Dynamic Geometry and Theorem Proving
a collection of ideas for cooperation

n.n.
Florian Haftmann, TU Munich
Predrag Janicic, Univ.Belgrade
Julien Narboux, Univ. Strasbourg
Walther Neuper, TU Graz
n.n
June 24, 2010

Abstract
This is a draft under construction which collects ideas for cooperation initiated at Castro Urdiales Feb.2010. The basic story is to integrate (certain aspects of) dynamic geometry systems, computer-aided theorems proving and computer algebra systems (see below for our motivation). The ideas, the texts and persons comprised in this draft probably will be integrated into a proposal for a larger project. The authors named above are all in the process of contacting further people which might be interested.

(((These are meta-comments, i.e comments on the text.)))

1 Motivation
The ideas collected here are driven by the motivation to bring together dynamic geometry systems and various aspects of computer-aided theorems proving. We think that both worlds have their individual strengths (appealing visualization and rigorosity respectively) which could benefit from each other. A system comprising a combination of both would form a suitable base for an educational system in geometric constructions.

(((FH: This was a first try to formulate this; I think that we don’t have to be overly precise here, but the reader will benefit if the different points not just fall down from the sky. The CAS aspect is still not very well represented, though.)))

1.1 Terminology

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Term and explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATP</td>
<td>Automated theorem proving, includes methods which usually are run on CAS (e.g. Gröbner bases)</td>
</tr>
<tr>
<td>CAS</td>
<td>Computer algebra system</td>
</tr>
<tr>
<td>CTP</td>
<td>Computer theorem proving, comprises ATP and ITP</td>
</tr>
<tr>
<td>DGS</td>
<td>Dynamic geometry system</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated development environment like Eclipse, jEdit, NetBeans</td>
</tr>
<tr>
<td>ITP</td>
<td>Interactive theorem proving</td>
</tr>
<tr>
<td>RP</td>
<td>Repository of theorems</td>
</tr>
</tbody>
</table>
1.2 Existing systems

The vision is that we integrate existing systems rather than starting from scratch. For convenience we give here a list of systems which could be involved. This list is not supposed to be a final decision, just an enumeration of candidates which we deem suitable according to our current knowledge and are represented to a certain extent among the authors; other suitable systems may be added accordingly, while not all systems listed here may turn out to be really suitable in the end.

DGS
- GeoGebra
- Cinderella
- GCLC Sect.

ITP
- Coq
- Isabelle/HOL [NPW02]

ISAC is an experimental system in an ongoing project \(^1\) which provided a proof of concept for the CTP-based tutoring technology described in Sect.3.5.

RP
- GeoThms

CAS

...\(^2\)

(((not yet wholly compiled)))

2 ATP and ITP in geometry

the link between Automated Deduction in Geometry and Interactive Theorem Proving (this would involve Nicolas Magaud):

2.1 A domain-specific proof language for proofs in geometry within existing infrastructure

In the early days of interactive theorem proving, the task of writing “proofs” meant to write down so-called tactic scripts which consisted of a programmatic combination of so-called tactics of which each would implement a particular proof strategy. This was successful but turned out to be inconvenient: tactic scripts seldom resemble the abstract human-invented ideas behind a proof, proof scripts are brittle, and a tactic script refers to an implicit goal state which is not represented in the script itself, making maintenance difficult.

To overcome this, declarative proof languages were developed, starting with the Mizar system [Rud92] and later on the proof languages Isar for Isabelle [Wen99] and C-zar for

\(^1\)http://www.ist.tugraz.at/projects/isac/

\(^2\)
Coq [Cor07]. The core idea is that instead of writing a programatic description 
how to accomplish a proof, the proof writer gives a proof text describing 
what to prove in each 
step, where each step needs to be justified by either a subproof or a terminal tactical proof. 
This means a proof writer is free to express his ideas about the proof in his text such that a 
reader can follow his thoughts accordingly.

FIXME: explain idea of domain-specific proof language here

2.2 Interactive formal proofs

Trying to have readable interactive formal proofs in geometry: this requires maybe a lan-
gage, but also some proof automation techniques. (this could involve Florian)

2.3 Formalisation of algebraic methods for ATP

Formalization of the Wu’s method and full angle method in Coq and maybe in Isabelle (this 
could involve Florian)

2.4 Automated generation of formal proofs

Generate readable formal proofs automatically: proof by symmetry, using automation to 
solve degenerated cases, ... (this could involve Florian)

There has been a lot of efforts invested in automated theorem provers in geometry 
and also in developing collections of formalized geometry knowledge. But, it is still to link 
these two and to develop efficient systems that automatically generate formal proofs. There 
are already systems that use the Groebner bases method or the area method for generating 
formal proofs of geometry conjectures. However, it would be welcome to develop systems 
that automatically generate synthetic (hopefully readable) proofs of geometry conjectures. One option is to use coherent logic as a framework for that.

