Declarative Characterization of a General Architecture for Constructive Geometric Constraint Solvers

R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, J. Vilaplana-Pastó
Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics
Av. Diagonal 647, 8a, E–08028 Barcelona

[robert,tonis,sebas,josep]@lsi.upc.es

Abstract

Geometric constraint solving is a growing field devoted to solve geometric problems defined by relationships, called constraints, established between the geometric elements. There are several techniques to solve geometric constraint problems. In this work we focus on the Constructive technique. Usually, it works in two steps. In a first step, the problem is analyzed symbolically. If the problem is solvable by the technique, the output is the construction plan, that is, a sequence of abstract geometric constructions which defines parametrically the solution to the problem. Then, the construction plan is applied to a set of specific values assigned to the parameters. If no numerical incompatibilities arise, instances of the solution are generated.

In this paper we present a general architecture for constructive geometric constraint solvers. The basic components of this architecture are three functional units: the analyzer, the index selector and the constructor. Each functional unit is specified in terms of the entities that manipulates such as geometric constraint problems and construction plans. These relevant entities are declaratively characterized and its precise semantic is stated.

Keywords Geometric constraints, Constructive geometric constraint solving, Declarative representations.

1 Introduction

In two-dimensional constraint-based geometric design, the designer creates a rough sketch of an object made out of simple geometric elements like points, lines, circles and arcs of circle. Then the intended exact shape is specified by annotating the sketch with constraints like distance between two points, distance from a point to a line, angle between two lines, line-circle tangency and so on. A geometric constraint solver then checks whether the set of geometric constraints coherently defines the object and, if so, determines the position of the geometric elements. The designer can now modify the values of constraints or ask the geometric constraint solver for alternative solutions that also satisfy the constraints.

Many techniques have been reported in the literature that provide powerful and efficient methods for solving systems of geometric constraints. For example, see [5] and references therein for an extensive analysis of work on constraint solving. Among all the geometric constraint solving techniques, our interest focuses on the one known as *constructive*, [1, 2, 3, 7, 11, 12, 15].

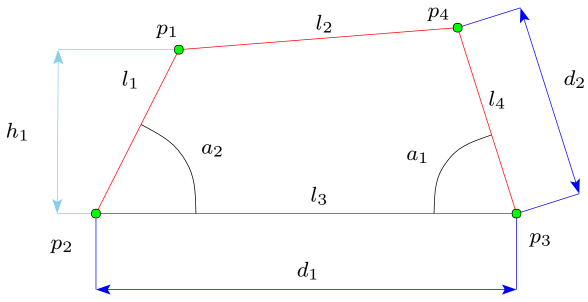
Constructive solvers have two major components, [7]: the *analyzer* and the *constructor*. The analyzer symbolically determines whether a geometric problem defined by constraints is solvable. If the problem is solvable, the output of the analyzer is a sequence of construction steps, known as the *construction plan*, that places each geometric element in such a way that all constraints are satisfied. After assigning specific values to the parameters, the constructor interprets the construction plan and builds an object instance, provided that no numerical incompatibilities arise.

The specific construction plan generated by an analyzer depends on the underlying constructive technique and on how it is implemented. For example, the ruler-and-compass constructive approach is a well-known technique where each constructive step in the plan corresponds to a basic operation solvable with ruler, compass and protractor. In practice, this simple approach solves most useful geometric problems. [8].

Although the constructive geometric constraint solvers proposed in the literature seems to share a common architecture, few efforts have been devoted to characterize it independently of the underlying constraint solving method proposed.

In this paper we present a general architecture for constructive geometric constraint solvers. We deliberately avoid focusing on solving methods. First, we identify a set of relevant entities in constructive geometric constraint solving such as abstract geometric constraint problems, abstract construction plans, parameters assignment and index assignment. We present a high level declarative characterization of these entities and their semantics. Next, we identify three functional units in which a geometric constraint solver is structured: the analyzer, the index selector and the constructor. These functional units are specified in terms of the entities that each one manipulates. Lastly, we assemble this functional units in a general architecture for constructive geometric constraint solvers.

The outline of the paper is as follows. In Section 2 we present preliminary concepts and definitions. In the following two sections we identify the entities relevant in geometric constraint solving. Geometric constraint problems and parameters assignments are defined in Section 3. Construction plans and index assignments are defined in Section 4. In Section 5 we interpret geometric constraint problems and construction plans in terms of first order logic formulae. In Section 6 we identify three basic functional units in constructive geometric constraint solving, namely the analyzer, the index selector and the constructor. We specify each functional unit in terms of its input, its output and the relationship between them. In particular, we give a definition for correctness and completeness of an analyzer. In Section 7 we give a general architecture for constructive geometric constraint solvers built on the concepts previously introduced. Finally, Section 8 offers a summary.



$onPL(p_1, l_1)$	$onPL(p_1, l_2)$
$onPL(p_2, l_1)$	$onPL(p_2, l_3)$
$onPL(p_3, l_3)$	$onPL(p_3, l_4)$
$onPL(p_4, l_2)$	$onPL(p_4, l_4)$
$distPP(p_2, p_3, d_1)$	$distPP(p_3, p_4, d_2)$
$distPL(p_1, l_3, h_1)$	$angleLL(l_3, l_1, a_2)$
$angleLL(l_3, l_4, a_1)$	

Figure 1: Geometric problem defined by constraints.

2 Preliminaries

In this section we present concepts and notational conventions that will be used throughout all the manuscript.

We assume that a constraint-based design is made of geometric elements like point, lines, circles and arcs of circle. The intended shape is defined by means of constraints like distance between two points, distance from a point to a line, angle between two lines, line-circle tangency and so on.

In what follows, the symbols to represent geometric elements will be taken from the set

$$\mathcal{L}_G = \{p_1, l_1, c_1, p_2, l_2, c_2, \dots, p_n, l_n, c_n, \dots\}$$

p_i denoting a point, l_i a straight line and c_i a circle. We assume that the number of different symbols available is unlimited.

