Knowledge Anticipation on Agents Relationship in the Geometry Proof System

Márcia R. Notare, Júlio P. Machado, Tiarajú A. Diverio, P. Blauth Menezes
Instituto de Informática and PPGC/UFRGS
Av. Bento Gonçalves 9500, Campus do Vale, Bloco IV, Caixa Postal 15064,
CEP 91501-970, Porto Alegre – RS, Brazil
E-mail: {notare, jhapm, diverio, blauth}@inf.ufrgs.br

Abstract
This paper describes the Geometry demonstration learning system LEEG (Learning Environment on Euclidean Geometry). This system was constructed on a learning environment composed of five agents that interact to promote the knowledge construction and evolution. The five agents are: *Mestre*, *Oráculo*, *Sonda*, *Cliente* and *Aprendiz* (or in English, respectively, Master, Oracle, Probe, Client, Apprentice), each one with distinctive and specific behavior. The focus of this work will be the specification of the *Mestre-Oráculo* and *Mestre-Sonda* relationships and the knowledge base specification.

Keywords: knowledge anticipation, learning system, automata theory, geometry demonstration, *MOSCA* protocol.

1 Introduction

This paper describes the Geometry demonstration learning system LEEG (Learning Environment on Euclidean Geometry). This system was constructed on a learning environment composed of five agents that interact to promote the knowledge construction and evolution. The five agents are: *Mestre*, *Oráculo*, *Sonda*, *Cliente* and *Aprendiz* (or in English, respectively, Master, Oracle, Probe, Client, Apprentice), each one with distinctive and specific behavior.

Cliente starts the learning process sending a proposition to *Aprendiz* to be demonstrated and awaits its solution. *Aprendiz* must construct the demonstration with help of examples/counter-examples sent by *Oráculo/Sonda*. *Mestre* controls the learning process by permanently accessing the knowledge base of *Aprendiz* and comparing to the knowledge base that contains all the correct demonstrations. This means that the agent *Mestre* knows previously the demonstration structure that was proposed to *Aprendiz*. In this way, it always signalizes to *Oráculo* and *Sonda* when it identifies an incoherence in the *Aprendiz* construction, in order to coordinate the sending of examples and counter-examples that help the correct demonstration construction.

Mestre has access to the complete demonstrations proposed by *Cliente*. So, it is, in some sense, able to anticipate the possible incorrectness committed by *Aprendiz*. Thus, it can require that examples have to be send to *Aprendiz*, avoiding the execution of this incorrectness. This permits a better control over the demonstrations construction developed by *Aprendiz*.

A demonstration on Euclidean Geometry is composed by a set of statements that must obey a hierarchical order and must be rigorously justified. By this reason, the knowledge base implementation was structured by Hyper-Automaton model [2], [3], [5]. This model enables an adequate structure for the demonstrations statements and possibility the verification of the learning process through comparison of automata substructures.

The focus of this work will be the specification of the *Mestre-Oráculo* and *Mestre-Sonda* relationships and the knowledge base specification. Also, initially, the anticipatory system attribute had not been specifically tackled in the project, but implicitly incorporated, which motivated us to make it explicit and then explore it in this work.

2 MOSCA Protocol

Formal theories of learning state that a minimal learning environment should comprise a learner in communication with an oracle in order to enable learning.

Based on these assumptions, Reitz [4] has proposed a learning environment known as MOSCA (**M**estre + **O**ráculo + **S**onda + **C**liente + **A**prendiz) as shown in Figure 1. This environment is composed by 5 distinct entities (considered as human or artificial agents), each with specific behavior, according to proposed aims and interacting in the learning process. MOSCA is a learning protocol based on learning by example, which have been used and adapted by many in the literature [7], [8], [9].

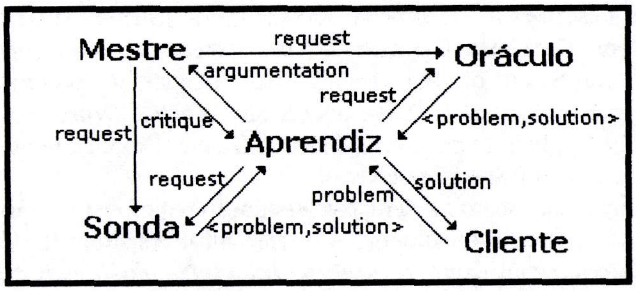

Fig. 1: MOSCA learning protocol

In summary, the initial proposal of the MOSCA protocol is as follows: the process starts with *Cliente* which submits a problem to be solved to the *Aprendiz*. The *Aprendiz* also receives problems solved by *Sonda*. However such solutions can be incorrect. The *Aprendiz* compares its solution with *Sonda*'s solution and defends its solution to the *Mestre*. For each argumentation produced by the *Aprendiz*, a critical argument is sent by the *Mestre*. Criticized arguments are memorized by the *Aprendiz*. Every negative argument leads the *Aprendiz* to present a new argumentation.

