A formalization of geometric constraint systems and their decomposition

Pascal Mathis, Simon E. B. Thierry

DOI 10.1007/s00165-009-0117-8
BCS © 2009
Formal Aspects of Computing (2010) 22: 129–151

Formal Aspects of Computing

A formalization of geometric constraint systems and their decomposition

Pascal Mathis[1,2] and Simon E. B. Thierry[1,2]

[1] Université de Strasbourg, Strasbourg, France
[2] LSIIT, UMR CNRS-UdS 7005, Strasbourg, France. E-mail: simon.thierry@unistra.fr

Abstract. For more than a decade, the trend in geometric constraint systems solving has been to use a geometric decomposition/recombination approach. These methods are generally grounded on the invariance of systems under rigid motions. In order to decompose further, other invariance groups (e.g., scalings) have recently been considered. Geometric decomposition is grounded on the possibility to replace a solved subsystem with a smaller system called *boundary*. This article shows the central property that justifies decomposition, without assuming specific types of constraints or invariance groups. The exact nature of the boundary system is given. This formalization brings out the elements of a general and modular implementation.

Keywords: Geometric constraints solving, Decomposition, Transformation groups, Parametric CAD, Formalization

1. Introduction

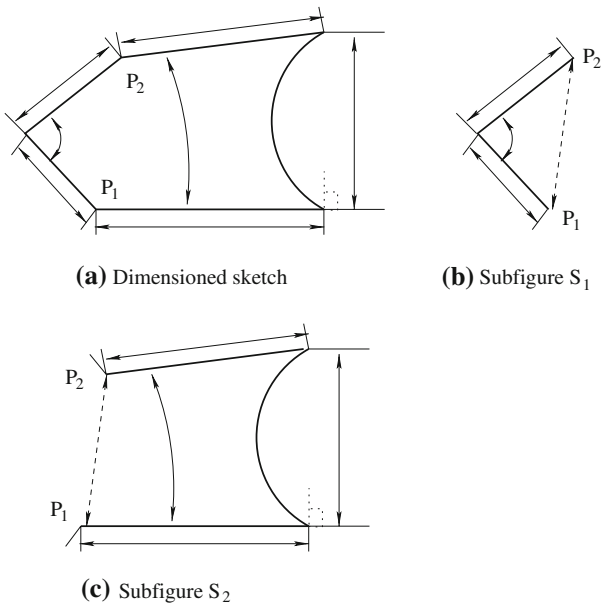
Solving geometrical 2D or 3D constraint systems is a key functionality in most CAD software. It aims at yielding a figure which meets some metric requirements (e.g., distances between points or angles between lines), usually specified under graphical form. Solutions are returned as coordinates of the geometric entities (points, lines, etc.), either directly or indirectly using a construction plan.

There is no general and complete method, but there exist many different approaches to solve the geometric constraints satisfaction problems. On one hand, through algebraic methods, the geometric nature of the problem is hidden and an equation system is solved, either with numerical methods [LM96a, LLG81] or with symbolic ones [Kon92, GC98] (which are not really used in CAD due to their exponential complexity). On the other hand, geometrical methods are based on well-known geometrical constructions. A locus method with some appropriate extensions [GHY02] is often applied and implemented as a graph analysis [LM96b] or a rule-based system [Ald88, Brü93, JAS97, Kra92, Sch94].

In contrary to the numerical methods, if a GCS has multiple solutions, geometric methods can often provide them.

For more than a decade, the trend to deal with large systems has been to make use of a divide and conquer approach [Sun86, VSR92, Owe91, DMS98, GLZ06, Sit06]. The different subproblems can be solved either by algebraic or geometric methods. The decomposition of geometric constraint systems has become an important part of a solver architecture, because it helps to reduce the complexity of the problem for algebraic solvers, and allows geometric solvers to deal with problems that they could not handle otherwise.

Correspondence and offprint requests to: S. E. B. Thierry, E-mail: simon.thierry@unistra.fr



(a) Dimensioned sketch **(b)** Subfigure S_1

(c) Subfigure S_2

Fig. 1. Decomposition for solving

This approach can be bottom-up: first, a decomposition is performed and then solvers (equational or geometric) yield solutions for subproblems. It can also be top-down: solvers cannot give solutions for the overall problem but can provide solutions for subproblems, then, subsolutions are combined. For a detailed study of existing decomposition methods, the reader may refer to [JTNM06].

In the dimensioned sketch of Fig. 1a, no basic geometric construction can help to solve the whole system. But, if one uses a decomposition/recombination mechanism, the constraints can be easily solved with some basic geometric constructions. First, the leftmost triangle, subsystem $\mathcal{S}_1$, is built (Fig. 1b) in a local coordinates system. Next, this triangle is removed from the system and replaced with the distance between P_1 and P_2 which can be computed in $\mathcal{S}_1$. Then, the remaining system, subsystem $\mathcal{S}_2$, shown in Fig. 1c can be geometrically built. And finally, $\mathcal{S}_1$ is added to $\mathcal{S}_2$ by relaxing what we call its local reference and computing the right transformation.

Thus, according to a solving strategy, some subsystems are computed and solved in local coordinates system. New information is brought to a remaining system and then subsystems are assembled. We call these new pieces of information the *boundary system*. *References* correspond to temporary constraints that must be relaxed to assemble subsystems.

This example illustrates the notions used in decomposition: subsystems, references, and boundary systems. As shown in [SM06], these notions are related to the *invariance group* of subsystems (rigid motions in 2D for the example above). Although most methods only consider transformations that are invariant under rigid motions, the use of several transformation groups allows to further decompose and thus enlarge the class of systems solvable by the means of geometrical methods. This is the idea underlying the work of [SS06] in 2D (with rigid motions and similarities) as well as the work of [vdMB08] in 3D (with rigid motions, similarities, and assemblies of systems invariant under scalings, called radial cluster).

The central result of this article is about the justification of the decomposition mechanism: if a system $\mathcal{S}$ is well-constrained *modulo* a group G, and if subsystem $\mathcal{S}'$, well-constrained *modulo* G', is solved, with $G \subset G'$, then by considering information from the boundary $\mathcal{B}$ of system $\mathcal{S}'$, the remaining system $\mathcal{S} - \mathcal{S}' + \mathcal{B}$ is well-constrained *modulo* G.

In order to get to this result, it is mandatory to separate the syntactic aspects of geometric constraint systems (set of terms) and the semantic ones where a geometric constraint system is considered as a set of geometrical figures.

Usually, decomposition algorithms are presented for a particular implementation (types of the geometric entities, types of constraints). One or several decomposition schemes are shown. Yet, in order to reach the general

justification mentioned above, we give a semantic definition of the boundary. It is then necessary to show that it is constructible in the chosen implementation.

As a corollary, the formalization we propose by clearly separating syntax and semantics allows to bring out the elements and data types for a really modular implementation and add new types of constraints, objects, and solvers in a multi-solver and multi-group architecture. The paper gives hints for a modular implementation and outlines our implementation.

The rest of this paper is organized as follows: Sect. 2 exposes the geometrical framework, constraint systems and solutions (figures), Sect. 3 describes how transformation groups take place in constraint systems and gives simplifications that come from well-constrainedness of CAD problems, Sect. 4 presents the elements of a multi-solver architecture to get a modular implementation.

2. Syntax and semantics of geometric constraint systems

A drawing specification is a statement consisting in a set of properties that the drawing must fulfill. A general way to define a statement is to consider it as a set of first-order logic terms. These terms are associated with a geometric meaning. Thus, statements have two aspects: syntax and semantics. *Constraint systems* capture the syntactic aspect, whereas geometric *figures* express the semantic one.

In this section, we provide a formalization of geometric constraint systems. We first explain what is a geometrical universe (Sect. 2.1), then detail the syntactic (Sect. 2.2) and the semantic aspects (Sect. 2.3) of geometric constraint systems solving.

2.1. Geometrical universe

The framework for describing constraint systems and figures is a pair $(\Sigma, \mathcal{E})$, where Σ is a many-sorted signature and $\mathcal{E}$ the geometric model for interpreting the variables. Such a pair is a *geometrical universe*.

Example 1 Consider the signature below:

sort
angle, length, point, line, circle
predicates

dist_pp: point point length	# distance between two points
angle_ll: line line angle	# angle between two lines
center : circle point	# center of circle
tangency_cl : circle line	# line tangent to circle
on_pl : point line	# incidence of line and point

A model could be the Euclidean plane, $\mathbf{E_2}$, with the classical interpretations: this universe has the carrier $\mathbb{R}^2$ for sort `point`, the sort `line` is interpreted by the set $[0, \pi[\times\mathbb{R}^+$ for the angle from x-axis and distance from origin, and the sort circle by the set $\mathbb{R}^2 \times \mathbb{R}^{+*}$ for coordinates of the center and a non-null radius.

The predicate symbol `tangency_cl`, for instance, is mapped to the set:

$$\{(xc, yc, r), (a, d) \mid \text{distpl}\,(xc, yc, a, d) = r\}$$

where *distpl* is the distance from a point to a line.

Such a framework is often extended by predicate symbols related to the usual constraints encountered in CAD and the considered model to the Euclidean 3D space $\mathbf{E_3}$.

2.2. Constraint systems

2.2.1. Definitions

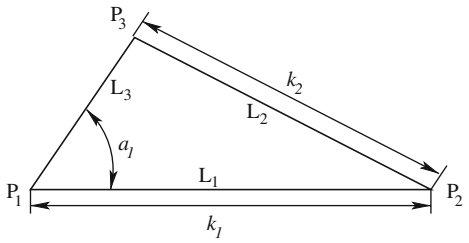
A constraint system is a conjunction of relational terms built on a set of sorted variables. Logical variables are either unknowns or parameters of the geometric statements.

Definition 2.1 constraint system: Let Σ be a many-sorted signature. A constraint system is a tuple $\mathcal{S} = (C, X, A)$, where C is a set of predicative terms built on Σ. X and A are two disjoint ordered sets of sorted variables of Σ, such that arguments appearing in C are either in $X \cup A$ or constants of Σ. X is the set of unknowns and A the set of parameters of the constraint system.

For a set of constraints C, *vars(C)* stands for the logical variables involved in C.

For the sake of understanding, unless mentionned otherwise, examples will be built on the signature of example 1 with 2D Euclidean space as carrier. Yet, results do not depend on this geometrical universe.

Example 2 The statement represented by the next figure is the constraint system below.




```
unknowns
   P1, P2, P3, L1, L2, L3
parameters
   k1, k2, a1
constraints
   on_pl(P1, L1)  on_pl(P2, L1)
   on_pl(P2, L2)  on_pl(P3, L2)
   on_pl(P3, L3)  on_pl(P1, L3)
   dist_pp(P1, P2, k1)
   dist_pp(P2, P3, k2)
   angle_ll(L1, L3, a1)
```

2.2.2. Operations on subsystems

Constraint systems express statements. In order to formalize decomposition principles, we describe subsystems, which are a key notion.

Definition 2.2 subsystem: Given a constraint system (C, X, A), a subsystem $\mathcal{S}_1$ of $\mathcal{S}$, denoted by $\mathcal{S}_1 \subset \mathcal{S}$, is a constraint system (C_1, X_1, A_1) where C_1 is a subset of C. X_1 and A_1 are the set of unknowns and parameters, respectively, appearing in C_1.

If two systems are based on the same signature, their addition or substraction corresponds, respectively, to union and difference of constraints sets.

Addition of constraint systems:

$(C_1, X_1, A_1) + (C_2, X_2, A_2) = (C_1 \cup C_2, X_1 \cup X_2, (A_1 \cup A_2) - (X_1 \cup X_2))$.

When a variable is an unknown in one system and a parameter in the other one, it is an unknown in the union.

Substraction (*precondition:* $(C_2, X_2, A_2) \subset (C_1, X_1, A_1)$):

$(C_1, X_1, A_1) - (C_2, X_2, A_2) = (C_1 - C_2, X_1 \cap vars(C_1 - C_2), A_1 \cap vars(C_1 - C_2))$.