Constraints will be represented by predicates relating geometric elements or geometric elements plus a symbolic value called *parameter*. For example,

$$\begin{aligned}\mathcal{L}_R = \{&onPL(p, l),\\ &distPP(p_i, p_j, d),\\ &distPL(p_i, l_j, h),\\ &angleLL(l_i, l_j, a), \dots\}\end{aligned}$$

Predicate names are self explanatory. The predicate $onPL(p, l)$ specifies that point p must lie on line l, $distPP(p_i, p_j, d)$ specifies a point-point distance, $distPL(p_i, l_j, h)$ defines the perpendicular distance from a point to a straight line and, $angleLL(l_i, l_j, a)$ denotes the angle between two straight lines. The number and syntax of available constraints are fixed. Symbols d, h and a are parameters. The symbols to represent parameters will be taken from the set

$$\mathcal{L}_P = \{d_1, h_1, a_1, d_2, h_2, a_2, \dots, d_n, h_n, a_n, \dots\}$$

d_i denoting a distance between two points, h_i a distance between a point and a line and a_i an angle between two lines. Figure 1 shows an example of a constraint-based design and the set of constraints defined between the geometric elements.

This work is centered on constructive geometric constraint solving. Thus, a chief entity is the construction plan. To illustrate the

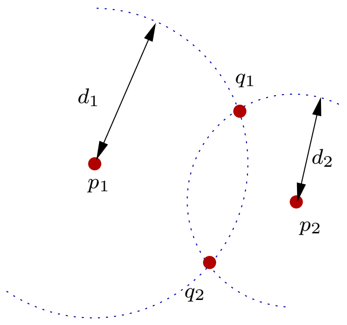


Figure 2: Possible placements of a point.

concepts, in what follows, we assume that a constructive ruler-and-compass based solver like that reported in [12] is available. Therefore, according to [8], the basic geometric constructions are

$$\begin{aligned}\mathcal{L}_{CB} = \{&pointXY(x, y),\\ &linePP(p_i, p_f),\\ &lineAP(l, a, p),\\ &circleCR(p, r),\\ &interLL(l_i, lj),\\ &interLC(l, c, s),\\ &interCC(c_i, c_j, s)\}\end{aligned}$$

The meaning of the basic construction names is the usual: point defined by its coordinates, straight line given by an ordered pair of points, straight line through a point at an angle with respect to another line, circle defined by the center and radius, and intersections between straight lines and circles.

The repertoire and syntax of available geometric operations depends on the specific constructive solving approach used and the implementation. However, it is considered fixed. In what follows, we assume that the set of available geometric operations $\mathcal{L}_C$ consists of those given in $\mathcal{L}_{CB}$ plus some additional simple operations that can be easily expressed as a sequence of operations in $\mathcal{L}_{CB}$, for example *lineLD*(l, h, s), which defines a straight line parallel to another one at a given signed distance, see [9].

Note that basic intersection operations involving circles may have more that one intersection point. We characterize each intersection point by using an additional *sign parameter* s with value in $\{+1, -1\}$. Therefore, this leads to operations like $interLC(l, c, s)$ and $interCC(c_i, c_j, s)$. For a full definition of the semantics of parameter s see [14]. The symbols to represent sign parameters will be taken from the set

$$\mathcal{L}_I = \{s_1, s_2, \dots, s_n, \dots\}$$

Example 2.1 The intersection between circle $c_1 = circleCR(p_1, d_1)$ and circle $c_2 = circleCR(p_2, d_2)$ in Figure 2 are the points $\{q_1, q_2\}$. According to the semantic of sign parameters defined in [14],

$$\begin{aligned}q_1 &= interCC(c_1, c_2, +1)\\ q_2 &= interCC(c_1, c_2, -1)\end{aligned}$$

$\diamond$

Given a set of symbols S and a set of values V, a *textual substitution* α is a total mapping from S to V. Let W be a set of predicates and α a textual substitution, we note by $\alpha.W$ the set of predicates obtained by replacing every occurrence of any symbol $s \in S$ found in W by $\alpha(s) \in V$.

Example 2.2 Let $S = \{a_1, h_1\}$ be a set of symbols and $V = \mathbb{R}$. Let α a textual substitution from S to V defined as

$$\alpha(a_1) = 0.57, \quad \alpha(h_1) = 4.0$$

and let W be a set of predicates in $\mathcal{L}_R$ with

$$\begin{aligned} W &= \{onPL(p_1, l_1), angleLL(l_1, l_3, a_1), \\ &\quad distPL(p_1, l_3, h_1)\}. \end{aligned}$$

Then $\alpha.W$ is

$$\begin{aligned} \alpha.W &= \{onPL(p_1, l_1), angleLL(l_1, l_3, 0.57), \\ &\quad distPL(p_1, l_3, 4.0)\}. \end{aligned}$$

$\diamond$

In this paper we will also apply textual substitutions to first order logic formulae and other syntactical descriptions.

3 Geometric Constraint Problems

We define and describe declaratively the concepts of abstract geometric constraint problem and of instance of a geometric constraint problem. Abstract entities are exclusively defined in terms of symbols like those in the sets $\mathcal{L}_G$, $\mathcal{L}_P$ and $\mathcal{L}_I$. Instance entities are abstract entities where some of the symbols occurring in them have been replaced by values.

3.1 Abstract Problem

An *abstract geometric constraint problem*, or *abstract problem* in short, is a tuple $A = \langle G, C, P\rangle$ where G is a set of symbols in $\mathcal{L}_G$ denoting geometric elements, C is a set of constraints taken from $\mathcal{L}_R$ and defined between elements of G, and P is the set of parameters taken from $\mathcal{L}_P$.

