrow Length$$
$$distpl : Point \times Line \rightarrow Length$$
$$angle : Line \times Line \rightarrow Angle$$
$$dir : Point \times Point \rightarrow Direction$$

correspond to the functions giving respectively the midpoint of two points, the centre of a circle, the distance between two points, the distance between a point and a line, the angle between two oriented lines and the direction defined by to distinct points. The same goes with the following predicative symbols

$$perp : Line \times Line \rightarrow$$
$$tgclp : Circle \times Line \times Point \rightarrow$$
$$\_ = \_ : \alpha \times \alpha \rightarrow$$

expressing the perpendicularity, the tangency and the equality. Note that the equality is polymorphic and with unfixed notation. We add another polymorphic function symbol *transf* : *Displacement* $\times\ \alpha \longrightarrow \alpha$

allowing the application of a displacement to each geometric object of type α.

We suppose that the set E_α of geometric objects of type α is bijectively determined by a system of real coordinates. More precisely, we suppose that the set E_α has a topological structure and that there is an homeomorphism $\gamma_\alpha : I_1 \times \cdots \times I_n \longrightarrow E_\alpha$ where each I_j is an interval of $\mathbb{R}$. We say that n is the *degree of freedom* of α or, as usually, that each object of type α has n degrees of freedom. The coordinate systems γ_α are used during the interpretation of the program construction in order to calculate numerically the solutions and to draw the geometric objects.

Usually, a *figure* is a set of geometric objects. But for some reasons that will be explained later on, we often prefer to consider a *figure* as an n-tuple, denoted by a *vector*, $f = (o_1, \ldots, o_n)$ of geometric objects of respective atomic types $\alpha_1, \ldots, \alpha_n$. That way, the type of f is simply the Cartesian product $\alpha_1 \times \cdots \times \alpha_n$.

Example 3.3. In the case of the Euclidian plane, types *Point*, *Point* $\times$ *Point* and *Point* $\times$ *Direction* have respectively a degree of freedom 2, 4 and 3.

Let $f = (o_1, \ldots, o_n)$ be a figure of type $\alpha_1 \times \cdots \times \alpha_n$. We say that $f' = (o_{i_1}, \ldots, o_{i_k})$ is a *subfigure* of f if it is one of its subvectors. We say that f is *proper* if no component o_i of f can be defined from its other components o_j of f, with $j \neq i$, using the functions of the universe.

3.2. Geometric constraint system

A figure can be specified by a logical formula concerning the geometric universe [15,22,47]. Since we are interested by the effective construction of figures, we prefer to use the constraint system terminology: thus, specified objects can be seen as solutions for such a system given as a statement.

Definition 3.4 (*Constraint system*). A *system of geometric constraints*—or *geometric constraint system*—S is a triple (X, A, C), where X is a set of *unknowns*, denoted by $\mathcal{I}(S)$, A a set of *parameters*, denoted by $\mathcal{A}(S)$, and C a set of *constraints*, denoted by $\mathcal{C}(S)$, of the form

$$C = \{p_1[X, A], \ldots, p_m[X, A]\},$$

where each $p_i[X, A]$ is a predicative term, namely a *constraint*. Unknowns from X and parameters from A are regarded with their sorts that are always geometric, i.e. in Θ_g.

We suppose that unknowns and parameters come from a referential set

$$\mathcal{V} = \bigcup_{\alpha \in \Theta_g} \mathcal{V}_\alpha$$

which is a disjoint union of countable sets $\mathcal{V}_\alpha$ of α-typed variables.

Thus, we always have $X \subset V$ and $A \subset V$. We suppose that any set Y of variables can be totally ordered into a vector also denoted by Y. Conversely, we can consider every vector Y of distinct variables as a set Y of variables. If necessary, we will specify the sort α of a variable x by using the notation $x : \alpha$. We use the notation $p_i[X, A]$ to denote an atomic positive formula which contains variables of X and A. In our framework, we request that every constraint should be algebraically expressed by *polynomial equations* using the coordinate systems of the geometric objects (Section 3.1). We note $var(p_i[X, A])$ and $var(C)$ the sets of all variables respectively in $p_i(X, A)$ and in the set of constraints C. We impose also the two natural conditions

$$X \cap A = \emptyset \quad \text{and} \quad X \cup A = var(C).$$

Finally, in order to simplify, we will often note C for the system (X, A, C).

3.3. *Solutions for a system*

Definition 3.5 (*Solution*). When there are no parameters, a *solution* for a geometric constraint system S is a valuation from the set of the unknowns to the set of the geometric objects of the Euclidian plane, i.e. an application

$$\sigma : \mathcal{I}(S) \to \bigcup_{\alpha \in \Theta_g} E_\alpha$$

which respects types and satisfies every predicate $p_i[\sigma(X), \emptyset]$.

If we fix an ordering on the unknowns, we can consider any solution for a geometric constraint system as a vector of geometric objects, i.e. as a figure. We note $\mathcal{F}(S)$ the set of figures which are solutions for S.

If the set $\mathcal{F}(S)$ is finite and non-empty, we say that S is *well-constrained*. If it is empty, we say that S is *over-constrained*. If it is infinite, we say that S is *under-constrained*. With the assumption that all of the constraints can be translated into polynomial equations, $\mathcal{F}(S)$ is an algebraic manifold which, in a way, defines the *type* of the solutions for S. The degree of freedom of this type is the dimension of the algebraic manifold. Such types could be formally described by a more elaborated typing system such as **T** or **F** [23], but an intuitive vision of such type constructions is enough in the present context.

Example 3.6. Consider the system S_5 defined by $\mathcal{I}(S_5) = \{x_1 : Point, x_2 : Point\}$ and $C(S_5) = \{distpp(x_1, x_2) = 5\}$. It defines an algebraic manifold of degree 3 which represents the type of 5 units long segments.

When the S contains parameters, we consider that the it has only one solution which is the function computing for each point a of the parameter space, the set $\mathcal{F}(S_a)$ of the solutions for the corresponding system S_a with a fixed. If the subset of the parameter space where S_a is well-constrained, i.e. where $\mathcal{F}(S_a)$ is finite, non-empty and contains some part homeomorphic with a paving stone of $\mathbb{R}^n$, we say that the parametric system S is well-constrained. If the system S_a is under-constrained (respectively over-constrained) for almost all of the points a of the parameter space, we say that the parametric system S is *under-constrained* (respectively *over-constrained*).

From our semantical point of view, the well-constrainedness of a system does not involve any relation between the number of unknowns and the number of constraints. Indeed, because of the underlying field of numbers, a well-constrained system can contain fewer constraints than degrees of freedom [15]. On the other hand, such a system can contain more constraints than degrees of freedom because of possible redundancy of constraints. Usually, this problem is avoided by adding some strong hypotheses as for instance the complexification of the underlying field of numbers, the algebraic independence of the constraints and/or the consideration of homogeneous coordinates [31]. For reasons of simplicity, we will try later to stay at the semantical level, and with the real field since we choose to solve constraints in Euclidian plane geometry. Proceeding this way does not suppress all difficulties, thus we denote by the expression *in general* any situation where the restrictive hypothesis concerning the algebraic independence of the constraints is required.

Inclusion of solution sets is as usual translated by a consequence relation.

Definition 3.7 (*Consequence and equivalence*). Let S and S' be two systems with the same unknowns and the same parameters. The set $\mathcal{F}(S)$ of the solutions for S is included in the set $\mathcal{F}(S')$ of the solutions for S' if for each point a of the parameters space, we have $\mathcal{F}(S_a) \subseteq \mathcal{F}(S'_a)$. S' is then called a *consequence* of S, which is denoted by $S \Rightarrow S'$. S and S' are *equivalent* if $S \Rightarrow S'$ and $S' \Rightarrow S$, which is denoted by $S \Leftrightarrow S'$.

It may seem unnecessary to compare two systems $S = (X, A, C)$ and $S' = (X', A, C')$ with different unknowns. However this is indispensable in two cases: first when intermediate unknowns are added and defined by new constraints, second when a subsystem of a given system is considered. In both cases, one of the unknowns sets is a subset of the other. If variables are ordered, we can say as well that one of the unknown vectors is a subvector of the other.

Definition 3.8 (*Extended consequence*). Let X be a subvector of X' and π the projection such that $\pi(X') = X$. Then, $S \Rightarrow S'$ if and only if $\mathcal{F}(S) \subseteq \pi(\mathcal{F}(S'))$, and $S' \Rightarrow S$ if and only if $\pi(\mathcal{F}(S')) \subseteq \mathcal{F}(S)$.

This definition means that any solution for S can be extended into a solution for S' in the first case, and any solution for S' can be projected into a solution for S in the second case.

3.4. Operations on constraint systems

Since we use formal manipulations of constraint systems, we must precisely specify some operations concerning constraints, unknowns and parameters.

Definition 3.9 (*Subsystem, sum, difference, disjunction*). Let $S = (X, A, C)$ and $S' = (X', A', C')$ be two constraint systems. S' is a *subsystem* of S if $C' \subseteq C$, $X' \subseteq X$ and $A' \subseteq A$. The *sum* $S + S'$ is the constraint system $S'' = (X'', A'', C'')$ where $X'' = X \cup X'$, $A'' = A \cup A' - X''$ and $C'' = C \cup C'$. If S' is a subsystem of S, the *difference* $S - S'$ is the system $S'' = (X'', A'', C'')$ where $X'' = var(C'') \cap X$, $A'' = var(C'') \cap A$ and $C'' = C - C'$.

It is important to note that unknowns of $S + S'$ which also appear in $\mathcal{A}(S)$ or $\mathcal{A}(S')$ are removed from A''. This fact corresponds somehow to parameter passing.

The *disjunction* of two systems S and S' containing the same unknowns and the same parameters is noted $S \otimes S'$. It is not a constraint system in the sense already defined, but the disjunction of, on the one hand, the conjunction of the constraints of S and, on the other hand, the conjunction of the constraints of S'. A solution for $S \otimes S'$ is a solution for S or a solution for S'. Therefore, we have $\mathcal{F}(S \otimes S') = \mathcal{F}(S) \cup \mathcal{F}(S')$. More generally, if $S_1, \ldots, S_p$ are p constraint systems containing the same unknowns and parameters, we note $\bigotimes_{i=1}^{p} S_i$ the disjunction of the p systems. Thus, we have

$$\mathcal{F}\left(\bigotimes_{i=1}^{p} S_i\right) = \bigcup_{i=1}^{p} \mathcal{F}(S_i).$$

3.5. *Solving constraint systems*

The aim of geometric construction is mainly to solve well-constrained geometric constraint systems. When a system does not contain parameters, it can be processed by producing some numerical solutions. For example, this is the way the classical Newton–Raphson method [35,39,43] or the Sunde method [51,54,55] work. To yield parametric figures solutions for a parametric system, a formal method is needed: no numerical solutions can be shown, and we must produce a construction process for the figures. This can be done by transforming a constraint system into a triangular system, that is a solved form in the following sense.

Definition 3.10 (*Triangular form, solved form*). A parametric constraint system S is *triangular* if the constraints and the unknowns can be ordered so that, for every i, $p_i[X, A]$ contains only unknowns of $\{x_1, \ldots, x_i\}$. Such a triangular form is said to be *solved* when all the constraints $p_i[X, A]$ are in the form

$$x_i = f_i[x_1, \ldots, x_{i-1}, A],$$

where $f_i[x_1, \ldots, x_{i-1}, A]$ denotes a functional term whose unknowns are in $\{x_1, \ldots, x_{i-1}\}$ and the parameters in A.

A solved triangular system has at most one solution. In fact, some functional symbols in $f_i[x_1, \ldots, x_{i-1}, A]$ may correspond to partial functions not defined for some values of the parameters space. A solved triangular system can be viewed in an operational way as a *construction plan* whose interpretation is the parametric figure solution for the system.

Definition 3.11 (*Solvable system*). We say that a parametric system S is *solvable* if there are $m \geqslant 1$ solved triangular systems $T_1, \dots, T_m$ with $\mathcal{A}(T_i) = \mathcal{A}(S)$, $\mathcal{I}(S) \subseteq \mathcal{I}(T_i)$, for every i, and such that

(i) $T_1 \Rightarrow S, \ \dots, \ T_m \Rightarrow S$,

(ii) $S \Rightarrow \bigotimes_{i=1}^{m} T_i$.

Each condition $\mathcal{I}(S) \subseteq \mathcal{I}(T_i)$ points out that some new, or intermediate, unknowns can appear in a triangular system. The condition (i) expresses the *correctness* of the construction and the condition (ii) its *completeness*. In other words, both conditions can be translated into equality:

$$\mathcal{F}(S) = \bigcup_{i=1}^{m} \mathcal{F}(T_i).$$

This notion of solvability is both syntactical and semantical. It is syntactical as it expresses that all solutions can be yielded formally using the geometric universe syntax. It is semantical as it requires the model we considered above, namely the Euclidian plane.

By definition, a solvable system is well-constrained. However the converse is false: this comes from the impossibility to axiomatize the real field in a finite way, which can be proved by Löwenheim–Skolem's theorem [49].

Example 3.12 (*Carver and Lesser* [12]). The famous problem of the circle quadrature with ruler and compass is insolvable. The zeroes of polynomials with degree greater than or equal to 5 cannot be written as terms built with elementary operations and radicals of their coefficients. For this reason, computer-aided designers often content themselves with approximate values.

Moreover, the complete and correct decomposition of a solvable system into a disjunction of solved systems is seldom carried out in practice. Thus, geometric reasoning by necessary conditions leads to the construction of figures that are not solutions for the initial system. This incorrectness must be rectified by a so-called discussion phase or a checking phase during the numerical interpretation. The Newton–Raphson method [39,43] often used in CAD gives a good example of an incomplete and incorrect method because only one solution can be found and this method can diverge even though there are solutions.

4. Solving modulo the displacements group

The notion of *isometry*, and more precisely of even isometry, is one of the significant notion our method is based on. Usually, an even isometry is called a displacement. That is an affine application that preserves distance and orientation of the plane. As usual, we extend this notion to all the considered types in our geometric universe.

4.1. Action of the displacements group over the geometric universe

Following a rigorous point of view, the action of a *displacement* φ over a geometric object should be denoted by *transf*(φ, o) according to the signature given above in Example 3.2. To simplify and to conform to the common habits, we shorten this notation into $\varphi(o)$.

