Available online at www.sciencedirect.com
ScienceDirect
Computers & Education 51 (2008) 1283–1303
COMPUTERS & EDUCATION
www.elsevier.com/locate/compedu

An algorithm for automatic checking of exercises in a dynamic geometry system: iGeom

Seiji Isotani [a,*], Leônidas de Oliveira Brandão [b]

[a] The Institute of Scientific and Industrial Research, Department of Knowledge Systems, Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka 567-0047, Japan
[b] The Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, São Paulo, SP, 05508-090, Brazil

Received 25 October 2007; received in revised form 6 December 2007; accepted 23 December 2007

Abstract

One of the key issues in e-learning environments is the possibility of creating and evaluating exercises. However, the lack of tools supporting the authoring and automatic checking of exercises for specifics topics (e.g., geometry) drastically reduces advantages in the use of e-learning environments on a larger scale, as usually happens in Brazil. This paper describes an algorithm, and a tool based on it, designed for the authoring and automatic checking of geometry exercises. The algorithm dynamically compares the distances between the geometric objects of the student's solution and the template's solution, provided by the author of the exercise. Each solution is a geometric construction which is considered a function receiving geometric objects (input) and returning other geometric objects (output). Thus, for a given problem, if we know one function (construction) that solves the problem, we can compare it to any other function to check whether they are equivalent or not. Two functions are equivalent if, and only if, they have the same output when the same input is applied. If the student's solution is equivalent to the template's solution, then we consider the student's solution as a correct solution. Our software utility provides both authoring and checking tools to work directly on the Internet, together with learning management systems. These tools are implemented using the dynamic geometry software, iGeom, which has been used in a geometry course since 2004 and has a successful track record in the classroom. Empowered with these new features, iGeom simplifies teachers' tasks, solves non-trivial problems in student solutions and helps to increase student motivation by providing feedback in real time.

Keywords: Dynamic geometry; Automatically checking exercises; Distance education; Geometry; iGeom

1. Introduction

The computer has been used in education since its earliest appearance in 1945.[1] However, it was only in the 1980s, with the emergence of "personal computers" (PCs), that the use of this machine and its resources (e.g.,

* Corresponding author. Tel.: +81 6 6879 8416; fax: +81 6 6879 2123.
E-mail addresses: isotani@acm.org (S. Isotani), leo@ime.usp.br (L. O. Brandão).

[1] To learn more about the history of the computer, visit the timeline of computing history at http://www.hofstra.edu/ComputingHistory.

0360-1315/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2007.12.004

software) significantly impacted teaching and learning processes (Oldknow, 1997). Today, in Brazil and other developing countries, there has been a large expansion in the use of the Internet for teaching. Consequently, the demand for research in this area has considerably increased (Litto, 2006). The large-scale distribution of cheap computers for teaching in Brazil has introduced a new stimulus for developing local software with the capability of functioning in computing equipment with low-processing capacities.

The development of environments for distance education and e-learning through the Internet have not only allowed for an easier and faster diffusion of information and knowledge, but has also helped with the introduction of courses using the flexibility of schedules and places (Hentea, Shea, & Pennington, 2003; Litto, 2006). In this context, these online environments that support teaching and learning processes have become virtual classrooms, where students and teachers can communicate and interact using tools, including chat rooms, forums, emails, wikis, virtual blackboards, etc.

The use of the Internet and computers can bring great benefits to the teaching of mathematics, but in order to achieve such goals, it is necessary to choose and create appropriate programs and methodologies that take advantage of the computer's positive characteristics. Good examples of this include dynamic geometry (DG) programs, which can benefit teaching and learning processes (Ruthven, Hennessy, & Deaney, in press; Santos & Sola, 2001).

Dynamic geometry can be understood as an alternative to traditional geometry, which makes use of a ruler and compass and produces static constructions. When using traditional methods to teach geometry, if a student, after accomplishing a construction, wants to analyze the same construction using some of the objects in another disposition, he or she needs to repeat the entire construction. However, with DG, used today to specify geometric equations implemented on the computer, objects can be freely manipulated across the screen, maintaining all of the constraints and properties initially established during construction.

DG programs have proven to be an excellent resource for teachers and students (Botana & Valcarce, 2002; Ruthven et al., in press; Sinclair, 2005). In spite of the great benefits for teaching and learning, as well as the existence of useful DG programs, including GSP (Jackiw, 1995), Cabri (2007), Cinderella (Kortenkamp, 1999), C.a.R. (Grothman, 1999), and Tabulae (Moraes, Santoro, & Borges, 2005), DG is not commonly used in adaptive distance education environments. The main reason for this is the lack of tools for authoring and automatically checking exercises that allow communication between servers and the adaptation of their interface (e.g., show/hide tools to draw objects), according to the learning context.

The importance of the evaluation or checking students' exercises is a reality in both classroom and e-learning environments. According to Hara and Kling (1999) and Hentea et al. (2003), one main frustration of students engaged in e-learning courses (including blended learning[2] courses) is the lack of feedback or the limitation of an immediate evaluation of completed exercises. In this context, using the usual DG programs generates some difficulties for students and teachers, including the fact that the student is unaware of whether his or her solution is correct and must save the solution and send it by email (or another medium) to the teacher in order to receive an evaluation. Moreover, in order to gain access to the student's solution, a teacher needs to receive the email (or other form of response), open it using a DG program and check each construction. Throughout this cumbersome process, the elapsed time eliminates one advantage of the Internet, which is to provide a faster and more interactive environment for learning. This loss of time makes a teacher's task of helping students improve learning through the use of technology much more difficult.

The answer to this problem is to make DG programs more effective when used in e-learning environments. The following can be implemented to facilitate that process: (a) generate tools that produce and automatically check student exercises, allowing for faster authoring and feedbacks; and (b) develop tools allowing for communication between DG programs and learning management systems (LMS) that facilitate distribution, management, adaptation, and reception of exercises or any other content.

Today, two DG programs possess tools for authoring and checking exercises, including Cinderella (Kortenkamp, 1999) and C.a.R. (Grothman, 1999). The contributions of these programs inspire many teachers to include DG programs in classroom activities. However, in the current versions of these programs, there are some limitations for their use on distance education courses over the Internet. For instance, it is not pos-

[2] Blended learning is broadly used to define a course that combines usual classroom activities and online activities.

sible to satisfactorily link these programs to LMS, allowing for effective management and/or adaptation of the created content. This means that these programs cannot personalize their content in real time nor consider teacher and student preferences that would increase success in the learning process.

The purpose of this paper is to present the development of algorithms and tools for authoring and automatically checking exercises in a DG program, called iGeom, which can be used for both web pages and as a stand-alone version. Conventional algorithms, based on automatic theorem proving techniques, to check geometry constructions, are very time- and machine-consuming and, sometimes, cannot check complex constructions in real time (Chou, Gao, & Zhang, 2000; Gallier, 2003; Gilmore, 1970). The developed algorithm of this paper is not as formal as a proof, but does enable the instantaneous checking of a geometric exercise by comparing it with a solution's template. The main benefits of this approach include a program that is (a) fast enough to be used over the Internet to support the use of DG programs in distance education courses; (b) able to check any kind of geometric construction and return whether the construction has the necessary properties to solve an exercise; and (c) able to run on computers with low-processing capabilities, which is very important in Brazil, where many computers used in public schools are out-of-date. Furthermore, we will present results using this algorithm, implemented on iGeom, together with an under development learning management system, referred to as SAW (Moura, Brandão, & Brandão, 2007). This paper is structured as follows: In Section 1, we will briefly describe the DG program, iGeom, compare its functionalities with other DG programs and outline its potential use when together with LMS. In Section 2, we show iGeom functionality for authoring and automatically evaluating exercises. In Section 3, we present the automatic evaluation algorithm. In Section 4, we show how geometric constructions can be exported to the Internet. In Section 5, we present the results of the application in a compulsory discipline offered for an undergraduate mathematics course at the University of Sao Paulo. Finally, in Section 6, we present the conclusions of our study.

1.1. iGeom: DG on the internet

The iGeom program is a complete, multi-platform, DG system (DGS) developed at the Institute of Mathematics and Statistics at the University of São Paulo (IME–USP). It has been under development since 2000, under the direction of Professor Leônidas O. Brandão (Brandão & Isotani, 2003). The iGeom program is implemented in Java and can be used as a stand-alone program or as an applet. Like other DG programs, the current version of iGeom allows users to perform all of the basic operations of DG, including the creation of geometric objects, such as points, lines, circumferences, and dynamic measures (e.g., angles); the modification of characteristics of the objects (e.g., color, size); options for saving or recovering constructions in different formats; and other advanced resources. Besides these, iGeom has special resources for creating scripts (functions used to create automatic geometric constructions) and fractals. Fig. 1 depicts the interface of the iGeom, including fractals created using a recursive script.

The iGeom has been developed considering the computational restrictions of computers in Brazil. This philosophy makes iGeom run efficiently in both residential and personal computers that have low-processing capabilities (e.g., computers using a microprocessor similar to Intel i386). Furthermore, it can be used without restrictions with any other learning management system (Brandão, Isotani, & Moura, 2005).

In Table 1,[3] we show some of the resources available with the iGeom program, in comparison with other DG programs. The column *Portability* refers to the DG programs that can be executed in any platform. The column *ADDs* refers to DG programs that allow for the opening of multiple drawing areas. The column *Script* contains those DG programs that possess resources for script creation. In the column *Rec,* those DG programs that allow for the creation of recursive scripts (allows for the construction of fractals) are listed. In the column labeled *Web* are DG programs that allow for the unrestricted Internet use of these resources. In the column labeled *Com* are DG programs that possess communication resources. In the column titled *AA* are DG programs that possess resources for authoring and the automatic evaluation of exercises. Finally, the column *License* refers to the license type each program possesses.