Initially, the anticipatory system attribute [1] had not been specifically tackled in the project, but implicitly incorporated, which motivated us to make it explicit and then explore it in this work. The *Mestre*, entity responsible for monitoring the actions of the *Aprendiz*, knows in advance the final and partial objectives that the *Aprendiz* should meet. In that way, the *Mestre* can in anticipation, warns the *Oráculo* and the *Sonda* what intervention are necessary in order to reach the final objective successfully. I.e., the *Mestre*, having a global view of the process can lead the *Aprendiz* to the correct learning path, aided by the *Oráculo* and the *Sonda*.

An example of this behavior can be observed in the application of the learning protocol in the context of arithmetic, specifically, in learning how to solve numerical expressions. Next, possible ways that can be followed by the *Aprendiz* in the solution of numerical expressions (Figure 2).

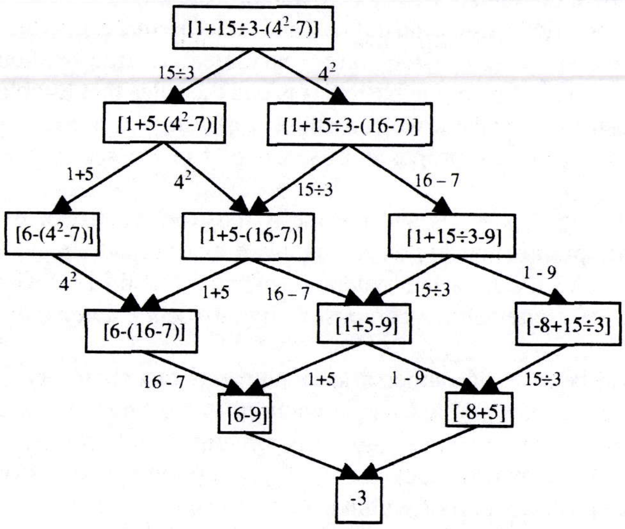

Fig. 2: Resolution of arithmetic expression

For each incorrect operation that the *Aprendiz* tries to execute, the *Oráculo* and *Sonda* agents intervene in the construction through messages, with the aim of telling the *Aprendiz* to repeat the operation. For instance, if in the first moment the *Aprendiz* calculate *1+15*, the *Sonda* agent intervenes, warning that division has priority over addition. And this should happen in all operations not available for resolution.

3 Learning Environment on Euclidean Geometry

The LEEG (Learning Environment on Euclidean Geometry) [11] is a learning environment for Deductive Euclidean Geometry, based on adaptations of the MOSCA learning protocol with the aim of helping the construction of theorem proving process in Euclidean Plane Geometry.

The system proposes the learning of geometric proofs constructions with the aid of examples and counterexamples, which characterize interventions of the system when inconsistent statements or incorrect use of terms are used.

As Euclidean Geometry is a classical example of an axiomatic system [10], its structural form can be proved from a certain number of base premises that give rise to derived propositions [6].

The terms involved in a deductive system are the following [10]:

- ***Definitions***: assertion that only requires a comprehension of the terms applied;
- ***Postulates***: principles or facts acknowledged but not demonstrated and admitted without demonstration;
- ***Axioms***: evident propositions and not subject to demonstration;
- ***Propositions***: object's property assertions (theorems) or steps or its construction steps (problems), that must be subject to demonstration.

In other words, the definitions, postulates and axioms make up the set of evident terms in an axiomatic system, which are considered as true without proof. The propositions must be proved from the basic terms and the rules that establish the system. Whenever a proposition is proved, we can admit it as true and use it in order to prove new propositions, i.e., this new proposition is now part of the set of true statements of the system.

In a proposition, the hypothesis and the thesis must be identified. In the case of a deductive proof, the proposition hypothesis is taken as true and should be used in the proof construction. The thesis is the statement of what should be proved, by a rigid logical sequence of statements, composed by evident statements and proved propositions.