2.3. Figures

2.3.1. Definitions

The notion of figure gives semantics to constraint systems. Given a universe $(\Sigma, \mathcal{E})$, a figure is a map where logic variables of a constraint system are mapped from Σ to $\mathcal{E}$.

Definition 2.3 figure: Given a universe $(\Sigma, \mathcal{E})$, for a map ρ from A to $\mathcal{E}$ (with respect to carriers of sorts), a *figure* of a constraint system $\mathcal{S} = (C, X, A)$ is a map $f_\rho : X \to \mathcal{E}$ such that all constraints in C are satisfied when interpreted in $\mathcal{E}$. The set of all figures of $\mathcal{S}$ according to ρ is denoted $F_\rho(\mathcal{S})$.

Thus, a figure can be seen as a vector of couples where the first member is an unknown and the second member is the set of coordinates. In what follows, among all possible values for parameters, we only consider those maximizing the dimension of the underlying variety, i.e., degenerate cases (with a distance equal to zero for example) will not be considered. For the sake of clarity in notation, if values of parameters are not important, the set of figures of system $\mathcal{S}$ will be denoted by $F(S)$ or simply F when no confusion can occur.

Restriction of a function allows to define the notion of *subfigure*.

Definition 2.4 restriction: Let be two figures $f : X \to \mathcal{E}$ and $f' : X' \to \mathcal{E}$ with $X' \subset X$ and such that for all $x \in X'$, $f(x) = f'(x)$. f' is the restriction of f to X, this is denoted by $f' = f|_{X'}$.
f' is also called a *subfigure* of f.

The notion of restriction can be slightly extended to sets X' which are not included in X by considering that $f|_{X'} = f|_{X' \cap X}$. Then of course, $f|_X = f$.

For a constraint system $\mathcal{S} = (C, X, A)$, the set of solutions F could be restricted to a subset X' of X : $F|_{X'} = \{f|_{X'} \mid f \in F\}$. Such a set contains only subfigures of F.

If $\mathcal{S}_1 = (C_1, X_1, A_1)$ is a subsystem of $\mathcal{S} = (C, X, A)$, we have $F(\mathcal{S})|_{X_1} \subseteq F(\mathcal{S}_1)$. Indeed, figures of $F(\mathcal{S})|_{X_1}$ meet all constraints of C_1 but could also meet constraints of C that are not in C_1 and that involve entities of X_1. This relation says that if it is possible to find solutions of a subsystem, then some subfigures may not be considered in the building of the solution set of the whole system.

2.3.2. Joint

A central operation is the *joint* of figures. This operation is useful for combining subfigures but can be done only if a compatibility condition holds.

Definition 2.5 figure compatibility: Two figures $f_1 : X_1 \to \mathcal{E}$ and $f_2 : X_2 \to \mathcal{E}$ with $X_e = X_1 \cap X_2$ are compatible iff $f_1|_{X_e} = f_2|_{X_e}$. This equivalence relation is denoted by $f_1 \equiv_{X_e} f_2$.

In other words, two figures are compatible if the unknowns they share are embedded in the same values. Two such figures can be joined. This semantic operation corresponds to the syntactic addition of systems.

Definition 2.6 joint of figures: Let $f_1 : X_1 \to \mathcal{E}$ and $f_2 : X_2 \to \mathcal{E}$ be two figures and $X_e = X_1 \cap X_2$ with $f_1 \equiv_{X_e} f_2$,

$$f_1 \otimes f_2(x) = \begin{cases} f_1(x) & \text{if } x \in X_1 \\ f_2(x) & \text{if } x \in X_2 \end{cases}$$

The restriction preserves the joint operation. If $f = f_1 \otimes f_2$, for any subset X of $X_1 \cup X_2$, $f|_X = f_1|_X \otimes f_2|_X$.
The joint operation is extended to sets of figures in the next definition.

Definition 2.7 joint of figure sets: Let be F_1 and F_2 two sets of figures, with F_i containing figures $f_i : X_i \to \mathcal{E}$. $F_1 \otimes F_2 = \{f = f_1 \otimes f_2 \mid f_1 \in F_1 \wedge f_2 \in F_2$ such that $f_1 \equiv_{X_e} f_2\}$, with $X_e = X_1 \cap X_2$.

Notice that if no couple of figures in F_1 and F_2 can be joined, $F_1 \otimes F_2$ is empty. This operation corresponds to combining subsystems after a decomposition process. One of the issues we deal with in this paper is to give a constructive way for joint operation.

The following result shows the close link between addition of systems and joining solutions of these systems.

Result 2.1 addition and joint: The solution set of a system $\mathcal{S}$ is the joint of any two subsystems solution sets, i.e., $F(\mathcal{S}_1 + \mathcal{S}_2) = F(\mathcal{S}_1) \otimes F(\mathcal{S}_2)$

Proof. Let $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ be a constraint system with $\mathcal{S} = (C, X, A)$, $\mathcal{S}_1 = (C_1, X_1, A_1)$, and $\mathcal{S}_2 = (C_2, X_2, A_2)$.

An element $f \in F(\mathcal{S})$ can be written as $f|_{X_1} \otimes f|_{X_2}$. We obviously have $f|_{X_1} \in F(\mathcal{S}_1)$ because $F(\mathcal{S})|_{X_1} \subseteq F(\mathcal{S}_1)$ and $f|_{X_1}$ is in $F(\mathcal{S})|_{X_1}$. As $f|_{X_2} \in F(\mathcal{S}_2)$, we have $F(\mathcal{S}) \subset F(\mathcal{S}_1) \otimes F(\mathcal{S}_2)$.

Let us now consider a figure $f \in F(\mathcal{S}_1) \otimes F(\mathcal{S}_2)$. It associates values to all elements in X. Assume that f is not in $F(\mathcal{S})$, so there exists a constraint $c \in C$ such that f makes c false. But c is in $\mathcal{S}_1$ or in $\mathcal{S}_2$ and f makes all constraints in these sets true. So the assumption that f is not in $F(\mathcal{S})$ is false and $f \in F(\mathcal{S})$.

Thus, we have the mutual inclusion: $F(\mathcal{S}) \subset F(\mathcal{S}_1) \otimes F(\mathcal{S}_2)$ and $F(\mathcal{S}_1) \otimes F(\mathcal{S}_2) \subset F(\mathcal{S})$. □

Example 3 In the Euclidean plane, Fig. 2 shows a constraint system $\mathcal{S}$ cut into two parts $\mathcal{S}_1$ and $\mathcal{S}_2$. The dotted line expresses symmetry in the top triangle. Infinite set $F(\mathcal{S}_2)$ contains all possible triangles $P_3P_4P_5$ where $\widehat{P_3P_4P_5} = a$ and segments P_3P_4 and P_4P_5 are of the same length. Infinite set $F(\mathcal{S}_1)$ contains all possible rectangles with dimensions k_1 and k_2. Figure f_3 is a solution of $\mathcal{S}$ and we have $f_3 = f_1 \otimes f_2$.

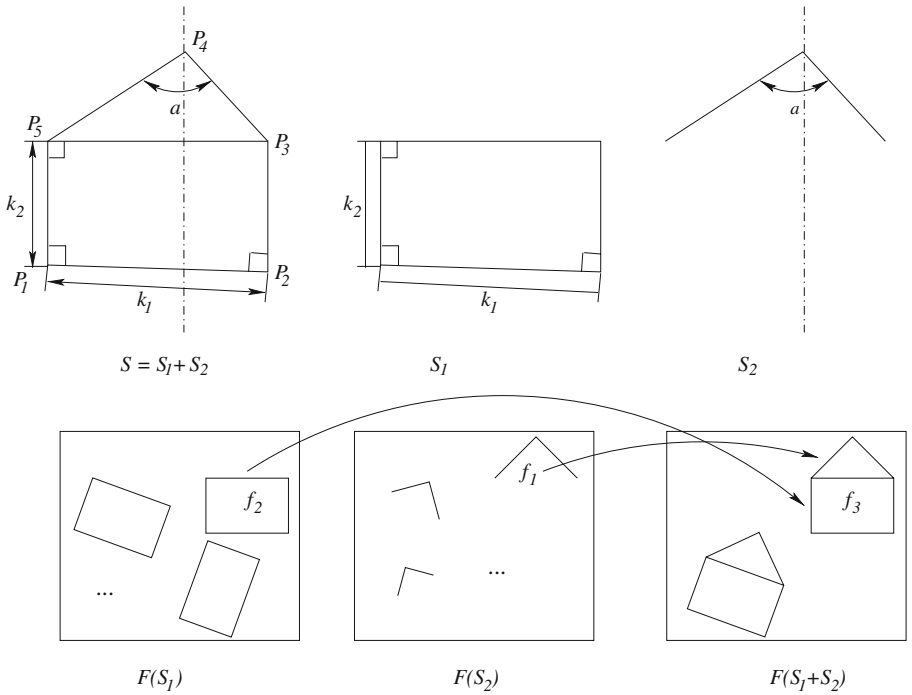


Fig. 2. Joint of two subfigures

The restriction does not preserve the joint operation for figure sets except in a specific case which is useful in decomposition.

Result 2.2 restriction and joint: For a system $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ with $X_e = X_1 \cap X_2$ where X_1 and X_2 are, respectively, unknowns of $\mathcal{S}_1$ and $\mathcal{S}_2$, for any $X \subset X_1 \cup X_2$, $F(\mathcal{S}_1 + \mathcal{S}_2)|_X = F(\mathcal{S}_1)|_X \otimes F(\mathcal{S}_2)|_X$ iff $X_e \subseteq X$.

Proof. Let be $f \in F(\mathcal{S}_1 + \mathcal{S}_2)$ such that $f = f_1 \otimes f_2$, with $f_1 \in F(\mathcal{S}_1)$ and $f_2 \in F(\mathcal{S}_2)$. We obviously have $f_1|_X \in F(\mathcal{S}_1)|_X$ and $f_2|_X \in F(\mathcal{S}_2)|_X$. Since we know that $f|_X = f_1|_X \otimes f_2|_X$, we can deduce that $F(\mathcal{S}_1 + \mathcal{S}_2)|_X \subset F(\mathcal{S}_1)|_X \otimes F(\mathcal{S}_2)|_X$. Notice that this relations stands even when $X \not\subset X_e$.

$F(\mathcal{S}_1)|_X \otimes F(\mathcal{S}_2)|_X \subset F(\mathcal{S}_1 + \mathcal{S}_2)|_X$ comes from the joint definition. Let be $f \in F(\mathcal{S}_1)|_X \otimes F(\mathcal{S}_2)|_X$ with $f = f_1 \otimes f_2$, $f_1 \in F(\mathcal{S}_1)|_X$ and $f_2 \in F(\mathcal{S}_2)|_X$, f is in $F(\mathcal{S}_1 + \mathcal{S}_2)|_X$ only if it fulfills the compatibility condition on X_e and that is the case if and only if $X_e \subseteq X$.

Thus, we have the mutual inclusion iff $X_e \subseteq X$. □

Example 4 In the previous example (Example 3), it is easy to see that the relation does not stand if $X_e \not\subset X$. Indeed, in Fig. 2, point P_1 is in $\mathcal{S}_1$ and P_4 belongs to $\mathcal{S}_2$. If $\mathcal{P}$ is the Euclidean plane, $F(\mathcal{S}_1)|_{\{P_1,P_4\}} = F(\mathcal{S}_1)|_{\{P_1\}} = \mathcal{P}$ and so it is for $\mathcal{S}_2$ and P_4. The joint of $F(\mathcal{S}_1)$ and $F(\mathcal{S}_2)$ leads to all couple of points of the plane, $F(\mathcal{S}_1)|_{\{P_1,P_4\}} \otimes F(\mathcal{S}_2)|_{\{P_1,P_4\}} = \mathcal{P}^2$, whereas $F(\mathcal{S})|_{\{P_1,P_4\}}$ contains all segments coming from projection of figures of $F(\mathcal{S})$.