[3] The analysis of resources was done in 2004.

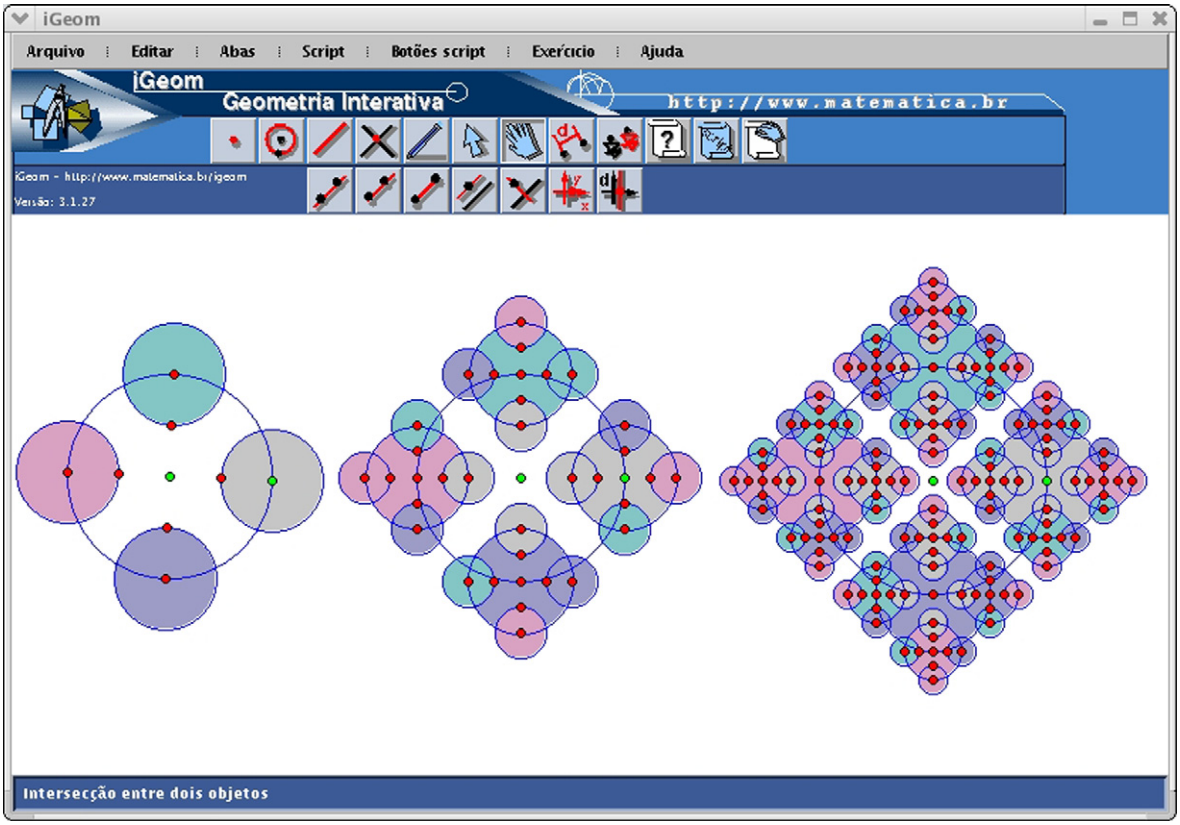

Fig. 1. The interface of the iGeom showing fractals created using recursive script.

Table 1
Comparison between the resources of some DGS

Program	Portability	ADDs	Script	Rec	Web	Com	AA	License
iGeom	X	X	X	X	X	X	X	Free
Cabri		X	X					Commercial
C.a.R	X		X		X		X	GNU
Cinderella	X		X		X		X	Commercial
GSP		X	X	X				Commercial
Tabulae	X	X			X	X		Commercial

In 2004, a project was initiated to develop a learning management system that could easily incorporate applets into educational modules. This system was referred to as SAW (Brandão, Isotani, & Moura, 2004; Moura et al., 2007). The iGeom program has been used together with SAW in different courses. One of these includes a course directed towards public school teachers and people interested in teaching mathematics using computers. Another use includes an obligatory discipline offered for a degree course in mathematics, titled "*Notions of teaching of mathematics using computers*". Each year, more than a hundred students and teachers are enrolled in these courses. The use of iGeom + SAW, including some of the obtained results using the automatic checking tool on the Internet, will be shown in detail in Sections 4 and 5.

2. Authoring and automatically evaluating exercises

The action of evaluating or checking exercises is a complicated process (Chou et al., 2000; Gelernter, 1963; Kortenkamp, 1999). Because of this, automation is not a trivial matter. It requires the application

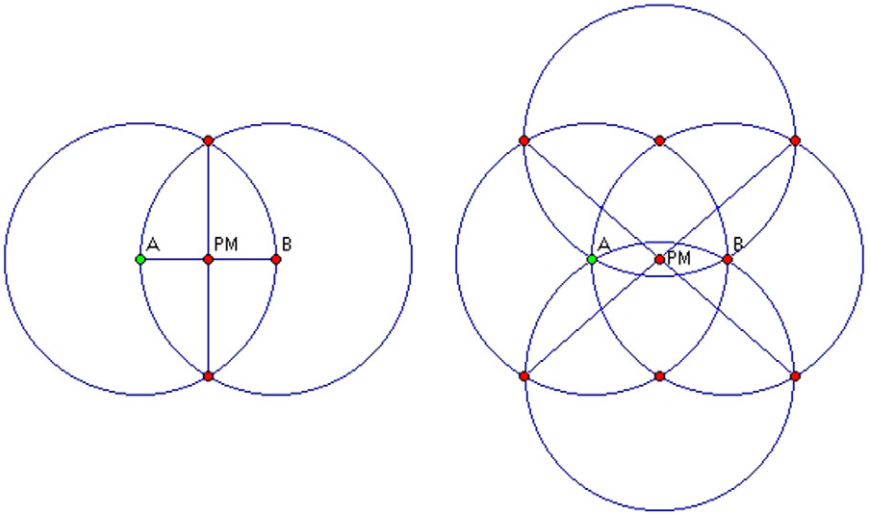

Fig. 2. Two different constructions of the middle point.

of heuristics or simplifications. One of the difficulties in developing an algorithm that enables automatic checking in GDS is the multiplicity of different (and correct) solutions for the same problem. To illustrate this point, consider the following problem: *Given two points, A and B, build a middle point between them.* Fig. 2 shows two different constructions used to solve this problem. Many other constructions are possible, whether minimal[4] or not.

We have identified three techniques used by DG programs in the automatic evaluation of exercises: the automated (geometry) theorem proof (Gallier, 2003), the technique based on the theory ACT (Atomic Component of Thought) (Anderson, 1993), and the numerical evaluation (Isotani & Brandão, 2004).

According to Gallier (2003), an automatic theorem proof (ATP) can be summarized as a computer program that shows whether or not a sentence (a conjecture) is a logical consequence of a group of sentences (axioms and hypotheses). The language used by these programs should be formal in order to eliminate ambiguity. The verification of a sentence produced by an ATP program is known as a proof. This proof describes a sequence of steps (logical consequences) that validate a conjecture. The steps, followed by an ATP program (also known as a proof tree), can be understood by other ATP programs or even a person. Some pioneering works in the development of automatic evaluators of geometry theorems have been reported by Gelernter (1963), Gelernter, Hanson, and Loveland (1963) and Gilmore (1970).

The DG programs based on ATP are capable of checking the validity of different constructions, which in turn, generate different types of proofs that depend on the methods used to prove a scenario (Botana & Valcarce, 2002; Chou et al., 2000). At present, several methods exist for accomplishing an automatic proof in geometry. Some of them are (Gao & Zhu, 1999): the method of Wu, the area method, the base of Groebner, the vector method and the angles method. Among the DG programs that use ATP methods are Geolog (Holland, 1993), GeometryExpert (Gao & Zhu, 1999) and Discover (Botana & Valcarce, 2002).

The ACT-R technique is a current version of the ACT theory, a general theory on the human cognition attempting to reproduce how an individual acquires knowledge. It was developed by a group of researchers under direction of John Anderson at Carnegie Mellon University (Anderson, 1993) and has been applied to intelligent tutoring programs focused on the teaching of Algebra and Geometry (Aleven, Koedinger, Sinclair, & Snyder, 1998).

Although these two techniques produce excellent results, they cannot be effectively applied to web-based courses that check geometry exercises. This dilemma is the result of these techniques being time- and

[4] Minimal constructions/solutions are those from which we cannot remove any object without compromising the properties that satisfy a given problem.

machine-consuming. Furthermore, if a teacher needs to create an exercise, depending on the complexity of the solution, these techniques have difficulties to determine, rapidly, whether the solution is correct or not (Chou et al., 2000; Gallier, 2003). For online courses, the time available for response is extremely limited. Feedback should be given almost instantaneously (real time). Additionally, in Brazilian public schools, most computers have low-processing capabilities, which demand programs and algorithms that do not consume a great deal of computer processing time.