The set of the plane displacements forms a group under composition of functions. This allows us to define for each geometric type α an equivalence relation $\equiv_\alpha$ over the set E_α in the following way: for each pair (f, f') of $E_\alpha \times E_\alpha$

$$f \equiv_\alpha f' \text{ if and only if there is a displacement } \varphi \text{ such that } \varphi(f) = f'.$$

Definition 4.1 (*Orbit, modisp degree of freedom*). The equivalence class modulo $\equiv_\alpha$, or merely *modulo the displacements*, abbreviated into *modisp*, of a geometric object of type α is named its *orbit*. The quotient set $E_\alpha/\equiv_\alpha$, i.e. the set of the orbits, is the set of the objects of type α *modisp*. For each type α, the set $E_\alpha/\equiv_\alpha$ can be fitted with the quotient topologic structure whose dimension is called the *modisp* degree of freedom of type α.

Considering the quotient set, two opposite cases may occur. First, if there is only one orbit, i.e. $E_\alpha/\equiv_\alpha$ is reduced to a single element, then the *modisp* degree of freedom is null. This means that the objects with this type are completely determined by their position. This happens with the points and the lines. Second, if each orbit is reduced to a single element, i.e. $E_\alpha/\equiv_\alpha$ is equal to E_α, then the *modisp* degree of freedom is equal to the degree of freedom. This means that the objects of this type are invariant under displacements. This happens with the lengths and the angles. The corresponding types are described as *metric types*. In the other cases, the *modisp* degree of freedom is not equal to zero and smaller than the degree of freedom.

Example 4.2. Circles have a degree of freedom equal to 3 (one for their radius and two for their centre) and a *modisp* degree of freedom equal to 1: each orbit contains the circles of the plane with the same radius and the set of all the orbits is homeomorphic to $\mathbb{R}_+$. Ellipses have a degree of freedom equal to 5 (one for each radius, one for the direction of their great axis, and two for their centre) and a *modisp* degree of freedom equal to 2: in this case, each orbit contains the ellipses with the same short and large radii. Triangles are figures with a degree of freedom equal to 6 (two for each vertex). Their *modisp* degree of freedom is equal to 3: each orbit contains triangles with corresponding edges of same length.

As it can be seen, the group of displacements acts also over figures and parametric figures. By passing to the quotient, the orbit of such a figure yields a so-called *modisp* figure. In the case of a parametric figure, a *modisp* parametric figure is a function relating a point—i.e. numerical values—of the parameter space to a *modisp* figure. To simplify, when no problem occurs, we will use the term figure for both "plain" and parametric figures. Likewise, we must consider the effects of the displacements on the functions of the geometric universe.

Definition 4.3 (*Stability under displacements*). A function g with n arguments is *stable under displacements* if, for each displacement φ, the equality $\varphi(g(z_1,\dots,z_n)) = g(\varphi(z_1),\dots,\varphi(z_n))$ holds for all $z_1,\dots,z_n$.

Note that this implies $g(z_1,\dots,z_n) = g(\varphi(z_1),\dots,\varphi(z_n))$ for the functions with values in a metric type. Such functions are called *metric functions*. They play an important role in CAD because they are linked with the systems of dimensions and allow the expression of some properties of already solved subfigures.

Example 4.4. The functions *midp*, *distpp* and *distpl* are stable under displacements. The two latter are metric functions because they return a length. The functions referring to the absolute coordinate system $(O,\vec{\imath},\vec{\jmath})$ such those yielding the x-coordinate or the y-coordinate are not stable under displacements.

We take the liberty to lightly misuse the notations by making the parameters of a functional term visible. So, a parametric function $g(z_1,\dots,z_n,A)$ is stable under displacements if, for each displacement φ, $\varphi(g(z_1,\dots,z_n,A)) = g(\varphi(z_1),\dots,\varphi(z_n),A)$. Finally, we must consider the effects of the displacements on the predicates and constraints of the geometric universe.

Definition 4.5 (*Invariance under displacements*). A constraint system $S = \{p_1[X,A],\dots,p_m[X,A]\}$ is *invariant under displacements* if it is equivalent to the system $S_\varphi = \{p_1[\varphi(X),A],\dots,p_m[\varphi(X),A]\}$ for each displacement φ. In particular, a single constraint of the form $p[X,A]$ is *invariant under displacements* if, for each displacement φ, we have $\{p[\varphi(X),A]\} \Leftrightarrow \{p[X,A]\}$.

Then, a system is invariant under displacements in particular if all of its constraints are invariant under displacements as it is the case in CAD.

Example 4.6. Let parameter k be a length and unknowns x_1 and x_2 be points. The equality $distpp(x_1,x_2) = k$ is invariant under displacements. More generally, the constraints defined by the equalities of the form $g(x_1,\dots,x_n,A) = h(y_1,\dots,y_m,A')$ where g and h are metric functions parameterized respectively by A and A', and where x_i and y_i are unknowns, are invariant under displacements.

The invariance under displacements of a constraint system is rendered by the form of the solutions as it can be seen in the next proposition.

Proposition 4.7 (Passing to the quotient). *A system S is invariant under displacements if and only if $\mathcal{F}(S)$ is a union of orbits. So, we say that the solutions for S are* modisp *figures.*

Proof. Let S be an invariant under displacements system (from now on, we simply will say "an invariant system"). If S has no solution, then this proposition is trivially true. Otherwise, let f be a solution for S and φ any displacement. Then, $\varphi(f)$ is a solution for $S_{\varphi^{-1}}$ which is equivalent to S. Thus, by Definition 3.7, $\varphi(f)$ is a solution for S too.

Conversely, let S be a constraint system such that $\mathcal{F}(S)$ is a union of orbits and φ a displacement. Since φ is a bijection, the system S_φ is such that $\varphi^{-1}(\mathcal{F}(S)) = \mathcal{F}(S_\varphi)$. Since $\mathcal{F}(S)$ is a union of orbits, we have $\varphi^{-1}(\mathcal{F}(S)) = \mathcal{F}(S)$, therefore $\mathcal{F}(S) = \mathcal{F}(S_\varphi)$ and so S is invariant under displacements. □

4.2. Positioning and reference

A *modisp* figure $f = (o_1, \ldots, o_p)$ is positioned by the choice of one of its representatives. In the case of the Euclidian plane, this is usually done setting a point and a direction. A coordinate system can be defined from such a pair, so we call it a *reference frame*—in short a *reference*—of the Euclidian plane.

Definition 4.8 (*Type Reference*). The type *Reference* is defined as the Cartesian product *Point* × *Direction*. We say that a figure *contains* a reference if an object of type *Reference* can be determined only from its components $o_1, \ldots, o_p$ using functions stable under displacements. We say that a family of figures with type α contains a reference if each figure of the family contains a reference and the way to determine this reference is the same for all the figures of the family. We say that a constraint system S contains a reference if $\mathcal{F}(S)$ contains a reference. In this case, note that the determination of this reference included in each figure of $\mathcal{F}(S)$ involves the same subvector of unknowns.

Example 4.9. A figure with two distinct points a and b contains, among others, the references $(a, dir(a,b))$ and $(midp(a,b), dir(a,b))$. An ellipse contains, among others, the reference constituted by its centre and the direction of its major axis.

The classical problem of the construction of a triangle $x_1x_2x_3$ from the length of its three edges leads to the constraint system

$$S = \{distpp(x_1,x_2) = k1, distpp(x_2,x_3) = k2, distpp(x_3,x_1) = k3\},$$

where $k1$, $k2$ and $k3$ are positive parameters. This system contains, among others, the reference that we denote by $(x_1, dir(x_1,x_2))$ indicating that all the solutions for S contain the reference defined substituting x_1 and x_2 by their value.

To point out the importance of the stability under displacements in Definition 4.8, we show that a figure $f = (d)$ containing only line d, does not contain a reference. Indeed, it is not possible to define a reference point only from a line, because there is no function g from the set of the lines to the set of the points stable under displacements. This fact can be proven by reducing it to the absurd. Let g be such a function, d a line and φ a translation whose vector is a direction vector of d. Then, on the one hand, we have $g(\varphi(d)) = g(d)$ because $\varphi(d) = d$, and on the other hand, we have $\varphi(g(d)) \neq g(d)$ because a translation distinct from identity function has no fixpoint. This refutes the hypothesis that the function g is stable under displacements.

Generally, the family of all the figures with the same type α does not contain stricto sensu a reference because of the so-called degenerate cases. This cannot be produced if only proper figures of type α are considered (see Section 3.1).

Checking that a system contains a reference is a matter for the semantical level and therefore hard to do. So we content ourself with syntactical criteria. The first one comes from the fact that a system S contains a reference if the type of vector $\mathcal{I}(S)$ contains the type *Reference*. As we have seen above, this is not a sufficient condition, but it is easy to check. In practice, it is enough to consider a single constraint whose unknowns contain a reference, because the nature of the constraint assures that the set of the subfigures solution for this constraint contains a reference. Typically, the constraint $distpp(x_1, x_2) = k$ ensures that the two points x_1 and x_2 are different and contain a reference.

To solve an invariant system, we make sure that it contains a reference and then, we impose a value to this reference. We say that we *fix a reference*. The constraints by which a reference is fixed are named *reference constraints*.

Example 4.10. We fix a reference for a figure containing two distinct points a and b imposing for instance that $a = O$ and $dir(a, b) = \pi/2$. We can fix another one by the constraints $midp(a, b) = O$ and $dir(a, b) = \pi/3$.

In order to simplify our notations, for an invariant system containing a reference determined by the unknowns $x_{i_1}, \ldots, x_{i_k}$, we condense the set of the constraints fixing a reference from these unknowns in the single generic constraint $ref(x_{i_1}, \ldots, x_{i_k})$.

Example 4.11. The two unknowns x_i and x_j of type *Point* linked by the constraint $distpp(x_i, x_j) = k$ determine a reference. We fix this reference by the constraints $x_i = O$ and $dir(x_i, x_j) = 0$ that we sum up $ref(x_i, x_j)$.

The notions of reference and displacement are closely linked. Thus, the degrees of freedom of the types *Reference* and *Displacement* are equal, and we have the following well-known properties:

- in an Euclidian space of dimension d, given two references, there is only one displacement translating one into the other; so the *modisp* degree of freedom of the type *Reference* is null;
- let be two figures with the same type containing no reference, typically lines as in Example 4.9, then there is an infinity of displacements translating one into the other. In such a case, we say that these figures *contain less than a reference*;
- if a figure f contains a reference, then for any pair of distinct displacements (φ_1, φ_2), we have $\varphi_1(f) \neq \varphi_2(f)$ and, for any infinite set of displacements Φ, the set $\{\varphi(f) \mid \varphi \in \Phi\}$ is infinite too.

4.3. Positioned figures and systems

With the notion of reference constraint, we can moreover distinguish one figure within a *modisp* orbit. More precisely, we have the following proposition refering to the absolute coordinate system $(O, \vec{\imath}, \vec{\jmath})$ given above.

Proposition 4.12 (Positioning by a reference). *Let α be a type of figures containing the type* Reference *corresponding to the reference constraint $ref(x_{i_1}, \ldots, x_{i_k})$. In each* modisp *orbit Ω of the figures of type α, there is one and only one figure satisfying the reference constraint $ref(x_{i_1}, \ldots, x_{i_k})$.*

Proof. We show this proposition in the case where $\alpha = \mathit{Point} \times \mathit{Point} \times \alpha_1 \times \cdots \times \alpha_p$ and $ref(x_1, x_2)$ defined as in Example 4.11. This does not restrict much the generality and permits to trim down the proof. Let Ω be a *modisp* orbit of E_α and $f' = (a', b', o'_1, \ldots, o'_p) \in \Omega$ a figure. There is only one displacement φ such that $\varphi(a') = O$ and $\varphi(dir(a', b')) = 0$: it is the composition of the translation of vector $\overrightarrow{a'O}$ with the rotation of centre O and angle of measure $-dir(a', b')$. The figure $\varphi(f')$ is one of the expected figures. Let us show that there is only one figure like this one. Let $f = (a, b, o_1, \ldots, o_p)$ and $g = (a', b', o'_1, \ldots, o'_p)$ be two figures from the same orbit and meeting the conditions of the proposition. On the one hand, there is a displacement φ such that $\varphi(f) = g$ and, on the other hand, we have $a = a' = O$ and $dir(a, b) = dir(a', b') = 0$. So, $\varphi(O) = O$ and $\varphi(\vec{\imath}) = \vec{\imath}$ (hence $\varphi(\vec{\jmath}) = \vec{\jmath}$). Then φ is the identity function and $f = g$. $\square$

This property is particularly useful in the case of an invariant system whose solutions are orbits.

Example 4.13. Resume from Example 4.9 the problem of the construction of a triangle $x_1x_2x_3$ given the lengths of its three edges. The corresponding system S is invariant under displacements. A particular solution for S is reached imposing, among other constraints, that the centre of the circumcircle ω is set onto O and the half-line with origin x_1 passing by the midpoint of (x_2, x_3) is equal to the x-axis. With our geometric universe, these unusual constraints have the form

$$\{interll(med(x_1, x_2), med(x_1, x_3)) = O, dir(x_1, midp(x_2, x_3)) = 0\}.$$

More commonly, we can fix a reference like in Example 4.9 setting a onto O and half-line x_1x_2 onto the x-axis.

Thus, to obtain a particular figure solution for an invariant system S, it is necessary and sufficient to add to S some reference constraints positioning one figure from each orbit solution for S. We say that we position such a system as the following definition and proposition point out.