2.4. Boundary system

In a system $\mathcal{S}$, a subsystem $\mathcal{S}_1$ has *inner* and *boundary* variables. Boundary variables are linked by constraints to variables of $\mathcal{S} - \mathcal{S}_1$, while inner variables are only connected to $\mathcal{S}_1$ variables. For instance, in Fig. 2 (Example 3), for subsystem $\mathcal{S}_1$, boundary variables are $\{P_3, P_5\}$ while inner variables are $\{P_1, P_2\}$.

Boundary variables were already encountered in compatibility relation. The system they induced plays a large part in decomposition of subsystems because the boundary system of a system $\mathcal{S}_1$, subsystem of $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ contains all the information needed to retrieve $F(\mathcal{S})|_{vars(\mathcal{S}_2)}$.

Definition 2.8 boundary system: Let $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ be a system with $\mathcal{S}_1 = (C_1, X_1, A_1)$ and $\mathcal{S}_2 = (C_2, X_2, A_2)$. The *boundary system* of $\mathcal{S}_1$ with respect to system $\mathcal{S}_2$ is the system $\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1) = (Ce, Xe, Ae)$ with $Xe = X_1 \cap X_2$, $Ae = A_1 \cap A_2$ and $F(\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1)) = F(\mathcal{S}_1)|_{(X_1 \cap X_2)}$.

To enhance clarity of the notations, when the border of a system $\mathcal{S}_1$ is computed with respect to a system $\mathcal{S}_2$ and when there is no ambiguity, $\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1)$ will be denoted simply by $\mathcal{B}(\mathcal{S}_1)$.

Note that the definition of the boundary system is semantical: there is no guarantee that the signature allows to express the corresponding contraints. This is discussed in Sect. 4.1.4.

The following result is essential for decomposition. It shows that removing a subsystem $\mathcal{S}_1$ from system $\mathcal{S}$ does not change the solutions of the remaining system if the boundary system of $\mathcal{S}_1$ is added. In other terms, it proves the validity of bottom-up decomposition methods: if the subsystem solvers are correct (i.e., yield only figures that satisfy the constraints) then the assembly of the subfigures will yield valid solutions.

Result 2.3 boundary system and solution set: Let $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ be a system with $\mathcal{S}_2 = (C_2, X_2, A_2)$. The restriction of $F(\mathcal{S})$ to variables of $\mathcal{S}_2$ is the solution of the system obtained by adding the boundary system of $\mathcal{S}_1$ to $\mathcal{S}_2$: $F(\mathcal{S})|_{X_2} = F(\mathcal{S}_2 + \mathcal{B}(\mathcal{S}_1))$

Proof. Consider a system $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$, with $\mathcal{S}_1 = (C_1, X_1, A_1)$ and $\mathcal{S}_2 = (C_2, X_2, A_2)$. Boundary variables of $\mathcal{S}_1$ are involved in constraints of $\mathcal{S}_1$ and constraints of $\mathcal{S}_2$, i.e., they are the set $X_e = X_1 \cap X_2$. We know that $F(\mathcal{S})|_{X_2} \subseteq F(\mathcal{S}_2)$, because some constraints applied on unknowns of X_e could be present in constraints of $\mathcal{S}_1$. So, subtracting $\mathcal{S}_1$ of $\mathcal{S}$ could remove constraints on X_2. The boundary system of $\mathcal{S}_1$ is the system such that $F(\mathcal{B}(\mathcal{S}_1)) = F(\mathcal{S}_1)|_{X_e}$. As $X_e \subset X_2$, from Sect. 2.2 we can deduce:

$$\begin{aligned} F(\mathcal{S})|_{X_2} &= F(\mathcal{S}_1)|_{X_2} \otimes F(\mathcal{S}_2)|_{X_2} \\ &= F(\mathcal{S}_1)|_{X_e} \otimes F(\mathcal{S}_2) \\ &= F(\mathcal{B}(\mathcal{S}_1) + \mathcal{S}_2) \end{aligned}$$

□

Example 5 In Example 3, X_2 is the set $\{P_3, P_4, P_5\}$ and $F(\mathcal{S}_2)$ contains all triangles whose two segments are of the same length and angle between them is fixed to parameter a. We can see that $F(\mathcal{S})|_{X_2}$ is the subset of $F(\mathcal{S}_2)$ where distance P_3P_5 of triangles is k_1. $F(\mathcal{S}_2)$ carries triangles that are not involved in any solutions.

The set X_1 is $\{P_1, P_2, P_3, P_5\}$. Boundary variables are $X_e = X_1 \cap X_2 = \{P_3, P_5\}$ and $F(\mathcal{B}(\mathcal{S}_1))$ contains all segments in Euclidean plane where length is k_1. Considering the signature of Example 1, this set can be syntactically expressed by the system $\mathcal{B}(\mathcal{S}_1) = (\{dist\_pp(P_3, P_5, k_1)\}, \{P_3, P_5\}, \{k_1\})$.

Thus, $\mathcal{S}_2 + \mathcal{B}(\mathcal{S}_1)$ restricts $\mathcal{S}_2$ to triangles where opposite segment of angle $\widehat{P_3P_4P_5}$ has value k_1. Notice that $F(\mathcal{B}(\mathcal{S}_2))$ is just all possible segments because boundary variables of $\mathcal{S}_2$ are $\{P_3, P_5\}$ and the distance between these two points is not set. Here, the relation is $F(\mathcal{S})|_{X_1} = F(\mathcal{B}(\mathcal{S}_2) + \mathcal{S}_1)$ but the boundary system $\mathcal{B}(\mathcal{S}_2)$ does not bring relevant information because $F(\mathcal{S})|_{X_1} = F(\mathcal{S}_1)$, meaning that removing $\mathcal{S}_2$ from $\mathcal{S}$ does not impact on X_1.

2.5. Decomposition

The decomposition of a constraint system relies on the "divide and conquer" strategy. The constraint system is split into subsystems that a solver can handle. The aim of the decomposition is the resolution of the global system more than a decrease of complexity. Nevertheless, as the complexity of the resolution of subsystems is the same as that of the global system, the total complexity of the resolution is only divided by a constant.

Definition 2.9 decomposition: A decomposition of a system $\mathcal{S}$ is a sequence $\mathcal{S}_1, \cdots, \mathcal{S}_n$ such that $F(\mathcal{S}) = F(\mathcal{S}_1 + \cdots + \mathcal{S}_n)$ and $\mathcal{S} \subset \mathcal{S}_1 + \cdots + \mathcal{S}_n$.

That is, decomposition is not only a partition of the constraints from $\mathcal{S}$ since new constraints can appear in subsystem $\mathcal{S}_i$. Yet, if there are additional constraints, they must be redundant with those from $\mathcal{S}$. Obviously, a decomposition is guided by semantics, because the interpretation of a sequence $\mathcal{S}_1 + \cdots + \mathcal{S}_n$ is the same as that of $\mathcal{S}$.

3. Invariance under the action of a transformation group

Geometric constraint systems are usually considered well designed when they are rigid and solvers often use the rigidity of a constraint system as an hypothesis. Yet, a solution of a rigid system can still be moved without violating any constraint. More generally, non-rigid systems can be seen as well designed if the user intended them to be scalable, for instance.

In this section, we formalize the notion of invariance under the action of a transformation group and extend the joint operation to a joint operation by transformation called transformation joint (Sect. 3.1). Then, we extend the classical notion of well-constrainedness to well-constrainedness *modulo*, a transformation group (Sect. 3.2).

3.1. Transformation groups

In CAD, solution set $F(\mathcal{S})$ is usually stable under some geometric transformations. Equivalence between figures with regard to some transformations naturally leads to consider group structure of transformations because action of group on figures induces an equivalence relation.

3.1.1. Group invariance, orbits

Definition 3.1 invariance by a group of transformations: A set of figures F is invariant by a group of transformations G (or G-invariant) if for any figure $f \in F$ and any transformation $\varphi \in G$, $\varphi(f) \in F$.

For a group G and a figure $f \in F$, the set $G.f = \{f' \mid \exists \varphi \in G, \varphi(f) = f'\}$ is the *orbit* of f. The set of orbits of F under the action of G form a partition of F. The associated equivalence relation states that f and f' are equivalent if there exists a transformation $\varphi \in G$ such that $f = \varphi(f')$. The orbits are the equivalence classes of this relation.

The set of all orbits of F under the action of G is written as F/G and $|F/G|$ denotes the number of orbits.

If $F(\mathcal{S})$ is invariant by groups G and G', it is invariant by $G \cap G'$ as well.

Example 6 Figure 3a shows the very simple constraint system of a triangle where length of all three sides are given. Given values for parameters k_1, k_2, and k_3, Fig. 3b–d represents the solution set $F(\mathcal{S})$ with different equivalence classes according to the transformation groups considered. In Fig. 3b, transformations are translations, thus number of orbits is infinite. In Fig. 3c, transformations are direct isometries or rigid motions, so $|F(\mathcal{S})/G| = 2$. In the last case, Fig. 3d, there is only one orbit, all solutions are equivalent *modulo* isometries.

By extension, a constraint system $\mathcal{S}$ is said to be G-invariant if $F(\mathcal{S})$ is so. This brings a semantic notion (invariance of figure sets) to the syntactic side (invariance of constraint systems).

When a constraint system is G-invariant, $F(\mathcal{S})$ can be characterized by a set of orbit representatives F_r containing one figure per orbit. In other words, $F(\mathcal{S}) = G.F_r$.

Example 7 In the previous example, the set of solutions can be defined by $F(\mathcal{S}) = G.F_r$ with G the group of rigid motions and F_r a set containing two figures, one from each orbit of $F(\mathcal{S})/G$. This description is not unique, e.g., $F(\mathcal{S})$ could also be expressed with G the isometries and F_r a set including a single figure meeting the constraints.

3.1.2. Joint and groups

Let us consider the cutting of a system $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2$ ($\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$), $\mathcal{S}_1 = (C_1, X_1, A_1)$ is $G1$-invariant and $\mathcal{S}_2 = (C_2, X_2, A_2)$ is $G2$-invariant, with $X_e = X_1 \cap X_2$ the set of common unknowns. $F(\mathcal{S}_1)$ and $F(\mathcal{S}_2)$ are described by a set of representatives, $F(\mathcal{S}_1) = G1.F_{r_1}$ and $F(\mathcal{S}_2) = G2.F_{r_2}$, respectively. Thus, each solution $f \in F(\mathcal{S})$ can be written as $f = \varphi_1(f_1) \otimes \varphi_2(f_2)$ with $f_1 \in F_{r_1}$, $f_2 \in F_{r_2}$, $\varphi_1 \in G1$ and $\varphi_2 \in G2$.

Given two representatives, $f_1 \in F_{r_1}$ and $f_2 \in F_{r_2}$, we note $(\Psi_1, \Psi_2)_{f_1,f_2} \subset G1 \times G2$ the set of couples (φ_1, φ_2) such that $\varphi_1(f_1) \equiv_{X_e} \varphi_2(f_2)$. So, each element $(\Psi_1, \Psi_2)_{f_1,f_2}$ allows to yield a solution of $\mathcal{S}$. We then have

$$F(\mathcal{S}) = \bigcup_{(f_1,f_2)\in F_{r_1}\times F_{r_2}} \{f \mid f = \varphi_1(f_1) \otimes \varphi_2(f_2) \wedge (\varphi_1, \varphi_2) \in (\Psi_1, \Psi_2)_{f_1,f_2}\}$$

In this notation, $G1$ and $G2$ are assumed.