Instead of attempting to prove a student's construction/solution, a numerical evaluation compares the student's answer-objects with the corresponding teacher's answers-objects. This comparison is made using a distance criterion between the objects. For instance, if the problem is to determine the middle point between points A and B, the distance is verified between the point of the student's construction and the point of the teacher's construction according to an established criterion. Programs that use a similar method include Cinderella (Kortenkamp, 1999), C.a.R. (Grothman, 1999), and iGeom (Isotani & Brandão, 2004). The numerical evaluation method is not a formal proof, like ATP or ACT-R, and demands a solution template (a previously provided correct construction). However, this type of evaluation possesses advantages by using less computational processes and by not restricting the application domain. This allows a proposed exercise to be solved using any technique without harming the evaluation process.

2.1. An overview of the authoring and checking processes of iGeom

The functionalities for authoring and automatically checking exercises in iGeom are incorporated so that an exercise may be completed using the application (which facilitates the execution of tests for the definition of the exercise) or web pages.

The authoring of exercises in the iGeom program has five steps: (a) the construction of a solution template; (b) the selection of initial objects (including the selection of a statement); (c) the selection of answer-objects; (d) the use of the unable/enable buttons; and (e) saving the exercise or exporting it into HTML format.

The construction of the solution template is accomplished the same as any other construction. For example, to construct the median line of two points, one possible construction requires the authors (a) to create points A and B; (b) to create a segment $s0$ that connects A and B; (c) to create a middle point M between A and B; and (d) to create the line r (median) perpendicular to the segment $s0$, which passes through M, shown in Fig. 3.

When construction is complete, the author can initiate an authoring window (to the right of Fig. 3) and select the initial objects (those that will appear at the beginning of the exercise) using the "*marcador*" button (in Fig. 3, the icon with a red square around it). The selection of the answer-object is completed in a similar fashion.

In the authoring window, it is possible to see which objects have been selected, as well as remove or add other objects. To disable/enable buttons in the student main interface, one can click on the buttons shown at the bottom of the authoring window. Thus, teachers can select tools that will be available to learners during the resolution of the exercise. For example, the buttons "*middle point*", "*perpendicular*" and "*parallel*" can be disabled and, thus, students will be unable to use these tools to solve the given problem. Finally, authors can save the exercise in a stand-alone version of iGeom, or save it as an HTML file.

Along with the authoring tool, a resource for automatically checking a student's solution has been developed. When a student opens an exercise in iGeom, similar to the example shown in Fig. 3, he or she will only see points A and B and the message, "*Given two points A and B, build the median line*". Furthermore, the buttons disabled by the teacher during the authoring phase will not appear in the menu. Thus, the student will be unable to use them.

After a student completes a given exercise, he or she needs to select the answer-object that he or she believes to be the correct answer (e.g., select the median line for his or her construction). Then, the evaluation algorithm begins. If the student's solution is evaluated as incorrect, the iGeom finds a counter-example or a configuration (instance) of the construction where the selected answer-objects do not satisfy the desired properties (more details on how to find the counter-example is shown in Section 3). The use of counter-examples aids both teachers verifying a student's construction and students who need to visualize their mistakes. Another

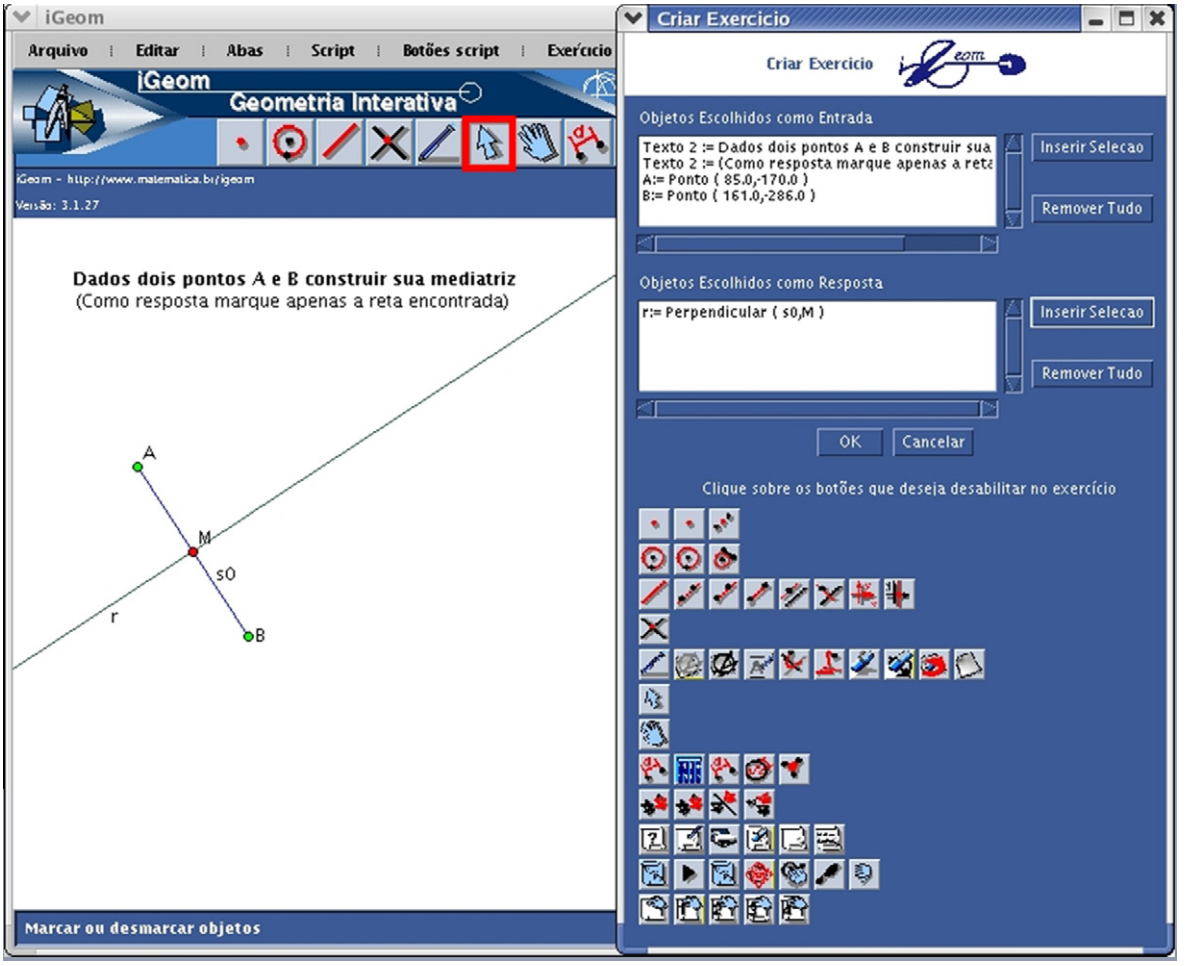

Fig. 3. Construction of a solution template for a median line exercise. In the left-hand pane of the display, the solution template is presented as the construction and the statement"Given Figure/table legends two points, *A* and *B*, build the median line" (in Portuguese). In the right is shown the authoring window.

advantage of the numerical method implemented in iGeom is that a student can use any geometrically valid solution to solve an exercise, even those not imagined by a teacher.

The tools of authoring and automatically checking, together with exporting resources for the Internet, facilitate the creation of interactive exercises that can be used for free access. By using this export feature, a teacher can produce a set of web pages using several exercises and offer their students an immediate evaluation of their constructions, without the need of personally verifying each construction.

A practical example of the use of these resources can be found with iMática, a project making digital materials for mathematics available in Portuguese (http://www.matematica.br/igeom/docs/exemplo1). This webpage possesses several activities (divided into classes, topics and exercises) that are helpful in teaching geometry (Fig. 4). All of them can be completed online and the evaluation (whether it is correct or not) of each solution is supplied by iGeom.

3. Proposing an algorithm for automatically checking exercises using a numerical evaluation

Proposing a fast and accurate algorithm for checking an exercise that can verify whether a construction is correct or not is intrinsically dependent on how the concept "solution" is defined. For example, in the C.a.R. program, a solution is defined as any static object. With such a definition, the algorithm used to check exer-