Definition 4.14 (*Positioned system*). Let r be a name for the reference constraint $ref(x_{i_1}, \ldots, x_{i_k})$. The system S *positioned* in r is the system noted $S + r$ and defined by

$$S + r = \{ref(x_{i_1}, \ldots, x_{i_k}), p_1[X, A], \ldots, p_m[X, A]\}.$$

The solutions for $S + r$ are particular solutions for S which permit to find, by action of the displacements, all the solutions for S. In particular, we say that a system S is *well-constrained modisp* if $\mathcal{F}(S)$ is a *finite* non-empty union of orbits. This notion is directly linked to the notion of well-constrained system when passing to the quotient. More precisely, we have:

Proposition 4.15. (Characterization of the well-constrained modisp systems, change of reference) *A constraint system is* well-constrained modisp *if and only if it is invariant under displacements and, for each reference constraint r, the positioned system $S + r$ is well-constrained. Moreover, if r and r' are two reference constraints, then for each solution f for $S + r$, there is a displacement φ translating it into a solution $\varphi(f)$ for $S + r'$.*

Proof. If S is a well-constrained *modisp* system, then according to the definition above, $\mathcal{F}(S)$ is a finite non-empty union of orbits. So, according to Proposition 4.7 it is invariant under displacements. Let f be a figure solution for $S + r$, f is a fortiori a solution for S and for each displacement φ, and it is the same for $\varphi(f)$. Conversely, if Ω is an orbit solution for S, then there is only one figure from Ω satisfying r because of Proposition 4.12. It can be deduced that there is a bijection between the set of solutions for $S + r$ and the set of the orbits—or *modisp* figures—solutions for S. This last set is finite and non-empty, therefore $S + r$ is well-constrained.

Conversely, if S is invariant under displacements, then $\mathcal{F}(S)$ is a union of orbits. Thus, if $S + r$ is well-constrained, $\mathcal{F}(S)$ is finite and non-empty and the bijection described above—which still works—implies that this union is finite and non-empty too.

Now, let f be a solution for $S + r$. The set $\{\varphi(f) \mid \varphi \text{ is a displacement}\}$ is an orbit solution for S containing only one figure f' satisfying the reference constraint r'. So, f' is a solution for $S + r'$, and to each solution f for $S + r$ correspond a displacement φ and a figure $f' = \varphi(f)$ solution for $S + r'$. □

Intuitively, this proposition indicates that reference constraints can be added to an invariant system and *relaxed* for a system positioned by another reference constraint. Thus, in our method, a system is positioned to be partially solved, then the reference constraint is released for another one which will permit to solve another part of the system, and so on until all the partial solutions can be assembled into a single one. The notion of partial construction of a subfigure plays an important role which becomes clearer in the next subsection.

Before that, it is advisable to define how to syntactically treat the unknowns in the case where several references are considered in the same construction process. So, for each reference constraint used, we give a new name which is used to *qualify* systematically the unknowns, as discussed in the following definition.

Definition 4.16 (*Located system*). Let S be an invariant system containing a reference and the associated reference constraint $r = \mathit{ref}(x_{i_1}, \ldots, x_{i_k})$. We note $S.r$ the system S *located in* r and corresponding to the system $S + r$ with its unknowns qualified in a pointed notation. More precisely, $S.r$ is defined by $\mathcal{I}(S.r) = \{x_1.r, \ldots, x_n.r\}$, $\mathcal{A}(S.r) = \mathcal{A}(S)$ and $\mathcal{C}(S.r) = \mathcal{C}(S + r).r$, where the notation $\mathcal{C}(S + r).r$ stands for the set of the constraints of $S + r$ where each unknown x_i is replaced by the qualified unknown $x_i.r$.

We also use the notation $\mathcal{I}'$ for the set of the de-qualified unknowns of a system, i.e. $\mathcal{I}'(S.r) = \mathcal{I}(S)$. We will see later on that a qualified unknown $x.r$ cannot be qualified again into $x.r.r'$. Note that the notion of well-constrainedness *modisp* exactly recovers the specification of rigid bodies, excluding non-connected or articulated bodies.

4.4. Partial solving

The notions of subfigure and subsystem introduced above pass to the quotient as well. So, we can talk about *modisp* subfigure and subsystem invariant under displacements. The following proposition results from this.

Proposition 4.17 (Well-constrained subsystem). *Let S be a well-constrained modisp system and S' be a well-constrained modisp subsystem of S. Then, to each modisp figure f solution for S corresponds a modisp subfigure f' which is solution for S'. Conversely, any solution f' for S' can be extended into a solution f for S.*

As we have seen, to solve a construction problem, our method works with representatives of *modisp* figures obtained by fixing local references. So, to pass from a positioned subfigure to another one, the method determines the displacement moving the former into the latter. This process is justified by the following proposition.

Proposition 4.18 (Replacing a subfigure). *Let S be a well-constrained modisp system located into $S.r$ and S' be a well-constrained modisp subsystem of S located into $S'.r'$. To each figure f solution for $S.r$ corresponds a figure f' solution for $S'.r'$ and a single displacement φ such that $\varphi(f')$ is a subfigure of f.*

This proposition, whose proof results immediately from the definitions and propositions above, points out that it is possible to consider at first only a part S' of the constraints of a system S to begin a construction with a local reference defined by a reference constraint r'. But, in general, the remaining constraints in $S - S'$ do not constitute a well-constrained *modisp* system: some metric informations coming from the solved part $S'.r'$ must be considered. We call such informations the border of the system $S'.r'$ within the system S. This generalizes the notion of *virtual bonds* used by Owen [41]. More precisely, we have the following definition.

Definition 4.19 (*Border*). Let S be an invariant system and S' be a subsystem of S located into $S'.r'$. The *border* of $S'.r'$ within S is the system $B = \mathit{border}(S'.r', S)$ defined by $\mathcal{I}(B) = \mathcal{I}(S - S') \cap \mathcal{I}(S')$, $\mathcal{A}(B) = \mathcal{I}(B).r'$ and

$$\mathcal{C}(B) = \{g(x_1, \ldots, x_m) = g(x_1.r', \ldots, x_m.r') \mid \\ g \text{ any metric function and } x_1, \ldots, x_m \in \mathcal{I}(B)\}.$$

Note that the unknowns provided by $S'.r'$ become parameters of B. These parameters are named *border parameters* and are meant to be substituted by solutions to $S'.r'$. This substitution can be numeric or formal if $S'.r'$ contains parameters.

Proposition 4.20 (Property of the border). *System $B = border(S'.r', S)$ is invariant under displacements. Moreover, if B contains a reference, the reference constraint associated with being named r, then $B + r$ is solvable.*

Proof. It is clear that B is invariant under displacements because it only contains equalities between metric functional terms as constraints. Suppose that B contains a reference associated with the reference constraint r. As in the proof of Proposition 4.12, we simplify our purpose without loss of generality by supposing that the reference r is $ref(x_1, x_2)$ where x_1 and x_2 are two unknown points. In order to show that $B + r$ is solvable, note first that $\mathcal{I}(B).r'$ is an obvious solution for B and thus, $B + r$ is not over-constrained. Moreover, there is only one solution for $x_1 = O$ and at most two solutions for x_2 in the intersection of the x-axis and the circle of centre O and radius $distpp(x_1.r', x_2.r')$. Examine now the other unknowns of $\mathcal{I}(B)$ according to their sort. Let x be such an unknown ($x \neq x_1$ and $x \neq x_2$):

- if x is of the sort *Point*, then $C(B)$ contains, among others, the constraints $distpp(x, x_1) = distpp(x.r', x_1.r')$ and $distpp(x, x_2) = distpp(x.r', x_2.r')$; therefore x is in the intersection of two known distinct circles;
- if x is of a sort *Line*, then $C(B)$ contains the constraints
$$distpl(x_1, x) = distpl(x_1.r', x.r') \quad \text{and}$$
$$angll(x, stlig(x_1, x_2)) = angll(x.r', stlig(x_1.r', x_2.r')),$$
where *angll* measures the oriented angle between oriented lines, and *stlig* gives the oriented line passing through two points; then x is a tangent line of circle of centre x_1 and radius $distpl(x_1.r', x.r')$ making a given angle with oriented line $stlig(x_1, x_2)$ (whose construction is not complicated);
- if x is of the sort *Circle*, then $C(B)$ contains the constraints
$$radius(x) = radius(x.r'),$$
$$distpp(centre(x), x_1) = distpp(centre(x.r'), x_1.r') \quad \text{and}$$
$$distpp(centre(x), x_2) = distpp(centre(x.r'), x_2.r');$$
hence it is easy to construct x;
- etc., for each sort.

In each case, we can build a finite number of solutions for x, so $B + r$ is solvable. □

We can now examine what happens with the remaining constraints. If S and its subsystem S' are well-constrained *modisp*, this means that S and S' have at least unknowns in common that define a reference. Let us clarify this phenomenon which is purely syntactic.

Definition 4.21 (*Common reference*). Two constraint systems S' and S'' have a *common reference* if S' and S'' contain a reference determined by the same vector of unknowns included in $\mathcal{I}(S') \cap \mathcal{I}(S'')$. We say that they have *exactly* a common reference if they have a common reference and $\mathcal{I}(S') \cap \mathcal{I}(S'')$ is of the type *Reference*. They have *more than* a common reference if they have a common reference but not exactly.

Hence, we have the following main proposition which makes clear the role of the border.

Proposition 4.22 (Use of the border). *Let S be a well-constrained modisp system and S' be a subsystem of S containing a reference and such that $S'' = S - S'$ contains as well a reference. If S' is well-constrained modisp, then the following assertions hold:*

(1) *S' and S'' have a common reference.*

(2) *If S' and S'' have exactly a common reference, then S'' is well-constrained modisp, else S'' is* in general *under-constrained modisp.*

(3) *If S' is located into $S'.r'$, then $S'' + border(S'.r', S)$ is well-constrained modisp.*

Proof. Note first that S'' cannot be over-constrained because any solution for S gives a solution for S''.

Examine assertion (1) and suppose that the common unknowns between S' and S'', say $x_1, \ldots, x_k$, do not form a reference. Let r' and r'' be some reference constraints added respectively to S' and S''. Then, let $f' = (a_1, \ldots, a_k, a_{k+1}, \ldots, a_m)$ be a solution for $S'.r'$ where $a_1, \ldots, a_k$ correspond respectively to $x_1, \ldots, x_k$ and $f'' = (b_1, \ldots, b_k, b_{k+1}, \ldots, b_p)$ a solution for $S''.r''$ where $b_1, \ldots, b_k$ correspond to $x_1, \ldots, x_k$. According to the second and third properties in Section 4.2, since the a_i and b_i do not contain a reference, there is an infinity of displacements φ such that $\varphi(b_1) = a_1, \ldots, \varphi(b_k) = a_k$. Moreover, since S'' contains a reference, there is an infinity of figures f_φ of the form $(a_1, \ldots, a_k, a_{k+1}, \ldots, a_m, \varphi(b_{k+1}), \ldots, \varphi(b_p))$: they are all different and solution for $S.r'$. So $\mathcal{F}(S.r')$ is infinite, refuting the assumption that S is well-constrained *modisp*.

Next, consider assertion (2) in the case where S' and S'' have exactly a common reference defined by $x_1, \ldots, x_k$ and suppose that S'' is under-constrained *modisp*. According to Proposition 4.15, for any reference constraint r, there is an infinity of solutions for $S''.r$. Then, call r the constraint fixing the unknowns $x_1, \ldots, x_k$ and let $f' = (a_1, \ldots, a_k, a_{k+1}, \ldots, a_m)$ be a solution for $S'.r$ where $a_1, \ldots, a_k$ correspond respectively to $x_1, \ldots, x_k$. So, each solution for $S''.r$ is of the form $(a_1, \ldots, a_k, b_{k+1}, \ldots, b_p)$ where there is an infinity of solutions for $b_{k+1}, \ldots, b_p$. All the figures of the form $(a_1, \ldots, a_k, a_{k+1}, \ldots, b_{k+1}, \ldots, b_p)$ are solution for $S.r$. Then, $\mathcal{F}(S.r)$ is infinite and this refutes our assumption that S is well-constrained *modisp*. Therefore S'' is well-constrained *modisp*.

Now, we show (2) in the case where S' and S'' have more than a common reference. In order to simplify the proof, we suppose that a common reference for S' and S'' is made up of two common points, say x_1 and x_2. We note r the common reference constraint $ref(x_1, x_2)$. If we suppose that S'' is well-constrained *modisp*, then $S'.r$ and $S''.r$ yield both a finite non-empty set of solutions such that $x_1 = O$ and $dir(x_1, x_2) = 0$. So, we have a finite non-zero number of solutions for x_2 from $S'.r$, say $\{a_2^1, \ldots, a_2^k\}$ and other solutions from $S''.r$, say $\{b_2^1, \ldots, b_2^p\}$. In general, these two sets are disjoint. In this case $S.r$ has no solution and there is a contradiction. The case where the two sets have a non-empty intersection implies that S' and S'' are not algebraically independent: with our hypothesis, this case is possible since we do not impose any condition over the form of S' and S''. But generally one tries to avoid this situation in CAD. So, we can say that S'' is *in general* under-constrained *modisp*.

Finally, we prove assertion (3). In order to show that $S'' + border(S'.r', S)$ is well-constrained *modisp*, note first that $S \Rightarrow S'' + border(S'.r', S)$, therefore $S'' + border(S'.r', S)$ is not over-constrained. We can show as in the proof of assertion (2) that each solution for $S'' + border(S'.r', S)$ can be extended into a solution for S such that $S'' + border(S'.r', S)$ is not under-constrained *modisp*. □

4.5. Assembling

In the previous subsection, we have shown that an invariant system can be partially solved using the border of the already solved subsystems. The proof of Proposition 4.22 points out how a global solution can be produced from the partial located solutions. We say that we assemble the subfigures, or the subsystems defining them, what is the same, as we will see below.

Definition 4.23 (*Assembling*). Let S' and S'' be two subsystems of the same system S respectively located into $S'.r'$ and $S''.r''$. Let φ be an unknown of type *Displacement* not in $Var(S)$. The *assembling of S' with S'' by φ* is a system $S_1 = asb(S'.r', S''.r'', \varphi)$ defined by

(i) $\mathcal{I}(S_1) = \mathcal{I}(S'.r') \cup \{y.r' \mid y \in \mathcal{I}(S'') - \mathcal{I}(S')\} \cup \{\varphi\}$;

(ii) $\mathcal{A}(S_1) = \mathcal{A}(S)$;

(iii) $\mathcal{C}(S_1) = \mathcal{C}(S'.r') \cup \mathcal{C}(S''.r'') \cup eqdisp(S'.r', S''.r'', \varphi)$

with $eqdisp(S'.r', S''.r'', \varphi) = \{y.r' = transf(\varphi, y.r'') \mid y \in \mathcal{I}(S'')\}$.

In this definition, S' and S'' play dissymmetric roles. However, it is easy to see that $asb(S'.r', S''.r'', \varphi)$ and $asb(S''.r'', S'.r', \varphi)$ are equivalent *modisp*. The assembling works therefore in a sense or in the other.

