Formal specifications and data exchanges in GCS domain: a Geometric Constraints Mark-up Language

Pascal Schreck, Pascal Mathis,
Arnaud Fabre, Julien Wintz

Université de Strasbourg - LSIIT, UMR CNRS 7005

July 2012

Geometric constraint solving: several domains, several contexts

- Education: Statement → program of construction;
- Technical drawing: sketch → precise drawing;
- Architecture, photogrammetry (projections → 3D-objects);
- molecule problem, robotic (distance geometry) ...

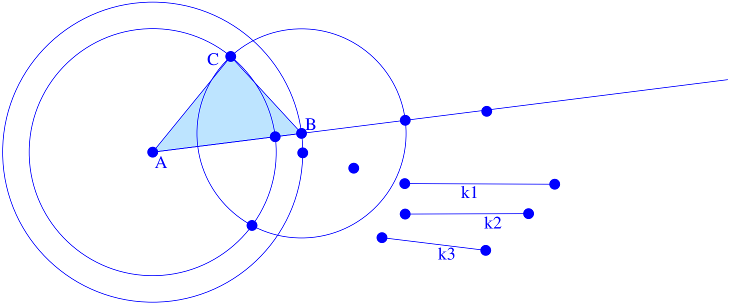
Even in a same domain, there are very different ways to express geometric constraint systems.

Examples in DGS

Statement

Given three lengths k_1, k_2 and k_3, construct triangle (A, B, C) such that $AB = k_1$, $AC = k_2$ and $BC = k_3$.

And a construction using kig:

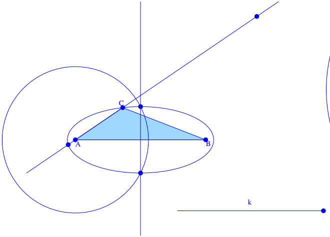


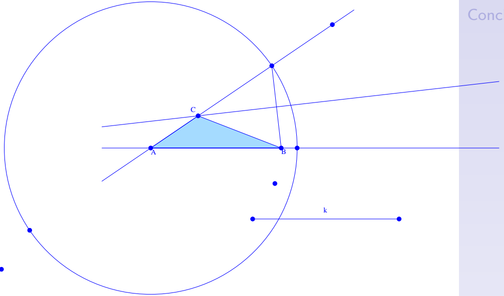
Example in education

Statement

Construct a triangle (A, B, C) given its base $[A, B]$, a base angle $\alpha = \angle(AB, AC)$ and sum $k = AC + CB$ of the other two sides.

And *two* possible constructions using kig:





Pascal Schreck, Pascal Mathis, Arnaud Fabre, Julien Wintz

Another example in education

Statement

Construct a circle tangent to a given circle Γ and to a given line Δ at a given point A on Δ.

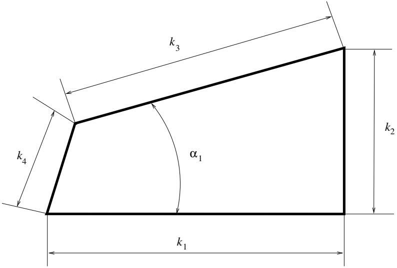
There is at least four constructions using four different principles:

- addition of distances;
- invariance by homothetic transformation;
- use of radical axis and radical center;
- use of an inversion.

Example of problem in CAD

Questions

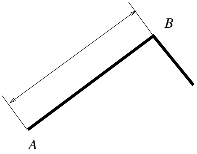
- implicit constraints?
- nature of the constraints (k_2?)
- involved objects?
- orientation?
- ...

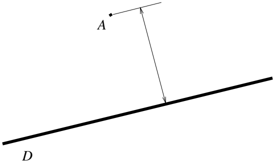


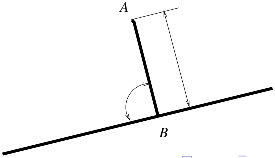
Example in CAD

Normalization?

Usually a double arrowed line indicates a distance constraint between two points. But according to the relative positions of objects it can indicates a constraint of distance between a point and a line. Some solver (Sunde method) are not able to natively take this kind of constraint into account:

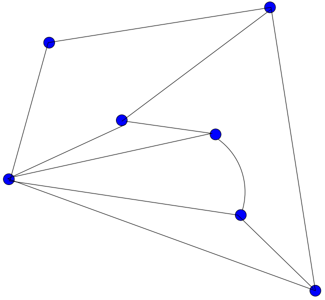


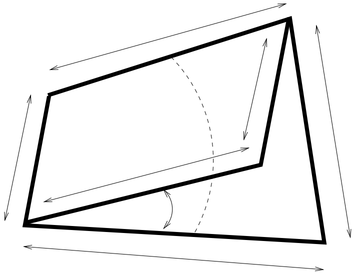




Example of CAD constraint solver

Constraint graph.