A proof then will be a set of mathematical statements that should be justified by the use of definitions, postulates and axioms in addition to the propositions already shown previously. These statements must be rigorously structured and ordered in a logical and hierarchical form. The demonstrations are rigidly structured and this structure must be followed in order to produce a proof within a axiomatic deductive system.

Thus, the LEEG system aims at helping an agent (*Aprendiz*), human or virtual, in the construction of proofs in Euclidean Plane Geometry, have as a basis the rigidity of the axiomatic system. The construction of the demonstrations developed by the *Aprendiz* must follow a logical sequence of construction. Each step of the *Aprendiz* is followed by the *Mestre* agent which identifies inconsistencies and mistakes and activates the message passing mechanism (examples and counter-examples) that intervene in the construction.

The deductive demonstrations in LEEG are developed in textual form, organized in a table composed by two columns, named *Statement* and *Deduction Rule*. The *Aprendiz* will develop its demonstration over them. For each field in the statement column there will be an equivalent field in the deduction rule column, which must be exactly filled by the deduction rule that concluded the corresponding statement. Only statements which cannot be deduced via deduction rules can have the field not filled in.

The LEEG accepted deduction rules comprise 23 definitions, 5 axioms and 5 postulates of Euclidean Geometry, applied to the right elements. For instance, Postulate 1 which states that *"It is possible to draw a straight line from any point to any point"* refers to the construction of a segment from two given points. This means it should be applied to exactly two points.

The first and last statements of the proof must be, respectively, the hypothesis and thesis of the proposition.

Table 1 presents the proof of Proposition 1 *(To construct an equilateral triangle over a give line segment)* as it should be constructed by the *Aprendiz* in the LEEG interface.

Table 1: Demonstration of Proposition 1

Statement	Deduction Rule
segment AB	(Hypothesis)
circle A, AB	Postulate 3 (A, AB)
circle B, AB	Postulate 3 (B, AB)
point C = intersection (circle A,AB; circle B,AB)	(Statements 2 and 3)
segment AC	Postulate 1 (A, C)
segment BC	Postulate 1 (B, C)
segment AC = segment AB	Definition 15 (AC, AB)
segment BC = segment AB	Definition 15 (BC, AB)
segment AC = segment BC	Axiom 1 (AB, AC, BC)
segment AC = segment AB = segment BC	(Statements 7, 8 and 9)
triangle ABC = equilateral	Definition 20 (AB, AC, BC)

The fields in the Deduction Rule column that are shown between round brackets do not need to be included in the demonstration developed in the system. The hypothesis falls into this case as it is not a statement deduced from a deduction rule (is extracted from the proposition) and the other statements that fall into this case, in the above example, are only facts with no deduction rules.

Each reasoning step of the *Aprendiz*, i.e., each line in the proof table is supervised by the *Mestre*. The next step is enabled only if the current reasoning is correct. Otherwise, the *Oráculo* and *Sonda* agents are warned in order to send examples and counterexamples that tell the *Aprendiz* about its mistake and make it possible for it to correct the mistake.

In this way, the permanent monitoring of the proof construction developed by the *Aprendiz* allows the *Mestre* to anticipate likely mistakes that could be made by the *Aprendiz*, warning the *Oráculo* and *Sonda* agents to send messages that influence the construction of a correct proof.

Figure 3 shows the MOSCA protocol adapted for the LEEG system.

In the following items, we describe the knowledge base of the system and the interaction between agents while learning how to make proofs about Geometry.

4 Agents Relationship

This section aims at specifying the interaction between the five agents that compose the LEEG system.