Note that this operation is useful only if the two systems have a common reference. According to Proposition 4.22, if they have less than a common reference, the assembling has an infinity of solutions, and if they have more than a common reference, a certain kind of border compatibility is required to get a solution for φ. In this case, we say that the two systems are *assemblable*.

The assembling operation is compatible with the solving of the invariant systems. The correctness of the assembling operation comes from the following theorem which indicates that the assembling operation yields no false solution.

Theorem 4.24 (Correctness). *Let S_1, S_2, S_1' and S_2' be four invariant constraint systems so that $\mathcal{I}(S_1) = \mathcal{I}(S_2)$ and $\mathcal{I}(S_1') = \mathcal{I}(S_2')$, respectively located into $S_1.r$, $S_2.r$, $S_1'.r'$ and $S_2'.r'$. If $S_1.r \Rightarrow S_2.r$ and $S_1'.r' \Rightarrow S_2'.r'$, then $asb(S_1.r, S_1'.r', \varphi) \Rightarrow asb(S_2.r, S_2'.r', \varphi)$.*

Proof. If $asb(S_1.r, S_1'.r', \varphi)$ has no solution, then there is nothing to say. Otherwise, noting that $eqdisp(S_1.r, S_1'.r', \varphi) = eqdisp(S_2.r, S_2'.r', \varphi)$, a solution F for $asb(S_1.r, S_1'.r', \varphi)$ can be broken into a solution f for $S_1.r$, a solution f' for $S_1'.r'$ and a displacement φ_0 permitting to solve $eqdisp(S_1.r, S_1'.r', \varphi)$. The figure f is therefore solution to $S_2.r$, the figure f' solution for $S_2'.r'$ and the assembling of these two figures by φ_0, giving F, is clearly a solution for $asb(S_2.r, S_2'.r', \varphi)$. $\square$

We can deduce immediately the following corollary.

Corollary 4.25. *If $S_1.r \Leftrightarrow S_2.r$ and $S_1'.r' \Leftrightarrow S_2'.r'$ then*

$$asb(S_1.r, S_1'.r', \varphi) \quad \Leftrightarrow \quad asb(S_2.r, S_2'.r', \varphi).$$

The completeness of the assembling operation is then given by the following theorem which expresses that all the solutions for the initial system S can be found by assembling the solutions for two subsystems whose union is S and whose solutions are found using the border of one of them.

Theorem 4.26 (Completeness). *Let S be a well-constrained modisp system and S' be a well-constrained modisp subsystem of S. Suppose that S' is located into $S'.r'$, which is solvable giving the solved forms $T_1', \ldots, T_{k'}'$, and that the system $S'' = S - S' + border(S'.r', S)$ is located into $S''.r''$, which is solvable too giving the solved forms $T_1'', \ldots, T_{k''}''$. Then:*

(1) *$asb(T_i', T_j'', \varphi_{j\to i})$ is solvable for each i and each j;*

(2) *$S.r' \Leftrightarrow \bigotimes_{j\to i} asb(T_i', T_j'', \varphi_{j\to i})$, where $\varphi_{j\to i}$ are new distinct unknowns neither contained in $\bigcup_i var(T_i')$ nor in $\bigcup_j var(T_j'')$.*

Proof. Assertion (1) is clear: the choice of a common reference between T_i' and T_j''—whose existence is given by Proposition 4.22—permits to compute $\varphi_{j\to i}$ within the system $eqdisp(T_i', T_j'', \varphi_{j\to i})$ and, then, the images of the solutions for T_j''.

For assertion (2), note that $S.r'$ is contained in $asb(S'.r', S''.r'', \varphi)$, thus it can be immediately seen that $S.r' \Leftrightarrow asb(S'.r', S''.r'', \varphi)$. For each i and each j such that $T_i' \Rightarrow S'.r'$ and $T_j'' \Rightarrow S''.r''$, we therefore have $asb(T_i', T_j'', \varphi_{j\to i}) \Rightarrow S.r'$ by the previous theorem. Conversely, each solution f for $S.r'$ gives a solution for $S'.r'$, say f_1, and another one to $S''.r'$ and therefore by displacement a solution, say f_2, to $S''.r''$. So, there are some i and j such that f_1 is a solution for T_i' and f_2 a solution for T_j''. Hence, the figure $(f_2, f_1, \varphi_{j\to i})$, containing f plus some auxiliary objects, is a solution for $asb(T_i', T_j'', \varphi_{j\to i})$. $\square$

We have shown in this section that in order to get all the solutions for a constraint system S, it is possible to solve locally two subsystems S' and S'' using the border of one of them, then to assemble it. But the success of the whole geometrical construction is subject to the correctness and the completeness of the local solving methods.

5. Strategies

The previous theory of constraint systems solving does not make any hypothesis about the way to concretely proceed. In order to make it operational, we must clarify strategies for the choice of subsystems, local methods, activation of methods and assembling.

5.1. Decomposition in subsystems

Two main strategies can be considered for the decomposition of a geometric constraint system into subsystems. The first one proceeds a priori by static analysis of the subsequent constraint graph connectivity to detect common references. That method is followed in [30,41], where notions like pairs of articulation or perfect matching play an essential role. Next, the solving is done locally on each of the subsystems and the partial solutions are assembled to obtain the global solution.

The second strategy proceeds dynamically by simultaneous exploration and solving of the constraint system. More precisely, at each step, a solving method is brought into play on the part of the system yet unsolved. The capabilities of the method allow the determination and solving of a subsystem which will be assembled. This strategy is related to the artificial intelligence approach based on a *blackboard* [13]. It is used in [7] with a unique numerical iterative method.

We can say that the first strategy is top-down, whereas the second is bottom-up. We have used the second one in our prototype, because it favours the conjoint use of several solving methods, which can be deeply different. Indeed, we can associate specialized knowledge-based systems, computer algebra systems and numerical iterative methods in the same solver. That is the idea of *multi-agent* systems, where agents are the local solving methods.

In our prototype, the subsystem of constraints to solve in one step is not always determined in the same way. Thus, the knowledge-based systems try to solve the maximum of the remaining constraints to formally obtain a triangular system. Conversely, the use of other methods is restrained, to avoid the propagation and amplification of approximations or partial solutions. That is the case for the Newton–Raphson method, which is restricted to take into account the minimum set of constraints necessary to solve only one of them.

In fact, in our prototype, at a given step, several constraint subsystems already solved can coexist without any possible assembling, because they have not any common reference. The prototype tries then to complete one of them using the border of the others. In case of failure, a new solving process is tried by fixing a new reference.

5.2. Activation of the local solving methods

The local solving methods included in a solver depend on the underlying geometric universe and on the type of constraints. Thus, constraints which can be expressed by ruler and compass can be handled by a geometric knowledge-based system, like *Progé* [18, 47,48]. Constraints, where more complex elements are included, e.g. areas of polygons or trigonometric functions, can be scarcely treated with any method but numerical iterative ones, e.g. Newton–Raphson. Classical procedural methods where a triangular system is directly produced can also be accepted. Thus, programs of interactive construction in CAI, like *LEGO* [21], the system described in [40], *Cabri-Geomètre* [3] or *Geometer's Sketchpad* [26] generate directed acyclic graphs of constructions which could be integrated in our framework.

As classically in the multi-agent systems, two strategies can be considered to activate the local solving methods. They can be sequentially triggered taking into account priorities. They can be activated in parallel with the usual problems of concurrency, synchronization and termination [13]. We have not yet worked in this direction, because the sequential case has always been complex enough to seize upon our attention. Indeed, we must say at first why and when trigger a method and not another one. The criteria to be considered are numerous : formal character or not, completion, quickness of solving.

Thus, in our prototype, where a formal result—more precisely a construction plan—is first required, we always favour geometric methods based on reasoning rather than numerical methods. The second ones are triggered only when the first ones block, what is realized by a technique of static priorities. The method having the highest priority is activated until it cannot make progress the solving anymore or an assembling is possible. In this case, the assembling is tried, with success or failure. Then, the methods are once more required in the decreasing priority order. The running is stopped when the formal construction is complete or no method can be activated and no assembling is possible.

5.3. Detection of common references

An important point is the detection of common references between two construction plans. We saw in Section 4.3 that two constraint systems have a common reference when the type of their common unknowns contains the *Reference* type. When the constraint systems are two construction subplans, these unknowns must be defined both in the two subplans.

For these common unknowns to determine a reference—or coordinate system—the correspondent figure must be proper (Section 3.1) and some relationships between the degrees of freedom must be satisfied. In a practical way, when the underlying polynomial equations are algebraically independent (see Section 3.3), these two conditions lead us *in general*, to check that

$$D - Dm - R > 3,$$

where D, Dm and R are respectively the sums of the degrees of freedom of the common objects built in the two subplans, of their degrees of freedom *modisp* and of the *degrees of restriction* of the relations between these objects. The degree of restriction of a relation, also called *valency* [30], corresponds to the number of degrees of freedom removed from the objects which are bound by the relation.

The previous relationship brings to the fore that it is not sufficient to test that $D - Dm \geqslant 3$ to know if there is exactly a common reference or more, but that a corrective term R must be subtracted, to measure the formal cleanness of the figure (see Section 3.1).

Example 5.1. Consider two subplans with the common points a and b. These two points are given explicitly by setting a constraint of the form $distpp(a, b) = k$ or implicitly because they are in the same subplan. Thus, for this configuration, we have $D = 4$, $Dm = 0$ and $R = 2$ or 1, whether we have detected that the two points are confound ($k = 0$) or not($k \neq 0$). In the first case, $D - Dm - R = 2$ and the configuration does not determine a reference, because the 2-tuple (a, b) is not a proper figure. In the second case, $D - Dm - R = 3$, the configuration determines a reference, and even more than a reference, because $D - Dm = 4$.

Consider now two subplans with common point a and circle c. Then, $D = 5$, $Dm = 1$ and $R = 2$ or 1, whether a is detected as to be bound to c, e.g. as its centre, or not. In the first case, $D - Dm - R = 2$ and this configuration does not determine a reference, because the 2-tuple (a, c) is not a proper figure. In the second case, $D - Dm - R = 3$, this configuration determines a reference, and even more than a reference, because $D - Dm = 4$.

The hypothesis of algebraic independence makes the test of formal equality possible, between points a and b in the first example, and between point a and the centre of c in the second one.

When the same reference has been detected in a configuration belonging to two subplans, its extraction and the corresponding displacement must be prepared to realize the assembling. In the prototype, different procedures of extraction are used depending on the nature of the configuration.

Let us notice however that even if an assembling is found in the formal phase, the numeric interpretation may abort with particular values, which enables a failure of the effective construction.

5.4. *Triggering of the assembling*

The case of the assembling of two solved systems was investigated in Section 4. This is insufficient since Section 5.3 showed that more than two systems can be candidate to an assembling. In this general case, the assembling possibilities and order must be studied.

Theorem 5.2 (Order of assembling). *Let $S_1.r_1$, $S_2.r_2$ and $S_3.r_3$ be three solved located subsystems stemming from a well-constrained modisp system S. If they are assemblable two by two, then we have*:

$$\begin{aligned}
&asb(asb(S_1.r_1, S_2.r_2, \varphi_{2\to 1}), S_3.r_3, \varphi_{3\to 1})\\
&\quad\Leftrightarrow\quad asb(asb(S_1.r_1, S_3.r_3, \varphi_{3\to 1}), S_2.r_2, \varphi_{2\to 1})\\
&\quad\Leftrightarrow\quad asb(asb(S_2.r_2, S_3.r_3, \varphi_{3\to 2}), S_1.r_1, \varphi_{1\to 2}).
\end{aligned}$$

This result can be generalized at any number of subsystems. This extension shows that when the assembling two by two of several solved subsystems is possible, it can be achieved at any moment, in any order, without prejudice for the completeness of the result. The possibilities of assembling can be examined in greater details.

When the assemblable systems have exactly a common reference, we can have the situation of three located subsystems $S_1.r_1$, $S_2.r_2$ and $S_3.r_3$ obtained by local solving of a well-constrained *modisp* system and such that $S_1.r_1$ and $S_2.r_2$ are assemblable and $S_3.r_3$ is assemblable with $asb(S_1.r_1, S_2.r_2, \varphi_{2\to 1})$, but neither with $S_1.r_1$ nor with $S_2.r_2$. In other words, an order is imposed by the assembling possibilities. This happens only when $S_3.r_3$ and $asb(S_1.r_1, S_2.r_2, \varphi_{2\to 1})$ have exactly a common reference.

When, as it is often the case, the geometric universe does not allow the definition of exact references, two subsystems are assemblable only if they have *strictly more than* a common reference. In these conditions, the previous situation cannot happen. The frame of the assembling is more restrictive, but the strategy can be simplified thanks to the following theorem, satisfied *in general* in the case of algebraic independence of the underlying polynomial equations (see Sections 3.3 and 5.3).

Theorem 5.3 (Assembling possibilities). *Let $S_1.r_1$, $S_2.r_2$ and $S_3.r_3$ be three solved located subsystems stemming from a well-constrained modisp system S. If $S_1.r_1$ and $S_2.r_2$ are assemblable into $asb(S_1.r_1, S_2.r_2, \varphi_{2\to 1})$ and this system is itself assemblable with $S_3.r_3$, then $S_3.r_3$ is assemblable with $S_1.r_1$ (case 1) or with $S_2.r_2$ (case 2).*

Proof. By hypothesis, if $S_3.r_3$ is assemblable with $asb(S_1.r_1, S_2.r_2, \varphi_{2\to 1})$, the two subsystems contain strictly more than a common reference. For $S_3.r_3$, this reference is common with $S_1.r_1$ (case 1), with $S_2.r_2$ (case 2) or else with $asb(S_1.r_1, S_2.r_2, \varphi_{2\to 1})$ in its entirety. The latter case is impossible, because it would enable the metric of this common reference to be defined both in $S_3.r_3$ and in $asb(S_1.r_1, S_2.r_2, \varphi_{2\to 1})$, which would impose, by Proposition 4.22, that the solving of $S_3.r_3$ needed a constraint of the border of $asb(S_1.r_1, S_2.r_2, \varphi_{2\to 1})$, excluding the borders of $S_1.r_1$ and $S_2.r_2$. Now, this is false by hypothesis, because this assembling was not available at the construction of $S_3.r_3$ and we are *in general* in the case of algebraic independence of the underlying polynomial equations (see Section 3.3). The two other cases enable $S_3.r_3$ to be assemblable with $S_1.r_1$ (case 1) or with $S_2.r_2$ (case 2). □

Thus, the assembling can be done according to different strategies, particularly when a system is completely decomposed and locally solved or, conversely, as soon as possible. In the first case, the assembling is *as early* and in the second one *at the latest*. For instance, in [41], a complete decomposition of the constraint graph in triangles is achieved before a local solving of the correspondent subsystems and an assembling by *reconstruction*.