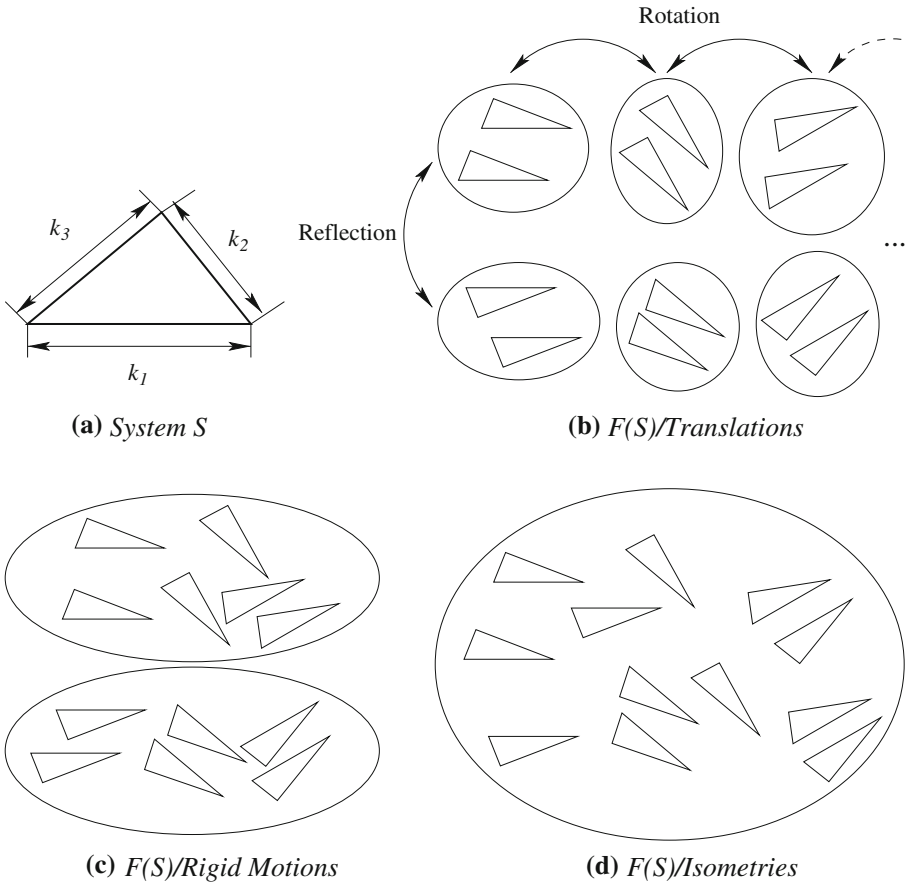


Fig. 3. Orbits of system $\mathcal{S}$ considering different transformations groups

From $G1$ and $G2$, it is not possible to state the invariance group of $F(\mathcal{S})$, because it also depends on variables shared by $\mathcal{S}_1$ and $\mathcal{S}_2$.

Let us consider $f = \varphi_1(f_1) \otimes \varphi_2(f_2)$, with X_e the set of common variables of f_1 and f_2. For each transformation $\varphi_1' \in G1$ such that $\varphi_1'(f|_{X_e}) = f|_{X_e}$, $\varphi_1(\varphi_1'(f_1)) \otimes \varphi_2(f_2)$ is also a solution.

Example 8 Figure 4a depicts a constraint system S made of two triangles corresponding to subsystems $\mathcal{S}_1$ and $\mathcal{S}_2$ as shown in the figure. These subsystems are invariant by rigid motions and share a common point, say P. Assume that these triangles are solved separately, f_1 is a representative solution of $F(\mathcal{S}_1)$ and f_2 a representative solution of $F(\mathcal{S}_2)$.

A solution f of $\mathcal{S}$ can be constructed by moving f_1 and f_2 so that P in each subfigure has the same coordinates. It amounts to find a couple (φ_1, φ_2) such that $f = \varphi_1(f_1) \otimes \varphi_2(f_2)$ with $\varphi_1(f_1)|_{\{P\}} = \varphi_1(f_2)|_{\{P\}}$ (see Fig. 4b). Any rigid motion φ_1' leaving $X_e = \{P\}$ unchanged (i.e., $\varphi_1'(f_1)|_{\{P\}} = f_1|_{\{P\}}$) leads to another solution $f' = \varphi_1\varphi_1'(f_1) \otimes \varphi_2(f_2)$ (see Fig. 4c).

This way of finding new solutions from a specific solution involves stabilization of common unknowns. As usual, we note $G_x = \{\varphi \,|\, \varphi(x) = x\}$ the stabilizer subgroup of x in G. If $f = \varphi_1(f_1) \otimes \varphi_2(f_2)$, then for any $g_1 \in G1_{f|_{X_e}}$ and $g_2 \in G2_{f|_{X_e}}$, we have $(g_1\varphi_1, g_2\varphi_2) \subset (\Psi_1, \Psi_2)_{f_1, f_2}$. Note that $G1_{f|_{X_e}}$ is isomorphic to $G1_{f_1|_{X_e}}$ because all stabilizers in a specific orbit are conjugate. This leads to say that, in the previous example, rotation around point P can be done either before or after applying φ_1.

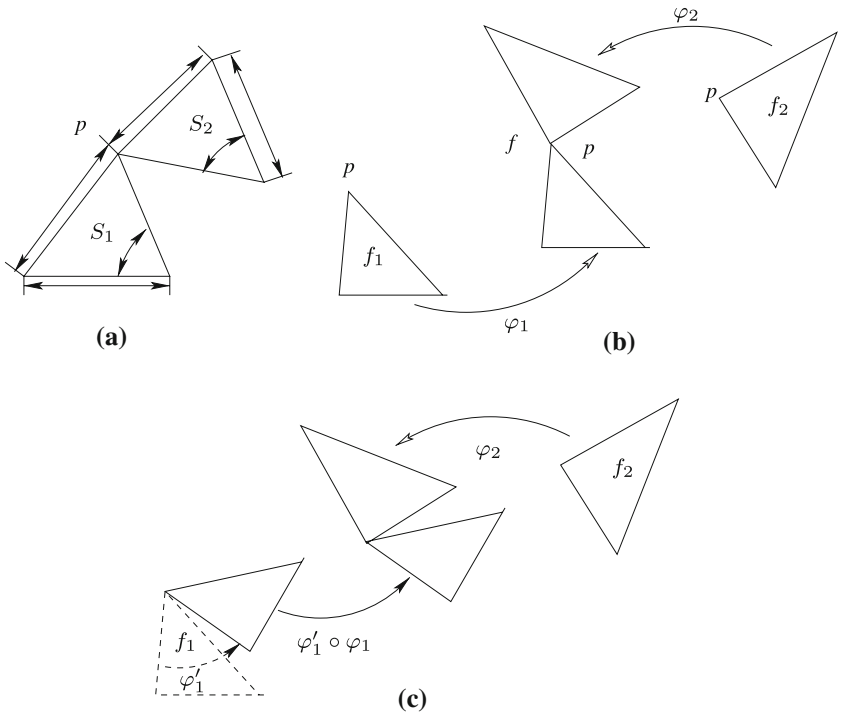


Fig. 4. Case of free rotation around a point

3.1.3. Bounded poset of transformation groups

More properties of joints come from a structure underlying the set of considered groups. In CAD, it is possible to gather transformation groups in a lower bounded *poset* (partial ordered set) structure. The set of groups is partially ordered by inclusion. In a lower bounded poset $(\mathcal{H}, \subseteq)$, there is a smallest element $G \in \mathcal{H}$ such that for any $G' \in \mathcal{H}$, we have $G \subset G'$. In the case of posets of groups, the trivial group made of the identity element can be this smallest element and bound the poset.

If we have $f \in (G1.f_1) \otimes (G2.f_2)$, we know that there exists (φ_1, φ_2) in $G1 \times G2$ such that $f|_{X_e} = \varphi_1(f_1|_{X_e}) = \varphi_2(f_2|_{Xe})$. If we also have $G2 \subseteq G1$, the transformation $\varphi_2^{-1}\varphi_1$ is in $G1$ and can therefore be applied on $f_1|_{Xe}$. Hence, we have:

$$\varphi_2^{-1}(f|_{X_e}) = f_2|_{X_e}$$

$$\varphi_2^{-1}\varphi_1(f_1|_{X_e}) = f_2|_{Xe}.$$

This situation is particularly interesting because it simplifies the expression of $(\Psi_1, \Psi_2)_{f_1,f_2}$. First, we consider a release of joint operation including transformation capability.

Definition 3.2 transformation joint: Given a geometrical universe and a transformation group G, the transformation joint of two figures f_1 and f_2 is the set:

$$f_1 \otimes_G f_2 = \{f \mid f = \varphi(f_1) \otimes f_2, \varphi \in G\}.$$

At first, one may think that this definition introduces an asymmetry because $f_1 \otimes_G f_2 \neq f_2 \otimes_G f_1$, but it is trivial to show that $f_1 \otimes_G f_2$ and $f_2 \otimes_G f_1$ are equivalent *modulo* G.

Next, we can see that the set of transformations $\varphi \in G$ involved in $\otimes_G$ operations is either empty (if f_1 and f_2 are not compatible) or isomorphic to $G_{f_1|_{X_e}}$ with X_e the common unknowns of f_1 and f_2. Indeed, let φ and φ' be two transformations of G such that $\varphi(f_1)|_{X_e} = f_2|_{X_e}$ and $\varphi'(f_1)|_{X_e} = f_2|_{X_e}$. As G is a group, there exists φ'' such

that $\varphi' = \varphi\varphi''$. $\varphi'(f_1)|_X = f_2|_X$ so $\varphi\varphi''(f_1)|_X = f_2|_X$. But $\varphi^{-1}(f_2)|_X = f_1|_X$ so $\varphi''(f_1)|_X = f_1|_X$. Thus, the set of φ in transformation joint is conjugate to $G1_{f_1|_{X_e}}$.

Assume that $\mathcal{S}$ is G-invariant and that a solver yields a G'-invariant substem $\mathcal{S}_1$ with $G \subseteq G'$. We know from Sect. 2.3 that the remaining system $\mathcal{S}_2 = \mathcal{S} - \mathcal{S}_1$ is generally not G-invariant, whereas $F(\mathcal{B}(\mathcal{S}_1) + \mathcal{S}_2)$ is since it is equal to $F(\mathcal{S})|_{X_2}$.

Result 3.1 joint in a bounded poset of groups: Let be a G-invariant system $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ with $F(\mathcal{S}_1) = G'.F_{r_1}$, $F(\mathcal{B}(\mathcal{S}_1) + \mathcal{S}_2) = G.F_{r_2}$ and $G \subseteq G'$. Stating that

$$F = \bigcup_{(f_{r_1}, f_{r_2}) \in F_{r_1} \times F_{r_2}} f_{r_1} \otimes_{G'} f_{r_2}$$

we have $F(\mathcal{S}) = G.F$

Proof. We argue by mutual inclusion.

$G.F \subseteq F(\mathcal{S})$ is the correctness aspect, i.e., the fact that the transformation joint operation does not yield wrong solutions (solutions that are not in $F(\mathcal{S})$). If we have $f \in F$, then $f = \varphi(f_{r_1}) \otimes f_{r_2}$ with $f_{r_1} \in F_{r_1}$ and $f_{r_2} \in F_{r_2}$. As $\varphi(f_{r_1})$ is in $F(\mathcal{S}_1)$, we have $f \in F(\mathcal{S})$. The inclusion is rather obvious, it comes from the definitions above.

$F(\mathcal{S}) \subseteq G.F$ is the completeness aspect, i.e., the fact that every solution is in $G.F$. Consider a figure of $F(\mathcal{S})$, $f = f_1 \otimes f_2$, and let f_{r_2} be the representative for the class containing f_2. There exists $\varphi \in G$ such that $\varphi(f_2) = f_{r_2}$. As $F(\mathcal{S})$ is G-invariant, $\varphi(f)$ is in $F(\mathcal{S})$. G contains ponctual transformations, hence $\varphi(f) = \varphi(f_1) \otimes \varphi(f_2)$. We have $\varphi(f_1) \in F(\mathcal{S}_1)$, so there exists a representative $f_{r_1} \in F_{r_1}$ for the class containing $\varphi(f_1)$. □

3.1.4. References

References are used to point out a specific figure in a set. First, recall that the action of group G is *free* if for any two different g and g' in G and any f in F, we have $g.f \neq g'.f$.