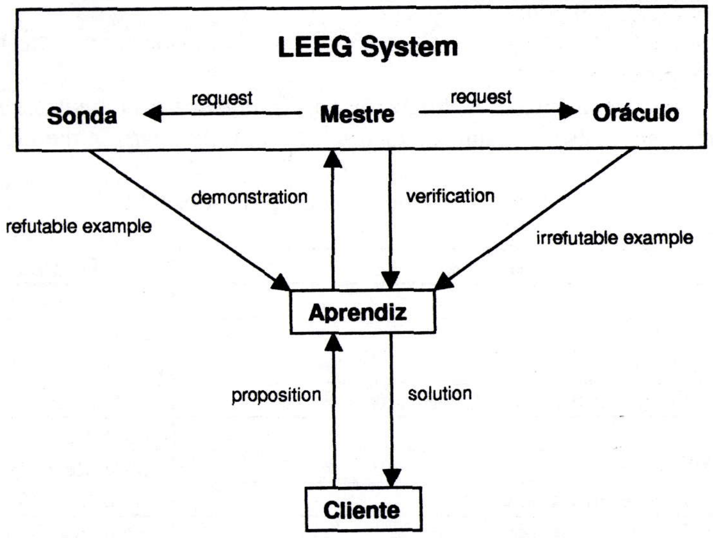

Fig. 3: LEEG's version of MOSCA protocol

4.1 Cliente ↔ Aprendiz Interaction

The communication between *Cliente* and *Aprendiz* agents establishes the beginning of the learning process. The *Cliente* sends a proposition to the *Aprendiz* (Cliente → Aprendiz), from the propositions available for a proof, and awaits the result.

In the chosen proposition, the thesis and hypothesis are identified. The hypothesis should be used by the *Aprendiz* as the initial statement of the proof. The thesis should be proved by a sequence of statements logically deduced and should be the last statement of the proof.

Once the proof is concluded, the *Aprendiz* returns the result of the proof to the *Cliente* (Cliente ← Aprendiz), presenting a complete demonstration.

4.2 Mestre ↔ Aprendiz Interaction

The interaction between the *Mestre* and *Aprendiz* agents is related to the construction (Mestre ← Aprendiz) and verification (Mestre → Aprendiz) of learning. Each state of the proof developed by the *Aprendiz* is sent to the *Mestre* which verifies the construction. Three situations may happen: correct construction, wrong construction of a statement or wrong construction of a deduction rule.

If the construction is correct, the *Mestre* send a signal to the *Aprendiz*, authorizing the next step of the construction.

If the construction of the statement or of the deduction rule is wrong, the *Mestre* do not authorizes the next step of the construction and send a message to the *Oráculo* and *Sonda* agents, which will warn the *Aprendiz*.

The verification of the steps developed by the *Aprendiz* is done though comparisons between its demonstration and the demonstration in the knowledge base of the system. As the *Mestre* has access to the correct construction in the knowledge base, it can anticipate likely mistakes in the construction of a proof by the *Aprendiz*, and prevent them by sending messages about correct steps in the demonstration.

4.3 Mestre → Oráculo Interaction

The communication between the *Mestre* and *Oráculo* agents is made in only one direction and it is related to the request by the *Mestre* to the *Oráculo* of messages to be sent to the *Aprendiz*. This warning happens in two situations: wrong construction of a deduction rule and correct construction.

If the *Mestre* identifies a mistake in the deduction rule constructed by the *Aprendiz*, then it warns the *Oráculo* so that it sends a message to the *Aprendiz* to correct the referred state of the proof.

On the other hand, if the *Mestre* checks the current state of the proof by the *Aprendiz*, and after the habilitation signal no action by the *Aprendiz* is identified by the *Mestre*, a signal is send to the *Oráculo* so that it can send a message encouraging the correct construction. Again, an anticipation about the next state of the proof happens and it helps the *Aprendiz* to construct a correct step.

4.4 Mestre → Sonda Interaction

This interaction is also unidirectional. The communication is established through a signal sent from the *Mestre* to the *Sonda* asking for a reflection message to be sent to the *Aprendiz*.

This signal is sent whenever the *Mestre* identifies an error in the statement of the *Aprendiz* construction. The reflection messages have the aim of helping the *Aprendiz* to identify and correct its error, but provide specific information about the correct construction of the statement.

4.5 Oráculo → Aprendiz Interaction

The communication established between the *Oráculo* and *Aprendiz* agents is related with the examples used to help in the learning process. The *Oráculo* aims to send a message to the *Aprendiz* that helps it in choosing the right deduction rules, for each step of the proof (correction messages) or messages that encourage the *Aprendiz* the continuation of the proof when the *Aprendiz* abandons a proof (incentive messages).

The *Oráculo* has access to a Module of Irrefutable Examples (which is in the Examples Base) which can be accessed depending on the current state of the proof by a signal from the *Mestre*.

4.6 Sonda → Aprendiz Interaction

The *Sonda* agent aims at sending reflection messages to the *Aprendiz* agent, whenever the *Mestre* identifies mistakes in the construction of statements. These reflection messages are determined according to the *Mestre* signal, which identifies the current state of the proof and the mistake made by the *Aprendiz*.