5.5. Exhaustive strategies

In our framework, we have a finite set of possible references—using the initial unknowns—, a finite set of local solving methods, and, at each step, a residual subsystem constraining only initial unknowns. Therefore, it is possible to finitely enumerate all the triggering possibilities of all local methods in all references in all the possible subsystems. A strategy bringing into play such an enumeration is said *exhaustive*. Our prototype use such a strategy.

When the local methods are correct and complete, whatever the exhaustive strategy used, the result is the same. That is, either all the strategies fail to solve the constraint system or all of them yield the set of all the numerical solutions. In practice, the local solving methods are at the best correct, so the global resulting solving method is itself incomplete.

Note that, although the assembling method is correct and complete, even if the local methods are correct and complete —on the subsystems that they determine theirselves— and the triggering strategy is exhaustive, the global solving method can fail to solve well-constrained systems. This comes from the lack of power of the local methods letting unsolved constraints, what limits the possibilities of assembling.

6. A prototype description

This section describes the characteristics of the prototype. It is based on the assembling idea and integrates a formal solving engine which works with several local resolution methods.

6.1. General presentation

Our prototype has been designed within the framework of a project on a topology-based modeller [33,34,36] named *Topofil* [4,5]. The interface of the modeller, which is used to draw sketches, has been augmented in order to enable the user to enter metric constraints. Topological constraints, i.e. incidence and adjacency constraints, are deduced from the sketches. So, the prototype can be seen as a formal constraint-based construction module added to the modeller.

Our approach is based on the assembling of subfigures obtained by application of different local solving methods federated by the prototype. As explained in Section 5, each method is applied from a fixed initial local reference. For this experimentation, the prototype had three local solving methods. Two of them are knowledge-based systems. The third one builds equation systems solved by the Newton–Raphson method. Thus, our prototype is similar to a multi-agent system.

Initially, a common base of facts contains the system of constraints, i.e. the description of the geometric objects and the constraints. Every solving method accesses and updates the base, which can be seen as the blackboard of our multi-agent system. During a solving phase, the blackboard contains also border informations, i.e. informations about the subfigures already constructed.

In the prototype, a general procedure, following an exhaustive strategy, launches the local solving and applies an *early* assembling strategy. Each time two subfigures have a common reference, the procedure immediately assembles them. We choose an early strategy for two main reasons. The first one is to avoid the production of irrelevant little subfigures contained in other bigger ones. The second reason is to allow a dynamic modification of the border informations. Each time a constructive definition is deduced, the prototype tries to detect a possible assembling. In this case, it stops the running method and performs the assembling. Theorem 5.2 attests that there is no possible assembling left after such an assembling phase.

At each step, the running method extracts from the blackboard the part of the problem not yet solved. Next, it completes the blackboard step by step with new pieces of knowledge. The assembling phase updates the blackboard too. The whole formal solving is achieved when the problem is solved or no new deduction can be produced. Note that the Newton–Raphson method ensures the success for a well-constrained problem. However, the solving may fail in the numerical phase, due to the classical reasons of non-convergence.

The result of the formal phase is a general construction plan which will be numerically interpreted. Some numerical results are not correct due to the use of *necessary* conditions by some local solving methods such as the knowledge-based systems. A verification of the constraints is necessary to filter the results. At this stage, our protoype systematically checks all the constraints, used or not used in the solving process. This way, constraint redundancy has no harmful effect and over-constrained cases are finally noticed if no numerical solution remains. Finally, heuristics are used in order to narrow down the set of solutions.

Like the modeller *Topofil*, the prototype was developed in C language for Silicon Graphics workstations using the GL graphic library. The next sections briefly present the blackboard and our local resolution methods. More technical details are given in [38].

6.2. Geometric universe and blackboard

For several reasons explained in [1, 10, 22], we only use predicates to formalize the constraints. Thus, all constraints are predicative terms of depth 1. Function symbols appear in our geometric universe only to express the construction.

Example 6.1. The function symbol *distpp* : *Point* × *Point* $\longrightarrow$ *Length* previously used becomes the relation symbol *distpp* : *Point* × *Point* × *Length* $\longrightarrow$. However, we sometimes need the function symbol *fdist* : *Point* × *Point* $\longrightarrow$ *Length* to define lengths in a construction plan. The relation symbol *angle* : *Point* × *Point* × *Point* × *Point* × *Angle* $\longrightarrow$ is needed to express constraints between two bi-points. The function symbol *initl* : *Real* $\longrightarrow$ *Length* converts a real number into a length. We also have *inv_angle*, *supp_angle*, *mod_angle* : *Angle* $\longrightarrow$ *Angle*, to express respectively the opposite, the supplementary, the value modulo 2π in $]-\pi, \pi]$ of an angle.

Unknowns and parameters of the system, seen as variables in Section 3, are considered in the prototype as typed logic constants. Each constraint is also formalized with a closed predicative term, which is a *fact*. Arguments of facts are either unknowns or parameters which represent geometric objects, i.e. points, lines, circles, lengths and so on. The depth of predicative terms is 1 for efficiency reasons, particularly in the unification of terms. In order to avoid functional terms in constraints, logic constants are set in their place. From a syntactical point of view, such a constant is a synonym of a functional term. From a semantical point of view, it is a geometric object defined by a functional term.

Example 6.2. With the function and predicate symbols of Example 6.1, a distance constraint of 4.5 units between two points p_1 and p_2 is written $k = initl(4.5)$ and $distpp(p_1, p_2, k)$, where k is a logical constant of type *Length*. If α_1 and α_2 are constants of type *Angle*, an angle constraint of $-(\pi - \alpha_1)$ between the directions (p_1p_2) and (p_3p_4) is written $\alpha_2 = inv\_angle(supp\_angle(\alpha_1))$ and $angle(p_1, p_2, p_3, p_4, \alpha_2)$.

Initially, the blackboard contains the statement of the problem. A solving method adds the local construction it produces and some new facts which are *private*. These private facts are eliminated when the running solving method stops. So, they correspond to a local base of facts. The border of a construction is not expressed by the equality of metric terms, as explained in Section 4, but just by predicative terms. Moreover, it is obvious that the complete evaluation of the border is not relevant due to a risk of combinatorial explosion of the number of facts. This is why, in our prototype, the border is only implicitly represented.

Indeed, the blackboard is not just a memory area, because access functions include deduction mechanisms to limit the number of explicit pieces of knowledge. These functions manage both the border and some particular properties of constraints. For instance, among other properties, permutativity and Chasles relation are realized by deduction mechanisms for the angle constraints. So, for the internal representation, we need particular data types. But, these ones are hidden by access functions to the local solving methods, which only see predicates of depth 1 in the base of facts.

6.3. Knowledge-based systems

The first two local solving methods are geometric formal methods in the way of Aldefeld [1]. They are based on the same logical theory and differ from each other by their solving strategy. Both use the method of loci [11,18,32,42,47,48] and infer new relations from initial constraints. The deduced relations are either *defining* incidence relations giving a geometric locus or new constraints. The construction of a geometric object is inferred from *defining* incidence relations.

Example 6.3. A defining incidence relation is for instance "point p is on line l" or "point p is on circle c". If line l and circle c are already defined, the construction of p is then defined by their intersection.

The axioms of our logic theory have the form of either closed atoms or implications with variables. Variables involved in the premises are always universally quantified. The others, only present in the conclusions of implications, are existentially quantified. They correspond to the naming of functional terms, as explained in Section 6.2.

The inference rules are the *modus ponens* and the *paramodulation* used to deal with the synonyms. The functional meaning of some relation symbols like *distpp* is used to detect synonymous objects, which are put in the same equivalence class.

Both of the knowledge-based systems contain *production rules* corresponding to the implications of the logical theory. Each production rule has the form

$$\textit{if } P1, \ldots, Pn \textit{ then } C1, \ldots, Cm \textit{ conditions } D1, \ldots, Dp,$$

where the P_i are premises, the C_i are conclusions, and the D_i are extra-logic conditions. The conclusions are either predicative terms of depth 1 or definitions like

$$Z = f(Z_1, \ldots, Z_k),$$

where f is a function symbol, the Z_i and Z are variables. The construction plan increases in size each time such a definition is produced. If Z is existentially quantified and replaced by a constant, then the functional term defines the constant. Since the function symbols are not in the logic theory, definitions are never reused to instantiate the premises.

Conditions in the production rules do not modify the meaning of the geometric construction. They avoid a proliferation of irrelevant relations. Indeed, the verification of conditions tests whether objects are already defined or not in the local reference. Applying a rule yields new relations only if the constants substituted for the variables validate the conditions.

The blackboard records all the deductions. The part of the blackboard corresponding to the local base of facts initialized for the running method increases with new relations. The construction plan, recorded in the general blackboard, is extended with new definitions.

The two knowledge based-systems have the same logic theory but differ by their search strategy, in relation with non-termination risks. Indeed, we have noticed that, with a usual search strategy, the production rules used to solve most of our problems do not produce loops. Thus, the knowledge base of the first system only contains this kind of rules, and, for this geometric framework, the most efficient strategy is the *depth-first search.*

However, the need to have a more complete module of geometric construction led us to consider in our second knowledge-based system more complicated rules that often produce loops. In this system, a *breadth-first search* with a limitation on the depth of the search tree is used. It is not complete, but suffices in most cases to achieve sophisticated constructions.

Thus, we have two levels of knowledge-based systems. But, in CAD, most of the problems are solved with the rules of the first level only. Therefore, it can begin a solving, and, when it fails, the second level system is required.

6.4. Newton–Raphson method

The Newton–Raphson method [39,43] finds a numerical solution for a non-linear equation system. This method is very useful in CAD software because it can solve most of the constraints set in this area. However, it has many well-known drawbacks, as convergence problems, unicity of the solution found and lack of information after a failure. So, when it is possible, we prefer others methods.

Our prototype contains the Newton–Raphson method to raise its solving capability and to show how it can federate several very different methods. In fact, the Newton–Raphson method is called to continue a solving when the two knowledge-based systems fail. When it is running, the problem will certainly be totally solved. However, in many cases, it is enough to solve a few parts of the problem determined from only one constraint by Newton–Raphson, and the other methods can work anew.

One can wonder why such a numerical method is used in a purely formal framework. In fact, in the first formal phase, our method only prepares a formal system of numerical equations, which is effectively solved in the second interpretative phase. The Newton–Raphson method being well known, we only give some indications about the setting up of the equations system to be solved.

A geometric constraint is chosen by a heuristics working on the not yet solved constraints. Next, a minimal system containing this constraint is constituted. The way of finding such a minimal system is based on irreductibility properties of equations systems. A complete explanation is given in [38].

A minimal system can be written

$$\{p_1[X, A], \ldots, p_k[X, A]\},$$

where $X = \{x_1, \ldots, x_n\}$ is the set of geometric unknowns and the $p_i[X, A]$ are the constraints at work. This system is used to produce the global definition

$$x_1, \ldots, x_n = NR(p_1, \ldots, p_k, x0_1, \ldots, x0_n),$$

where $x0_1, \ldots, x0_n$ are the initial values of the concerned geometric objects extracted from the sketch and *NR* is the function of the Newton–Raphson numerical resolution. A triangular form is obtained with the projection functions pr_i:

$$\begin{aligned} x_1 &= pr_1(NR(p_1, \ldots, p_k, x0_1, \ldots, x0_n)), \\ &\vdots \\ x_n &= pr_n(NR(p_1, \ldots, p_k, x0_1, \ldots, x0_n)). \end{aligned}$$

These definitions are actually added to the construction plan. In fact, the references defined in Section 3.1 are used to extract a numerical system with real unknowns $t_1, \ldots, t_m$ from the minimal system, and the equations

$$\begin{aligned} f_1(t_1, \ldots, t_m) &= 0, \\ &\vdots \\ f_m(t_1, \ldots, t_m) &= 0, \end{aligned}$$

where f_i are polynomial functions. This formal system, which is prepared in the formal phase, is numerically solved by the Newton–Raphson method with the initial values $t0_1, \ldots, t0_m$ deduced from $x0_1, \ldots, x0_n$. When a solution is found, $t_1, \ldots, t_m$ are the real coordinates of the geometric unknowns $x_1, \ldots, x_n$. So, the Newton–Raphson method is well integrated in our formal way of solving by triangulation.

7. Some examples

We present here three examples solved with our prototype. In each case, the sketch is interactively entered with the modeller *Topofil*, that allows us to fix the topology. The user interactively gives the metric constraints. Our prototype automatically names the geometric objects like points, lines, circles. It puts the constraints into the blackboard. Then, the solving system produces the formal construction plan and solves it numerically. The solutions are also drawn on the screen.

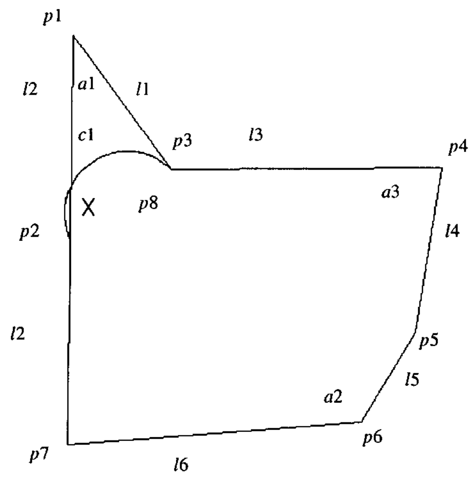

Fig. 3. Sketch of Example 1.

