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, they work 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 arises, instances of the solution are generated.

In this paper we present a declarative representation for each component in the constructive geometric constraint solving technique. We illustrate the definitions with an specific example taken from the ruler-and-compass constructive approach.

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

1 Introduction

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, constructive solvers work 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 arises, instances of the solution are generated.

In this paper we first 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 its semantics. Next we identify three functional units, the analyzer, the index selector and the constructor, which are specified in terms of the entities that each functional unit manipulates. Lastly, we assemble this functional units in a general architecture for constructive geometric constraint solvers and devise how these functional units can also be used to solve other related problems.

The outline of the paper is as follows. In Sections 2 we recall the basic concepts of constructive geometric constraint solving. We present declarative definitions for several entities relevant in geometric constraint solving. Geometric problems defined by constraints 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 addition, we precisely define the set of geometric values that satisfy a geometric constraint problem and the geometric values which a construction plan is evaluated to. In Section 6 we identify three functional units in which a constructive geometric constraint solver can be structured: the analyzer, the index selector and the constructor. We specify each of these functional units in terms of the entities defined in Sections 3 and 4. We also define the correctness and completeness of an analyzer. In Section 7 we collect in a general architecture for constructive geometric constraint solvers all the concepts previously introduced. Finally, Section 8 offers a summary.

2 Constructive Geometric Constraint Solving

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.

If geometric elements and constraints are like those above, a constraint-based design can be represented by a set of points along with a set of constraints drawn from distance between two points, distance from a point to a line and angle between two lines, [11]. In what follows, the symbols to represent geometric elements will be taken from the set

$$\mathcal{G} = \{ p_1, l_1, c_1, \ldots, p_2, l_2, c_2, \ldots, p_n, l_n, c_n, \ldots \}$$

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

Constraints will be represented by predicates relating geometric elements or geometric elements plus a symbolic value called pa-
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 a ruler, compass and protractor. In practice, this simple approach solves most useful geometric problems. [4].

Predicate names are self explanatory. The predicate onPL(p,l) specifies that point p must lie on line l, distPP(p1,p2,d) specifies a point-point distance, distPL(p1,l1,h) defines the perpendicular distance from a point to a straight line and, angleLL(l1,l2,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

\[ \ell_P = \{d_1,h_1,a_1,d_2,h_2,a_2,d_3,h_3,a_3,\ldots \} \]

\( d_i \) 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 sketch of a constraint-based design and the set of constraints defined between the geometric elements.

Many techniques have been reported in the literature that provide powerful and efficient methods for solving systems of geometric constraints. For example, see [3] 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 geometric constraint solving techniques, our interest focuses on the geometric constraints. For example, see [3] and references therein for an vide powerful and efficient methods for solving systems of geometric problems. [4].

After assigning specific values to the parameters, the constructor builds a geometric element in such a way that all constraints are satisfied. Distance from a point to a straight line and, a point and a line and an intersection between a line and a circle defined by the center and radius, and intersections between straight lines and circles.

The repertoire and syntax of available geometric operations depend on the specific constructive solving approach used and the implementation. However, it is considered fixed. In what follows, we assume in what follows that a constructive ruler-and-compass solver like that reported in [8] is available. Therefore, the basic geometric constructions are

\[ \mathcal{L}_{CB} = \{ \text{pointXY}(x,y), \text{linePP}(p_1,p_2), \text{lineAP}(l,a,p), \text{circleCR}(p,r), \text{interLL}(l_1,l_2), \text{interLC}(l,c,s), \text{interCC}(c_1,c_2,s) \} \]

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.

To illustrate the concepts, we assume in what follows that a constructive ruler-compass solver that in [8] is available. Therefore, the basic geometric constructions are

\[ \mathcal{L}_{CB} = \{ \text{pointXY}(x,y), \text{linePP}(p_1,p_2), \text{lineAP}(l,a,p), \text{circleCR}(p,r), \text{interLL}(l_1,l_2), \text{interLC}(l,c,s), \text{interCC}(c_1,c_2,s) \} \]

\( \text{interCC}(c_1,c_2,s) \) is explained in [12]. The symbols to represent sign parameters will be taken from the set

\[ \ell_I = \{s_1,s_2,\ldots,s_n,\ldots \} \]