In an orbit of figures $G.f$, a specific figure f' can be identified by giving one of its subfigure $f'|_X$ as long as G acts freely on $G.f'|_X$. If not, more than one figure of $G.f$ can contain $f'|_X$.

For instance, consider a system $\mathcal{S}$ describing a 2D triangle ABC defined by the length of its three sides, it is invariant by $G =$ rigid motions for instance. Given coordinates for point A, there is an infinite number of figures satisfying the position of A, all in free rotation around A. In terms of transformations, this comes from the fact that rigid motions do not act freely on a point. If we take coordinates for point A and line (AB), only two figures meet the length requirements, each figure being the reflection about line AB of the other. But in any orbit of $F(\mathcal{S})/G$ (here two orbits), G acts freely on figures.

Definition 3.3 reference: A constraint system R such that group G freely acts on elements of $F(R)/G$ is a *G-reference*.

If R is a reference of G and $\mathcal{S}$ is a system such that $R \subset \mathcal{S}$, there is a one-to-one map between $F(R)$ and each elements of $F(\mathcal{S})/G$. A reference is generally not unique, for example, the whole system $\mathcal{S}$ is obviously a reference.

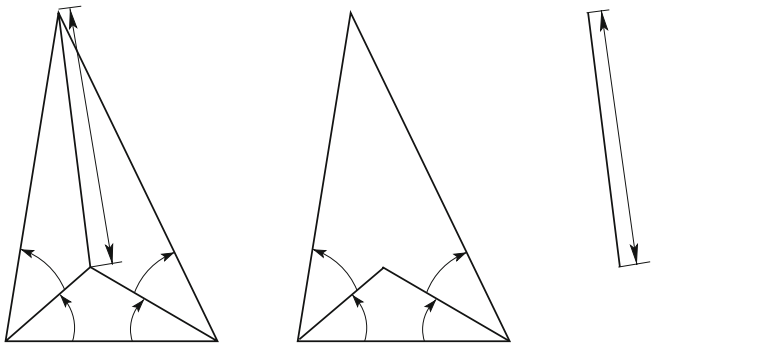
Knowing references for each group will be mainly useful in implementation to build specific figures (a representative of each orbit) and computing transformation by passing from a reference to another.

3.2. Well-constrainedness *modulo* a transformation group

In CAD, one usually assumes that constraint systems are rigid. Most solvers can only deal with such systems and fail otherwise. Rigid systems are thus said to be well-constrained. Yet, because these systems are invariant by rigid motions, there is an infinity of solutions built by applying rigid motions on a particular solution. Considering group invariance allows to alter the definition of well-constrainedness and solve this discrepancy.

3.2.1. Definitions

Definition 3.4 **G-well-constrained system:** A constraint system $\mathcal{S}$ is well-constrained *modulo* a group G (*G-wc* for short), if $|F(\mathcal{S})/G|$ is finite and greater than zero.



(a) Sketch **(b)** Scaling invariant part **(c)** Rigid motions invariant remaining part

Fig. 5. Decomposition under similarities and rigid motions

One can extend this definition to under-constrainedness: $\mathcal{S}$ is G-under-constrained if $|F(\mathcal{S})/G|$ is infinite; it is over-constrained if $|F(\mathcal{S})| = 0$, it is not group dependant. Obviously, a system $\mathcal{S}$ can be well-constrained *modulo* several groups. In Example 3, the system is well-constrained *modulo* rigid motions (two orbits) and also *modulo* isometries (one single orbit).

The definition induces some natural properties:

- for system $\mathcal{S} = (C, X, A)$ and $X' \subset X$, if $F(\mathcal{S})$ is G-wc then $F(\mathcal{S})|_{X'}$ is G-wc
- if $\mathcal{S}$ is G-wc and G'-wc, $\mathcal{S}$ is G''-wc with $G'' = G \cap G'$ and $|F(\mathcal{S})/G''| \leq |F(\mathcal{S})/G| * |F(\mathcal{S})/G'|$

Note that for any system $\mathcal{S}$, there always exists a group G such that $\mathcal{S}$ is G-wc: the group of permutations of the solution set. Of course, finding this group means knowing all solutions of $\mathcal{S}$ so this is not useful for solving or decomposition methods.

3.2.2. Well-constrainedness joint

Decomposition of G-well-constrained systems is based on Sect. 3.1. Let us take a G-well-constrained system $\mathcal{S}$. Suppose that a solver can only yield solutions for a G'-well-constrained subsystem $\mathcal{S}_1$ included in $\mathcal{S}$ such that group G' is taken in a bounded poset structure with G as least element, i.e., $G' \subseteq G$. Remaining subsystem is $\mathcal{S}_2 = \mathcal{S} - \mathcal{S}_1 + \mathcal{B}(\mathcal{S}_1)$ and it is G-well-constrained since its properties are the same as $\mathcal{S}$. We saw in Sect. 3.1 that $F(\mathcal{S}) = G.F$ with

$$F = \bigcup_{(f_{r_1}, f_{r_2}) \in F_{r_1} \times F_{r_2}} f_{r_1} \otimes_{G'} f_{r_2}$$

where F_{r_1} and F_{r_2} are respectively sets of representative for $F(\mathcal{S}_1)/G'$ and $F(\mathcal{S}_2)/G$. Given two representatives $f_{r_1} \in F_{r_1}$ and $f_{r_2} \in F_{r_2}$ with X_e common unknowns of f_{r_1} and f_{r_2}, $f_{r_1} \otimes_{G'} f_{r_2}$ is either empty or isomorphic to $G'_{f_{r_1}|_{X_e}}/G$. As $F(\mathcal{S})$ is G-well-constrained, $G'_{f_{r_1}|_{X_e}}/G$ is finite.

There are two cases to consider: either $G'_{f_{r_1}|_{X_e}}$ is finite or not. If $\mathcal{S}_1$ and $\mathcal{S}_2$ share a reference, according to the definition of a reference, $G'_{f_{r_1}|_{X_e}}$ is finite. If they share "less" than a reference $G'_{f_{r_1}|_{X_e}}$ could be infinite, i.e., an infinite number of transformations can be applied to join f_{r_1} and f_{r_2}. But in this case, as $\mathcal{S}$ is G-wc, $(G'_{f_{r_1}|_{X_e}})/G$ is finite and $X_2 = X_e$.

Example 9 First, we consider an example where two subsystems share a reference for both groups. Let us use the example of Fig. 5 (p. 140). This system is $\mathbb{D}$-wc ($\mathbb{D}$ being the group of rigid motions, also called $SE(n)$ in nD). If subsystem $\mathcal{S}_1$ is Fig. 5b, $F(\mathcal{S}_1)$ could be $\mathbb{S}.F_{r_1}$ where $\mathbb{S}$ is the group of similarities (rigid motions and scaling) and F_{r_1} the set carrying a single representative because $|F(\mathcal{S}_1)/\mathbb{S}| = 1$. If f_{r_1} is a solution for $\mathcal{S}_1$ and f_{r_2} a solution for

$\mathcal{S}_2$, there exists a single similarity transformation φ such that $\varphi(f_{r_2})|_{X_e} = f_{r_1}|_{X_e}$, where X_e are the two common points.

Then, let us take an example where common unknowns are "less" than a reference. Suppose that $\mathcal{S}_1$ is $\mathbb{D}$-wc and $\mathcal{S}_2$ is G-wc with G the group of the rotations around point P, global system $\mathcal{S}$ is G-wc. Also, suppose that $X_e = \{P\}$. Given two representatives f_{r_1} for $\mathcal{S}_1$ and f_{r_2} for $\mathcal{S}_2$, there is an infinite number of rigid motions φ such that $\varphi(f_{r_2})|_{X_e} = f_{r_1}|_{X_e}$. However, all the figures obtained by joining are equivalent *modulo* G as long as unknowns of $\mathcal{S}_2$ are only point P, otherwise, there is a free articulation of f_{r_1} and f_{r_2} around P and $\mathcal{S}$ is not G-wc. We see that among all possible rigid motions, any one can be chosen to join any f_{r_1} and f_{r_2}.

So, in well-constrained cases, joining subsystems involve computing a finite number of transformations. This computation is straightforward provided that two subsystems share a reference and that, for each group and each reference type, symbolic transformation from a reference to another one is given. If common subsystem is less than a reference, a random transformation can be chosen.

3.3. Decomposition under transformation groups

Generally, in the field of geometric constraints solving, a decomposition is to be understood with rigid motions in mind : solutions of a subsystem are invariant *modulo* rigid motions. With a multi-group approach, Definition 2.9 can be extended so that rigid motions are not implicitly considered.

Definition 3.5 decomposition under transformation groups: Given a set of transformations groups $\mathcal{G}$, a $\mathcal{G}$-decomposition of a constraint system $\mathcal{S}$ is a decomposition into $\mathcal{S}_1, \ldots, \mathcal{S}_n$ such that each $\mathcal{S}_i$ is G_i-invariant, with $G_i \in \mathcal{G}$.

This definition adds a semantic condition (invariance) to the syntactic decomposition. In a specific geometrical universe, there are a lot of possible decompositions for a system. In terms of solving complexity, they are not equivalent but, by definition, they all lead to the same solutions.

4. Modular implementation

The formalization given in this paper leads to a general and modular implementation. In order to achieve this, the notions presented in Sects. 2 and 3 must appear explicitly as elements of the implementation. After giving details about the main components, a classical decomposition/recomposition algorithm is described, where our operations on systems appear. We then show two commented examples and finally discuss time complexity of this algorithmic scheme.

4.1. Elements of implementation

4.1.1. Geometrical universe, constraint systems, and figures

The decomposition principles such as they were defined are not related to a particular geometrical universe. In order to keep genericity, the geometrical universe is not fixed but is a parameter of our architecture. The signature is described in a textual file. After the analysis of this file, constraint systems based on this signature can be entered. They appear as a conjunction of predicative terms.

Example 10 The left of the code shown in Appendix shows a description of a geometrical universe in a pseudo-XML format called GCML and described in [WSMF06]. In the following, this geometrical universe will be referred to as UG. A 2D constraint system corresponding to Fig. 6 is given on the right of the statement of Appendix.

Semantics (i.e., the geometrical representation of the sorts and the evaluation of the functional and predicative symbols) is also given in a textual way. Thus, the same signature can be interpreted in various ways. This aspect is further described in [WSMF06].

Constraint systems are tuples (C, X, A) together with their operations, such as described in Sect. 2.2. A figure is a map from the set of unknowns X to a semantic domain (usually 2D or 3D coordinates). These maps are parameterized by the set of parameters A.