Table 1
Constraint system of Example 1

$onl(p1, l1)$	% point $p1$ is on line $l1$	$fixpl(p7, l2, p2)$	% sets the initial reference
$onl(p1, l2)$		$distpp(p1, p2, k1)$	% the distance from $p1$ to $p2$ is $k1$
$onl(p2, l2)$		$distpp(p7, p2, k2)$	
$onl(p3, l1)$		$distpp(p4, p3, k3)$	
$onl(p3, l3)$		$distpp(p5, p4, k4)$	
$onl(p4, l3)$		$distpp(p7, p6, k5)$	
$onl(p4, l4)$		$distpl(p5, l6, k6)$	% the angle of lines $p1p2$, $p1p3$ is $a1$
$onl(p5, l4)$			
$onl(p5, l5)$		$angle(p1, p2, p1, p3, a1)$	
$onl(p6, l5)$		$angle(p6, p5, p6, p7, a2)$	
$onl(p6, l6)$		$angle(p4, p3, p4, p5, a3)$	
$onl(p7, l2)$		$tgclp(c1, l1, p3)$	% circle $c1$ is tangent to line $l1$ at $p3$
$onl(p7, l6)$		$tgclp(c1, l2, p2)$	
$onc(p2, c1)$	% point $p2$ is on circle $c1$	$centre(c1, p8)$	% $p8$ is the center of circle $c1$
$onc(p3, c1)$			

7.1. Example 1

We first return to the introductive example of Section 2. The sketch, where a voluntarily incorrect intersection of circle and line has been drawn, is in Fig. 3. The geometric objects have names chosen by the prototype, *pi*, *li*, *ci*, *ai* and *ki*, respectively for points, oriented straight lines, circles, angles and lengths. The names are not the same as in Section 2.

The corresponding constraint system lies in Table 1. In this table, we can remark the constraint $fixpl(p7, l2, p2)$ which plays the role of our $ref(x_{i_1}, \ldots, x_{i_k})$ and sets a local reference with $p7$ as origin and $l2$ as x-axis oriented towards $p2$. In fact, this first local reference is extracted from the sketch and will be the final reference for the whole figure. This forces the prototype to work in a desired way. In the following table, explanations about newly introduced symbols lie after %. The whole construction plan, which is automatically elaborated by the prototype, is in Table 2. First, we can notice the fixation of the metric parameters at the begin of the plan. Next, the construction is obtained by successive assemblings of three subplans, which are built by the first level knowledge-based system.

Each subplan is numbered and the corresponding geometric definitions are suffixed by the number in dotted notation. For instance, the definition of $p7$ in the subplan 1 is named $p7.1$. A subplan begins with the definitions of a point and an oriented line to set a local reference. For instance, for the subplan 1, we have:

$$p7.1 = initp(-296.958130, -248.927719)$$
$$l2.1 = initd(p7.1, 1.561722)$$

The local origin is at $(0, 0)$ except for the first subplan, which solves the particular constraint $fixpl(p7, l2p2)$ fixing point $p7.1$ and line $l2.1$ passing through $p7.1$ and $p2.1$ at their values in the sketch by functions *initp* and *initd*. Note that, in the subplans 2 and 3, this reference constraint is relaxed for the benefit of a new one. The border appears in the construction subplan of the subfigure 3 with the definitions of lengths $k9$ and $k10$ by *fdist*:

$$k9 = fdist(p7.1, p3.1)$$
$$k10 = fdist(p5.2, p3.2)$$

These definitions are extracted from the subplans 1 and 2, as indicated by the suffixes .1 and .2, respectively for points $p7$, $p3$ and $p5$, $p3$. They are used in the definitions of circles $c15.3$ and $c16.3$, whose intersection gives $p3.3$.

The complete construction is finally brought back by displacements in the first reference. Note that the formal solution founded by the prototype is different from the one given in Section 2, which was obtained by fixing another reference for the whole figure.

The formal execution time is 0.15 sec on a Silicon Graphics Indigo 2 workstation. The interpretation with the user's data yields several numerical solutions. The topological constraints and the dimensions are satisfied by some of these, like those in Fig. 4, which have been selected. Note that the second drawing in this figure perfectly respects the statement, even if it is probably not in accordance with the user's wishes. Other solutions, which do not satisfy the constraints, are eliminated.

7.2. Example 2

Here is an example where usual graph decomposition methods fail and for which we propose two ways of solving. At first, the prototype requires the two knowledge-based systems, particularly the powerful so-called *parallelogram rule* [54,55]. Next,

Table 2
Construction for Example 1

% fixing of the metric parameters	
$k5 = \mathit{initl}(250.000)$	% initialization of length $k5$
$k4 = \mathit{initl}(158.000)$	
$a3 = \mathit{inita}(1.413717)$	% initialization of angle $a3$
$k3 = \mathit{initl}(304.000)$	
$a2 = \mathit{inita}(2.809980)$	
$k2 = \mathit{initl}(200.000)$	
$k1 = \mathit{initl}(300.000)$	
$a1 = \mathit{inita}(0.558505)$	
$k6 = \mathit{initl}(70.000)$	
% subplan suffixed by .1	
$p7.1 = \mathit{initp}(-296.958, -248.927)$	% initialization of point $p7.1$
$l2.1 = \mathit{initd}(p7.1, 1.561722)$	% initialization of line $l2.1$
$a12 = \mathit{inv\_angle}(\mathit{supp\_angle}(a1))$	% definition of angle $a12$
$c3.1 = \mathit{mkcir}(p7.1, k2)$	% circle with centre $p7.1$ and radius $k2$
$p2.1 = \mathit{interlc}(l2.1, c3.1)$	% intersection of $l2.1$ and $c3.1$
$c4.1 = \mathit{mkcir}(p2.1, k1)$	
$p1.1 = \mathit{interlc}(l2.1, c4.1)$	
$a7.1 = \mathit{inita}(1.570796)$	
$l7.1 = \mathit{lpla}(p2.1, l2.1, a7.1)$	% line passing through $p2.1$ and making angle $a7.1$ with $l2.1$
$a9.1 = \mathit{bissect}(a1)$	% $a9.1$ is half $a1$
$l8.1 = \mathit{lpla}(p1.1, l2.1, a9.1)$	
$p8.1 = \mathit{interll}(l8.1, l7.1)$	% intersection of $l8.1$ and $l7.1$
$l1.1 = \mathit{lppa2}(p1.1, p2.1, p1.1, a12)$	% line passing through $p1.1$ and making angle $a12$ with $p1.1p2.1$
$c1.1 = \mathit{mkcir2}(p8.1, p2.1)$	% circle with centre $p8.1$ and passing through $p2.1$
$p3.1 = \mathit{interlc}(l1.1, c1.1)$	
$a55 = \mathit{inv\_angle}(a3)$	
% subplan suffixed by .2	
$p5.2 = \mathit{initp}(0.000, 0.000)$	
$l4.2 = \mathit{initd}(p5.2, 0.000)$	
$c6.2 = \mathit{mkcir}(p5.2, k4)$	
$p4.2 = \mathit{interlc}(l4.2, c6.2)$	
$c7.2 = \mathit{mkcir}(p4.2, k3)$	
$l3.2 = \mathit{lpla}(p4.2, l4.2, a55)$	
$p3.2 = \mathit{interlc}(l3.2, c7.2)$	
$a105 = \mathit{inv\_angle}(a2)$	
% subplan suffixed by .3	
$p7.3 = \mathit{initp}(0.000, 0.000)$	
$l6.3 = \mathit{initd}(p7.3, 0.000)$	
$l9.3 = \mathit{ldl}(l6.3, k6)$	
$c11.3 = \mathit{mkcir}(p7.3, k5)$	
$p6.3 = \mathit{interlc}(l6.3, c11.3)$	
$l5.3 = \mathit{lpla}(p6.3, l6.3, a105)$	

Table 2 — continued

$p5.3 = \mathit{interll}(l5.3, l9.3)$	
$k9 = \mathit{fdist}(p7.1, p3.1)$	
$k10 = \mathit{fdist}(p5.2, p3.2)$	
$c15.3 = \mathit{mkcir}(p7.3, k9)$	
$c16.3 = \mathit{mkcir}(p5.3, k10)$	
$p3.3 = \mathit{intercc}(c16.3, c15.3)$	% intersection of $c16.3$ and $c15.3$
% definition of the displacement $dep1.3$ from 2 to 3	
$dep1.3 = \mathit{make\_dep\_pp}(p3.3, p5.3, p3.2, p5.2)$	
% use of displacement $dep1.3$	
$p4.3 = \mathit{transfp}(p4.2, dep1.3)$	% displacement of $p4.2$ into $p4.3$
% now the subplan .1 is completed	
% definition of the displacement $dep2.1$ from 3 to 1	
$dep2.1 = \mathit{make\_dep\_pp}(p3.1, p7.1, p3.3, p7.3)$	
$p6.1 = \mathit{transfp}(p6.3, dep2.1)$	% displacement of $p6.3$ into $p6.1$
$p5.1 = \mathit{transfp}(p5.3, dep2.1)$	% displacement of $p5.3$ into $p5.1$
$p4.1 = \mathit{transfp}(p4.3, dep2.1)$	% displacement of $p4.3$ into $p4.1$

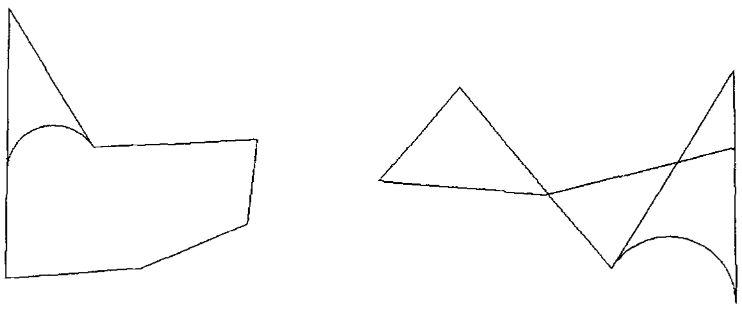

Fig. 4. Two numerical solutions for Example 1.

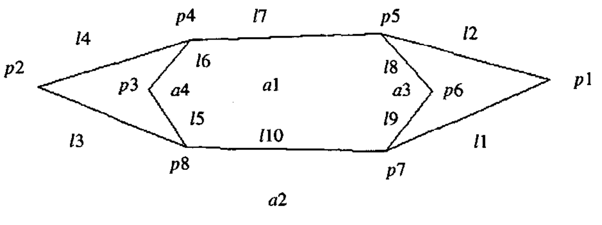

Fig. 5. Sketch of Example 2.

Table 3
Constraint system of Example 2

$onl(p1, l1)$	$onl(p8, l5)$
$onl(p1, l2)$	$onl(p8, l10)$
$onl(p2, l3)$	
$onl(p2, l4)$	$distpp(p5, p1, k1)$
$onl(p3, l5)$	$distpp(p8, p2, k2)$
$onl(p3, l6)$	$distpp(p4, p2, k3)$
$onl(p4, l4)$	$distpp(p3, p8, k4)$
$onl(p4, l6)$	$distpp(p4, p3, k5)$
$onl(p4, l7)$	$distpp(p5, p4, k6)$
$onl(p5, l2)$	$distpp(p6, p5, k7)$
$onl(p5, l7)$	$distpp(p7, p6, k8)$
$onl(p5, l8)$	$distpp(p8, p7, k9)$
$onl(p6, l8)$	
$onl(p6, l9)$	$angle(p8, p7, p4, p5, a1)$
$onl(p7, l1)$	$angle(p8, p2, p7, p1, a2)$
$onl(p7, l9)$	$angle(p6, p7, p6, p5, a3)$
$onl(p7, l10)$	$angle(p3, p8, p3, p4, a4)$
$onl(p8, l3)$	

we forget the second level knowledge-based system, and the prototype requires the Newton–Raphson method. The sketch with the names of the geometric objects chosen by the prototype is presented in Fig. 5.

The system of geometric constraints is in Table 3 and the first geometric construction is presented in Table 4. Note that the subplan numbers 2, 4, and 6 are not visible, because they correspond to partial solving attempts, which have aborted after fixing a local reference without discovering new definitions. Both of the knowledge-based systems are required. The first level system builds the beginning of the subplan 1 and the subplans 3, 5, 7. Subplan 1 is completed with the following definitions deduced by the second level system applying the parallelogram rule.

$$k18 = fdist(p8.3, p4.3)$$

$$k19 = fdist(p7.7, p5.7)$$

$$c65.1 = mkcir(p8.1, k18)$$

$$l12.1 = lppa(p7.1, p8.1, a1)$$

$$c68.1 = mkcir(p7.1, k6)$$

$$p10.1 = interlco(l12.1, c68.1) \quad \% \text{ intersection of } l12.1 \text{ and } c68.1$$

$$c69.1 = mkcir(p10.1, k19)$$

$$p4.1 = intercc(c69.1, c651)$$

Let us explain this fragment of plan in which the prototype tries to construct the quadrilateral $p8p4p5p7$ in the subplan numbered by 1. The points $p7$ and $p8$, the distance $k6$ between $p5$ and $p4$, and the angle $a1$ between oriented lines $p8p7$ and $p4p5$

Table 4
Construction 1 for Example 2

$k9 = initl(200.000)$	$c21.5 = mkcir(p4.5, k6)$
$k8 = initl(100.000)$	$p5.5 = interlc(l7.5, c21.5)$
$k7 = initl(100.000)$	
$k6 = initl(200.000)$	$a472 = inv\_angle(a3)$
$k5 = initl(100.000)$	$p5.7 = initp(0.000, 0.000)$
$k4 = initl(100.000)$	$l8.7 = initd(p5.7, 0.000)$
$a4 = inita(2.024582)$	$c30.7 = mkcir(p5.7, k7)$
$a3 = inita(-2.164208)$	$p6.7 = interlc(l8.7, c30.7)$
$a1 = inita(0.034907)$	$c33.7 = mkcir(p6.7, k8)$
$k3 = initl(150.000)$	$l9.7 = lpla(p6.7, l8.7, a472)$
$k2 = initl(180.000)$	$p7.7 = interlc(l9.7, c33.7)$
$k1 = initl(210.000)$	
$a2 = inita(-3.141593)$	$k18 = fdist(p8.3, p4.3)$
	$k19 = fdist(p7.7, p5.7)$
$p8.1 = initp(0.000, 0.000)$	$c65.1 = mkcir(p8.1, k18)$
$l10.1 = initd(p8.1, 0.000)$	$l12.1 = lppa(p7.1, p8.1, a1)$
$c1.1 = mkcir(p8.1, k9)$	$c68.1 = mkcir(p7.1, k6)$
$p7.1 = interlc(l10.1, c1.1)$	$p10.1 = interlco(l12.1, c68.1)$
	$c69.1 = mkcir(p10.1, k19)$
$p8.3 = initp(0.000, 0.000)$	$p4.1 = intercc(c69.1, c65.1)$
$l5.3 = initd(p8.3, 0.000)$	$dep1.1 = make\_dep\_pp(p4.1, p8.1, p4.3, p8.3)$
$c10.3 = mkcir(p8.3, k4)$	$p3.1 = transfp(p3.3, dep1.1)$
$p3.3 = interlc(l5.3, c10.3)$	$p2.1 = transfp(p2.3, dep1.1)$
$c11.3 = mkcir(p8.3, k2)$	$l1.1 = lppa2(p7.1, p8.1, p2.1, a2)$
$c12.3 = mkcir(p3.3, k5)$	$l7.1 = lppa2(p4.1, p8.1, p7.1, a1)$
$l6.3 = lpla(p3.3, l5.3, a4)$	$dep2.1 = make\_dep\_pl(p4.1, l7.1, p4.5, l7.5)$
$p4.3 = interlc(l6.3, c12.3)$	$p5.1 = transfp(p5.5, dep2.1)$
$c13.3 = mkcir(p4.3, k3)$	$dep3.1 = make\_dep\_pp(p5.1, p7.1, p5.7, p7.7)$
$p2.3 = intercc(c13.3, c11.3)$	$p6.1 = transfp(p6.7, dep3.1)$
	$c70.1 = mkcir(p5.1, k1)$
$p4.5 = initp(0.000, 0.000)$	$p1.1 = interlc(l1.1, c70.1)$
$l7.5 = initd(p4.5, 0.000)$	

are known. By the two first lines of the fragment, two other edges of the quadrilateral are constrained by the distances $k18$ and $k19$ deduced from the border informations of subplans 3 and 7. The quadrilateral is built thanks to the easy construction of an auxiliary point $p10$ in such a way that $p4p5p7p10$ is a parallelogram. This point permits to directly obtain $p4$.