The *Sonda* agent has access to a Module of Refutable Examples (also included in the Examples Base), where each message is associated to a state in the proof.

5 Knowledge Data Structure

Data in the LEEG system are structured as follows: the *Knowledge Base*, including definitions, axioms, postulates and proved propositions; the *Propositions Base* which contains propositions to be demonstrated; the *Examples Base*, composed by two Modules (*Irrefutable Examples* and *Refutable Examples*). Figure 4 presents the data structure of the system.

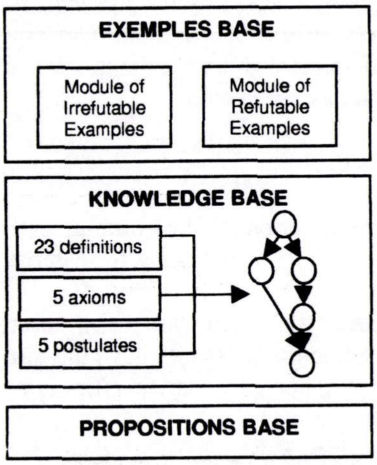

Fig. 4: LEEG's data structure

5.1 Knowledge Base

The knowledge base of the system is the complete set of all possible statements that can be used by the *Aprendiz* in the proof of each one of the propositions. The proofs of the propositions are developed in the basis of the system, obeying the logical sequence of the statements and justifications that should be reproduced by the *Aprendiz*. In the case of LEEG, the knowledge base may be formed by 23 definitions, 5 axioms, 5 postulates and 48 proposition theses.

The structure of a proof can be represented by a graph, where all the statements of Geometry being used in the proof are hierarchically organized and the edges between statements are the terms that justify their use (the deduction rules).

In the LEEG's knowledge base the structure of the proofs is organized and implemented by a Finite Automata model, which allows an efficient data organization [2], [3], [5].

5.2 Propositions Base

The propositions base or possible propositions databank which are to be submitted to the *Aprendiz* is determined by the scope of the knowledge base. In the case of Euclidean Plane Geometry, there are 48 possible propositions to be submitted by the *Cliente* to the *Aprendiz*. The *Cliente* agent is responsible for the access to this set of propositions, ànd chooses a proposition to be submitted to the *Aprendiz* to be proven. The proposition can be chosen *randomly* or *structurally*.

In the latter case the dependency order between propositions is followed, i.e., a proposition will be available for submission if the other propositions needed for its proof were already proved by the *Aprendiz*. In that way, knowledge acquisition is done step by step, in an *inductive* way.

5.3 Examples Base

The examples base of the system is composed by the set of message interventions send by the system (though the *Oráculo* and *Sonda* agents) to the *Aprendiz*.

The interventions send by the system to the *Aprendiz* have two aims:

- broadcasting of examples containing enough hints on deduction rules, in order to lead the *Aprendiz* agent to a correct proof construction, i.e., making it possible the inclusion of deduction rules consistent with the current proof;
- broadcasting of counter-examples, whenever the *Aprendiz* agent makes a statement mistake, in order to warn it about the mistake make and helping in the correction.

The errors make by the *Aprendiz* can be related to the statements or to the deduction rules. *Incoherence* of a demonstration is a consequence of statements derived from incorrect deduction rules and the *inconsistency* of a demonstration is the introduction of statements with no application of a deduction rule.

The *Mestre* agent is responsible for deciding whether to broadcast messages since it follows and checks, step by step, the reasoning developed by the *Aprendiz* agent. It is by the *Mestre* request that the *Oráculo* and *Sonda* agents lead the *Aprendiz* to the correct construction.

In that way, the *Oráculo* helps the *Aprendiz*, by a *Mestre* signal, by sending examples that lead to the correct deduction rule. On the other hand, the *Sonda* agent is warned by the *Mestre* whenever the *Aprendiz* makes a mistake in the statement of a proof, so that *Sonda* sends a counter-example to warn the *Aprendiz* about its mistake and allows a reflection about it and then a correction.

5.3.1 Irrefutable Examples Module

The irrefutable examples module is accessed only by the *Oráculo*, which is responsible for sending messages containing examples and hints. The messages in this module are irrefutable, i.e., they are correct and can be accepted by the *Aprendiz* with total confidence.

The messages in this module are divided into two categories, identified by their functionality: *Correction Messages* and *Incentive Messages*.