Example 3.1 Consider the sketch with annotated dimension lines shown in Figure 1. It can be seen as an abstract problem $A = \langle G, C, P\rangle$ where the set of geometric elements is

$$G = \{p_1, p_2, p_3, p_4, l_1, l_2, l_3, l_4\},$$

C is the set of constraints listed in Figure 1 and, the set of parameters is

$$P = \{d_1, d_2, a_1, a_2, h_1\}.$$

$\diamond$

A convenient way to fully describe an abstract problem is the algorithm-like notation. In this notation, the abstract problem in Example 3.1 can be expressed as

```
gcp A
  param
    $d_1, d_2, a_1, a_2, h_1$ : real
  endparam
  geom
    $p_1, p_2, p_3, p_4$ : point
    $l_1, l_2, l_3, l_4$ : line
  endgeom
  $onPL(p_1, l_1)$
  $onPL(p_1, l_2)$
  $onPL(p_2, l_1)$
  $onPL(p_2, l_3)$
  $onPL(p_3, l_3)$
  $onPL(p_3, l_4)$
  $onPL(p_4, l_4)$
  $onPL(p_4, l_2)$
  $distPP(p_2, p_3, d_1)$
  $distPP(p_3, p_4, d_2)$
  $distPL(p_1, l_3, h_1)$
  $angleLL(l_3, l_1, a_2)$
  $angleLL(l_3, l_4, a_1)$
endgcp
```

Note that an abstract problem defines a family of geometric constraint solving problems parameterized by the set P.

3.2 Instance Problem

A *parameters assignment* is a textual substitution α from a set of parameters P to $\mathbb{R}$.

Let $A = \langle G, C, P\rangle$ be an abstract problem and α be a parameters assignment from P. We say that $\alpha.A = \langle G, \alpha.C, P\rangle$ is an *instance problem* of A. Note that given an abstract problem, each different parameters assignment defines a different instance problem.

Example 3.2 Consider the abstract problem $A = \langle G, C, P\rangle$ described in the Example 3.1. An example of parameters assigment α is

$$\begin{aligned} \alpha(a_1) &= -1.222 \\ \alpha(a_2) &= 1.0472 \\ \alpha(h_1) &= 160.0 \\ \alpha(d_1) &= 290.0 \\ \alpha(d_2) &= 130.0 \end{aligned}$$

A description for the instance problem $\alpha.A$ is

```
gcp $\alpha.A$
  param
    $d_1, d_2, a_1, a_2, h_1$ : real
  endparam
  geom
    $p_1, p_2, p_3, p_4$ : point
    $l_1, l_2, l_3, l_4$ : line
  endgeom
  $onPL(p_1, l_1)$
  $onPL(p_1, l_2)$
  $onPL(p_2, l_1)$
  $onPL(p_2, l_3)$
  $onPL(p_3, l_3)$
  $onPL(p_3, l_4)$
  $onPL(p_4, l_4)$
  $onPL(p_4, l_2)$
  $distPP(p_2, p_3, 290.0)$
  $distPP(p_3, p_4, 130.0)$
  $distPL(p_2, l_2, 160.0)$
  $angleLL(l_3, l_1, 1.0472)$
  $angleLL(l_3, l_4, -1.222)$
endgcp
```

$\diamond$

Instance problems are no longer parameterized because the parameters have been replaced by the corresponding actual values.

Figure 3 shows a graphical representation for the instance problem $\alpha.A$ given in Example 3.2. Now parameters are no longer symbolic but actual values defined by the assignment α. Figure 3 is a

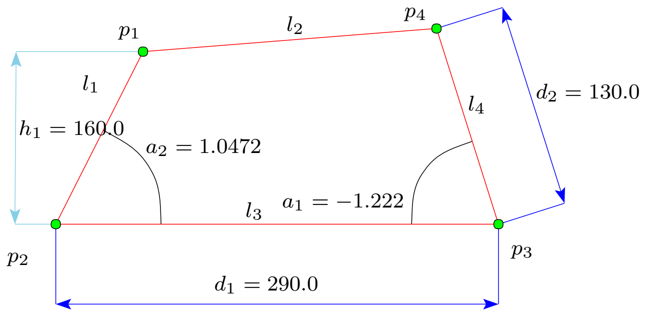


Figure 3: Instance problem

graphical representation of a declarative description of the geometric elements and constraints and, thus the actual geometry is irrelevant. For instance, the actual values of h_1 and d_2 in the figure do not match the values defined by $\alpha(h_1)$ and $\alpha(d_2)$.

Abstract problems precisely describe a set of geometric elements and the constraints that must fulfill, but they do not define how to place the geometric elements to satisfy the constraints. In the next section we will present the construction plan which describes how actually carry out the construction.

4 Construction Plan

A construction plan is a procedure that describes how to place the geometric elements with respect to each other. First we formalize the notion of abstract construction plan then we derive the concepts of instance plan and indexed plan.

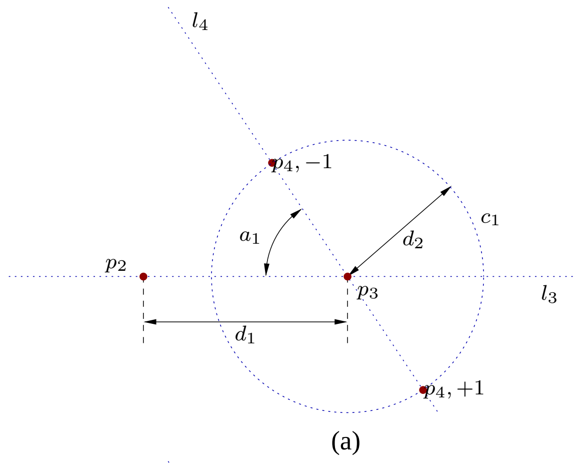
4.1 Abstract Plan

An *abstract construction plan*, or *abstract plan* in short, is a tuple $S = \langle G, P, L, I\rangle$ where G is a set of symbolic geometric elements taken from $\mathcal{L}_G$, P is a set of parameters taken from $\mathcal{L}_P$, the *index* I is a set of sign parameters taken from $\mathcal{L}_I$, and L is a sequence of basic construction operations taken from $\mathcal{L}_C$ and parameterized by P and I. L defines how to place with respect to each other the elements in G.