The second geometric construction is presented in Table 5. The first level knowledge-based system builts the beginning of the subplan 2 and the subplans 3, 5, 7. The local method based on Newton–Raphson is used to complete the subplan 2, giving formally the definitions of $p5.2$ and $p7.2$:

$$p5.2 = NR(0)$$
$$p7.2 = NR(0)$$

Table 5
Construction 2 with Newton–Raphson for Example 2

$k9 = initl(200.000)$	$p8.3 = initp(0.000, 0.000)$
$k8 = initl(100.000)$	$l10.3 = initd(p8.3, 0.000)$
$k7 = initl(100.000)$	$c12.3 = mkcir(p8.3, k9)$
$k6 = initl(200.000)$	$p7.3 = interlc(l10.3, c12.3)$
$k5 = initl(100.000)$	
$k4 = initl(100.000)$	$p4.5 = initp(0.000, 0.000)$
$a4 = inita(2.024582)$	$l7.5 = initd(p4.5, 0.000)$
$a3 = inita(-2.164208)$	$c26.5 = mkcir(p4.5, k6)$
$a1 = inita(0.034907)$	$p5.5 = interlc(l7.5, c26.5)$
$k3 = initl(150.000)$	
$k2 = initl(180.000)$	$a908 = inv\_angle(a3)$
$k1 = initl(210.000)$	$p5.7 = initp(0.000, 0.000)$
$a2 = inita(-3.141593)$	$l8.7 = initd(p5.7, 0.000)$
	$c35.7 = mkcir(p5.7, k7)$
$p8.2 = initp(0.000, 0.000)$	$p6.7 = interlc(l8.7, c35.7)$
$l5.2 = initd(p8.2, 0.000)$	$c38.7 = mkcir(p6.7, k8)$
$c6.2 = mkcir(p8.2, k4)$	$l9.7 = lpla(p6.7, l8.7, a908)$
$p3.2 = interlc(l5.2, c6.2)$	$p7.7 = interlc(l9.7, c38.7)$
$c7.2 = mkcir(p8.2, k2)$	
$c8.2 = mkcir(p3.2, k5)$	$p5.2 = NR(0)$ % Newton–Raphson method
$l6.2 = lpla(p3.2, l5.2, a4)$	$p7.2 = NR(0)$
$p4.2 = interlc(l6.2, c8.2)$	$dep1.2 = make\_dep\_pp(p7.2, p5.2, p7.7, p5.7)$
$c9.2 = mkcir(p4.2, k3)$	$p6.2 = transfp(p6.7, dep1.2)$
$p2.2 = intercc(c9.2, c7.2)$	$l1.2 = lppa2(p7.2, p8.2, p2.2, a2)$
$c67.2 = mkcir(p5.2, k1)$	
$p1.2 = interlc(l1.2, c67.2)$	

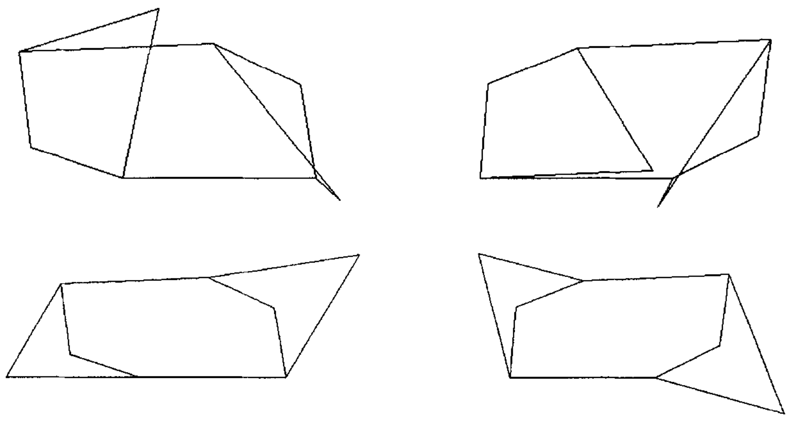
Fig. 6. Four solutions for Example 2.

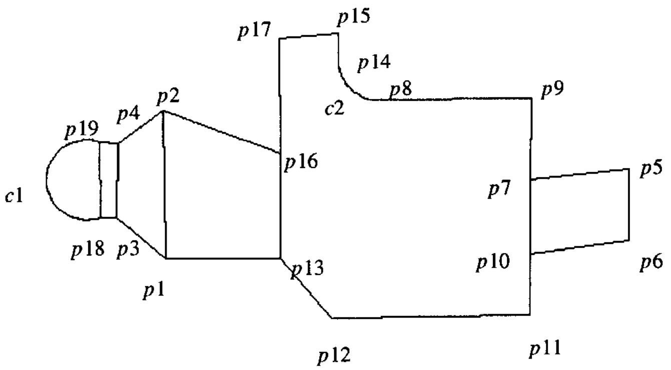

Fig. 7. Sketch of Example 3.

Table 6
Constraint system for Example 3

$onl(p1,l1)$	$onl(p10,l11)$	$onc(p19,c1)$	$distpp(p17,p15,k16)$
$onl(p1,l2)$	$onl(p10,l12)$	$onc(p8,c2)$	$distpp(p17,p16,k17)$
$onl(p1,l3)$	$onl(p11,l12)$	$onc(p14,c2)$	$angle(p3,p1,p4,p2,a1)$
$onl(p2,l4)$	$onl(p11,l14)$	$centre(c1,p20)$	$angle(p16,p2,p16,p13,a2)$
$onl(p2,l2)$	$onl(p12,l14)$	$centre(c2,p21)$	$angle(p1,p2,p1,p13,a3)$
$onl(p2,l5)$	$onl(p12,l15)$		$angle(p2,p1,p2,p16,a4)$
$onl(p3,l3)$	$onl(p13,l1)$	$distpp(p2,p1,k1)$	$angle(p18,p19,p18,p3,a5)$
$onl(p3,l6)$	$onl(p13,l16)$	$distpp(p2,p16,k2)$	$angle(p19,p18,p19,p4,a6)$
$onl(p3,l7)$	$onl(p13,l15)$	$distpp(p3,p1,k3)$	$angle(p5,p7,p5,p6,a7)$
$onl(p4,l5)$	$onl(p14,l17)$	$distpp(p4,p2,k4)$	$angle(p6,p10,p6,p5,a8)$
$onl(p4,l7)$	$onl(p15,l17)$	$distpp(p4,p3,k5)$	$angle(p10,p7,p10,p6,a9)$
$onl(p4,l8)$	$onl(p15,l18)$	$distpp(p19,p4,k6)$	$angle(p9,p8,p9,p7,a10)$
$onl(p5,l9)$	$onl(p16,l4)$	$distpp(p19,p18,k7)$	$angle(p11,p12,p11,p10,a11)$
$onl(p5,l10)$	$onl(p16,l16)$	$distpp(p9,p8,k8)$	$angle(p12,p13,p12,p11,a12)$
$onl(p6,l10)$	$onl(p17,l18)$	$distpp(p7,p9,k9)$	$angle(p15,p17,p15,p14,a13)$
$onl(p6,l11)$	$onl(p17,l16)$	$distpp(p6,p5,k10)$	$angle(p17,p16,p17,p15,a14)$
$onl(p7,l9)$	$onl(p18,l19)$	$distpp(p10,p6,k11)$	$radius(c1,k18)$
$onl(p7,l12)$	$onl(p19,l19)$	$distpp(p10,p11,k12)$	$radius(c2,k19)$
$onl(p8,l13)$	$onl(p19,l8)$	$distpp(p12,p11,k13)$	$tgclp(c2,l13,p8)$
$onl(p9,l12)$	$onl(p18,l6)$	$distpp(p13,p12,k14)$	
$onl(p9,l13)$	$onc(p18,c1)$	$distpp(p15,p14,k15)$	

In the numerical pass, the call $NR(0)$ prepares the equations system number 0 and solves it. The unknowns are the coordinates of points $p5.2$ and $p7.2$. The subplans number 1, 4, 6 correspond once again to aborted solving attempts.

The solving times on our workstation are respectively 1.37 sec and 1.12 sec. for the first and the second formal constructions. The prototype provides several solutions

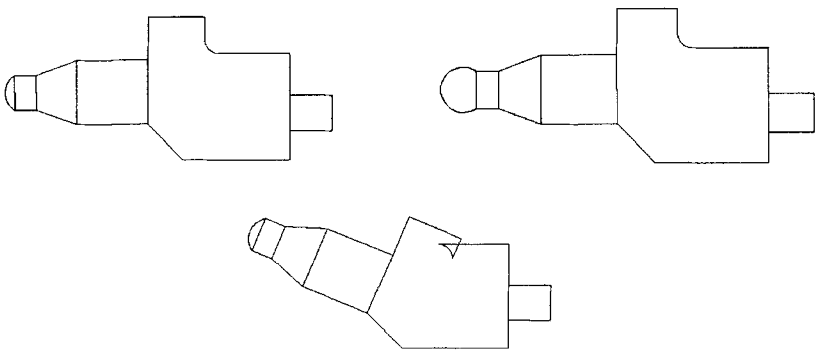

Fig. 8. Three numerical solutions for Example 3.

presented in Fig. 6. The solving with the knowledge-based systems gives all these solutions. The resolution with the Newton–Raphson based method gives only the first solution of Fig. 6.

7.3. Example 3

The third example is the biggest, with 47 topological constraints and 34 dimensional constraints. The sketch is presented in Fig. 7 with points and arcs of circles names, but without lines, angles and dimensions names to simplify. The corresponding formal constraint system is in Table 6.

The formal solving time is 2.67 sec on our workstation. The numerical interpretation yields several solutions, three of them being shown in Fig. 8. The first and second drawings differ by the choice of the centre of the arc of circle on the left side in the figures. The third is valid but will certainly be eliminated by the user which does not accept a self-intersecting solution.

Larger examples with more than 200 constraints have been solved with our prototype. This experimentation can be found in [17,38].

8. Discussion

The different approaches in geometric construction based on constraints can be roughly classified into two main categories, depending on whether they disregard or not the invariance under displacements. The approaches of the first category have often been described in the literature. Thus we content ourselves to summarily classify them into:

- methods using geometric knowledge-based systems, like in CAD the work of Aldefeld [1,2] or Suzuki et al. [53], and in CAI the prototype *Progé* for ruler and compass constructions [18,47,48] and the expert system of polyhedra generation described in [37];

- methods founded on computer algebra, like the Maple-toolbox called *Lt-Geol* [19] and the implementation attempt in [14] of Lebesgue's method [32];
- methods using numerical iterations, such as the Newton–Raphson one [35,39], the method of linear approximation of [9] used in the *Viking* system [44] or the method based on homotopies of [30];
- methods based on propagation of degrees of freedom restriction in a constraint graph or in a collection of objects, like in the picture editors *Sketchpad* [52], *ThingLab* [6], *PictureEditor* [27], *Juno*-1 or *Juno*-2 [25]. These methods are often completed by numerical ones for the case of failure of the propagation process. Note that *Juno*-2 uses sophisticated techniques to localize, unpack, repack and separate constraints to be solved by the Newton–Raphson method [25].

The approaches of the second category, which use displacements, are more or less closely related to the theory we have outlined in Sections 3 and 4. They can use the methods of the first category locally and assemble partial solutions thanks to common references and displacements. In fact, all known approaches but ours use a unique local method, which is elementary or is based on numerical or formal calculus. Moreover, the notion of border, which is essential to our framework, is often reduced to metric constraints on common references in these approaches, which limits the resolution power. Here we briefly review them.

The approach of Owen [41] consists in describing a constraint system by a constraint graph. The vertices are geometric objects of degrees of freedom equal to 2, more precisely points, lines and circles of given radius. The edges are binary constraints. That restricts the geometric universe, for the objects as well as for the constraints. Thus, circles necessarily have a known radius and the constraints are so simple that the egality of two lengths cannot be taken into account. The graph is decomposed in a top-down way, searching the pairs of articulation and adding *virtual bonds* if necessary, until tri-connected components, triangles or isolated edges can be obtained. The local solving is limited to triangles and isolated edges and fails if there are tri-connected components. The process of assembling is the inverse operation of the decomposition. Simple tri-connected graphs are not solved by this method, though they can be with ruler and compass. For instance, the problem of constructing a triangle which is defined by its three perpendicular heights can be translated into the formalism of Owen, but is not solvable by his method. The class of solvable problems is therefore a strict subset of the constructions with ruler and compass.