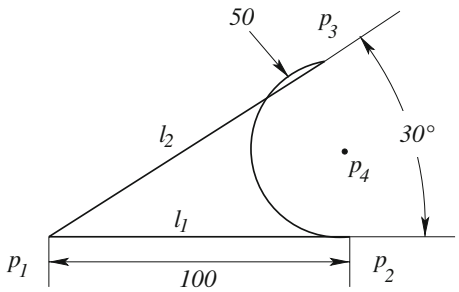
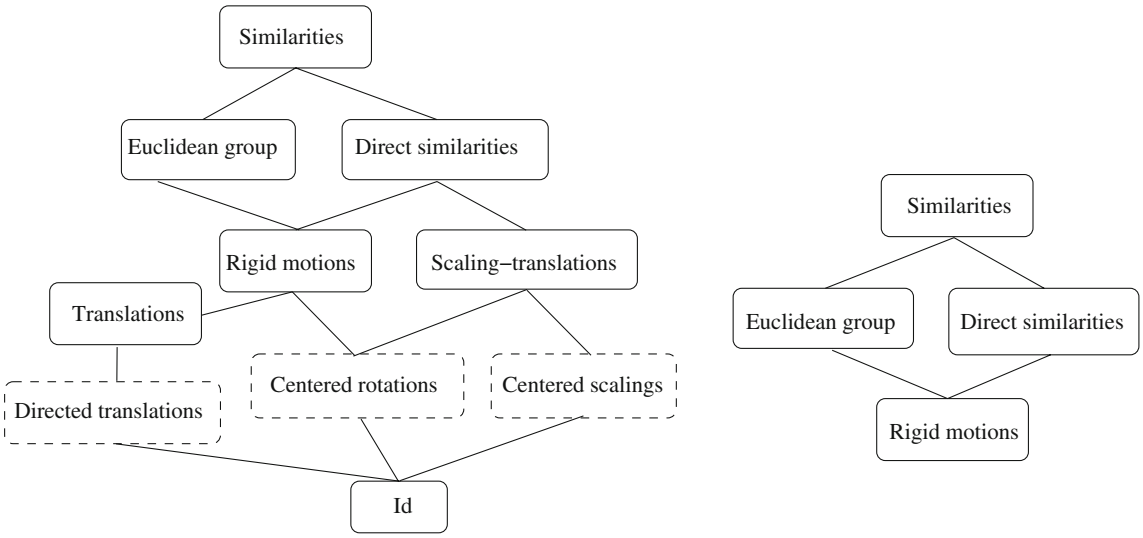


Fig. 6. Graphical representation of the constraint system of Appendix



(a) Group hierarchy **(b)** Down–set for rigid motions

Fig. 7. Ordered groups set

In the current implementation, two syntaxic signatures could be used. The first one allows to express 2D statements with points, lines, and circles for entities and distances, angles, incidences, and tangencies for constraints. The second one is used for 3D statements and also consider planes and spheres entities. These signatures are combined with semantics needed by solvers: dof semantic, 2D and 3D numerical semantics for real coordinates, and invariance group semantic.

4.1.2. Groups hierarchy

To a given semantics, a set of groups together with a partial-ordered relation is attached. Usually, transformation groups are subgroups of a more general invariance group and this set is lower bounded by the identity group. Figure 7a shows an example of an ordered set of groups for a classical 2D semantics and with similarities as most general group. In this figure, each box stands for a group while dotted boxes represent families of groups.

The poset associated with a semantics is a *hierarchy*. For a G-invariant constraint system with G belonging to the hierarchy, the lower bounded poset from which invariance subgroups are taken is a down-set of G, i.e., the subgraph where G is the least element. For instance, with the hierarchy of Fig. 7a, the decomposition of a constraint system invariant by rigid motions is made in the down-set starting from rigid motions, which is the lower bounded poset shown in Fig. 7b.

Table 1. Invariance groups attached to constraints types

c: Predicate (see Appendix)	**gr(c): Group**
on_pl, on_pc, center, tangency_cl	Similarities
dist_pp, radius	Euclidean group
angle_ll	Direct similarities
fix_p	Rotation around a given center
fix_l	Translation along a given direction

More groups could be added in the hierarchy of Fig. 7, such as glide reflections for instance. The choice of groups comes from the considered universe; it depends on the underlying geometry and on the type of constraints.

4.1.3. Invariance of systems

Each type of constraint is linked to an invariance group making well-constrained a minimal system containing such a constraint. For example, if `dist_pp` is the predicate for constraints of distance between two points with known length, the minimal system $\mathcal{S} = (\{\text{dist\_pp}(P_1, P_2, K)\}, \{P_1, P_2\}, \{K\})$ is considered for any P_1, P_2, and K assuming that P_1 is different from P_2. This system is well-constrained *modulo* 2D Euclidean group $\mathbb{D}$, so the latter is associated with predicate symbol dist_pp. This is denoted by $gr(dist\_pp) = \mathbb{D}$.

In a hierarchy, a unique group can be the well-constrainedness group for a type of constraint. If two groups $G1$ and $G2$ are appropriate, the group $G = \langle G1 \cup G2 \rangle$ (generated union) is added to the hierarchy. The group G becomes the invariance group for the considered type of constraints.

The hierarchy has to be closed for intersection. Indeed, assuming that a constraint system $\mathcal{S}$ is well-constrained under a group belonging to the hierarchy, the well-constriction group of $\mathcal{S} = (C, X, A)$ is given by $\bigcap_{c \in C} \text{gr(c)}$.

Table 1 gives invariance groups for the predicates of Example 10 with respect to the hierarchy of Fig. 7.

In order to compute global invariance group, our implementation includes the intersection for each couple of groups, explicitly given in a 2D array.

4.1.4. Boundary system computation

The boundary computation is made by exhaustive enumeration of constraint types. Assume that we need the boundary $\mathcal{B}(\mathcal{S}_1)$ for $\mathcal{S}_1 = (C_1, X_1, A_1)$ G-invariant and $\mathcal{S}_1$ is a subsystem of $\mathcal{S}$. First, the subsystem $\mathcal{S}_2 = \mathcal{S} - \mathcal{S}_1 = (C_2, X_2, A_2)$ is computed. The boundary entities are $X = X_1 \cap X_2$. Then, for every type of constraints invariant by G, all the possible metric constraints from X are generated.

For example, with G the group of rigid motions and if $X = \{p_1, p_2, p_3\}$ (all p_i being points), we add in $\mathcal{B}(\mathcal{S}_1)$ the distance contraints $dist\_pp(p_1, p_2, k_1)$, $dist\_pp(p_1, p_3, k_2)$, and $dist\_pp(p_2, p_3, k_3)$ with values for k_1, k_2, and k_3, respectively, computed in the solutions of system $\mathcal{S}_1$. The graph of groups inclusions show that the ratio constraints which are invariant by similarities are also invariant by rigid motions. So, the constraint $ratio(p_1, p_2, p_1, p_3, k_4)$ is added, k_4 being also computed from $\mathcal{S}_1$.

This saturation process generally makes $\mathcal{B}(\mathcal{S}_1)$ structurally over-constrained in the sense that there are too many constraints with regard to the number of entities. However, because many of these constraints are redundant, $\mathcal{B}(\mathcal{S}_1)$ is G-well-constrained as long as $\mathcal{S}_1$ is so. Besides, a high number of constraints increase the chances of success of geometrical solvers.

On the contrary, the geometrical universe does not always guarantee the possibility of expressing the boundary (see boundary system definition at Sect. 2.4). In the example of Sect. 4.3, without the constraint of ratio the boundary cannot be built.

4.1.5. Solvers

Solvers are algorithms which input a constraint system $\mathcal{S}$ and output a tuple $(\mathcal{S}', G, F_r)$ with $\mathcal{S}' \in \mathcal{S}$ and where $\mathcal{S}'$ is the subsystem the solver can deal with, and $G.F_r$ is the set of solutions of $\mathcal{S}'$. If the constraint set of $\mathcal{S}'$ is empty, the solver cannot solve any part of $\mathcal{S}$.

With this general definition of a solver, many algorithms, possibly using additional parameters, are considered as solvers:

- Location solvers: they use G-well-constrained systems as input and fix some elements to output systems which are well-constrained *modulo* identity (for instance, such a 2D solver for similarities pins two points down).

Table 2. Generic references for groups

Groups	References
Similarities	$(\emptyset, \{P_1, P_2\}, \emptyset)$ P_1, P_2 : point
Direct similarities	$(\emptyset, \{P, C\}, \emptyset)$ P : point, C : circle
Rigid motions	$(\{on\_pl(P, L)\}, \{L, P\}, \emptyset)$
Isometries (Euclidean group)	
Scaling-translations	$(\emptyset, \{L, P\}, \emptyset)$ P : point L : line and not $on\_pl(P, L)$
Translations	$(\emptyset, \{P\}, \emptyset)$ P is a point
Directed translations	
Rotations around center O	$(\{on\_pl(O, L)\}, \{L, O\}, \emptyset)$
Scaling of center O	$(\{center(O, C)\}, \{O, C\}, \emptyset)$ $(\emptyset, \{P\}, \emptyset)$ P is a point
Identity	$(\emptyset, \emptyset, \emptyset)$

- Classical solvers (knowledge-based solvers, graph analysis solvers, Newton-Raphson,. . .) which use a location solver as parameter.
- Boundary solvers: each boundary solver is specialized in its own transformation group and computes the boundary of a system $\mathcal{S}'$ with respect to system $\mathcal{S} - \mathcal{S}'$.
- Assembly solvers: the assembly process (see 4.1.4) can be seen as a particular solver which has, as a parameter, a strategy to choose solvers.

Currently, several solvers are implemented in our solving platform:

- Location solvers for each group in 2D and 3D.
- Two classical solvers: a knowledge-based system for rule-based geometric reasoning that can provide several solutions, and a numerical solver based on homotopy [LM96a] yielding the closest solution to the sketch. The latter is applied when the knowledge-based system failed.
- A boundary solver which computes constraints for boundary entities according to functional symbols of the signature and invariance group semantic.
- A general decomposition solver implementing algorithm of Sect. 4.2.

4.1.6. References

References are used in two ways : first, by location solvers (called by classical solvers) to build particular solutions, then, during the assembly process to calculate geometrical transformations. Indeed, recall that common elements share a reference for the higher group. So, the transformation is computed from the common reference of the two figures to assemble.

The geometrical universe does not always allow to express references exactly. For instance, with directed translations, a reference could be one coordinate of a point, but as the sort *coordinate* is not included in the signature, reference will be a point.

Sometimes, a *loose* definition for references can be considered and "less than" a reference is given. For example, a reference for rigid motions ($\mathbb{D}$) could be a point and an incident line. A reference for the Euclidean group ($\mathbb{D}^+$: $\mathbb{D}$ and symmetries) could be a point and an oriented incident line. If the universe does not contain the sort *oriented line*, reference for the Euclidean group can also be used because $|\mathbb{D}^+/\mathbb{D}| = 2$ is finite. In such a case, two transformation patterns must be given, one for each reference. The loose definition implies that G acts *finitely* transitively on figures.

These considerations lead to same reference for different groups. Table 2 shows generic references for the hierarchy of Fig. 7 with respect to the 2D universe of Appendix.

4.1.7. Decomposition strategy

The decomposition algorithm presented in the next section is based on a strategy that has to choose an invariance group and a solver at each iteration. The strategy is a parameter of the decomposition algorithm. At each step, if the previously chosen solver failed in solving, the strategy chooses a new solver and/or group.

In our implementation, to each group of the hierarchy corresponds a location solver which is called by classical solvers. The invariance group is computed by the following way. To each type of constraint of the geometrical

universe is associated the information of its greatest invariance group. This is of course a semantic information. For a given system, the global invariance group is the intersection of the invariance groups of each constraint.

Our strategy consists in attempting a geometric resolution for the global invariance group. In case of failure, smaller transformation groups are tried, using a breadthfirst traversal of the group hierarchy. At each step, the geometric rule-based system attempts to solve the system. If, finally, the system is not solved, the homotopic solver is called right after the location solver of the global invariance group.

When a group G is selected, the system supplied to the selected solver is the current system $\mathcal{S}$ where we get rid of all the constraints that are not G-invariant.

The solvers produce one or more particular solutions. For that, each solver starts by choosing a reference in the figure. For example, for rigid motions in 2D, a solver starts the construction by placing a point and the direction of a line passing through this point. For similarities, one will choose two points, etc.

Of course, other strategies could be used, and different strategies could lead to different decompositions. So, the central question here is: which strategy allows to discover all the solutions that solvers can produce with decomposition and assembly?

At first, one can make two comments. If a constraint system is solvable by the available solvers, an exhaustive strategy, that is to say one which tries all the groups with all the solvers for all references, will lead to the discovery of a solution. Then, it is clear that the process of joining is not involved in this question. Indeed, joint is correct because assembly does not give false solutions and complete because if all solutions are yielded for subsystems, the joint operation gives all solutions for the global system.