Example 4.1 If O denotes a reference point, an example of abstract construction plan that specifies how to build the geometric object given in Figure 1 is

```
cp S
  param
     $d_1, d_2, a_1, a_2, h_1$ : real
  endparam
  index
     $s_1, s_2$ : sign
  endindex
  geom
     $p_1, p_2, p_3, p_4$ : point
     $l_1, l_2, l_3, l_4$ : line
  endgeom
  $p_2 = pointXY(O_x, O_y)$
  $p_3 = pointXY(d_1, O_y)$
  $c_1 = circleCR(p_3, d_2)$
  $l_3 = linePP(p_2, p_3)$
  $l_4 = lineAP(l_3, a_1, p_3)$
  $p_4 = interCL(l_4, c_1, s_1)$
  $l_1 = lineAP(l_3, a_2, p_2)$
  $l_8 = lineLD(l_3, h_1, s_2)$
  $p_1 = interLL(l_1, l_8)$
  $l_2 = linePP(p_1, p_4)$
endcp
```

Note that L contains auxiliary symbols, $\{c_1, l_8\}$, which do not belong to G. These symbols are introduced to increase readability. Nonetheless, these symbols can be replaced by their definitions. For instance, symbol l_8 is defined as $l_8 = lineLD(l_3, h_1, s_2)$. If we replace l_8 in the definition of p_1 we have $p_1 = interLL(l_1, lineLD(l_3, h_1, s_2))$. This procedure can be repeated for every auxiliary symbol occurring in L.



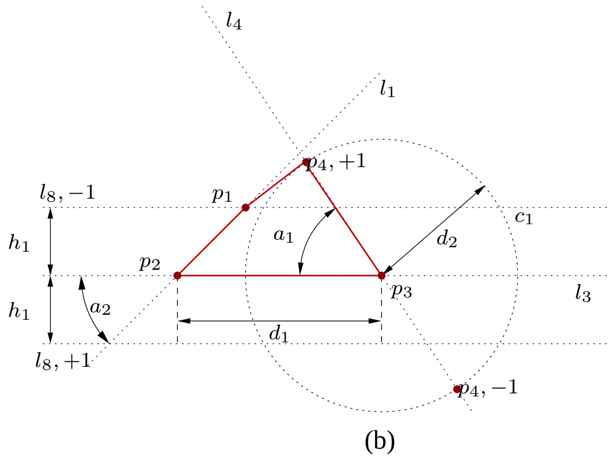


Figure 4: Step by step interpretation of the abstract plan given in Example 4.1.

Figure 4(a) and Figure 4(b) illustrate step by step how the plan is interpreted. First an arbitrary point O is chosen to start the construction. This point is labeled p_2, see Figure 4(a). Then, point p_3 whose y coordinate is coincident with O_y and at a distance d_2 from p_2 is created. Next the circle c_1, with center on p_3 and radius d_2, the straight line l_3, through points p_2 and p_3, and the line l_4, through point p_3 and at angle a_1 with, l_3 are created. Finally point p_4 is defined as the intersection of c_1 and l_4. Note that there are two possible locations for point p_4. Every possible location is distinguished with a sign from the set $\{+1, -1\}$, following the semantics of signs defined in [14]. Assuming that point p_4 is located in $p_4 - 1$, and that the line l_8 chosen is $l_8 - 1$, Figure 4(b) shows the interpretation of the rest of the plan. ◇

An abstract construction plan is parameterized by two sets: P and I.

In the following sections we present the concepts of instance plan and indexed plan. In an instance plan we fix the values of parameters in P and in an indexed plan we fix the values of signs in I.

4.2 Instance Plan

An abstract plan can be instantiated by applying a parameters assignment in the same way it has been done for abstract problems. Let $S = \langle G, P, L, I\rangle$ be an abstract plan and α a parameters assignment for P. The instance plan $\alpha.S$ is defined as $\alpha.S = \langle G, P, \alpha.L, I\rangle$.

Example 4.2 Applying the parameters assignment given in Example 3.2 to the abstract plan in Example 4.1, yields the instance plan

```
cp $\alpha.S$
  param
     $d_1, d_2, a_1, a_2, h_1$ : real
  endparam
  index
     $s_1, s_2$ : sign
  endindex
  geom
     $p_1, p_2, p_3, p_4$ : point
     $l_1, l_2, l_3, l_4$ : line
  endgeom
  $p_2 = pointXY(O_x, O_y)$
  $p_3 = pointXY(290.0, O_y)$
  $c_1 = circleCR(p_3, 130.0)$
  $l_3 = linePP(p_2, p_3)$
  $l_4 = lineAP(l_3, -1.222, p_3)$
  $p_4 = interCL(l_4, c_1, s_1)$
  $l_1 = lineAP(l_3, 1.0472, p_2)$
  $l_8 = lineLD(l_3, 160.0, s_2)$
  $p_1 = interLL(l_1, l_8)$
  $l_2 = linePP(p_1, p_4)$
endcp
```

Figure 5 shows the four possible evaluations of the instance plan in Example 4.2 obtained by changing the values of signs s_1 and s_2. $\diamond$

4.3 Indexed Plan

An *index assignment*, denoted ι, is a textual substitution from an index I to the set $\{+1, -1\}$.

Let $S = \langle G, P, L, I\rangle$ be an abstract plan and ι an index assignment from I. The *indexed plan* $\iota.S$ is defined as $\iota.S = \langle G, P, \iota.L, I\rangle$.

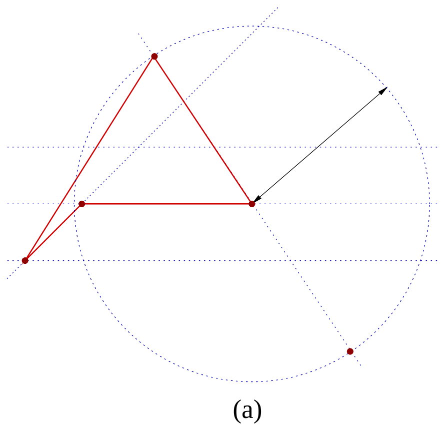
Example 4.3 Let the index assignment ι be

$$\iota(s_1) = -1, \quad \iota(s_2) = +1.$$

Applying ι to the abstract plan in Example 4.1, yields the indexed plan

```
cp S
  param
     $d_1, d_2, a_1, a_2, h_1$ : real
  endparam
  index
     $s_1, s_2$ : sign
  endindex
  geom
     $p_1, p_2, p_3, p_4$ : point
     $l_1, l_2, l_3, l_4$ : line
  endgeom
  $p_2 = pointXY(O_x, O_y)$
  $p_3 = pointXY(d_1, O_y)$
  $c_1 = circleCR(p_3, d_2)$
  $l_3 = linePP(p_2, p_3)$
  $l_4 = lineAP(l_3, a_1, p_3)$
  $p_4 = interCL(l_4, c_1, -1)$
  $l_1 = lineAP(l_3, a_2, p_2)$
  $l_8 = lineLD(l_3, h_1, +1)$
  $p_1 = interLL(l_1, l_8)$
  $l_2 = linePP(p_1, p_4)$
endcp
```