Example 2.1 The intersection between circle \( c_1 = \text{circleCR}(p_1,d_1) \) and circle \( c_2 = \text{circleCR}(p_2,d_2) \) in Figure 2 are the points \( \{q_1,q_2\} \). According to the semantic of sign parameters defined in [12].

\[ q_1 = \text{interCC}(c_1,c_2,+1) \]
\[ q_2 = \text{interCC}(c_1,c_2,-1) \]

In the following sections we will define abstract and instance entities. Abstract entities are exclusively defined in terms of symbols like those in the sets \( \mathcal{L}_G, \ell_P \) and \( \ell_I \). Instance entities are abstract
entities where some of the symbols occurring in them have been replaced by values. Different types of substitutions will be considered entities by themselves. Now we define the general concept of textual substitution which will be specialized further on.

Given a set of symbols $S$ and a set of values $V$, a textual substitution $\alpha$ is a total mapping from $S$ to $V$. Let $W$ be a set of parameters and $\alpha$ 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 $\alpha$ a textual substitution from $S$ to $V$ defined as

- $\alpha(a_1) = 0.57$
- $\alpha(h_1) = 4.0$

and let $W \subset \mathcal{L}_R$ be

$$W = \{\text{onPL}(p_1, l_1), \text{angleLL}(l_1, l_3, a_1), \text{distPL}(p_1, l_3, h_1)\}.$$  
Then $\alpha.W$ is

$$\alpha.W = \{\text{onPL}(p_1, l_1), \text{angleLL}(l_1, l_3, 0.57), \text{distPL}(p_1, l_3, 4.0)\}.$$  
\[\Diamond\]

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

In the following two sections we will identify seven entities relevant in geometric constraint solving. Abstract problems and abstract plans are abstract entities; instance problems, instance plans and indexed plans are instance entities; and parameters assignments and index assignments are textual substitutions.

### 3 Geometric Constraint Problems

We define and describe declaratively the concepts of abstract geometric constraint problem and of instance of a geometric constraint problem.

#### 3.1 Abstract Problem

An abstract geometric constraint problem, or abstract problem in short, is a tuple $A = (G, C, P)$ 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 $A = (G, C, P)$ 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**

$$d_1, d_2, a_1, a_2, h_1 : \text{real}$$

**endparam**

**geom**

$$p_1, p_2, p_3, p_4 : \text{point}$$

$$l_1, l_2, l_3, l_4 : \text{line}$$

**endgeom**

This is declarative description of a geometric object built from a number of geometric elements where the relative position of each element is defined by the properties (constraints) that the object must fulfill.

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 $\alpha$ from a set of parameters $P$ to $\mathbb{R}$.

Let $A = (G, C, P)$ be an abstract problem. We say that $\alpha.A = (G, \alpha.C, \alpha.P)$ 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 $A = (G, C, P)$ described in the Example 3.1. An example of parameters assignment $\alpha$ is

- $\alpha(a_1) = -1.222$
- $\alpha(a_2) = 1.0472$
- $\alpha(h_1) = 160.0$
- $\alpha(d_1) = 290.0$
- $\alpha(d_2) = 130.0$

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

**gcp A**

**param**

$$d_1, d_2, a_1, a_2, h_1 : \text{real}$$

**endparam**

**geom**

$$p_1, p_2, p_3, p_4 : \text{point}$$

$$l_1, l_2, l_3, l_4 : \text{line}$$

**endgeom**

\[\text{onPL}(p_1, l_1)\]
on $PL(p_1, l_4)$
on $PL(p_1, l_2)$
dist $PP(p_2, p_3, 290.0)$
dist $PP(p_3, p_4, 130.0)$
dist $PL(p_2, l_3, 160.0)$
angle $LL(l_1, l_1, 1.0472)$
angle $LL(l_1, l_4, -1.222)$
endcp

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 in Example 3.2. Note that parameters have acquired actual values.