Answering the question of which strategy is the best is very difficult and requires the analysis of solvers abilities. It can be made for simple solvers but not with rule-based solvers where new rules can be added. The same is true of algebraic solvers that are sensitive to numerical instability problems.

Thus, in our implementation we use the heuristic consisting in going down in the group hierarchy from the greatest invariance group. It also consists in using the geometrical solvers first of all because they could supply more than a single solution—in contrary to the homotopy which is based on Newton-Raphson—and, to finish, elements establishing a reference are chosen among those involved in most constraints.

Note that the simplest strategy could be to always use the same solver.

4.2. Decomposition/recombination algorithm

The results of Sect. 3 lead to a bottom-up decomposition/recombination algorithm. The recombination of the subsystems is performed through a two-by-two assembly in the reverse order of the decomposition. Adding the boundary system at each decomposition step allows to have only one assembly rule: if two subsystems share at least a reference for their greatest invariance group, then they can be assembled according to the geometric entities of the reference (joint operation).

First, this section gives an algorithm to find a decomposition that allows a straightforward recombination in well-constrained cases and then presents the corresponding recombination algorithm.

4.2.1. Decomposition algorithm

This algorithm gives an expression of a constraint system into a stack of group invariant subsystems (considered as solved). Parameters are a constraint system and a strategy for the choice of the solvers. The way to assemble solutions of subsystems, in order to yield a solved overall system, is shown in Sect. 4.2.2.

```
Input : S : constraint system
        strategy : choice strategy for solvers
Output : sp : stack of solved constraint system: (solution representative, group)

boolean success = true;
stack sp = empty_stack
while strategy.newSolverPossible(S, success) and S ≠ (∅,∅,A)
do
  solve = strategy.chooseSolver(S)
  (S1, G, Fr) = solve(S)                    # S1 subsystem of S, F(S1) = G.Fr
  if S1 == (∅,∅,A) then                     # solving failed
    success = false
```

```
else
        push(sp, (Fr, G))                          # record solved subsystem
        S_2 = S - S_1                              # compute remaining system
        B_S_1 = boundary_system(S, S_1, G, F_r)    # compute boundary system
        S = S_2 + B_S_1                            # residual system to solve
   fi
done
if success == true return sp else return empty_stack
```

The strategy acts as described in Sect. 4.1.7. It contains a method for deciding whether the decomposition can go on or not and also a method providing the current solver. The boundary is taken into consideration in the decomposition algorithm: after removing a solved subsystem, the border of this subsystem is added to the remaining system. This means that there is redundant information in the systems: the border can be computed from one system and is explicitly given in another. Yet, because both systems are solved separately, this redundancy does not imply the over-constrainedness of the overall system.

For simplicity reasons, this algorithm does not include the verification that the border of the solved subsystem $\mathcal{B}(\mathcal{S}_1)$ is not equal to $\mathcal{S}_1$ itself. This verification is of course mandatory in a real implementation, otherwise the program may loop infinitely.

4.2.2. Joining algorithm

Once a system is decomposed (into a stack), the solutions, representatives are assembled by a transformation joint operation. Solved systems are popped out in the opposite order of resolution and then assembled.

```
Input :  sp : stack of solved constraint systems : (solution representative, group)
Output : Fr : solution representatives
         G : invariance group

while height(sp) > 1 do
   (Fr1, G1) = pop(sp)                  # get solutions of two subsystems
   (Fr2, G2) = pop(sp)                  # out of stack
   F = Fr2 ⊗_G2 Fr1                     # join them
   push(sp, (F, G1))                    # push result which is G1-invariant
done
```

4.3. Examples

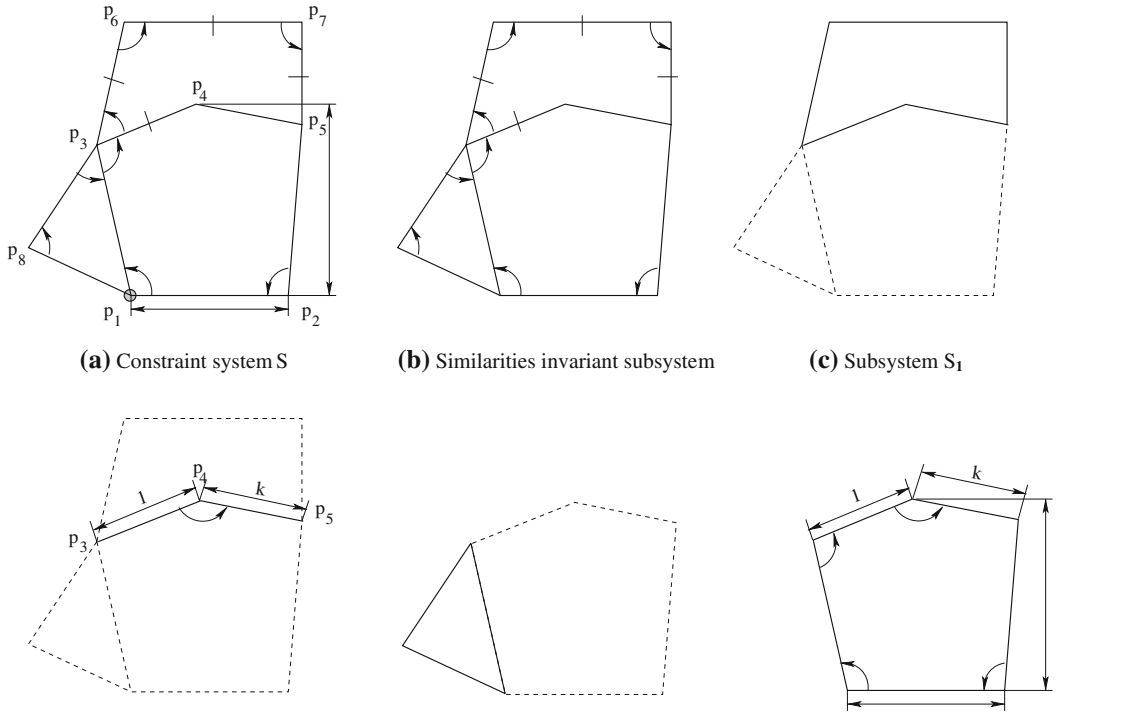
The two following examples were successfully solved using our multi-solver resolution platform based on decomposition. As was explained in Sect. 4.1.7, our strategy is based on location solvers for each transformation group, on a boundary system computation solver, on a rule-based geometric solver and on a numerical solver performing homotopy using Newton–Raphson iterations. The latter is not used in the examples below, because a geometric resolution is possible.

4.3.1. 2D Example

Figure 8a presents the statement of a constraint system based on the 2D geometrical universe UG. Recall that the types of constraints are: point-line incidence, distance between points, between a point and a line, angles, fixed point (point p_1), and distance ratio. The small lines across segments mean that these segments have all the same length, so that the ratio between any pair of these segments is 1.

By computing the intersection of the invariance groups of every type of constraints appearing in $\mathcal{S}$, we assume that the whole problem is invariant to $\mathcal{R}_{p_1}$, the group of rotations around point p_1. All considered groups are thus in the down-set extracted from Fig. 7 with $\mathcal{R}_{p_1}$ as lowest element. We assume that the system is $\mathcal{R}_{p_1}$-wc. If not, a decomposition will be made but the joint will not be done because the subsystems to be assembled will have "less than" a common reference.

The decomposition algorithm seen in Sect. 4.2 uses a strategy that first chooses the most general group. So, we first consider the subproblem containing only constraints invariant by similarities (Fig. 8c). A solver specialized in angle and ratio constraints is called. It succeeds in resolving a subsystem $\mathcal{S}_1$ made of points p_3, p_4, p_5, p_6, and p_7.



(a) Constraint system S **(b)** Similarities invariant subsystem **(c)** Subsystem S_1

(d) Boundary system from S_1 **(e)** Subsystem S_2 **(f)** Displacements invariant subsystem S_3

Fig. 8. Decomposition of a 2D system into three subsystems

The boundary of $\mathcal{S}_1$ in $\mathcal{S}$, system $\mathcal{B}(\mathcal{S}_1)$, includes points p_3, p_4, and p_5, which are the only ones connected by constraints to entities not in $\mathcal{S}_1$. System $\mathcal{B}(\mathcal{S}_1)$ then consists of these points to which are added an angle constraint and a ratio constraint. $\mathcal{B}(\mathcal{S}_1)$ is well-constrained *modulo* similarities. Notice that without the ratio type of constraint, UG was not complete and would not allow the expression of a well-constrained boundary. The boundary $\mathcal{B}(\mathcal{S}_1)$ is represented on Fig. 8d with the constraint of ratio indicated by stating that if length of segment p_3p_4 is 1, then length of segment p_4p_5 is k, with k computed in solutions of $\mathcal{S}_1$.

$\mathcal{S}_1$ is removed from current statement and $\mathcal{B}(\mathcal{S}_1)$ is added to the remaining of the system. The solving goes on with triangle p_1, p_3, and p_8 (system $\mathcal{S}_2$) which is well-constrained *modulo* similarities. The boundary $\mathcal{B}(\mathcal{S}_2)$ contains points p_1 and p_3. In UG, there is no constraints types involving two points and which would be similarity invariant. Thus, $\mathcal{B}(\mathcal{S}_2)$ contains no constraint.

No more similarity invariant subsystem can be calculated (apart from trivial ones reduced to a reference). The solving process continues for subgroups of the similarities. By going down in the groups, hierarchy, we can continue with rigid motions. The residual system $\mathcal{S}_3$ is shown in Fig. 8f. In the latter, only the constraint fixing point p_1 does not appear because it is obviously not invariant by rigid motions. A simple geometrical construction can resolve the remaining system.

The boundary $\mathcal{B}(\mathcal{S}_3)$ only contains point p_1. In UG, there is no type of constraint invariant by rigid motions and involving a single point, thus we have $\mathcal{B}(\mathcal{S}_3) = (\emptyset, \{p_1\}, \emptyset)$. No more construction considering the rigid motions is possible, so we go down in the hierarchy to group $\mathcal{R}_{p_1}$. The remaining system is $\mathcal{S}_4 = (\{fix\_p(p_1, 0., 0.\}, \{p_1\}, \emptyset)$. It is a reference for $\mathcal{R}_{p_1}$ and is trivially solved.

The joint is then performed on the resolved systems in the reverse order. The solved system $\mathcal{S}_3$ is assembled to the system $\mathcal{S}_4$ by computing the right rigid motion. Then, subsystem $\mathcal{S}_2$ and finally $\mathcal{S}_1$ are joined by similarities.

4.3.2. 3D example

The same principles apply to the resolution of 3D problems. The syntax of the geometrical universe is completed with sorts plane, spheres and with the corresponding angle, and tangencies constraints. Sorts and symbols of the signature are associated with a numerical 3D semantics.

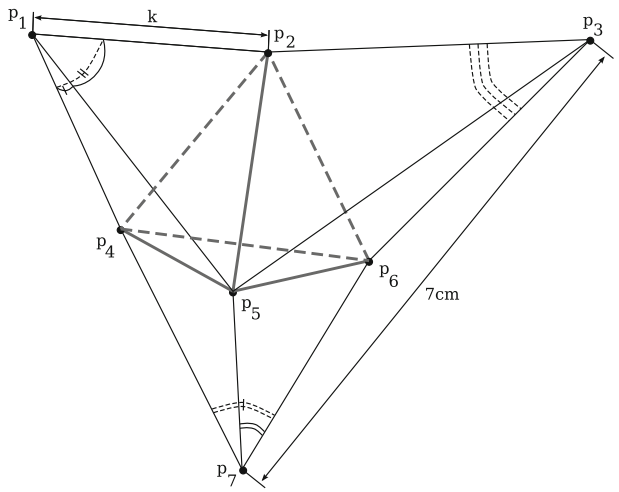


Fig. 9. Rigid 3D system decomposable in 4 scalable tetrahedrons and a rigid segment

Figure 9 shows a 3D statement with a single distance constraint, angle constraints, and distance ratio constraints that are not given in the figure. In order to help the reader, the edges of the central tetrahedron T_c were represented with thick lines. Dotted lines represent hidden edges and dotted curves represented angle constraints on hidden faces. Thin lines between two points represent distance ratio constraints. All ratios are expressed with respect to distance p_1p_2, we thus represent a distance constraint with a non-fixed value k and hereafter express all ratios with respect to k.