Figure 6 shows two different objects generated by changing the value of parameter d_2 in the plan. $\diamond$



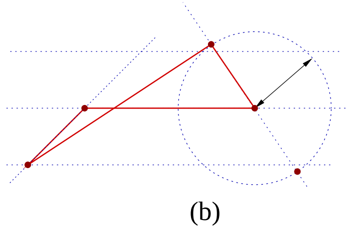


Figure 6: Distinct constructions encoded into the same abstract plan

Note that the application of a parameters assignment α and an index assignment ι to an abstract plan S commute. That is $\alpha.\iota.S = \iota.\alpha.S$.

5 Characteristic Formulae

We will interpret geometric constraint problems and construction plans by means of first order logic formulae. This will allow us to precisely characterize the set of placements of the geometric elements for which the set of constraints hold and, the set of placements actually generated by a construction plan.

5.1 Geometric Problems

Let $A = \langle G, C, P\rangle$ be an abstract geometric constraint problem with

$$C = \{c_1, c_2, \ldots, c_m\}$$

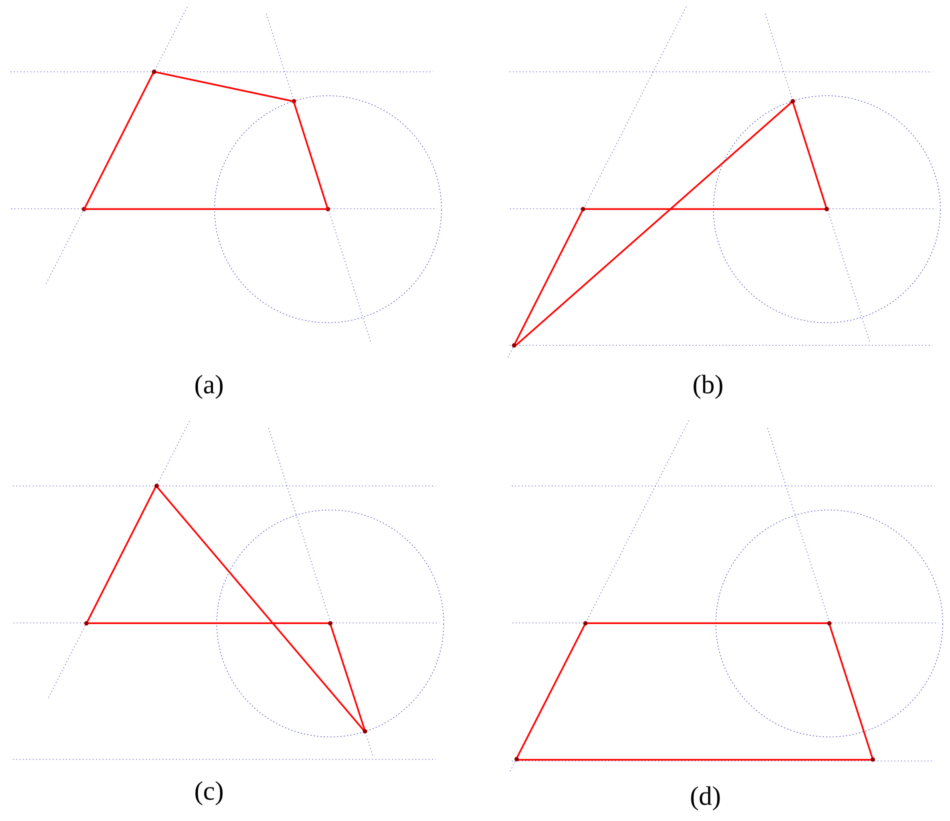


Figure 5: The four possible evaluations of the instance plan in Example 4.2

Then the *characteristic formula* of A is the first order logic formula,

$$\Psi(A) \equiv \bigwedge_{i=1}^{m} c_i$$

where the geometric elements of G and the parameters of P occurring in Ψ are interpreted as free variables.

Example 5.1 The characteristic formula of the abstract problem A given in the Example 3.1 is

$$\begin{aligned}\Psi(A) \quad \equiv \quad & (\mathit{onPL}(p_1, l_1) \wedge \mathit{onPL}(p_1, l_3) \wedge \\ & \mathit{onPL}(p_2, l_1) \wedge \mathit{onPL}(p_2, l_4) \wedge \\ & \mathit{onPL}(p_3, l_3) \wedge \mathit{onPL}(p_3, l_4) \wedge \\ & \mathit{onPL}(p_4, l_2) \wedge \mathit{onPL}(p_4, l_4) \wedge \\ & \mathit{distPP}(p_2, p_3, d_1) \wedge \mathit{distPP}(p_3, p_4, d_2) \wedge \\ & \mathit{distPL}(p_1, l_3, h_1) \wedge \mathit{angleLL}(l_1, l_3, a_2) \wedge \\ & \mathit{angleLL}(l_4, l_3, a_1))\end{aligned}$$

$\diamond$

Let α be a parameters assignment for P, and $\alpha.A$ the corresponding instance problem. Then the first order formula $\Psi(\alpha.A)$ expresses the instance problem. Note that a textual substitution α can be applied interchangeably to an abstract problem or to a first order logic formula. Therefore the relation $\Psi(\alpha.A) = \alpha.\Psi(A)$ is well defined.