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 $L_G$, $P$ is a set of parameters taken from $L_P$, the index $I$ is a set of sign parameters taken from $L_I$, $L$ is a sequence of basic construction operations taken from $L_C$ 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

\[
\begin{align*}
\text{cp } S & \; \text{param } \\
& \quad d_1, d_2, a_1, a_2, h_1 : \text{real} \\
& \quad \text{endparam} \\
& \quad \text{index } \\
& \quad \quad s_1, s_2 : \text{sign} \\
& \quad \quad \text{endindex} \\
& \quad \text{geom } \\
& \quad \quad p_1, p_2, p_3, p_4 : \text{point} \\
& \quad \quad l_1, l_2, l_3, l_4 : \text{line} \\
& \quad \quad \text{endgeom} \\
& \quad p_2 = \text{pointXY}(O_x, O_y) \\
& \quad p_3 = \text{pointXY}(d_1, O_y) \\
& \quad c_1 = \text{circleCR}(p_3, d_2)
\end{align*}
\]

Note that $L$ contains auxiliary symbols $\{c_1, l_5, l_6, l_7, l_8\}$ 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 $l_5$ is defined as $l_5 = \text{linePP}(p_2, p_3)$.

If we replace $l_5$ in the definition of $l_6$ we have $l_6 = \text{lineAP}(\text{linePP}(p_2, p_3), a_1, p_3)$. This procedure can be repeated for every auxiliary symbol occurrence in $L$.

In the figure 4 we show graphically the meaning of the abstract plan. The figure closely follows the step by step execution of the plan. This execution has been split into two sub-figures in the sake of clarity. In the first step of figure 4(a), we choose an arbitrary point $O$ to begin the construction. This point is labeled $p_2$. Then, a new point is created at the same $y$ coordinate than $O$ and at $d_2$ distance of it. This point is named $p_3$. Following, we trace an auxiliary circle $c_1$ with center in $p_3$ and radius $d_2$. An auxiliary line $l_5$ is created laying on points $p_2$ and $p_3$. A new auxiliary line $l_6$ is defined that goes through point $p_3$ and the angle between it and $l_5$ is
Finally we compute a new point \( p_4 \) as the intersection between \( c_1 \) and \( l_6 \). Note that there are two possible points \( p_1 \) because the circle and the line intersect on two points. Every possible point is labeled with a sign from the set \( \{+1, -1\} \). Figure 4(b) shows the execution of the rest of the plan. Notice that in figure (b) we choose the option \(-1\) for point \( p_4 \). The option \(+1\) can also be chosen thus leading to other result. The same consideration must be done too for line \( l_8 \). \( \Diamond \)

Two important things must be concluded from this example. The first one is that an abstract plan denotes a family of geometry placements because parameters are not substituted by actual values. Therefore, by following exactly the same construction of example 4(b) with distinct values for parameters \( d_1, d_2, a_1, a_2 \) and \( h_1 \) we obtain distinct constructions. The second aspect is that an abstract plan encodes one or more constructions. In fact the plan of the example encodes four distinct constructions that are obtained by choosing different values for signs \( s_1 \) and \( s_2 \). In figure 5 there are the distinct constructions encoded by the example plan. Labels have been omitted for clarity.

### 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 \( S = (G, P, L, I) \) be an abstract plan and \( \alpha \) a parameters assignment from \( P \). The instance plan \( \alpha.S \) is defined as \( \alpha.S = (G, P, \alpha.L, I) \).

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

```plaintext
cp \( \alpha.S \)
param
\( d_1, d_2, a_1, a_2, h_1 : \text{real} \)
endparam
index
\( s_1, s_2 : \text{sign} \)
endindex
gem
\( p_1, p_2, p_3, p_4 : \text{point} \)
\( l_1, l_2, l_3, l_4 : \text{line} \)
endgeom
\( p_2 = \text{pointXY}(O_x, O_y) \)
\( p_3 = \text{pointXY}(290.0, O_y) \)
\( c_1 = \text{circleCR}(p_3, 130.0) \)
\( l_5 = \text{linePP}(p_2, p_3) \)
\( l_6 = \text{lineAP}(l_5, -1.222, p_3) \)
\( p_4 = \text{intersectCL}(l_6, c_1, -1) \)
\( l_7 = \text{lineAP}(l_4, a_2, p_3) \)
\( l_8 = \text{lineLD}(l_7, h_1, -1) \)
\( p_1 = \text{intersectLL}(l_2, l_8) \)
\( l_3 = \text{linePP}(p_2, p_3) \)
\( l_1 = \text{linePP}(p_1, p_2) \)
\( l_4 = \text{linePP}(p_3, p_4) \)
\( l_2 = \text{linePP}(p_1, p_4) \)
endcp
```

In the figure 6 there is one of the four possible executions of the instance plan described before. Note how this instance is exactly the same object described in figure 3. \( \Diamond \)

### 4.3 Indexed Plan

An index assignment, denoted \( \iota \), is textual substitution from an indexed \( I \) to the set \( \{+1, -1\} \).