The statement contains a central tetrahedron T_c made up with points $p_2p_4p_5p_6$. Note that no explicit constraint is given upon T_c. Yet, it is linked to three other tetrahedrons: $T_{p_1} : p_1p_2p_4p_5$, $T_{p_7} : p_4p_5p_6p_7$, and $T_{p_3} : p_2p_3p_5p_6$. The constraints on each tetrahedron are angle constraints and distance ratios. The numerical values of these constraints are not relevant here. The constrained metrics are:

- $T_{p_1} : \widehat{p_2p_1p_5}$, $\widehat{p_2p_1p_4}$, $\widehat{p_5p_1p_4}$, $\frac{p_1p_5}{k}$, $\frac{p_1p_4}{k}$,
- $T_{p_7} : \widehat{p_6p_7p_5}$, $\widehat{p_6p_7p_4}$, $\frac{p_6p_7}{k}$, $\frac{p_5p_7}{k}$, $\frac{p_4p_7}{k}$, and
- $T_{p_3} : \widehat{p_2p_3p_6}$, $\frac{p_2p_3}{k}$, $\frac{p_3p_5}{k}$, $\frac{p_3p_6}{k}$

Tetrahedrons T_{p_1} and T_{p_7} are well-constrained *modulo* similarities. Tetrahedron T_{p_1} is well-constrained *modulo* similarities. The addition of its boundary system adds the angles of the triangle $p_2p_4p_5$ and the ratios between k and distances p_2p_4, p_2p_5, and p_4p_5. With these new pieces of information, T_{p_7} becomes well-constrained *modulo* similarities and can thus be solved. Adding its boundary adds, among other things, the ratio between k and distance p_5p_6. T_{p_3} + T_c is then solvable *modulo* similarities.

No similarity invariant constraints remain, so the strategy chooses to consider rigid motions. Segment p_3p_7 is constructed. The rest of the system is well-constrained *modulo* similarities and it is thus possible to compute the similarity transformation such that the distance p_3p_7 satisfies the distance constraint.

4.4. Time complexity

The complexity of the decomposition/recombination scheme can be computed from the complexity of the decomposition/recombination algorithms, the complexity of the solvers (classical and location) and the complexity of the boundary computation.

The complexity of the decomposition algorithm is as follows. In case of failure, each solver is tried for each transformation group. In case of success, in the worst case, trying every solver leads iteratively to the construction of one element (point, line,...). The recombination consists in searching for the common references of the subsystems, following the order of the decomposition stack. These operations all have a polynomial complexity.

Location solvers, which temporarily pin down geometric entities, proceed in a linear time. As to the classical solvers, their complexity may vary according to their algorithm, but most of them have a polynomial complexity. This is the case of both solvers used in our implementation: a homotopic solver based on Newton-Raphson and a rule-based solver.

The weak point in terms of complexity is the boundary computation. Indeed, exhaustively producing all the constraints allowed by the signature between the boundary elements, according to the invariance group of the system, is a process with an exponential time complexity. For 2D problems, the boundary is often reduced to less than four elements and the exponential complexity is then not a problem. However, for 3D problems where the number of boundary entities can easily reach more than ten unknowns, this complexity can be a real handicap. Usual statements are well-fitted for decomposition and lead to small boundaries, but one may imagine heuristics for boundary systems with too many elements. For instance, the boundary constraints could be constructed only on demand when needed by other solvers.

5. Conclusion

Geometrical constraints solvers generally proceed by joining solved subsystems. This mechanism of decomposition/recombination allows geometric solvers (which produce all the solutions) to solve a larger class of problems, and allows numerical solvers to lower their runtime. The current trend is to consider a decomposition in subsystems invariant under the action of known transformation groups.

This article presents a formalization of this approach and shows its validity: if a subsystem is solved *modulo* a transformation group, the remaining subsystem and the boundary system form a restriction of the original system: no solution has been added or lost through the removal of the solved subsystem and the addition of the boundary system. The boundary system is defined semantically: in order for it to be expressed, the geometrical universe must contain the appropriate constraint types.

This formalization also brings out the main elements and data types for a general and modular implementation of decomposition/recombination resolution platform.

The formalization presented in this paper applies to all decomposition methods, although it focuses a bit on bottom-up approaches. The general decomposition/recombination scheme that naturally results from this formalization is general and modular and, thus, requires time to be implemented. Indeed, it involves many data types as well as the programming of the solving strategy.

In the future, we intend this formalization to be a basis to express geometric constraints solving methods and to compare them in terms of robustness, completeness, and efficiency. A precise expression of the geometrical universe is a prerequisite to any formal comparison of algorithms.

We see that the notion of well-constrained system depends on the considered group. For example, numerous solvers consider as under-constrained a triangle defined by two angles, whereas this problem is well-constrained under similarities. In a more general way, any system could be declared as well-constrained if it is possible to express an invariance group. Here, we consider groups of well-known geometrical transformations acting on the whole system. To characterize the invariance of any system, it is necessary to be able to express an appropriate group of invariance for the system by considering local groups. This approach is the perspective of this work to deal with under-constrained problems, as was begun in [TMS07].

Appendix: 2D GCML example

This appendix contains an example of the description, using the Geometric Constraints Markup Language presented in [WSMF06], of a 2D geometrical universe (see below, left) and of the geometric constraint system graphically shown in Fig. 6 (see below, right).

```
<syntax>
     <sorts>
        point
        line
        circle
        length
        angle
        scalar
     </sorts>

     <fsymbols>
        initp : scalar scalar -> point
        initl : scalar scalar -> line
        initc : point scalar -> circle
        initlg : scalar -> length
        inita : scalar -> angle
     </fsymbols>

     <psymbols>
          <!-- incidence point line -->
         on_pl     : point line
           <!-- incidence point circle -->
         on_pc     : point circle
           <!-- center of circle -->
         center     : point circle
           <!-- distance between two points -->
         dist_pp  : point point length
           <!-- distance between a point and a line -->
         dist_pl  : point line length
           <!-- angle between lines -->
         angle_ll : line line angle
           <!-- fix radius for circle -->
         radius   : circle length
           <!-- set coordinates for a point -->
         fix_p : point scalar scalar
           <!-- set slope of a line -->
         fix_l : line scalar
           <!-- tangency of circle and line -->
         tangency_cl : circle line
           <!-- tangency of circles -->
         tangency_cc : circle circle
           <!-- ratio between couple of points -->
         ratio : point point point point scalar
     </psymbols>

</syntax>
```

```
<gcs>
    <syntax>
       <unknowns>
          point p1 p2 p3 p4
          line l1 l2
          circle c1
       </unknowns>

       <parameters>
           length k1 k2
           angle a1
       </parameters>

       <constraints>
            on_pl(p1,l1) on_pl(p2,l1)
            on_pl(p1,l2) on_pl(p3,l2)
            on_pc(p3,c1) on_pc(p2,c1)
            center(p4, c1)

            dist_pp(p1,p2,k1)
            angle_ll(l1,l2,a1)
            radius(c1,k3)
            tangency_cl(c1,l1)
       </constraints>

   </syntax>
   <valuation>
       k1=initlg(100)
       k2=initlg(50)
       a1=inita(0.52)
   </valuation>
</gcs>
```

References

[Ald88] Aldefeld B (1988) Variations of geometries based on a geometric-reasoning method. Comput Aided Des 20(3):117–126

[Brü93] Brüderlin B (1993) Using geometric rewrite rules for solving geometric problems symbolically. Theor Comput Sci 116(2):291–303

[DMS98] Dufourd J-F, Mathis P, Schreck P (1998) Geometric construction by assembling solved subfigures. Artif Intell 99(1):73–119

[GC98] Gao X-S, Chou S-C (1998) Solving geometric constraint systems II. A symbolic approach and decision of Rc-constructibility. Comput Aided Des 30(2):115–122

[GHY02] Gao X-S, Hoffmann CM, Yang W-Q (2002) Solving spatial basic geometric constraint configurations with locus intersection. In: SMA'02: Proceedings of the seventh ACM symposium on solid modeling and applications. ACM, Saarbrücken, germany pp 95–104

[GLZ06] Gao X-S, Lin Q, Zhang G-F (2006) A C-tree decomposition algorithm for 2D and 3D geometric constraint solving. Comput Aided Des 38(1):1–13

[JAS97] Joan-Arinyo R, Soto A (1997) A correct rule-based geometric constraint solver. Comput Graph 5(21):599–609

[JTNM06] Jermann C, Trombettoni G, Neveu B, Mathis P (2006) Decomposition of geometric constraint systems: a survey. Int J Comput Graph Appl 16(5,6):379–414

[Kon92] Kondo K (1992) Algebraic method for manipulation of dimensional relationships in geometric models. Comput Aided Des 24(3):141–147

[Kra92] Kramer GA (1992) A geometric constraint engine. Artif Intell 58(1–3):327–360

[LLG81] Light R, Lin V, Gossard DC (1981) Variational Geometry in CAD. Comput Graph 15(3):171–175

[LM96a] Lamure H, Michelucci D (1996) Solving geometric constraints by homotopy. IEEE Trans Vis Comput Graph 2(1):28–34

[LM96b] Latham RS, Middleditch AE (1996) Connectivity analysis: a tool for processing geometric constraints. Comput Aided Des 28(11):917–928

[Owe91] Owen JC (1991) Algebraic solution for geometry from dimensional constraints. In: SMA'91: proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications. ACM, Austin, pp 397–407

[Sch94] Schreck P (1994) A knowledge-based for solving geometric constructions problems. In: Brahan JW, Lasker GE (eds) Proceedings of the seventh international conference on systems research, Informatics and Cybernetics, pp 19–24

[Sit06] Sitharam M (2006) Well-formed systems of point incidences for resolving collections of rigid bodies. Int J Comput Geometry Appl 16(5,6):591–615

[SM06] Schreck P, Mathis P (2006) Geometrical constraint system decomposition: a multi-group approach. Int J Comput Geometry Appl 16(5,6):431–442

[SS06] Schreck P, Schramm E (2006) Using the invariance under the similarity group to solve geometric constraint systems. Comput Aided Des 38(5):475–484

[Sun86] Sunde G (1986) A CAD system with declarative specification of shape. In Proceedings of the IFIP WG 5.2 on Geometric Modeling, Rensselaerville

[TMS07] Thierry SEB, Mathis P, Schreck P (2007) Towards an homogeneous handling of under-constrained and well-constrained systems of geometric constraints. In: SAC'07: Proceedings of the 2007 ACM symposium on Applied computing. ACM, Seoul, pp 773–777

[vdMB08] van der Meiden HA, Broonsvort WF (2008) Solving systems of 3D geometric constraints using non-rigid clusters. In: GPM'08: advances in geometric modelling and processing, Hangzhou, China. Lecture Notes in Computer Science, vol 4975. Springer, Berlin

[VSR92] Verroust A, Schonek F, Roller D (1992) Rule-oriented method for parameterized computer-aided design. Comput Aided Des 24(10):531–540

[WSMF06] Wintz J, Schreck P, Mathis P, Fabre A (2006) A framework for geometric constraint satisfaction problem. In: SAC'06: proceedings of the 2006 ACM symposium on applied computing. ACM, Dijon, pp 974–978

Received 2 July 2008
Accepted in revised form 5 June 2009 by D.A. Duce
Published online 8 July 2009