Correction messages are sent whenever an identification error in the *Aprendiz* deduction rules is detected and aims at providing hints to warn the *Aprendiz* about the error and to direct a correction. Figure 5 shows the interaction between the *Aprendiz*, *Mestre* and *Oráculo* agents during the error identification process and the correction message production.

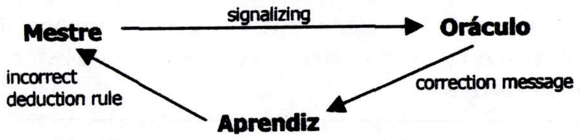

Fig. 5: Production of correction message

In summary, the *Aprendiz* produces a wrong deduction rule, the *Mestre* identifies the error by comparing it to the LEEG knowledge base and tells the *Oráculo* to send a correction message about the current state of the proof to the *Aprendiz*. These messages can still be classified into four distinct levels, according to the error identified:

- *hint about the type of element that the deduction rule should be applied* – sent whenever the *Aprendiz* makes a mistake related to the type of geometric element (point, segment,...) on which the deduction rule should be applied. I.e., the *Aprendiz* chooses the right deduction rule, but not the right element. (Example message: "Postulate 3 must be applied to a point and a segment").
- *hint that specifies the elements over which a deduction rule must be applied* - sent whenever the *Aprendiz* makes a mistake in the element(s) (and not in the type of element), specifying which element(s) that the rule should be applied to. (Example message: "Apply postulate 3 to point A and segment AB").
- *message warning about incoherence between statement and deduction rule* - sent to warn the *Aprendiz* that the inserted deduction rule is incoherent with the respective statement, i.e., the statement cannot be deduced from the rule. (Example message: "Deduction rule incoherent with statement").
- *message warning about inconsistency in deduction* - sent whenever the *Aprendiz* does not insert the deduction rule for a statement that should be deduced from a rule. (Example message: "Inconsistent Deduction").

Incentive Messages are sent when two facts are detected: the statement and corresponding deduction rule are correct, but no action from the *Aprendiz* for a certain amount of time (usually set to 3 minutes) is detected, which is interpreted as undecided behavior or withdrawal from learning. These messages aim at stimulating the *Aprendiz* to continue with the proof when indecision or detachment are detected, by sending hints about the next state of the demonstration. Figure 6 presents the interaction between the *Aprendiz, Mestre* and *Oráculo* agents during the incentive message production process.

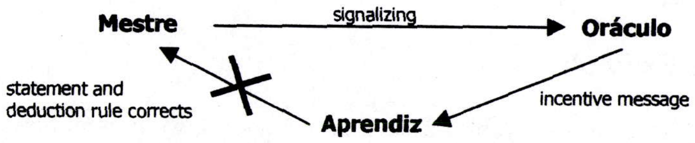

Fig. 6: Production of incentive message

Each state of the proofs may have more than one associated incentive message, anticipating the different reasoning strategies that can be adopted by the *Aprendiz*. An example message in this category is the following: *"You can check the equality of segments AB and AC"*.

5.3.2 Refutable Examples Module

The refutable examples module is accessed only by the *Sonda* agent, which sends counter-examples that may be refuted by the *Aprendiz*.

The messages in this module, known as *Reflection Messages*, are sent only in situations in which the *Aprendiz* makes a statement error. These messages aims to warn the *Aprendiz* about the error, creating situations which make him think about the concluded statement. Figure 7 presents the interaction between the *Aprendiz, Mestre* and *Sonda* agents during the reflection message production process.

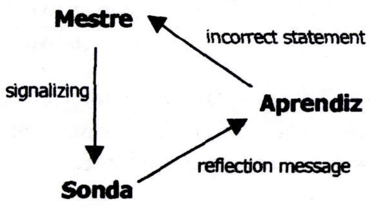

Fig. 7: Production of reflection message

These messages can be classified into four distinct levels, according to their objective:

- use of elements (points, segments,...) which have not been previously constructed, i.e., that do not exist. (Example message: "What is the BC segment?")
- incorrect use of variables, for instance the construction of a segment using three points or the construction of a triangle with only one point. (Example message:

"How many vertices determine a triangle? Think for a while...")