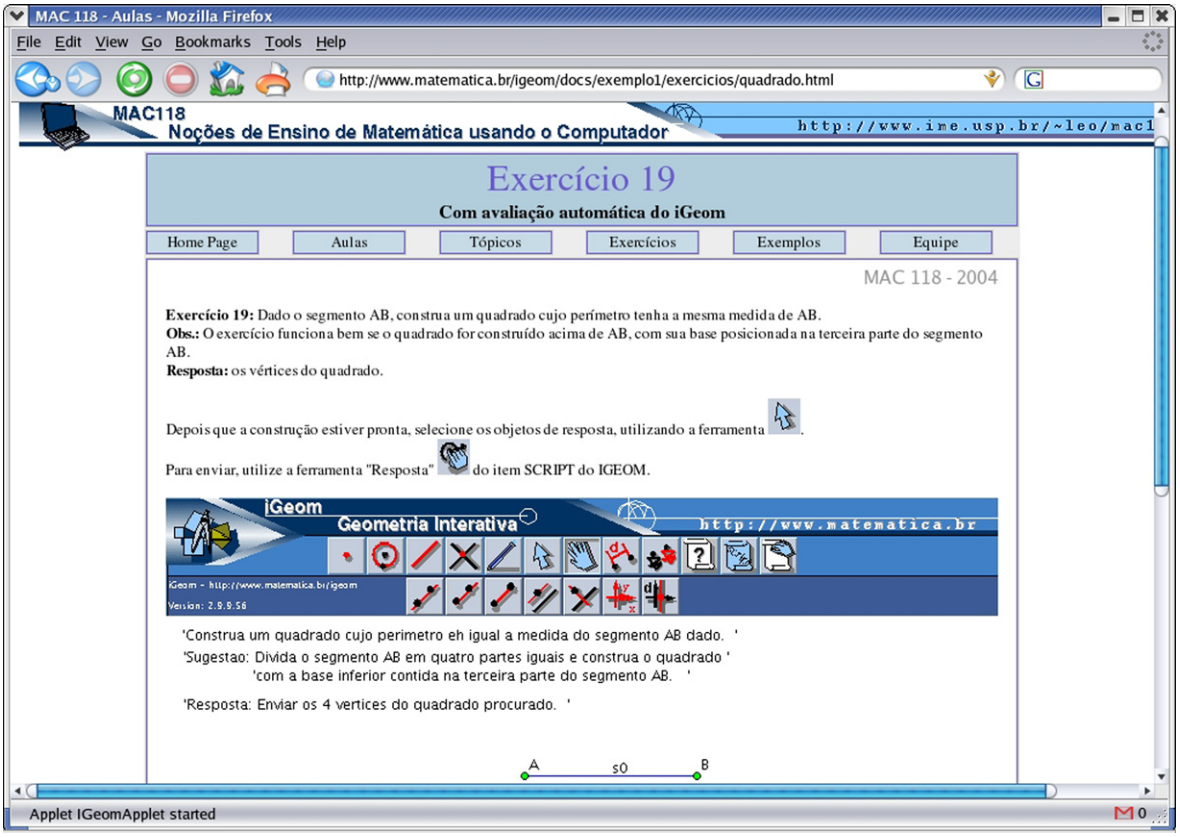

Fig. 4. Activity example for use through Web, with the iGeom program, developed by students at IME-USP in 2004.

cises in C.a.R. basically tests if the objects selected as a solution for an exercise have similar objects in the solution template. This algorithm somewhat works in practice, however, it constrains the dynamicity of the system. Because a solution is defined as a static object, C.a.R. does not allow for the movement of objects while an exercise is being solved. In order to simplify the automatic checking of exercises, such algorithms ignore the term "dynamic" of a DG system. An easy test can exemplify one weakness of this algorithm. For example, if an exercise asks an individual to find a middle point between the given points A and B, then one way of cheating is to create a point with the same coordinates as the middle point. Although this solution is not desirable, the automatic checking in C.a.R. will determine that such a construction is correct, even though no valid construction has been completed.

In Cinderella, the concept "solution" is defined as a theorem that needs to be checked. Thus, the algorithm for checking exercises is known as "automatic theorem checking", based on the Schwartz–Zippel Theorem (Schwartz, 1980). This theorem states that if it is possible to estimate the probability of choosing a counter-example (if there is one), it is then possible to bound the probability of failing if a theorem is true, based on a few random examples. By utilizing such a theorem when checking an exercise, the Cinderella algorithm tries to prove if objects in a construction coincide by chance or because a theorem drives them to coincide.

Finally, in iGeom, a solution is defined as a function applied to any object (initial objects) and returning objects (answer-objects). Such a definition allows for high flexibility when comparing two different constructions. Thus, when given initial objects, it is possible to verify if different constructions provide equivalent results without the necessity of comparing each object of every construction or the construction steps. To establish the definition of solution and equivalence between solutions, and thus, creating an algorithm based on numerical evaluation, it is necessary to measure the distance among objects of a construction. In the following sections, the basic formalization for these definitions and an algorithm for checking exercises are proposed.

3.1. Numerical evaluation

The result of the evaluation is a measurement of the distance between the student's solution and the teacher's solution (a correct solution given for a problem). To complete this evaluation, it is necessary to define the distance criterion among the geometric objects. We have defined the distance criterion only for pairs of objects of the same family or type. To simplify, we only name examples from the more common families in a construction, including the family of points (F_p), the family of circumferences (F_c) and the family of segments (F_s). The sets of all families of objects will be represented by F_{og}.

By doing so, the distance criterion is the function *dist* that receives a pair of geometric objects ($og1$, $og2$) $\in F_{og} X F_{og}$ and returns a value of $\Re_+$:

$$\text{dist} : (og1, og2) \rightarrow \Re_+ \tag{1}$$

The computational description of the objects is defined for each family of F_{og} as a list of numerical values. For instance, a point can be represented by (x, y), where x and y are the coordinates of a point; a circumference can be represented by (x, y, r), where (x, y) are the coordinates of its center and r is its radius; and a segment $s = [(x_1, y_1), (x_2, y_2)]$ can be represented by (x_1, y_1, x_2, y_2). Thus, if we consider just the families of points, circumferences and segments, given two objects of a same family, $og1$ e $og2$, if $\left(\ell_1^1, \ell_2^1, \ldots, \ell_i^1\right)$ and $\left(\ell_1^2, \ell_2^2, \ldots, \ell_i^2\right)$, represent $og1$ e $og2$, respectively, then we can define the function *dist* according to the Eq. (2).

$$\text{dist}(og1, og2) = \begin{cases} |\ell_1^1 - \ell_1^2| + |\ell_2^1 - \ell_2^2|, & (og1, og2) \in F_p X F_p \\ |\ell_1^1 - \ell_1^2| + |\ell_2^1 - \ell_2^2| + |\ell_3^1 - \ell_3^2|, & (og1, og2) \in F_c X F_c \\ \min\left\{ \dfrac{\sum_{i=1}^{4} |\ell_i^1 - \ell_i^2|,}{\sum_{i=1}^{4} |\ell_i^1 - \ell_{(i+1)\%4+1}^2|} \right\}, & (og1, og2) \in F_s X F_s \end{cases} \tag{2}$$

The symbol % is used to obtain the rest of the division (function module).

The need of the minimum (min), when objects are from the family of segments, is based on the desire to compare not only the distance between segments AB and CD, but also their permutations AB and DC, BD and CD, BA and DC, without differentiation between them. In other words, for segments, the distance criterion does not distinguish segment AB from segment BA.

Once the distance among geometric objects is defined, we can define the distance between pairs of geometric constructions, OG_p and OG_a. If OG_p and OG_a are two constructions and their objects are represented by $(og_1^p, og_2^p, \ldots, og_i^p)$ and $\left(og_1^a, og_2^a, \ldots, og_i^a\right)$, respectively, then the distance between OG_p e OG_a, can be expressed as shown in Table 2.

A *solution* (geometric construction) can be represented as a function that receives a list of geometric objects (input) and returns another list of geometric objects (output):

$$S : OG_i \rightarrow OG_f \tag{6}$$

An instance of construction S is the application of S over a given configuration of objects. Once the distance among pairs of constructions and the representation of a solution is determined, we can decide when the two constructions (solutions) are equivalent, as shown in Table 3.

It is worth noting that if two constructions are equivalent, the distance between them is invariant in relation to the initial configurations. In other words, the distance computed for any instance is always zero. Therefore, if we do not consider numerical errors, a good evaluation criterion states that construction S_a is correct whenever it is *equivalent* to a correct given construction, S_p (for example, a student's solution, S_a, can be considered correct if it is equivalent to solution S_p, given by the teacher). However, there is a practical problem: *how to implement a version that is sufficiently fast and takes into account numerical errors?*

As different constructions applied to the same initial objects generate the same answer-objects (output) with small numerical differences, we have opted to relax the equivalence criterion that allows, for instance, two points very close in a number of different instances of the initial objects to be considered *almost equivalent*. In principle, this solution allows for *false positives* (a wrong exercise evaluated as correct) or *false negatives*

Table 2
Definition of distance among pairs of constructions

If the cardinality of the lists OG_p and OG_a ($\#OG_p$ e $\#OG_a$) are different, then

$$\mathrm{dist}(OG_p, OG_a) = +\infty \tag{3}$$

If $\#OG_p = \#OG_a = n$,
then $I_p = (p_1, \ldots, p_n)$ and $I_a = (a_1, \ldots, a_n)$ two permutations on the first natural n,

$$\mathrm{dist}(I_p, I_a) = \begin{cases} \sum_{i=1}^{n} \mathrm{dist}\left(og_{p_i}^{p}, og_{a_i}^{a}\right), & \text{if } \langle tipo\left(og_{p_i}^{p}\right) = tipo\left(og_{a_i}^{a}\right), i \in \{1, \ldots, n\}\rangle \\ +\infty, \quad c.c. \end{cases} \tag{4}$$

tipo(og) being a function that returns the type of geometric object *og*.
Then,
if P_n is the set of all of the permutations of first natural n:

$$\mathrm{dist}\left(OG_p, OG_a\right) = \min\left\{\mathrm{dist}\left(I_p, I_a\right), \forall\left(I_p, I_a\right) \in P_n X P_n\right\} \tag{5}$$

In other words, among all of the permutations of objects of OG_p e OG_a, $\mathrm{dist}\left(OG_p, OG_a\right)$ is the sum of the distances among each pair of objects of same type that results in the smallest value.

Table 3
Definition of equivalence of solutions/constructions

Definition of equivalence:
Be S_p and S_a two constructions applicable over the same list of geometric objects OG. Then S_p andS_a are equivalent if, and only if, for any configuration OG_0 of the list OG, $\mathrm{dist}\left(S_p(OG_0), S_a(OG_0)\right) = 0$.

(a correct exercise evaluated as incorrect), but as we show in session 4, in practice, it works well. It is also possible to increase the number of instances tested or implement other modifications of parameters that would reduce or eliminate wrong evaluations.

3.2. The evaluation algorithm

From now on, when it is implicit or indifferent as to which list of geometric objects should be used, the simplified notation S will be used.