Let \( S = (G, P, L, I) \) be an abstract plan and \( \iota \) an index assignment from \( I \). The indexed plan \( \iota, S \) is defined as \( \iota, S = (G, P, \iota.L, I) \).

**Example 4.3** Let the index assignment \( \iota \) be

\[ \iota(s_1) = -1, \; \iota(s_2) = -1. \]

Applying \( \iota \) to the abstract plan in Example 4.1, yields the indexed plan

```plaintext
cp \( S \)
param
\( d_1, d_2, a_1, a_2, h_1 : \text{real} \)
endparam
index
\( s_1, s_2 : \text{sign} \)
endindex
gem
\( p_1, p_2, p_3, p_4 : \text{point} \)
\( l_1, l_2, l_3, l_4 : \text{line} \)
endgeom
\( p_2 = \text{pointXY}(O_x, O_y) \)
\( p_3 = \text{pointXY}(290.0, O_y) \)
\( c_1 = \text{circleCR}(p_3, 130.0) \)
\( l_5 = \text{linePP}(p_2, p_3) \)
\( l_6 = \text{lineAP}(l_5, a_1, p_3) \)
\( p_4 = \text{intersectCL}(l_6, c_1, -1) \)
\( l_7 = \text{lineAP}(l_4, a_2, p_3) \)
\( l_8 = \text{lineLD}(l_7, h_1, -1) \)
\( p_1 = \text{intersectLL}(l_2, l_8) \)
\( l_3 = \text{linePP}(p_2, p_3) \)
\( l_1 = \text{linePP}(p_1, p_2) \)
\( l_4 = \text{linePP}(p_3, p_4) \)
\( l_2 = \text{linePP}(p_1, p_4) \)
endcp
```

The column labeled \( s_1 = -1, s_2 = -1 \) in Figure 5 shows graphical representations obtained when the indexed plan above is evaluated choosing different parameters assignments. \( \Diamond \)

Note that the application of a parameters assignment \( \alpha \) and an index assignment \( \iota \) 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 define the set of geometric values which satisfy an instance problem and the set of geometric values which an instance plan evaluates to.

### 5.1 Geometric Problems

Let \( A = (G, C, P) \) be an abstract geometric constraint problem with

\[ C = \{c_1, c_2, \ldots, c_m\} \]

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 \( \Psi \) are interpreted as free variables.

**Example 5.1** The characteristic formula of the abstract problem \( A \) given in the Example 3.1 is

\[
\Psi(A) \equiv (\text{onPL}(p_1, l_1) \land \text{onPL}(p_1, l_3) \land \\
onPL(p_2, l_1) \land \text{onPL}(p_2, l_4) \land \\
onPL(p_3, l_3) \land \text{onPL}(p_3, l_4) \land \\
onPL(p_4, l_2) \land \text{onPL}(p_4, l_4) \land \\
distPP(p_2, p_3, d_1) \land distPP(p_3, p_4, d_2) \land \\
distPL(p_1, l_3, h_1) \land angleLL(l_1, l_3, a_2) \land \\
angleLL(l_1, l_3, a_1))
\]

\( \diamond \)

Let \( \alpha \) 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. Since \( \alpha \) is a substitution, \([9] \), the relation \( \Psi(\alpha.A) = \alpha.\Psi(A) \) trivially holds.

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

\[
\Psi(\alpha.A) \equiv (\text{onPL}(p_1, l_1) \land \text{onPL}(p_1, l_3) \land \\
onPL(p_2, l_1) \land \text{onPL}(p_2, l_4) \land \\
onPL(p_3, l_3) \land \text{onPL}(p_3, l_4) \land \\
onPL(p_4, l_2) \land \text{onPL}(p_4, l_4) \land \\
distPP(p_2, p_3, 290.0) \land \\
distPP(p_3, p_4, 130.0) \land \\
distPL(p_1, l_3, 160.0) \land \\
angleLL(l_1, l_3, 1.0472) \land \\
angleLL(l_1, l_3, -1.2222))
\]

\( \diamond \)

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

Let \( A = (G, C, P) \) be an abstract problem and \( \kappa \) an anchor from \( G \). We define \( \kappa.A \) as \( (G, \kappa.C, P) \).