Lamure and Michelucci [30] also follow an approach by graph decomposition. They use homotopies to numerically realize the processes of assembling. Their method is more general that the previous one, for the geometric universe as well as for the class of solved problems. Although the homotopy gives a certain stability to the terminal solving method, contrary to the Newton–Raphson method, this approach has the usual drawbacks of the numerical iterative ones.

The approach of Bouma et al. [7] consists in solving subsystems into subfigures, or *clusters*, with a constraint graph in a bottom-up way. The figures are then combined with the help of assembling templates and displacements. The templates are given a priori, but their number can easily be extended. They include a triangle configuration, such as in the approach of Sunde, and more complex systems, which are triangulated by a mechanism using Gröbner bases. The local solving method is very simple. It solves triangular systems by a method of loci which is directly translated into algebraic equations. The emphasis is therefore put on the method of assembling.

Our approach, in common with that of Bouma et al., employs the bottom-up strategy and the use of displacements, but has other major differences. First, it solves non-triangular systems with different local methods which are as powerful as possible. That is feasible by an exploitation of borders of any complexity. Second, its assembling method, founded on the use of common references, is very simple. Third, our approach is formal, while that of Bouma et al. stays numeric.

The approach of Sunde [50,51], extended by Verroust et al. [54,55], corresponds to a bottom-up strategy on a graph. The vertices are points or segments and the edges are constraints of distances and of angles. The graph is progressively made rigid, starting from the edges and using production rules to assemble two subfigures. In this extreme approach, where the rules do most of the work, the local solving method is reduced to its simplest expression. Because of the use of ad hoc data structures to manage the different kinds of constraints, mainly distances and angles constraints by *CD-sets* and *CA-sets*, this interesting approach can only treat a rather restricted geometric universe.

All these methods have the drawback of encoding a geometric problem in the form of a constraint graph, which is not general and imposes restrictions on the geometric universe. Thus, the fact that circles must have fixed radii is not an usual CAD constraint, but it is present only to satisfy constraints which are imposed by the graph structure.

To conclude this discussion, it seems the more local solving methods are advanced, the less the assembling methods need to be developed, and vice versa. We prefer a simple assembling allowing the integration of complex methods. This allows us to increase the solving power or to dedicate it to a specialized domain, without having to reconsider the heart of the system, which is based on an immutable assembling. To do the same, the alternative method needs to define new assembling templates or to dedicate them to applicative domains, which seems more difficult.

9. Conclusion

We have presented a mathematical framework to formalize the solving of constraint systems which are well-constrained modulo the displacements in a geometric universe. From this study we have developed a formal solving approach for the CAD area by local solving, displacement and assembling. We have indicated how constraints of reference, which are added for a local solving, are relaxed for a displacement. We have studied the essential role of the border of subplans in the achievement of the constructions. Finally, we have shown how this approach can be used in a prototype integrated into a topological-based modeller.

Classical geometry and logic, which are the foundations of our approach, allow us to express the different notions simply and to directly prove the correctness and completeness of the solving by assembling. Because no transformation in terms of graphs is necessary, this framework allows us to work in any geometric universe. Moreover, it can integrate any of the solving methods proposed in the literature and can easily extend its solving scope without reconsidering the principle of assembling. Thus, although formal, our approach can also use numerical methods.

One can ask about the performances of software based on this approach when dealing with specialized problems. For such applications, an unadapted method could lead to frequent and unnecessary displacements. Therefore, the choice of a good solving method, when included in our system, could be due to a more sophisticated heuristic than our simple scale of priorities.

In the future, we will expand the notion of coordinate system—or reference, in order to extend the possibilities of assembling in 2D and prepare the transition to 3D. We hope that this daunting step will be simplified with the help of our assembling techniques. Less specifically, our framework seems to be convenient for other universes, geometric or not, where an algebraic group of transformations more general than the displacements group acts.

Our prototype must be enriched by new methods, solving strategies, heuristics for selecting methods and clever strategies to place the references. Besides, the mode of triggering and the use of the methods can be revised in a parallel and cooperative strategy. Moereover, the under-constrained cases must be taken in account either by automatic completion of the constraint system to make it well-constrained, as in [2], or by allowing the definition of articulated systems, as in [28].

Our prototype will need to be better integrated in the *Topofil* modeller [4,5] than it is today. Thus, our 2D frame must be extended to take advantage of the 3D facilities of *Topofil*. Moreover, in the light of our experiment, a more convivial user interface can be designed and realized. For instance, the constraints can be better visualized, entered or modified, especially by taking into account implicit ones [2].

A final, but natural extension of such a modeller will consist in managing a rich parametrization of geometric objects, particularly in terms of our construction plans. An application would then be the building of objects with the same topology, but with different forms and dimensions. Another one, as in [28,29], would be the generation of animated sequences by a step by step progression of one of the parameter values.

References

[1] B. Aldefeld, Variations of geometries based on a geometric-reasoning method, Computer-Aided Design 20 (3) (1988) 117–126.

[2] B. Aldefeld, H. Malberg, H. Richter, K. Voss. Rule-based variational geometry in Computer-Aided Design, in: D.T. Pham (Ed.), Artificial Intelligence in Design, Springer, Berlin, 1992, pp. 27–46.

[3] Y. Baulac, Un micro-monde de géométrie—Cabri-Géomètre, Ph.D. Thesis, Université J. Fourier, Grenoble, 1990.

[4] Y. Bertrand, J.-F. Dufourd, Algebraic specification of a 3D-modeller based on hypermaps, Computer Vision—GMIP 56 (1) (1994) 29–60.

[5] Y. Bertrand, J.-F. Dufourd, J. Françon, P. Lienhardt, Algebraic specification and development in geometric modeling, in: Proceedings TAPSOFT Conference, Lecture Notes in Computer Science, Vol. 668, Springer, Berlin, 1993, pp. 75–89.

[6] A. Borning, R. Duisberg, Constraint-based tools for building user interfaces, ACM Trans. Graphics 5 (4) (1986) 345–374.

[7] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, R. Paige, A geometric constraint solver, Computer-Aided Design 27 (6) (1995) 487–501.

[8] B. Brüderlin, Constructing three-dimensional geometric objects defined by constraints, in: Proceedings ACM Siggraph Workshop in Interactive 3D Graphics, 1986, pp. 111–129.
[9] L.G. Bullard, L.T. Biegler, Lp strategies for constraint simulation, in: Proceedings AIChE Conference, 1989, pp. 69–84.
[10] A. Bundy, Computer Modelling of Mathematical Reasoning, Academic Press, London, 1983.
[11] M. Buthion, Un programme qui résout formellement des problèmes de constructions géométriques, RAIRO Informatique 13 (1) (1979) 73–106.
[12] J.-C. Carega, Théorie des Corps—La Règle et le Compas, Hermann, Paris, 1989.
[13] N. Carver, V. Lesser, Evolution of blackboard architectures, Expert Systems with Applications 7 (1994) 1–30.
[14] G. Chen, Les constructions à la règle et au compas par une méthode algébrique, Tech. Rept., Rapport de DEA, Université Louis Pasteur, Strasbourg, France, 1992.
[15] S.C. Chou, Mechanical Geometry Theorem Proving, Mathematics and its Applications, Reidel, Dordrecht, 1988.
[16] J.-F. Dufourd, P. Mathis, P. Schreck, Constructions géométriques dans un modeleur à base topologique, Revue Internationale de CFAO et d'Informatique Graphique 10 (4) (1995) 398–411.
[17] J.-F. Dufourd, P. Mathis, P. Schreck, Formal resolution of geometrical constraint systems by assembling, in: Proceedings 4th ACM-Siggraph Solid Modeling and Applications, Atlanta, GA, ACM Press, New York, 1997, pp. 271–284.
[18] J.-F. Dufourd, P. Schreck, Un système à base de connaissances pour les constructions géométriques, in: Actes de la Conférence AFCET-RFIA, 1994, pp. 351–361.
[19] L.W. Ericson, Lt-Geol 1.0—a Maple package for constrained planar euclidean geometric structures, Tech. Rept. 101, INRIA, Rocquencourt, 1988.
[20] L.W. Ericson, C.K. Yap, The design of Linetool, a geometric editor, in: Proceedings Computational Geometry Conference, Lecture Notes in Computer Science, Vol. 333, Springer, Berlin, 1988, pp. 83–92.
[21] N. Fuller, P. Prusinkiewicz, Geometric modeling with Euclidean constructions, in: Proceedings Computer Graphics International Conference, Springer, Berlin, 1988, pp. 379–391.
[22] H. Gelernter, Realization of a geometry-theorem proving machine, in: E.A. Feigenbaum, J.A. Feldman (Eds.), Computers and Thought, McGraw-Hill, New York, 1963, pp. 134–152.
[23] Y. Lafont, J.-Y. Girard, P. Taylor, Proofs and types, in: Tracts in TCS, Vol. 7, Cambridge University Press, Cambridge, 1989.
[24] J.A. Goguen, Modular specification of some basic geometrical constructions, in: Computational Computer Geometry (special issue), Artificial Intelligence 37 (1987) 123–153.
[25] A. Heydon, G. Nelson, The Juno-2 constraint-based drawing editor, Tech. Rept. SRC-131a, Digital, System Research Center, 1994.
[26] N. Jackiw, The Geometer's Sketchpad, Visual Project, Key Curriculum Press, San Francisco, CA, 1992.
[27] N. Kin, T. Noma, T.L. Kunii, PictureEditor: a 2D picture editing system based on geometric constructions and constraints, in: Proceedings Computer Graphics International Conference, Springer, Berlin, 1989, pp. 83–92.
[28] G.A. Kramer, Geometric reasoning in the kinematic analysis of mechanisms, Ph.D. Thesis, University of Sussex, 1990.
[29] G.A. Kramer, Using degrees of freedom analysis to solve geometric constraint systems, in: Proceedings First ACM Symposium of Solid Modeling and CAD/CAM Applications, ACM Press, New York, 1991, pp. 371–378.
[30] H. Lamure, D. Michelucci, Decomposition of 2D constraints graphs, Tech. Rept., Ecole des Mines, Saint-Etienne, 1995.
[31] S. Lang, Algebra, Addison-Wesley, Reading, MA, 1965.
[32] H. Lebesgue, Leçons sur les Constructions Géométriques, Gauthier-Villars, Paris, 1950.
[33] P. Lienhardt, Subdivisions of N-dimensional spaces and N-dimensional generalized maps, in: Proceedings 5th ACM Symposium on Computational Geometry, 1989, pp. 228–236.
[34] P. Lienhardt, Topological models for boundary representation: a comparison with n-dimensional generalized maps, Computer-Aided Design 23 (1) (1991) 59–81.
[35] R. Light, D.C. Gossard, Modification of geometric models through variational geometry, Computer-Aided Design 14 (4) (1982) 209–214.

[36] M. Mäntylä, A note on the modeling space of Euler operators, Computer Vision—GMIP 26 (1) (1984) 45-60.
[37] D. Martin, P. Martin, An expert system for polyhedra modelling, in: Proceedings Eurographics Conference, North-Holland, Amsterdam, 1988, pp. 221-232.
[38] P. Mathis, Un système de résolution de contraintes par assemblage en modélisation géométrique, Ph.D. Thesis, Université de Strasbourg, 1997.
[39] J.L. Moris, Computational Methods in Elementary Numerical Analysis, Wiley, New York, 1983.
[40] T. Noma, T.L. Kunii, H. Enomoto, E. Aso, T. Yamamoto, Drawing input through geometrical constructions: specifications and applications, in: Proceedings Computer Graphics International Conference, Springer, Berlin, 1988, pp. 403–415.
[41] J. Owen, Algebraic solution for geometry from dimensional constraints, in: Proceedings 1st ACM Symposium of Solid Modeling and CAD/CAM Applications, ACM Press, New York, 1991, pp. 397-407.
[42] J. Petersen, Problèmes de Constructions Géométriques, J. Gabay, new ed., 1990.
[43] W.H. Press, S.A. Teukolsky, W.T Werling, B.P. Flannery, Numerical Recipes in CS, Cambridge University Press, Cambridge, 1992.
[44] D. Pugh, Using interactive sketch interpretation to design solid objects, Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA, 1993.
[45] D. Roller, F. Schonek, A. Verroust, Dimension driven geometry in CAD: a survey, Tech. Rept., LIENS, Ecole Normale Supérieure, Paris, 1988.
[46] P. Schreck, Automatisation des constructions géométriques à la règle et au compas, Ph.D. Thesis, Université de Strasbourg, 1993.
[47] P. Schreck, Implantation d'un système à base de connaissances pour les constructions géométriques, Revue d'Intelligence Artificielle 8 (3) (1994) 223-247.
[48] P. Schreck, A knowledge-based for solving geometric constructions problems, in: J.W. Brahan, G.E. Lasker (Eds.), Proceedings 7th International Conference on Systems Research, Informatics and Cybernetics, 1994, pp. 19–24.
[49] J.R. Shoenfield, Mathematical Logic, Addison-Wesley, Reading, MA, 1973.
[50] G. Sunde, A CAD system with declarative specification of shape, in: Proceedings IFIP WG 5.2 on Geometric Modeling, Rensselaerville, 1986.
[51] G. Sunde, Specification of shape by dimensions and other geometric constraints, in: Proceedings Eurographics Workshop on Intelligent CAD Systems, Noordwijkerhout, 1987.
[52] I.E. Sutherland, Sketchpad: a man-machine graphical communication system, in: Proceedings IFIP Spring Joint Computer Conference, 1963, pp. 329-346.
[53] H. Suzuki, H. Ando, F. Kimura, Geometric constraints and reasoning for geometrical cad systems, Computer and Graphics 14 (2) (1990) 211-224.
[54] A. Verroust, Etude de problèmes liés à la définition, la visualisation et l'animation d'objets complexes en informatique graphique, Thèse d'Etat, Université de Paris-Sud, Orsay, 1990.
[55] A. Verroust, F. Schonek, D. Roller, Oriented method for parametrized computer-aided design, Computer-Aided Design 24 (10) (1992) 531-535.
[56] W.T. Wu, Basic principles of mechanical theorem proving in elementary geometries, J. Symbolic Computation 4 (1984) 207-235.