Based on the numerical evaluation presented in the previous section, we have created an algorithm composed of four main steps. These steps include *numerical transformation*, *evaluation*, *instantiation, and validation*. This algorithm has been implemented into iGeom as a way of verifying its performance and accuracy. As mentioned in Section 2, in order to check an exercise, it is necessary to create a solution template by providing a correct construction for the exercise and selecting which objects will be checked (answer-objects). At the end of a resolution, a student selects which are their answer-objects. The lists of the student's answer-objects and the solution template's answer-objects will be the input data for the evaluator algorithm. The result will be a natural number between 1 and 3, with (1) being the correct solution, (2) being a partially incorrect solution and (3) being an incorrect solution.

When applied on a geometric object, the first step of our algorithm, the *numerical transformation*, returns a list of scalars that represent the object. In most cases, this transformation is simple. However, besides the numerical transformation, some objects, such as the polygon, need the ordination of points representing it. With this list of scalars, we can make a comparison between objects. Then, in order to compare the solution template, S_p, and the student's solution, S_a, we transform objects marked as answers in lists of scalars and then begin the evaluation, as schematized in Fig. 5.

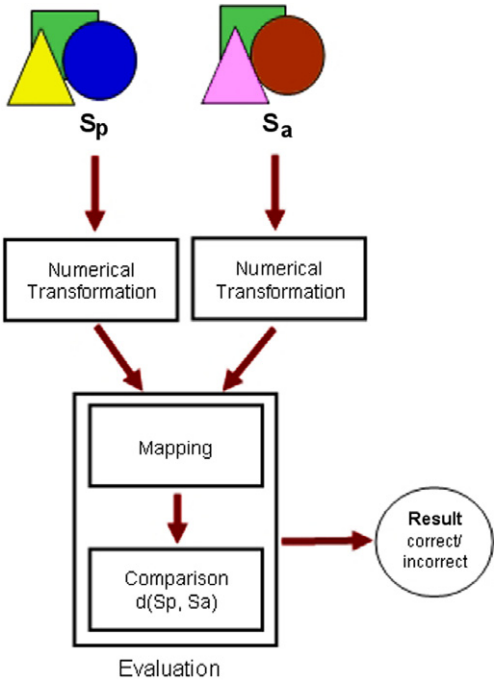

Fig. 5. Numerical transformation and evaluation.

The *evaluation* is made in two stages. The first stage consists of *mapping* between the lists, while the second is the *comparison* of the distance criterion. The first stage allows students to have the freedom of making a selection of answer-objects in any order. For instance, if the solution template contains points A and B and the student's answer-objects are points C and D, the mapping stage will identify if points C or D correspond to A or B.

The mapping of objects S_a e S_p is accomplished by comparing each *og* element of S_a, with all elements of S_p that belong to the same type (family), while minimizing the distance between them. Then, an object of S_a will be mapped with an object of S_p if both belong to the same family of geometric objects, have not been mapped before and the distance among them is the smallest possible in relation to other objects of S_a. This mapping is made only for the first instance (the initial configuration of an exercise). The second stage of the evaluation consists of a comparison between pairs of mapped geometric objects (og_i^a, og_i^p). In comparison, we use a relaxed version of the definition of equivalence, presented in Table 3, that takes into consideration the numerical imprecision of using different solutions. Then, we propose a definition of *Quasi Equivalence,* as shown in Table 4.

The pseudo-algorithm implemented in iGeom that corresponds to these two steps (mapping and evaluation), is shown in Table 5. Lines 1 through 12 represent the mapping of objects, while lines 13 through 19 represent the exercise evaluation.

Notice that the part of the algorithm shown in Table 5 represents just one instance of the solution, which considers a fixed position for each input object. If this algorithm is used to check only the initial configuration, it can frequently generate mistakes, such as false positives. For instance, in the problem using the middle point (Section 2), a student could try to put a free point on segment AB and move it so that it

Table 4
Definition of quasi-equivalence of solutions/constructions

Definition of quasi-equivalence: Fixed a list of input objects OG, be S_p a construction on OG and S_a another construction on the same OG. Then S_p and S_a are quasi equivalent if, and only if, for any configuration OG_0 of the list OG, we have $\text{dist}(S_p(OG_0), S_a(OG_0)) < \varepsilon$.

Table 5
An pseudo-algorithm to solve the problem of checking exercises in DG

1. Receive two lists S_p e S_a of geometric objects
2. Creates a new list of geometric objects $S_t \leftarrow \phi$
3. For each element og_i^p of the list S_p
4. $SmallestFoundDistance \leftarrow \infty$
5. $MappedObject \leftarrow \phi$
6. For each element og_j^a of the list S_a
7. If og_i^p e og_j^a belong the same family of geometric objects then
8. If dist$\left(og_i^p, og_j^a\right) < SmallestFoundDistance$
9. $MappedObject \leftarrow og_j^a$
10. $SmallestFoundDistance \leftarrow dist\left(og_i^p, og_j^a\right)$
11. If $MappedObject = \phi$
12. Return*False* // mapping between S_p and S_a failed
13. If not
14. $S_t \leftarrow S_t$ $MappedObject$ // concatenation
15. Remove $MappedObject$ de S_a // object is mapped
16. For each element og_i^p of the list S_p
17. Be og_j^t the first element S_t
18. If dist$\left(og_i^p, og_j^t\right) < \varepsilon$ then
19. Remove og_j^t of S_t
20. If not
21. Return *False* //solutions are not equivalent
22. Return *True* //solutions are equivalent

is close enough to the position of the medium point to generate a false positive, without making a valid geometric construction. In spite of the difficulty involved in positioning the point so that the algorithm evaluates the solution as correct, it is possible to identify such problems in the algorithm implemented in C.a.R. This false positive problem is illustrated with the following example: *Given two points, A and B, build an equilateral triangle* ΔABC.

In Fig. 6, two solutions for this problem are shown. The first construction is correct and is shown in Fig. 6a. The second, shown in Fig. 6b, is an incorrect solution, but contains the same properties of an equilateral triangle. In this case, the second solution depicts the construction of an isosceles triangle, with the free point C on the straight line s. Coincidentally, in this construction configuration (instance), point C is in such a position that $\overline{AB} = \overline{AC} = \overline{BC}$. Thus, the construction of the isosceles triangle can be erroneously used to produce an equilateral triangle. To avoid such problem, it is possible to move all free points of the construction, in particular, point C of the isosceles triangle, so that it becomes clear that triangle ΔABC in Fig. 6b is not the correct construction for an equilateral triangle.

Therefore, a simple way to detect such mistakes is to create a mechanism that analyzes the exercise in several instances (*instantiation*) and only after a considerable number of evaluations does the system return the result of the evaluation (*validation*). In Fig. 7, we show the procedure that is employed by the evaluation algorithm. For each application of the automatic checking algorithm, the positions of the geometric objects are

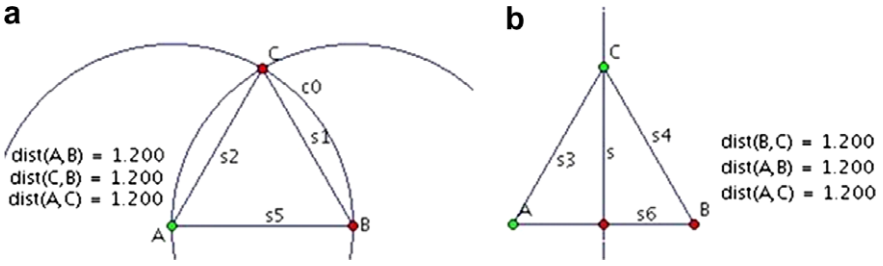

Fig. 6. Two constructions of an equilateral triangle. The construction (a) is the correct construction for an equilateral triangle and the construction (b) is the construction of an isosceles triangle in a instance where all the sides of the triangle have the same measure.

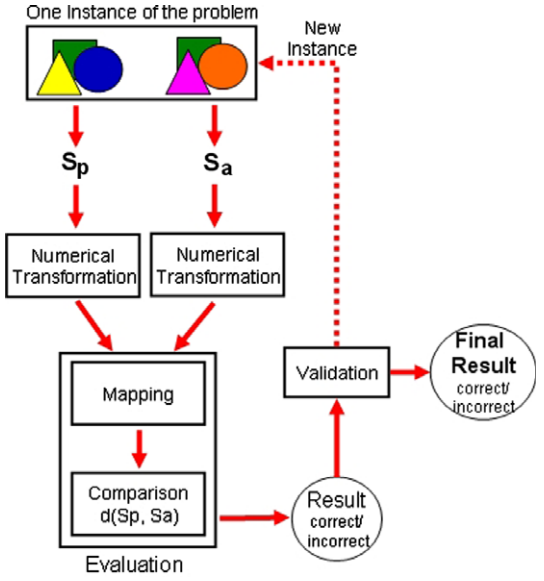

Fig. 7. Sequence of steps of the proposed algorithm for automatic checking of geometry exercises.

changed. This alteration is made through the random movement of all free points of the construction. The choice for the new position of a point is made by modifying the coordinates (x, y) for (x', y') and $x - k < x' < x + k$ for $y - k < y' < y + k$, where k is a random value such that $0 < k \leqslant 20$. The value of k was empirically determined providing evidences that the modification of the coordinates is enough to identify possible problems in a construction.

At the end of the automatic checking process, the algorithm returns an integer value: (1) means the solution is correct; (2) means the solution is incorrect, however, it found instances considered correct and incorrect; and (3) means the solution is incorrect.