**Example 5.3** If we represent points by pairs \((x, y) \in \mathbb{R}^2\) and straight lines by its equation in normal form \( ax + by + c = 0 \) and \( a^2 + b^2 = 1 \), then an example of anchor \( \kappa \) is

\[
\begin{align*}
\kappa(p_1) &= (92.38, 160) \\
\kappa(p_2) &= (0, 0) \\
\kappa(p_3) &= (290, 0) \\
\kappa(p_4) &= (245.54, 122.16) \\
\kappa(l_1) &= (-0.87x + 0.5y + 0 = 0) \\
\kappa(l_2) &= (-0.24x - 0.97y + 177.48 = 0) \\
\kappa(l_3) &= (0x - 1y + 0 = 0) \\
\kappa(l_4) &= (0.94x + 0.34y - 272.51 = 0)
\end{align*}
\]

The characteristic formula \( \Psi \) after applying the anchor \( \kappa \) to the instance problem \( \alpha.A \) in Example 5.2 is

\[
\Psi(\kappa.\alpha.A) \equiv (\text{onPL}(92.38, 160), (-0.87x + 0.5y + 0 = 0)) \land \\
onPL(92.38, 160), (0x - 1y + 0 = 0)) \land \\
onPL(0, 0), (-0.87x + 0.5y + 0 = 0)) \land \\
onPL(0, 0), (0.94x + 0.34y - 272.51 = 0)) \land \\
onPL(245.54, 122.16), (-0.24x - 0.97y + 177.48 = 0)) \land \\
onPL(245.54, 122.16), (0.94x + 0.34y - 272.51 = 0)) \land \\
distPP(0, 0), (290, 0), 290.0)) \land \\
distPP(290, 0), (245.54, 122.16), 130.0)) \land \\
distPP(92.38, 160), (0x - 1y + 0 = 0), 160.0)) \land \\
distPL(l_1, l_3, 160.0) \land \\
angleLL(l_1, l_3, 1.0472) \land \\
angleLL(l_1, l_3, -1.2222))
\]

\( \diamond \)

Since the sets of symbols \( P \) and \( G \) are disjoint, \( \alpha \) and \( \kappa \) commute, that is, \( \kappa.\alpha.A = \alpha.\kappa.A \).

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

\[
V(\alpha.A) = \{ \kappa \mid \Psi(\kappa.\alpha.A) \}
\]
Figure 6: A construction (of four) obtained by interpreting an instance plan.
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 \(\alpha.A\).

**Example 5.4** Figure 6 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 \(S \equiv (G, P, L, I)\) be an abstract plan with \(L = \{o_1, o_2, \ldots, o_n\}\). The characteristic formula of \(S\) is the first order logic formula,

\[
\Phi(S) \equiv \bigwedge_{i=1}^n \alpha_i
\]

where the geometric elements of \(G\), the parameters of \(P\) and signs of \(I\) occurring in \(\Phi\) are considered free variables.

**Example 5.5** The characteristic formula of the abstract plan \(S\) given in Example 4.1 is

\[
\Phi(S) \equiv (p_2 = pointXY(O_2, O_y) \\
\land p_3 = pointXY(d_1, O_y) \\
\land c_1 = circleCR(p_3, d_2) \\
\land l_5 = linePP(p_2, p_3) \\
\land l_6 = lineAP(l_5, a_1, p_3) \\
\land p_4 = interCL(l_6, c_1, s_1) \\
\land l_7 = lineAP(l_5, a_2, p_2) \\
\land l_8 = lineLD(l_5, h_1, s_2) \\
\land p_1 = interLL(l_7, l_8) \\
\land l_3 = linePP(p_2, p_3) \\
\land l_4 = linePP(p_1, p_2) \\
\land l_2 = linePP(p_1, p_4))
\]