Example 5.2 The characteristic formula of the instance problem in Example 3.2 is

$$\begin{aligned}\Psi(\alpha.A) \quad \equiv \quad & (\mathit{onPL}(p_1, l_1) \wedge \mathit{onPL}(p_1, l_3) \wedge \\ & \mathit{onPL}(p_2, l_1) \wedge \mathit{onPL}(p_2, l_4) \wedge \\ & \mathit{onPL}(p_3, l_3) \wedge \mathit{onPL}(p_3, l_4) \wedge \\ & \mathit{onPL}(p_4, l_2) \wedge \mathit{onPL}(p_4, l_4) \wedge \\ & \mathit{distPP}(p_2, p_3, 290.0) \wedge \\ & \mathit{distPP}(p_3, p_4, 130.0) \wedge \\ & \mathit{distPL}(p_1, l_3, 160.0) \wedge \\ & \mathit{angleLL}(l_3, l_1, 1.0472) \wedge \\ & \mathit{angleLL}(l_3, l_4, -1.222))\end{aligned}$$

$\diamond$

A *geometry assignment* or *anchor* κ is a textual substitution such that assigns an actual geometry to each geometric element in a set of geometry symbols G.

Let $A = \langle G, C, P\rangle$ be an abstract problem and κ an anchor for G. We define $\kappa.A$ as $\langle G, \kappa.C, P\rangle$.

Example 5.3 If we represent a point by the pair $(x, y) \in \mathbb{R}^2$ and a straight line by (a, b, c), the coefficients of the normasl form $ax + by + c = 0$ with $a^2 + b^2 = 1$, then an example of anchor κ is

$$\begin{aligned}
\kappa(p_1) &= (92.38, 160)\\
\kappa(p_2) &= (0, 0)\\
\kappa(p_3) &= (290, 0)\\
\kappa(p_4) &= (245.54, 122.16)\\
\kappa(l_1) &= (-0.87, 0.5, 0)\\
\kappa(l_2) &= (-0.24, -0.97, 177.48)\\
\kappa(l_3) &= (0, -1, 0)\\
\kappa(l_4) &= (0.94, 0.34, -272.51)
\end{aligned}$$

The characteristic formula Ψ after applying the anchor κ to the instance problem $\alpha.A$ in Example 5.2 is

$$\begin{array}{l}
\Psi(\kappa.\alpha.A)\\
\quad\equiv\\
(\mathit{onPL}((92.38, 160), (-0.87, 0.5, 0)) \wedge\\
\mathit{onPL}((92.38, 160), (0, -1, 0)) \wedge\\
\mathit{onPL}((0, 0), (-0.87, 0.5, 0)) \wedge\\
\mathit{onPL}((0, 0), (0.94, 0.34, -272.51)) \wedge\\
\mathit{onPL}((290, 0), (0, -1, 0)) \wedge\\
\mathit{onPL}((290, 0), (0.94, 0.34, -272.51)) \wedge\\
\mathit{onPL}((245.54, 122.16),\\
\qquad (-0.24, -0.97, 177.48)) \wedge\\
\mathit{onPL}((245.54, 122.16),\\
\qquad (0.94, 0.34, -272.51)) \wedge\\
\mathit{distPP}((0, 0), (290, 0), 290.0) \wedge\\
\mathit{distPP}((290, 0), (245.54, 122.16), 130.0) \wedge\\
\mathit{distPL}((92.38, 160), (0, -1, 0), 160.0) \wedge\\
\mathit{angleLL}((0, -1, 0),\\
\qquad\quad (-0.87, 0.5, 0), 1.0472) \wedge\\
\mathit{angleLL}((0, -1, 0),\\
\qquad\quad (0.94, 0.34, -272.51), -1.222))
\end{array}$$

$\diamond$

Note that α and κ commute, that is, $\kappa.\alpha.A = \alpha.\kappa.A$.

Let κ be an anchor for G. The set of anchors for which the formula $\Psi(\kappa.\alpha.A)$ holds

$$V(\alpha.A) = \{\kappa \quad | \quad \Psi(\kappa.\alpha.A)\}$$

define the set of anchors which are solution to the instance geometric constraint problem $\alpha.A$. We refer to the anchors in $V(\alpha.A)$ as *realizations* of the instance problem $\alpha.A$.

Figure 5 shows a graphical representation of the set of realizations $V(\alpha.A)$ for the instance problem $\alpha.A$ in Example 3.2.

5.2 Construction Plans

Let $S = \langle G, P, L, I\rangle$ be an abstract construction plan with $L = \{o_1, o_2, \ldots, o_n\}$. The *characteristic formula* of S is the first order logic formula,

$$\Phi(S) \equiv \bigwedge_{i=1}^{n} o_i$$

where the geometric elements of G, the parameters of P and signs of I occurring in Φ are considered free variables.

Example 5.4 The characteristic formula of the abstract plan S given in Example 4.1 is

$$\begin{array}{lll}
\Phi(S) & \equiv & (p_2 = \mathit{pointXY}(O_x, O_y)\\
& \wedge & p_3 = \mathit{pointXY}(d_1, O_y)\\
& \wedge & c_1 = \mathit{circleCR}(p_3, d_2)\\
& \wedge & l_3 = \mathit{linePP}(p_2, p_3)\\
& \wedge & l_4 = \mathit{lineAP}(l_3, a_1, p_3)\\
& \wedge & p_4 = \mathit{interCL}(l_4, c_1, s_1)\\
& \wedge & l_1 = \mathit{lineAP}(l_3, a_2, p_2)\\
& \wedge & l_8 = \mathit{lineLD}(l_3, h_1, s_2)\\
& \wedge & p_1 = \mathit{interLL}(l_1, l_8)\\
& \wedge & l_2 = \mathit{linePP}(p_1, p_4))
\end{array}$$

$\diamond$

Let α be a parameters assignment for P, and $\alpha.S$ the corresponding instance plan. Then the first order formula $\Phi(\alpha.S)$ expresses the instance plan. Note that $\Phi(\alpha.S) = \alpha.\Phi(S)$ trivially holds.