3.3. Identification of ambiguity

The concept of functions has been explored in order to deal with the problem of automatically checking geometric exercises. These functions are constructions that receive a set of input objects and return a set of answer-objects. Such functions are considered *ambiguous* when they are not bijective functions, or in other words, they possess more than one set of answers for the same input data. In DG, this problem can appear whenever the proposed exercise has an implicit or explicit free point needed to solve the problem. An example is presented in the following problem: *Given two points, A and B, build an isosceles triangle* ΔABC.

This problem is intrinsically ambiguous, possessing infinite solution instances, because point C can be located on any position of the median line A and B. Independent of A and B, point C can be moved, changing the instance of triangle ΔABC, yet maintaining the desired properties, as can be observed in Fig. 8.

To find this ambiguity problem during the authoring of an exercise, we have implemented an algorithm that checks the construction of the solution template. It informs the author which geometric object makes the construction ambiguous. This warning is very useful when correcting teachers' mistakes during the creation of an exercise. Other programs, such as C.a.R. and Cinderella, lack similar algorithms or functionalities.

During the construction, when selecting the initial objects and answer-objects, the algorithm of ambiguity verifies if some answer-object is dependent of some free point, P. If such point P does not belong to the selected initial objects and cannot be determined by them, then, we verify whether moving point P will change the position of some answer-objects. In the example of Fig. 8, during the creation of the solution template selection, only points A and B are initial objects, while segments $\overline{AC}$ and $\overline{BC}$ are answer-objects. The algorithm

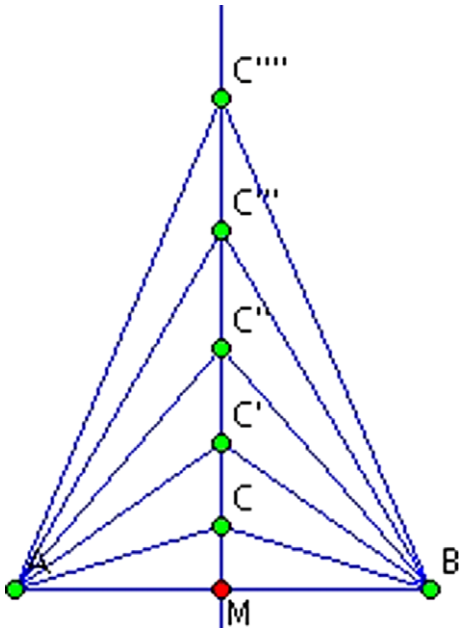

Fig. 8. For a same position of the points *A* and *B*, potentially we have infinite configurations for the problem of construction isosceles triangles.

identifies the ambiguity and returns a message to the teacher informing him or her that point *C* needs to be selected as an initial object to remove the ambiguity of the solution template.

It is worth noting that an answer-object, dependent on a free point that does not belong to the initial objects of the solution template, is not always modified when the free point is moved. An example can be observed by creating the solution template for the following problem: *Given the straight line r and point A, build a straight line s by passing through A and forming an angle of 60° to r.*

The solution for this problem can be seen in Fig. 9. The objects selected as initial objects are straight line *r* and point *A*, while the object selected as the answer will be straight line *s*. Note that in this construction, straight line *s* is dependent on point *C*. Yet, this point is not selected as an initial object. However, when moving point *C*, the position of straight line *s* does not modify and, therefore, the solution is not ambiguous. The developed algorithm detects this case using part of the mechanism of the automatic evaluation, making the comparison of the construction in different configurations.

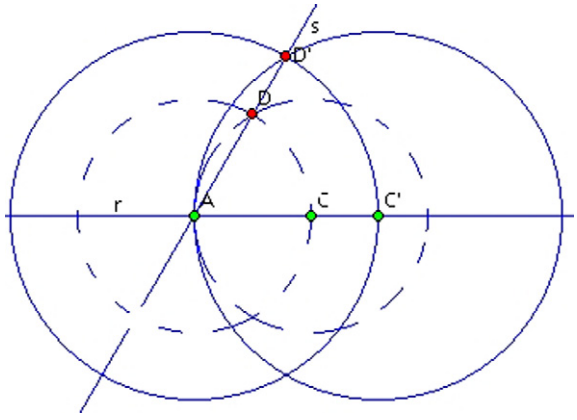

Fig. 9. Construction of an angle of 60°. In this construction the movement of the point C (free point) does not interfere in the result.

4. Communication between the iGeom program and a web server

The tools for automatically checking and authoring exercises have been built directly into the iGeom application. To fulfill our goal of effectively using DG programs in distance education courses, it was necessary to build a tool that could communicate with learning management systems (LMS). Among the several advantages of the communication between iGeom and a LMS we emphasize the following:

(a) Teacher can produce the exercises with his or her machine and send them to the server or create them directly on the Internet through an iGeom applet, working together with a learning management system (LMS);
(b) iGeom can exchange data related to the resolution of an exercise with a LMS;
(c) By receiving data from a server, iGeom is able to personalize the content according to learner's behavior.

Today, there are several complex systems used to manage Web-based courses. Some of these are commercial, while others are freeware, including SAW (Moura et al., 2007), AulaNet (Lucena, Fuks, Raposo, Gerosa, & Pimentel, 2006) and Moodle (Moodle, 2007). Usually, these environments lack specialized resources that would boost learning specific contents (e.g., geometry). Thus, the use of iGeom allows any LMS that supports Java applets to incorporate specific functionalities that will help the teaching–learning of geometry in a very interactive way.

The possibility of integrating iGeom into LMS offers many advantages for students and teachers. For students, besides the possibility of interactively accomplishing exercises online, solutions can be checked almost instantaneously and a counter-example can be given when a solution is incorrect. For teachers, we offer resources that facilitate authoring interactive content, evaluating exercises and collecting data related to the student's interaction with the content (exercises, examples, etc.) for a subsequent analysis.

The exchange of messages between iGeom and the server (LMS) is made using the method POST (Raggett, Hors, & Jacobs, 1999). This method allows iGeom to send variables in the form of chains of characters. In this way, different information is sent for the server using the same message. In this case, the program located in the server should be responsible for the treatment of the received data, in order to recover the values of the sent variables.

An example of sending messages happens after the evaluation of an exercise is accomplished on the Internet. When the exercise is solved, iGeom requests a connection to an address on the server. Data are stored using different variables. Each variable, defined by a group of characters, is packed using the method POST. After the reception of the data, the server stores the messages received in local variables so that they can later be manipulated by the program manager. Yet, through the same connection, the server can request changes in the active Web page interface or open a new Web page. Some of the variables manipulated by iGeom include the following:

$*envWebValor*: This variable indicates the result of the evaluation of the exercise. If its value is equal to 0 (zero), then the exercise is evaluated as incorrect; if it is equal to 1 (one), the exercise is evaluated as correct.

$*envWebArq*: This variable possesses two functionalities. The first relates to when a teacher is creating an exercise and the other to when a student is solving an exercise. When a teacher creates an exercise, directly on the Internet, the variable stores all of the information of the exercise, including the solution template that will be used later. For students, it stores every construction accomplished during the resolution of an exercise. If the solution is considered incorrect, then the counter-example is stored to show that the student has made a mistake. If the solution is correct, it stores the construction in any configuration.

$*envWebGeoResp*: This variable is used to indicate which geometric objects have been selected as answers to an exercise.

$*envWebGeoOuvidor*: This variable stores the data created from interactions between the user and iGeom. For instance, the buttons frequently used may be stored. It can also recall specific situations in which the user used such buttons.

5. Results

In order to analyze the efficiency, efficacy and accuracy of our algorithm, since 2004, iGeom has integrated a learning manager system, referred to as SAW. iGeom together with SAW has been used in several didactic experiments to offer geometry e-courses for undergraduate students, math teachers, and students in private high schools. The analysis of *efficiency* refers to the possibility of applying our algorithm to check exercises in class or over the Internet, using personal computers with low-processing capabilities and offering immediate feedback to alleviate student frustration. The analysis of *efficacy* means to determine if the algorithm can check a student's solution, given any exercise in which the solution is not ambiguous. Finally, checking *accuracy* refers to checking how many times our algorithm will return false positive and false negative results.

The tools of authoring and automatically checking exercises in iGeom have been tested on several occasions. The first instance occurred in 2004 for an obligatory discipline, titled "*Notions of teaching of mathematics using computers*" (MAC118), for undergraduate students in mathematics. The goal of this discipline is to teach future teachers how to use technologies that support students learning mathematics. This discipline comprised of two teachers, three teaching assistants and more than 150 students, divided into three groups. The authoring and checking tools were used during a period of six months, until the end of the semester. Due to the success of the use of iGeom and SAW, these tools were incorporated into this discipline. Another test was done at the beginning of 2005 for courses for senior math teachers. This test included the participation of more than 25 junior high and high school teachers. These teachers acted as authors of content and problem solvers. During a two month period, they were instructed on how to use iGeom and SAW to construct e-courses for geometry, as well as how to benefit from the functionalities of DG programs to create more interactive content for Web-based courses. In 2006 and 2007, a qualitative and quantitative analysis of SAW, using the iGeom program, was completed by Moura et al. (2007), following the methodology proposed by Yin (2002).