Let \(\alpha\) 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 5.6** The characteristic formula of the instance plan in Example 4.2 is

\[
\Phi(\alpha.S) \equiv (p_2 = pointXY(O_2, O_y) \\
\land p_3 = pointXY(290.0, O_y) \\
\land c_1 = circleCR(p_3, 130.0) \\
\land l_5 = linePP(p_2, p_3) \\
\land l_6 = lineAP(l_5, -1.222, p_3) \\
\land p_4 = interCL(l_5, c_1, s_1) \\
\land l_7 = lineAP(l_5, 1.0472, p_2) \\
\land l_8 = lineLD(l_5, 160.0, s_2) \\
\land p_1 = interLL(l_7, l_8) \\
\land l_3 = linePP(p_2, p_3) \\
\land l_1 = linePP(p_1, p_2) \\
\land l_4 = linePP(p_3, p_4) \\
\land l_2 = linePP(p_1, p_4))
\]

Let \(S \equiv (G, P, L, I)\) be an abstract plan and \(\kappa\) an anchor from \(G\). We define \(\kappa.S\) as \(\langle G, P, \kappa.L, I \rangle\).

**Example 5.7** Let \(\kappa\) be the anchor in Example 5.3 and \(\alpha.S\) the instance plan in Example 4.2. The characteristic formula \(\Phi\) after applying the anchor \(\kappa\) to the instance problem \(\alpha.S\) is

\[
\Phi(\kappa.\alpha.S) \equiv ((0, 0) = pointXY(O_2, O_y) \\
\land (290, 0) = pointXY(290.0, O_y) \\
\land c_1 = circleCR((290, 0), 130.0) \\
\land l_5 = linePP((0, 0), (290, 0)) \\
\land l_6 = lineAP(l_5, -1.222, (290, 0)) \\
\land (245.54, 122.16) = interCL(l_6, c_1, s_1) \\
\land l_7 = lineAP(l_5, 1.0472, (0, 0)) \\
\land l_8 = lineLD(l_5, 160.0, s_2) \\
\land (92.38, 160) = interLL(l_7, l_8) \\
\land (0x - 1y + 0 = 0) = linePP((0, 0), (290, 0)) \\
\land (-0.87x + 0.5y + 0 = 0) = linePP((92.38, 160), (0, 0)) \\
\land (0.94x + 0.34y - 272.51 = 0) = linePP((290, 0), (245.54, 122.16)) \\
\land (-0.24x - 0.97y + 177.48 = 0) = linePP((92.38, 160), (245.54, 122.16)))
\]

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

\[
V(\alpha.S) = \{\kappa \mid \exists \iota \Phi(\iota.\kappa.\alpha.S)\}
\]

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 \(\alpha.S\).

**Example 5.8** Figure 6 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 for all index assignment \(\iota\) and for all parameters assignment \(\alpha\), there is at most one anchor \(\kappa\) that satisfies \(\Phi(\iota.\kappa.\alpha.S)\). In other words, for a fixed parameters assignment \(\alpha\), there is a one to one correspondence between an index assignment \(\iota\) and an indexed anchor \(\kappa\).

### 6 Constructive Solvers

According to the main concepts identified in the geometric constraint solving process, we propose to organize constructive geometric constraint solvers into three functional units: the analyzer, the index selector and the constructor.

#### 6.1 The Analyzer

The analyzer is the functional unit that computes an abstract plan \(S \equiv (G, P, L, I)\) from an abstract problem \(A \equiv (G, P, C, P)\).

The set of abstract problems \(A\) for which an analyzer is able to compute a construction plan \(S\) is the analyzer domain.

The only 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 $A$ must coincide with those in $S$. However, we are interested in analyzers that solve the abstract problem. This statement can be formalized as follows.

We say that an analyzer is correct if and only if for every abstract problem $A$ in its domain, and for every parameters assignment $\alpha$, $V(\alpha.S) \subseteq V(\alpha.A)$.

We say that an analyzer is complete if and only if for every abstract problem $A$ in its domain, and for every parameters assignment $\alpha$, $V(\alpha.S) = V(\alpha.A)$.

Of course, we are interested in complete analyzers but correct analyzers are sufficient. Apparently, the analyzers described in [1, 2, 7, 13] are complete.

Example 6.1 Since the abstract plan $S$ in Example 4.1 has been generated by a complete analyzer from the abstract problem $A$ in Example 3.1, the set of indexed anchors of the instance plan $\alpha.S$ in Example 4.2 coincides with the set of realizations of the instance problem $\alpha.S$ in Example 3.2. See Figure 6.