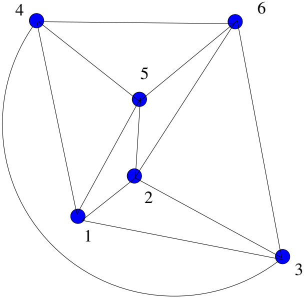


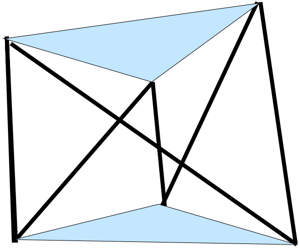


Example of molecular modeling/robotics

Steward plateform and Cayley-Menger determinant

two 3D solid blocks linked by 6 hydraulic cylinders.





Classically it would take $6 \times 3 - 6 = 10$ equations to fully modelize the constraints. Using Cayley-Menger resultants reduces this number to 2 :

$D(1, 2, 5, 6, 3) = 0$

$D(1, 2, 5, 6, 4) = 0$

Example in axiomatization of geometry

Lines and collinearity

In incidence geometry, one can consider only points with a collinearity relation or points *and* lines with an incidence relation.
As a consequence, the pseudo-associativity of collinearity is treated very differently according to that choice. Moreover, particular (or degenerate) cases are very different.

Problematics

Previous examples could be seen as avatars of the data exchange problem (usually solved by standard(s) for industry and common file format(s)). We focus here on GCS problem and we want to deal with the following questions

Problems

- how to understand the examples given by our colleagues in order to test their benchmarks on our solvers?
- how to translate a problem in a context in another context automatically (compilation) or by hand with the help of some specifications?
- how to let some agents collaborate into a x-agents with blackboard architecture?
- how to easily extend a solver prototype?

Problematics

Since the beginning of our works about GCS, we choose to consider a meta-framework within a very pragmatic viewpoint. Then, we designed a meta-language (a rather general one) so-called GCML using XML framework:

- based on multi-sorted logic,
- with the possibility of describing several different semantics
- with descriptions of morphisms *à la* OBJ3.

The language comes with some tools with usually a short cycle of life (< 3 years $=$ mean time of a thesis). The current tools are developed and maintained by Pascal Mathis.

Syntax and semantic

$$x^2 + 1 = 0$$

Signature and syntax

A (first order) signature Σ is a triple $< S, F, P >$:

- S: set of sorts;
- F: $S^* \times S$-indexed set of functional symbols;
- P: S^*-indexed set of predicative symbols.

With such a signature, we can construct terms and formula as usual. Then, a constraint system is a conjunctive of positive formula, or a set of predicative terms (C, X, A) where parameters and unknowns are distinguished.

GCML

```
<gcml>
<syntax>
<doc> very simple 2D universe </doc>
<sorts>
   point
   scalar
</sorts>
<fsymbols>
   initScalar : -> scalar
   initPoint : scalar scalar -> point
   initPoint2 : point scalar -> point
   mkPoint_2p2s : point point scalar scalar -> point
</fsymbols>
<psymbols>
    distpp : point point scalar
</psymbols>
```

A constraint system

```
   <gcml> <include ref="3d_universe.gcml"/> <gcs>
    <syntax>
 <unknowns>
 Point p1 p2 p3 p4
 </unknowns>
 <parameters>
 Length k1 k2 k3 k4 k5 k6
 </parameters>
 <constraints>
 distance(p1,p2) = k1
 distance(p2,p3) = k2
 distance(p1,p3) = k3
 distance(p1,p4) = k4
 distance(p2,p4) = k5
 distance(p3,p4) = k6
 </constraints>
 </syntax>
```

axioms

```
<axioms>
<properties>
  <property> distpp(p1,p2,l)=distpp(p2,p1,l)
  </property>
</properties>

<construction>
<if>
  distpp(p1,p2,l1)
  distpp(p1,p3,l2)
  distpp(p1,p4,l3)
</if> <then>
  p1=mkPoint(p2,l1,p3,l2,p4,l3)
</then>
</construction>
</axioms>
```

Morphisms

Not implemented yet, some ideas come from OBJ3:

The simpler ones are signature morphisms (for instance by forgetting lines)

But we also could consider more complicated things;

- relational to functional :
  $\text{distpp}(A, B, k) \mapsto \text{dist}(A, B) = k$.
- use of constructions :
  $\text{tgt}(\Gamma, \Delta) \mapsto \text{distpl}(\text{center}(\Gamma), \Delta) = \text{radius}(\Gamma)$

Multi-semantics framework

Usually, we have to consider a Σ-algebra where sorts are interpreted by sets, functional symbols by functions, predicative symbols by relations.

In addition, the interestiong Σ-algebra are models of the specification (the axioms are fulfiled).