In MAC118, before the implementation of the automatic checking tool, the exercises were solved using iGeom (or a similar program), but exercises were corrected manually by teaching assistants and teachers. Correcting consumed a great deal of the teaching assistant's time and feedback of these corrections was given two or three weeks later. Before 2004, approximately 20 exercises were applied and solved per semester. With the use of automatic checking and SAW, it was possible to reduce the teachers' and assistants' work. Both were able to spend more time taking care of student's personal difficulties raised during the attempt to solve an exercise. Furthermore, in 2004, it was possible to apply and solve more than 40 exercises in class and over the Internet with immediate feedback. In 2006, there were more than 70 exercises completed in one semester. Through the application of the automatic checking tool, students have eliminated misconceptions immediately after the resolution of an exercise. As affirmed by one student, "...in the case that the construction is evaluated as correct, it was already recorded (in the SAW), and in the case it was wrong, I could start it again and would remove my doubts immediately...".

According to the work of Moura et al. (2007), which analyzes the impact of iGeom and SAW in the learning process, the increased number of exercises given in the course is "... a consequence of the use of automatic assessment resources which have encouraged learners in their studies (in 2006, 82% of students that answered the second questionnaire think that the use of SAW + iGeom has encouraged them during the course). Also, it reduced tutor workload and encouraged him/her to introduce preparatory exercises to serve as an auxiliary for the solution of the main problem, as well as leaving related exercises as homework."

The experiments completed in MAC118 have shown that our algorithm is efficacious in checking exercises and efficiently runs in computers with low-processing capabilities. In 2004, the computers used in class had processors similar to a K6II 450 MHZ with 256 Mb of RAM memory. For any proposed exercise, our algorithm responded almost instantly (milliseconds), immediately returning output. In comparison, GEOLOG (Holland, 1993), another geometry software program that uses an algorithm based on theorem-proving methods, required more than five seconds to check the exercise of a middle point, using the solutions presented at the right of Fig. 2. For more complex exercises, such as the construction of a pentagon (see Table 6), GEOLOG did not answer until several minutes had passed. Such a delay (ten seconds and greater with no response) is not acceptable when exercises must be accomplished over the Internet, where instantaneous feedback is key to maintaining student motivation (Hentea et al., 2003).

Table 6
Sequence of steps (algorithm) performed by one student to construct a pentagon. The result is show on Fig. 10.

$c0$: = Circle ($P1$, $P2$);
$c1$: = Circle ($P2$, $P1$);
A: = Intersection south ($c0$, $c1$);
B: = Intersection north ($c0$, $c1$);
$s0$: = Segment (A, B);
$c2$: = Circle (A, $P1$);
C: = Intersection ($c1$, $c2$);
D: = Intersection ($s0$, $c2$);
r: = line (C, D);
E: = Intersection ($c0$, $c2$);
s: = line (E, D);
F: = Intersection ($c0$, r);
$c3$: = Circle (F, $P1$);
G: = Intersection north ($c1$, s);
$c4$: = Circle (G, $P2$);
$s1$: = Segment ($P1$, F);
H: = Intersection north ($c3$, $c4$);
$s2$: = Segment (F, H);
$s3$: = Segment (H, G);
$s4$: = Segment (G, $P2$);

Finally, the accuracy of our algorithm has been proven robust enough to not provide false positive and false negative evaluations. Different instructors from schools and universities have produced hundreds of exercises and thousands of solutions which have been checked by our algorithm. So far, none of them have been given a false positive or false negative result. A very good example of the accuracy of our algorithm occurred in MAC118 in 2005. The teacher had asked the students to create a pentagon, starting with a given segment (initial object). One of the students tried to solve the exercise using a solution (construction) that he knew and believed to be correct. The steps to create this solution are presented in Table 6 and the corresponding geometric construction is shown in Fig. 10. This solution was considered incorrect by our algorithm. The student resent it again and again, complaining that our algorithm was not accurate enough, given the numerical imprecision. In this case, even the teacher was in doubt about the veracity of the answer given by our algorithm. However, a bibliographic research made by one of the teaching assistants verified that the solution given by the student was, in fact, wrong. According to Fourrey (1924), this construction is an approximation of the correct solution. In this solution (Fig. 10), the internal angles of points F and G have $109°2'28''$, and not 108° (in a regular pentagon all internal angles should have 108°). In such situations, without the automatic checking tool, the student would never know that such a construction is incorrect.

The difficulties and obstacles regarding the use of iGeom have been analyzed using answers from distributed questionnaires. Through the application of automatic checking, students have the possibility of removing any doubt they may have concerning about the resolution of the exercise rapidly. Ninety-seven percent of students are very satisfied with this functionality. However, we observe that it is necessary to provide a more precise statement for each exercise. Many students have difficulties solving some exercises due to an imprecision in statements. According to the data obtained during the use of iGeom during MAC118, about 69% of the students whose exercises are evaluated as incorrect have difficulty to understand the statements. Other difficulties listed by the students include the selection of correct answer-objects and difficulty in building the solutions due to the complexity of the exercises.

Another implemented resource that contributes to the success of iGeom in the classroom and over the Internet is the integrated communication capability. With communication, the teacher can create the exercises directly on the Internet and store them in a database using any LMS that supports our protocols. Thus, iGeom has facilitated the reuse of created exercises for different teachers and courses. Besides these benefits, each exercise accomplished by a student and other information about its solution also can be stored. When an exercise is considered incorrect by our algorithm, the iGeom sends the student's solution

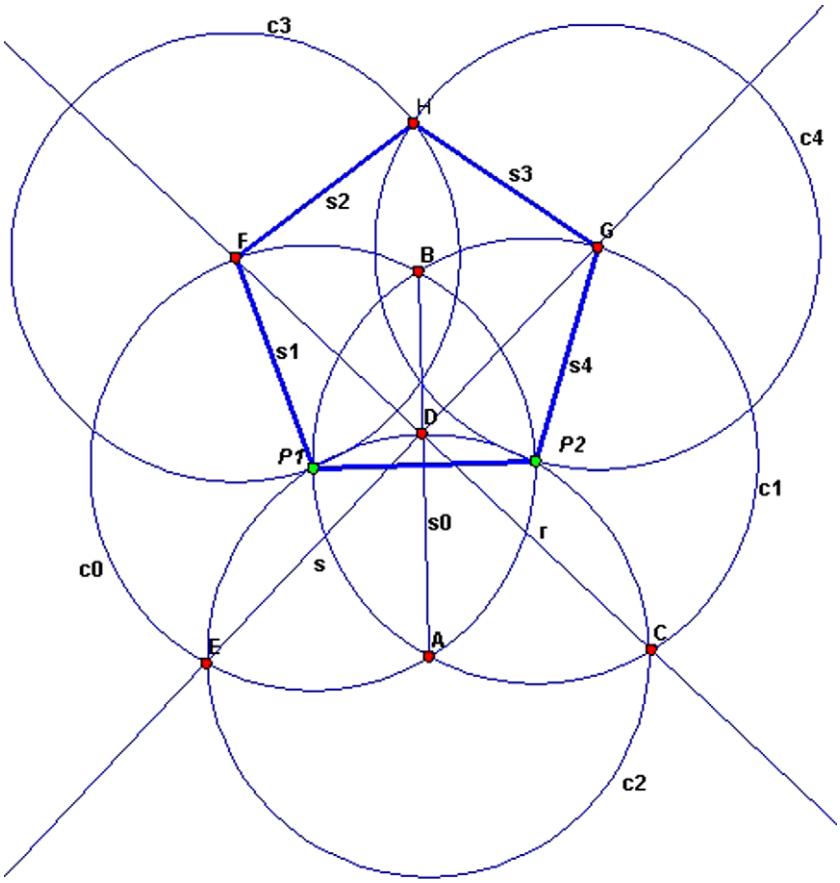

Fig. 10. Result of an incorrect construction of a pentagon sent by one student. The internal angles in *F* and *G* have 109°2′28″ and not 108°. In Table 6 is shown the steps (algorithm) used to construct the figure.

to the server/database in a configuration that facilitates the visualization of the mistake (a counter-example). Thus, a teacher can verify a student's solution and provide more personalized feedback. It is also possible to build a library of different solutions for the same problem, including one illustrating mistakes most frequently made.

6. Conclusions

The possibility of increasing the benefits of DG programs to be used in computers with low capabilities, over the Internet, and in distance education courses, has been a great challenge, as well as a strong motivation for developing and implementing these resources. In this context, the resources of communication, authoring, and automatically checking exercises have been intentionally developed to increase the dissemination of DG programs to teachers and students.

According to Ruthven et al. (in press), one of the biggest difficulties in encouraging a wider use of DG programs is the difficulty of creating content and evaluating and guiding students during learning activities. To overcome such difficulties for teachers, this work offers resources that facilitate the authoring and checking of exercises which can be used on simple Web pages or with an LMS. Thus, teachers' time spent creating and evaluating exercises is reduced. Moreover, resources for storing and reusing exercises are provided. For students, the use of DG is provided directly on Web pages and, because of automatic checking, fast feedback is given for each accomplished exercise. With this, students' doubts can be immediately eliminated, which reduces student frustration over a lack of feedback after the conclusion of an exercise.