Example 5.5 The characteristic formula of the instance plan in Example 4.2 is

$$\begin{array}{lll}
\Phi(\alpha.S) & \equiv & (p_2 = \mathit{pointXY}(O_x, O_y)\\
& \wedge & p_3 = \mathit{pointXY}(290.0, O_y)\\
& \wedge & c_1 = \mathit{circleCR}(p_3, 130.0)\\
& \wedge & l_3 = \mathit{linePP}(p_2, p_3)\\
& \wedge & l_4 = \mathit{lineAP}(l_3, -1.222, p_3)\\
& \wedge & p_4 = \mathit{interCL}(l_4, c_1, s_1)\\
& \wedge & l_1 = \mathit{lineAP}(l_3, 1.0472, p_2)\\
& \wedge & l_8 = \mathit{lineLD}(l_3, 160.0, s_2)\\
& \wedge & p_1 = \mathit{interLL}(l_1, l_8)\\
& \wedge & l_2 = \mathit{linePP}(p_1, p_4))
\end{array}$$

$\diamond$

Let $S = \langle G, P, L, I\rangle$ be an abstract plan and κ an anchor for G. We define $\kappa.S$ as $\langle G, P, \kappa.L, I\rangle$.

Example 5.6 Let κ be the anchor in Example 5.3 and $\alpha.S$ the instance plan in Example 4.2. The characteristic formula Φ after applying the anchor κ to the instance problem $\alpha.S$ is

$$\begin{array}{l}
\Phi(\kappa.\alpha.S)\\
\quad\equiv\\
((0, 0) = \mathit{pointXY}(O_x, O_y) \wedge\\
(290, 0) = \mathit{pointXY}(290.0, O_y) \wedge\\
c_1 = \mathit{circleCR}((290, 0), 130.0) \wedge\\
(0, -1, 0) = \mathit{linePP}((0, 0), (290, 0)) \wedge\\
(0.94, 0.34, -272.51) =\\
\quad \mathit{lineAP}((0, -1, 0), -1.222, (290, 0)) \wedge\\
(245.54, 122.16) =\\
\quad \mathit{interCL}((0.94, 0.34, -272.51), c_1, s_1) \wedge\\
(-0.87, 0.5, 0) =\\
\quad \mathit{lineAP}((0, -1, 0), 1.0472, (0, 0)) \wedge\\
l_8 = \mathit{lineLD}((0, -1, 0), 160.0, s_2) \wedge\\
(92.38, 160) = \mathit{interLL}((-0.87, 0.5, 0), l_8) \wedge\\
(-0.24, -0.97, 177.48) =\\
\quad \mathit{linePP}((92.38, 160), (245.54, 122.16)))
\end{array}$$

$\diamond$

Let κ be an anchor for G and α a parameters assignment for P. The set of anchors for which there is an index assignment ι such that the formula $\Phi(\iota.\kappa.\alpha.S)$ holds

$$V(\alpha.S) = \{\kappa \quad | \quad \exists\, \iota\ \Phi(\iota.\kappa.\alpha.S)\}$$

define the set of anchors which are computed by the instance plan $\alpha.S$. We refer to the anchors in $V(\alpha.S)$ as *indexed anchors* of the instance plan $\alpha.S$.

Figure 5 shows a graphical representation of the set of indexed anchors $V(\alpha.S)$ for the instance plan $\alpha.S$ in Example 4.2.

Note that given an index assignment ι and a parameters assignment α, there is at most one anchor κ for which $\Phi(\kappa.\iota.\alpha.S)$ holds.

6 Constructive Solvers

In the preceding sections we have identified a set of entities relevant in the constructive geometric constraint solving process: abstract problems, parameters assignments, instance problems, abstract plans, instance problems, index assignments and anchors. In this section we advocate an architecture for constructive geometric constraint solvers based on three functional units: the analyzer, the index selector and the constructor. We will specify the functionality of each unit by stating the input, the output and the relationships between them.

6.1 The Analyzer

The *analyzer* is the functional unit that computes an abstract plan $S = \langle G, P, L, I\rangle$ from an abstract problem $A = \langle G, C, P\rangle$. The relationship between the abstract problem A and the abstract plan S established by the definition of analyzer is that the sets G and P in A and S are the same.

The set of abstract problems A for which an analyzer computes a construction plan S is the *analyzer domain*.

We say that an analyzer is *correct* if and only if for every abstract problem A in its domain, and for every parameters assignment α, $V(\alpha.S) \subseteq V(\alpha.A)$. That is, each anchor for which the construction plan S is feasible corresponds to one realization of the instance problem A.

We say that an analyzer is *complete* if and only if for every abstract problem A in its domain, and for every parameters assignment α, The set of anchors computed by the construction plan S and the set of realizations of the instance problems A are coincident, $V(\alpha.S) = V(\alpha.A)$. Analyzers described in [2, 4, 11, 15] are complete.

> **Example 6.1** Since the abstract plan S in Example 4.1 has been generated from the abstract problem A in Example 3.1 by a complete analyzer, the set of indexed anchors of the instance plan $\alpha.S$ in Example 4.2 and the set of realizations of the instance problem $\alpha.S$ in Example 3.2 are the same set. Figure 5 shows this set. $\diamond$

6.2 The Index Selector

An *index selector* is a functional unit exclusively characterized by its output which is an index assignment ι.

The input to an index selector depends on the selection method it implements. Here we enumerate some methods.

1. A *trivial* index selector returns an index assignment ι fixed *a priori*. For instance, $\iota(s) = +1$ for all s in I.
2. An index assignment ι from $I = \{s_1, \ldots, s_n\}$ can be represented by the binary number $d_1 d_2 \ldots d_n$ where $d_i = 0$ if $\iota(s_i) = -1$ and $d_i = 1$ if $\iota(s_i) = 1$. The order relation in binary numbers induces an order in the index assignments. Therefore, we can define a *successor* (*predecessor*) index selector to compute the next (previous) index assignment ι' from a give index assignment ι.
3. An *anchor-based* index selector computes an index assignment ι from an anchor κ and an abstract plan $S = \langle G, P, L, I\rangle$. The output is an index such that defines a realization where the placement of geometric elements preserves the orientations defined by the anchor κ, [2, 6, 13].

See [13] for an extensive analysis of methods for implementing index selectors.

6.3 The Constructor

The *constructor* is the functional unit that computes an anchor κ from an abstract plan S, a parameters assignment α and an index assignment ι. The anchor κ is a realization in $V(\alpha.A)$ provided that the abstract plan S has been computed from the abstract problem A by a correct analyzer.

7 Solvers Software Architecture

In this section we present a software architecture useful for building a geometric constraint solving tool-box. The aim of such a tool-box is to provide the software engineer with a set of tools to design and implement software applications founded on constraint solving.