Here, we have a very loose notion of semantic (remember: pragmatism): a sort could be "interpreted" by an explicative sentence, or a number representing its degree of freedom ...

```
<semantics>
 <semantic name="textuelle">
   <point>Let %name be a point.</point>
   <scalar>Let %name be a real.</scalar>
   <distpp>
    The distance between %arg1 and %arg2 is %arg3.
   </distpp>
  </semantic>
 <semantic name="cartesienne">
   <point> real @x,@y,@z </point>
   <length> real @l </length>
 </semantic>
```

```
<semantic name="C++"> <use name="cartesienne">
 <point>
  class Point
  {
   private :
    float %x, %y, %z;
   public :
    Point(float x,y,z) : %x(%), %y(y), %z(z)
  };
 </point>
 <initPoint>
  Point initPoint(float x, float y, float z)
  {
   return Point(x,y,z)
  }
 </initPoint>
```

```
<semantic name="graphique"> <use name="C++"/>
  <point>
    glVertex(%x, %y, %z)
    glPrint(%name, %x + 0.1, %y, %z)
  </point>
...
  <distpp>
    glBegin(GL_LINE)
    $point(%arg1)
    $point(%arg2)
    glEnd()
    glPush()
    glTranslate((%arg1.%x + %arg2.%x)/2,
    (%arg1.%y + %arg2.%y)/2,
    (%arg1.%z + %arg2.%z)/2)
    $length(%arg3)
    glPop()
  </distpp>
</semantic>
```

```
<semantic name="combinatoire">
    <scalar>1</scalar>
    <measure>1</measure>
    <point>3</point>
    <line>4</line>
    <plane>4</plane>
    <onl>2</onl>
    <onP>1</onP>
    <norm>1</norm>
    <onlP>2</onlP>
    <angle2l>1</angle2l>
    <angle3p>2</angle3p>
    <distpp>1</distpp>
...
    <distlP>1</distlP>
    <distPP>3</distPP>
</semantic>
```

```
<semantic name="equations">
  <measure>(m)</measure>
  <point>(x,y,z)</point>
  <distpp>
    <param>
       (x_1,y_1,z_1) (x_2,y_2,z_2) (m)
    </param>
    <equations>
    (x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2 - m^2
    </equations>
  </distpp>
</semantic>
```

Statement, sketch and semantics

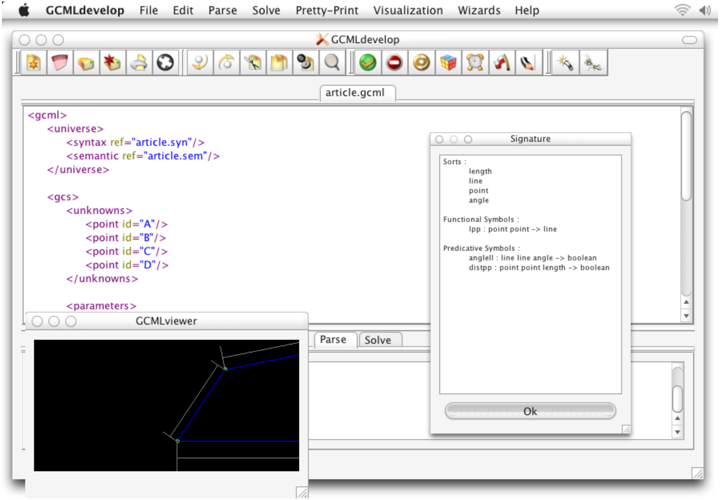
```
 <semantic>
  <sketch>
   p1=initPoint(0,0,3)
   p2=initPoint(1.67,0,3)
   p3=initPoint(1,3,4)
   p4=initPoint(0,2,1)
   l1=1
   l2=2
   l3=2
   l4=1
   l5=2
   l6=1
  </sketch>
</semantic>
```

Statement, sketch and semantics

```
<semantic>
  <valuation>
   l1=3
   l2=4
   l3=5
   l4=1.01
   l5=0.1
   l6=3.87
   p1=initPoint(0,0,3)
   p2=initPoint(l1,0,3)
   p3=initPoint(l1,l2,3)
   p4=mkPoint(p1,l4,p2,l5,p3,l6)
  </valuation>
</semantic>
```

Tools

Gcml-writer/sketcher



Tools (continued)

syntactic analyzer(s)

- strict xml: Julien Wintz (2004)
- an ad hoc one written in Ruby: Arnaud Fabre (2007)
- xml with qt library: Pascal mathis (present)

Tools (continued)

muc: a meta-universe compiler

- muc is written in Perl
- input: a gcml description of a geometric universe with the semantic(s) needed to have a geometric formal solver (rules and numerical interpretation) or a numerical one (equationnal semantic).
- output: C++ code

Conclusion

- We followed a pragmatic way;
- gcml is not used as a suport for data exchange;
- but this approach is still used in our team ... and elsewhere