In order to develop an automatic checking tool in iGeom that could be used in computers with low-processing capabilities and over the Internet, this work proposes and implements an algorithm that checks the validity of a construction, using the idea of function. Each geometric construction is considered a function that receives some geometric objects (input) and returns other geometric objects (output). Thus, for a given problem, if one function (construction) is known to solve the problem, we can compare it with another function to check whether these functions are equivalent or not. Two functions are equivalent if, and only if, for the same input they have the same output. Because of the numerical imprecision of geometric constructions, some numerical differences between outputs of each function are accepted (quasi-equivalence). By using such definitions to check an exercise, we can compare a solution given by a student with the solution given by a teacher to solve the same problem. If the student's solution is considered equivalent (or quasi-equivalent) to the teacher's solution for a variety of different inputs, then we consider the student's solution correct. This algorithm is implemented into iGeom and is shown to be faster and consume less machine resources than formal approaches (rule-based approaches), which allows for an increased use in classrooms, with computers with low-processing capabilities and via the Internet (Isotani, 2005). The main concern of the presented algorithm is the possibility of giving false positive and false negative answers. This means that our algorithm could indicate that a given solution is correct when it is, in fact, incorrect (false positive) or indicate that a given solution is incorrect answer when it is, in fact, correct (false negative). Nevertheless, an extensive evaluation of our algorithm shows that it works well in practice. So far, we have not found any case where a solution has been given a false positive or false negative evaluation.

The communication protocol implemented in iGeom allows for the exchange of information over the Internet. Using such capabilities, the iGeom has been used together with a learning management system, called SAW, to create a very interactive environment to learn geometry. Thus, teachers can create exercises directly through a Web page (or upload the exercise) and students can submit their solutions that will be stored on the server. Furthermore, students receive immediate feedback that determines whether their solutions are correct or not. According to Moura et al. (2007), the use of iGeom in classrooms and in distance education courses can increase student's motivation and help teachers identify student misconceptions.

iGeom is continuously under development. Each semester, users of the program offer feedback and suggestions to improve its functionalities and remove technical problems (*bugs*). As mentioned at the end of Section 5, we intend to create a library with different solutions for the same problems, including good solutions and frequent mistakes committed by students. One of the biggest challenges facing this program is the augmentation of our automatic checking algorithm. We intend to use this library to improve feedback given by our program.

Today, our algorithm runs only if the user clicks the "checking exercise" button. Otherwise, no action is taken. By using a library made up of correct solutions and common mistakes, our algorithm could run in the background and identify whether the user's solution is similar to a possible mistake, and then offer some sort of hint to help the student realize his or her mistake (or avoid it). If the student's solution is similar to a correct solution, then the program can offer praise for partial accomplishment to increase his or her motivation. Usually, during the problem-solving process, a user's solution is not completely wrong. Thus, this approach allows a student to reuse part of his or her solution that is considered correct (i.e., partially similar to a good solution in the library), to guide the user or to give some suggestions/hints in order to help him or her "find" the correct solution to the problem.

To achieve this goal, our library needs to be formal and organized in terms of (a) a common vocabulary with structured definition of geometric concepts; (b) semantic interoperability and a high expression of each concept; and (c) a variety of contexts by which a problem can be solved. One possibility in fulfilling these requirements is to use ontologies and ontological engineering techniques that support the development of our library (Mizoguchi & Bourdeau, 2000). By using these ontologies, we may feasibly augment our automatic checking algorithm by providing a better way of identifying necessary and desirable concepts for solving an exercise and identifying similarities between the actual construction done by the user and the constructions recorded in the library. Thus, we believe that with such improvements, it will be possible to offer a meaningful and more personalized system for users, which will con-

tribute to an increase in motivation and the avoidance of possible frustrations during interactions with the system.

The latest version of the program iGeom, with all resources discussed in this paper, is freely available by visiting http://www.matematica.br/igeom.

References

Aleven, V., Koedinger, K. R., Sinclair, H. C., & Snyder, J. (1998). Combatting shallow learning in a tutor for geometry problem solving. In *Proceedings of the International Conference on Intelligent Tutoring Systems. Lecture Notes in Computer Science* (Vol. 1452, pp. 364–373). Berlin: Springer.

Anderson, John (1993). *Rules of the Mind. Hillsdale*. New Jersey: Lawrence Erlbaum Associates.

Botana, F., & Valcarce, J. L. (2002). A dynamic-symbolic interface for geometric theorem discovery. *Computer and Education, 49*(2), 27–34.

Brandão, L. O., & Isotani, S. (2003). A tool for teaching of dynamic geometry in the Internet: iGeom. *Proceedings of the Brazilian Computer Society Congress*, 1476–1487.

Brandão, L. O., Isotani, S., & Moura, J. G. (2004). A Plug-in based adaptive System: SAAW. In *Proceedings of the International Conference on Intelligent Tutoring Systems. Lecture Notes in Computer Science* (Vol. 3220, pp. 791–793). Berlin: Springer.

Brandão, L. O., Isotani, S., & Moura, J. G. (2005). Immersing dynamic geometry in distance education environments: iGeom e SAW. *Brazilian Journal of Informatics in Education, 14*(1), 41–49.

Cabri II Plus (2007). http://www.cabri.com.

Chou, S. C., Gao, X. S., & Zhang, J. Z. (2000). A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning, 25*(3), 219–246.

Fourrey, E. (1924). *Procédés Originaux de Instructions Géométriques* (pp. 86–89), Paris: Librairie Vuibert.

Gallier, J. H. (2003). *Logic for computer science: Foundations of automatic theorem proving*. New York, NY: Harper and Row Publishers. Available at http://www.cis.upenn.edu/~jean/gbooks/logic.html.

Gao, X.-S., & Zhu, C. (1999). Building dynamic mathematical models with geometry expert – iii a geometry deductive database. *Proceedings of Asian Technology Conference in Mathematics*, 153–162.

Gelernter, H. (1963). Realization of a geometry theorem-proving mechanism. In *Computers and Thought* (pp. 134–152). New York: McGraw-Hill.

Gelernter, H., Hanson, J. R., & Loveland, D. W. (1963). *Empirical explorations of the geometry-theorem proving machine. Computers and Thought*. New York: McGraw-Hill.

Gilmore, P. C. (1970). An examination of the geometry theorem-proving machine. *Artificial Intelligence, 1*, 171–187.

Grothman, R. (1999). C.a.R.: Compass and Rules. http://www.z-u-l.de/doc_en/index.html.

Hara, N., & Kling, R. (1999). Students' frustrations with a web-based distance education course. *First Monday: Journal on the Internet, 4*(12). Available at http://www.firstmonday.dk/issues/issue4_12/index.html.

Hentea, M., Shea, M. J., & Pennington, L. (2003). A perspective on fulfilling the expectations of distance education. *Proceedings of the 4th conference on Information technology curriculum*, 160–167.

Holland, G. (1993). *Geolog Geometrische Konstruktionen mit dem Computer*. Dümmler Verlag. Available at http://www.uni-giessen.de/~gcp3/Geolog/geolog.htm.

Isotani, S. (2005). *Developing tools in iGeom: Using the dynamic geometry in the classroom and distance learning, master dissertation*. Institute of Mathematics and Statistics of the University of São Paulo, 2005. Available at http://www.teses.usp.br/teses/disponiveis/45/45134/tde-17092005-063522/.

Isotani, S., & Brandão, L. O. (2004). Tool of automatic evaluation in the iGeom. In *Proceedings of the Brazilian symposium of informatics in education*, 328–337.

Jackiw, N. (designer) (1995). *The Geometer's Sketchpad*. Berkeley, CA: Key Curriculum Press.

Kortenkamp, U. (1999). *Foundations of dynamic geometry*. Ph.D. Thesis No. 13403, Swiss Federal Institute of Technology, Zurich, Switzerland. Available at http://kortenkamps.net/papers/diss.pdf.

Litto, F. M. (2006). Learning with technology in Brazil: A study in contrasts and conquests. *Advanced Technology for Learning, 3*(2), 62–68.

Lucena, C. J. P., Fuks, H., Raposo, A., Gerosa, M. A., & Pimentel, M. (2006). Communication, coordination and cooperation in computer-supported learning: The AulaNet experience. *Advances in Computer-Supported Learning*, 274–297.

Mizoguchi, R., & Bourdeau, J. (2000). Using ontological engineering to overcome AI-ED problems. *International Journal of Artificial Intelligence in Education, 11*(2), 107–121.

Moodle (2007). http://moodle.org/.

Moraes, T. G., Santoro, F. M., & Borges, M. R. S. (2005). Tabulae: educational groupware for learning geometry. In *IEEE international conference on advanced learning technologies*, Kaohsiung, 750–754.

Moura, J. G., Brandão L. O., & Brandão, A. A. F. (2007). A web-based learning management system with automatic assessment resources. In *Proceedings of ASEE/IEEE Frontiers in education conference*, Session F2D, 1–6.

Oldknow, A. (1997). Dynamic geometry software – a powerful tool for teaching mathematics, not just geometry. In *Proceedings of international conference on technology in mathematics teaching*. Available at http://citeseer.ist.psu.edu/341832.html.

Raggett, D., Hors, A. L., & Jacobs, I. (1999). HTML 4.01 Specification. Available at http://www.w3.org/TR/REC-html40/.

Ruthven, K., Hennessy, S., & Deaney, R. (in press). Constructions of dynamic geometry: A study of the interpretative flexibility of educational software in classroom practice. Computers and Education http://dx.doi.org/10.1016/j.compedu.2007.05.013.

Santos, E. T., & Sola, J. I. R. (2001). A proposal for an on-line library of descriptive geometry problems. *Journal for Geometry and Graphics, 5*(1), 93–100.

Schwartz, J. T. (1980). Probabilistic algorithms for verification of polynomial identities. *Journal of the ACM, 27*(4), 701–717.

Sinclair, M. P. (2005). Peer interactions in a computer lab: Reflections on results of a case study involving web-based dynamic geometry sketches. *Journal of Mathematical Behavior, 24*(1), 89–107.

Yin, R. K. (2002). *Case study research, design and methods* (3rd ed.). Newbury Park: Sage Publications.