3 Ruler & compass geometry constructions

3.1 Ruler and compass construction in ITP

Define and formalize the notion of ruler and compass construction in a proof assistant. ITP 
systems could be used to ensure that construction steps given by the user are applicable and 
consistent.

3.2 Check constructions correct by ATP

Check that the constructions given by an oracle (a student or a software) are correct. 

Check by ATP (and possibly by ITP, following links from 2.3) that the construction 
meets the given specification.

3.3 Generate constructions automatically

For a given specification, generate a ruler & compass construction automatically. This 
automated problem solving could be aided by ATP systems (based on existing methods or 
some new ones)
3.4 Constructions and ATP/ITP

The link between Constructions and Interactive/Automated Theorem Proving (this could involve Pascal Schreck, Predrag, Florian... )

This work is expected to provide prerequisites for Sect.3.5, in particular for Pt.2c and Pt.2e below.

3.5 User guidance in stepwise construction

Established concepts and technologies shall be combined to a novel kind of tutoring technology:

1. We take an existing special purpose language for geometry, for instance [\ldots]
2. We adapt the Lucas-interpreter [n.n], which has undergone the proof of concept in the $\Sigma$4C-project [iT] for algebra, to geometry. This interpreter automatically sets break-points at each tactic $^2$. At a break-point a dialogue module takes over control and decides which of these choices are given to the learner during construction of a composed geometric object:
   
   (a) the learner just might request the next step of construction from the system: then the interpreter just resumes execution.
   
   (b) the learner inputs an arbitrary geometric object as suggestion for the next step towards the solution. The construction expanded by this object is proven equivalent with the intermediate constructions, which are generated successively in the continuation of the program. For such proofs (or disproofs) we expect to take profit from ATP and from results envisaged in Sect.3.2.
   
   (c) the learner asks the system to explain one of the construction steps. This is a concern of ITP equipped with the notion of ruler and compass construction as introduced in Sect.3.1. In particular, we expect to take profit from ATP/ITP-linkage envisaged in Sect.3.4.

3. Given a particular geometric construction example, provide context (sensitive to particular steps) access to
   
   (a) axioms and theorems mechanised in Sect.3.1, which are human readable
   
   (b) respective proofs according to Sect.2.2, which are human readable.

Authoring in such a system is programming in the language introduced in Pt.1. Thus authoring concerns only the mathematical aspects of tutoring. Aspects of learning theory are separated to the dialogue module introduced in Pt.2 above.

Authoring also might take profit from results obtained in Sect.3.3 and Sect.2.4.

4 Presentation of geometry knowledge

4.1 Electronic geometry textbooks

Build electronic (not necessarily an online) book about geometry which would be both dynamic and (and at least partly) formally proved, and link it with exercises supported with ATP. (this could involve everyone) Also, (at least some) conjectures form textbooks should be automatically proved by ATP.

$^2$Tactics are those statements in the language, which construct a geometric object.
4.2 Build geometry repositories

Build a TPTP collection of geometry problems (starting with GeoThms). Conjectures should be accompanied by (when possible):

- formal proofs (not necessarily automatic)
- automatically generated proofs (not necessarily formal)
- dynamic figures (for different DGS)

(this could involve Pedro)

5 Interoperability between components

This collection of ideas involves quite a couple of software components from different disciplines — and all of them are expected to take advantage of interaction with at least one of the other components. So interoperability is an issue.

5.1 Interchange format between DGS

TODO intergeo

5.2 Interaction between CTP and DGS

A recent development in ITP user interfaces could have beneficial impact on our aims: the document model presented in [Wen10]. It is motivated by the observation that the typical ITP platforms barely provide support for contemporary editing technology and it would be waste of resources to build editors from scratch. Therefore it is proposed to design ITP systems in a way that they can interact with existing editor frameworks. One should refrain from the temptation to model arbitrary details of the ITP system in the user interface since this would be bare duplication with all negative effects like keeping versions consistent etc. Instead the interaction between ITP system and user interface is kept to a simple document model which supports basic activities like open new document, insert document fragment, drop document fragment, conclude document. The ITP system judges which parts of the document are to be considered valid or not and reports on that to the user interface which visualizes this to the end user accordingly (e.g. red underlining), thus mimicking the look-and-feel of contemporary IDEs for programming languages. Hence the user interface need only the knowledge whether something is valid or not, but not why. A concrete instance of this document model is presented using Isabelle/HOL and the JVM-based jEdit editor framework, but the ideas can be transferred to similar components.

In our scenario this would mean that a visual user interface (DGS) would map actions in a geometry scenario (draw circle, connect dots, construct triangle, . . .) on changes of a underlying document; the checking is done within the ITP which gives feedback which steps are valid etc.

References


[n.n] n.n. Tutoring as a side-effect of programming. under joint construction with the Complang Group at TU Vienna.