The architecture has functional units and data entities. The data entities are geometric constraint problems, constructions plans, parameters assignments, geometry assignments and index assignments. The functional units are analyzers, index selectors and constructors. All these components relate each other following the data-flow diagram shown in Figure 7.

This architecture exhibits a number of advantages:

1. The architecture is precisely and concisely defined.
2. It is independent of any particular implementation of the functional units. All what is needed is to define the specific grammar and semantics of $\mathcal{L}_R$ and $\mathcal{L}_C$ given in Section 2.
3. The nature of the computations in each step are quite different. The analyzer requires symbolic computation while the constructor only performs numerical computations.
4. Determining whether the problem can be symbolically solved or not is performed in the analysis step and it does not depend neither on the actual parameter values nor on the geometric computations.
5. When computing instances for different parameter values, only the second step needs to be carried out. This allows to skip the analysis step which is computationally the most expensive.
6. Once a construction plan and a parameters assignment is fixed, navigation in the solutions space is governed just by the index selector that computes index assigments which define different realizations.
7. Given an abstract plan, a parameters assignment and and index assignment, an anchor can be computed if there are not numerical impossibilities.
8. The functional units are reusable to solve problems which are not geometric constraint solving problems but are related. For example, in [10] the tool-box is applied to deal with requirements of concurrent engineering applications.

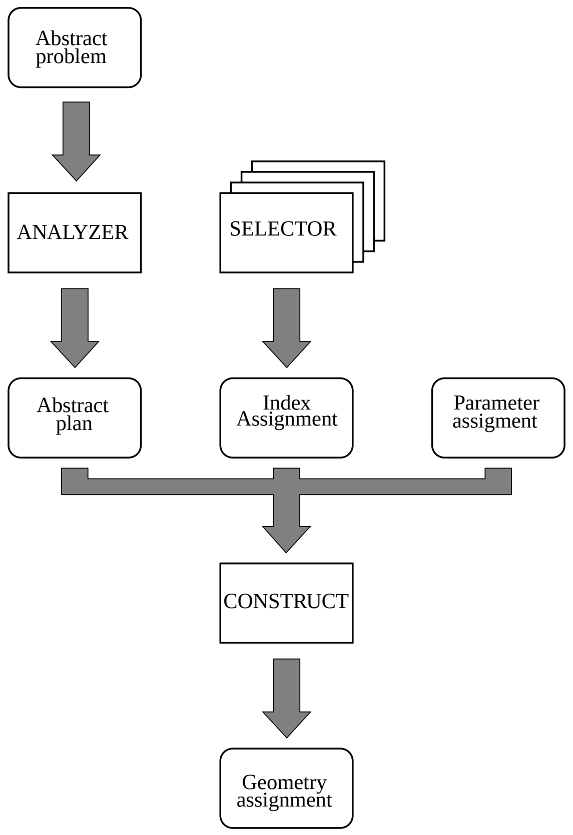


Figure 7: Architecture data-flow diagram.

8 Summary

We have presented a general architecture for constructive geometric constraint solvers. The architecture is based on three functional units: the analyzer, the index selector and the constructor. Functional units have been precisely defined in terms of the entities which are their input and output. These entities are: the abstract problem, the abstract construction plan, the parameters assignment and the index assignment.

To illustrate the concepts, we have used functional capabilities which are specific to the ruler-and-compass constructive geometric constraint solving technique. But the concepts apply to any constructive approach. All what is needed is to replace the set of geometric elements, the constraints available and the set of basic constructions with those in the constructive approach of interest.

Acknowledgements

This research has been supported by the CICYT under the project TIC2001-2099-C03-01.

References

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

[2] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. Geometric constraint solver. *Computer-Aided Design*, 27(6):487–501, June 1995.

[3] B.D. Brüderlin. Symbolic computer geometry for computer aided geometric design. In *Advances in Design and Manufacturing Systems*, Tempe, AZ, Jan. 8-12 1990. Proceedings NSF Conference.

[4] B.D. Brüderlin. Using geometric rewrite rules for solving geometric problems symbolically. In *Theoretical Computer Science 116*, pages 291–303. Elsevier Science Publishers B.V., 1993.

[5] C. Durand. *Symbolic and Numerical Techniques for Constraint Solving*. PhD thesis, Purdue University, Department of Computer Sciences, December 1998.

[6] C. Essert-Villard, P. Schreck, and J.-F. Dufourd. Skecth-based pruning of a solution space within a formal geometric constraint solver. *Artificial Intelligence*, 124:139–159, 2000.

[7] I. Fudos and C.M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, April 1997.

[8] D.J.H. Garling. *A course in Galois theory*. Cambridge University Press, 1986.

[9] R. Joan-Arinyo. Triangles, ruler and compass. Technical Report LSI-95-6-R, Department LiSI, Universitat Politècnica de Catalunya, 1995.

[10] R. Joan-Arinyo, A. Soto, S. Vila, and J. Vilaplana. A framework to support multiple views in geometric constrain-b ased models. In E. Dekneuvel, editor, *Proceedings of the 8th. IEEE International Conference on Emerging Technologies and Factory Automation ETFA'2001*, Antibes-Juan les Pins, France, October 2001. 8th. IEEE.

[11] R. Joan-Arinyo and A. Soto-Riera. A correct rule-based geometric constraint solver. *Computers & Graphics*, 21(5):599–609, 1997.

[12] R. Joan-Arinyo and A. Soto-Riera. Combining constructive and equational geometric constraint solving techniques. *ACM Transactions on Graphics*, 18(1):35–55, January 1999.

[13] M.V. Luzón. *The root identification problem in constructive geometric constraint solving*. PhD thesis, Universidade da Vigo, 2001. Writen in spanish.

[14] N. Mata. *Constructible Geometric Problems with Interval Parameters*. PhD thesis, Dept. LSI, Universitat Politècnica de Catalunya, Barcelona, Spain, 2000.

[15] J.C. Owen. Algebraic solution for geometry from dimensional constraints. In *ACM Symp Foundations of Solid Modeling*, pages 397–407, Austin, TX, 1991.