6.2 The Index Selector

An index selector is a functional unit exclusively characterized by its output. The output of an index selector is an index assignment $\iota$.

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

1. A trivial index selector computes an index assignment $\iota$ from the scratch. For instance, $\iota(s) = +1$ for all $s$ in $I$.

2. A successor (predecessor) index selector computes the next (previous) index assignment $\iota'$ from an index assignment $\iota$. For instance, the index assignment $\iota$ from $I = \{s_1, \ldots, s_n\}$ can be represented by the binary number $d_1d_2\ldots d_n$ where $d_i = 0$ if $\iota(s_i) = -1$ and $d_i = 1$ if $\iota(s_i) = 1$. Now we have an order relation on index assignments induced by the order relation on the binary numbers and, thus, the next (previous) index assignment is well defined. 

3. An anchor-based index selector computes an index assignment $\iota$ from an anchor $\kappa$ and an abstract plan $S = (G, P, L, I)$. Let $\alpha_\kappa$ be a parameters assignment taken from the anchor $\kappa$. The index assignment $\iota$ is such that $\Phi(\kappa, \iota, \alpha_\kappa, S)$ holds. See [10] for an extensive analysis of methods for implementing index selectors.

6.3 The Constructor

The constructor is the functional unit that computes an anchor $\kappa$ from an abstract plan $S$, a parameters assignment $\alpha$ and an index assignment $\iota$. The anchor $\kappa$ 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 are going to present a software architecture useful for building a geometric constraint solving toolbox. The aim of such a toolbox is to provide the software engineer with a set of tools to design and implement software applications founded on constraint solving. We have successfully implemented this toolbox and a 2D parametric profile editor layering on it. Additionally, the toolbox design is easily extendable. In [10] we extend the capabilities to provide new kinds of selectors and in [6] it is shown how to extend the toolbox to cope with requirements of concurrent engineering applications.

The architecture have two main components: functional units and data items. Both kinds of components closely follows the entities described in the preceding sections. The data items are geometric constraint problems, constructions plans, parameters assignments, geometry assignments and index assignments. These data items are composed of smaller items as shown in the previous sections. The functional units are analyzers, selectors and constructors. All these components relates each other following the data flow diagram of figure 7.

To illustrate how this toolbox works we are going to roughly explain how to implement a parametric 2D profile editor using it. Assume that the editor is an interactive application. By using its graphical interface, the user draws a sketch of the profile that he wants to design without paying too much attention to the real dimensions. After that, symbolical dimensions are given by drawing labeled dimension lines on the sketch. We should assign actual values to this symbolical dimensions manually or automatically measuring them on the sketch. In figure ?? we show an image of our prototype editor with an sketch being defined.

The following step is to solve the sketch. This is done by obtaining from the graphical sketch the corresponding abstract problem and the parameters assignment. If the abstract problem is in the solver domain, it is solved by the analyzer and an abstract plan is obtained. An index assignment for this abstract plan can be computed by using the trivial selector described in section 6. By using the abstract plan, the parameters assignment obtained from the sketch and the index assignment, the constructor obtains a realization of the original sketch. That is an anchor that fulfills the constraints given in this sketch.
In general, this is not the only existing realization. By using the graphical interface of the application the user can ask it to compute the next realization. This is done by using the successor selector on the actual index assignment to obtain the index assignment corresponding to the next realization and running again the constructor. Note that it is not mandatory to run again the analyzer because the abstract problem have not changed. This process can be repeated to visit all the possible realizations.

In many cases the user interest is to obtain a realizations which looks like the sketch. This can be done applying the anchor-based selector and directly computing the index assignment of this particular realization. In [10] the implementation of this and more powerful selectors is discussed.

Finally we describe how the editor proceeds when the user modifies the actual value of any parameter. This is usually done by using the graphical interface of the application. In response to this interaction, the application recomputes the corresponding parameters assignment and apply again the constructor. A new realization is obtained in which the corresponding dimension has changed. Note that the new realization looks like the previous one because the index assignment has not changed. Observe too that only the constructor functional unit has been executed to resolve the change.

8 Summary

To illustrate the concepts, we have used functional capabilities which are specific of the ruler-and-compass constructive geometric constraint solving technique. But they 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