- to write out correct statements before their derivations. (Example message: "You cannot conclude that yet")
- any other situation not foreseen in the previous situations. (Example message: "Incorrect statement")

6 Interactions Example

In the example that follows (Tables 2, 3, 4, 5, 6, 7 and 8), we present how messages are used in the LEEG system by illustrating a possible interaction between the *Aprendiz*, *Oráculo* and *Sonda* agents during a proof of Proposition 1. One of the possible correct constructions for this proof is illustrated in Table 1 and is taken as a basis for the interactions presented below. The statements and the deduction rules written in boldface represent error situations that render the messages sent by the *Oráculo* and the *Sonda* agents.

Table 2: Example of deduction

Statement	**Deduction Rule**
Segment AB	
Circle A, AB	**Postulate 3 (B, AB)**

Observe that the second statement was correctly inputted, but the deduction rule is applied to an incorrect element (the center of the circle should be A and not B, as it was written out). In this case the *Mestre* agent identifies the error and signalize the *Oráculo* so that it sends the following message: *"Apply postulate 3 to point A and segment AB"*.

Table 3: Example of deduction

Circle A, AB	Postulate 3 (A, AB)
Circle B, AB	Postulate 3 (B, AB)

Using an anticipatory message, the *Mestre* could tell the *Oráculo* to send the message *"It is necessary to determine the circle intersection point A, AB and circle B, AB"*. This message anticipates the next step of the proof, avoiding possible construction mistakes.

Table 4: Example of deduction

Point C = intersection (circle A,AB; circle B,AB)	
Segment AC	Postulate 1 (A, C)
Segment BC	Postulate 1 (B, C)
Segment AC = Segment BC	

The statement Segment AC = Segment BC was inserted out of a logical and hierarchical ordering. Then the following message is sent by the *Sonda*: *"You cannot conclude that yet"*.

Table 5: Example of deduction

Segment AC = Segment AB = Segment BC	

If the *Aprendiz* makes a mistake over the same statement, the following message is sent: *"You can check the equality of segments AB and AC"*.

Table 6: Example of deduction

Segment AC = Segment AB	Definition 15 (AC, AB)
Segment BC = Segment AB	Definition 15 (BC, AB)
Segment AC = Segment BC	**Axiom 3 (AB, AC, BC)**

In this case, the deduction rule is incorrect, even though the statement was inputted correctly. Then the following message is sent: *"The deduction rule and the statement are not coherent."*

Table 7: Example of deduction

Segment AC = Segment BC	Axiom 1 (AB, AC, BC)

Again, an anticipatory message may be sent: *"You can now conclude the equality of the three sides of the triangle"*.

Table 8: Example of deduction

Segment AC = Segment AB = Segment BC	
Triangle ABC = equilateral	Definition 20 (AB, AC, BC)

When the proof is finished, the following message is sent: *"Proof concluded with success!"*.

Obviously, not all possible messages were presented in this example. We just illustrated how the system works during the knowledge construction process.

7 Concluding Remarks

The development of the work presented here is part of a conception of a proof learning system for Euclidean Plane Geometry which, as a first instance, was not thought of as an anticipatory system. However, its formalization led us to notice that we could consider it as such.

The anticipatory behavior can be observed in the *Mestre-Oráculo* and *Mestre-Sonda* interactions which follows step by step the *Aprendiz* reasoning and send messages aiming at avoiding or correcting errors.

In the LEEG system the *Aprendiz* agent was prototyped as a human entity. However, its structure allows one to study its structure as an artificial entity which should learn the logically ordered path of geometrical proofs.

Even though the system was conceived to proof learning in Geometry, its structure supports other application domains if appropriate adaptations are taken into consideration.

In the implementation of the first prototype some characteristics considered in the specification were simplified thus, the implemented agent presents a reactive behavior. We are current working on the implementation of the full specified agent.

References

[1] Dubois D. (1999). *Review of Incursive, Hyperincursive and Anticipatory Systems – Foundation of Anticipation in Electromagnetism.* Computing Anticipatory Systems: CASYS'99 – Third International Conference, edited by D. M. Dubois, published by The American Institute of Phisics, New York, AIP Conference Proceedings 517, p.3-30.

[2] Machado J.P., De Morais C.T. & Menezes P.B. (1999). *Finite Automata: a formalism for web courses.* 13th Brazilian Symposium on Software Engineering: SBES'99, D. J. Nunes and M. S. Camargo (Eds.), published by SBC, Florianópolis, Brazil, p.213-223.

[3] Machado J.P., De Morais C.T., Menezes P.B. & Reis, R. (2000). *Structuring Web Course Pages as Automata: revising concepts.* Recherche D'Informations Assistee par Ordinateur: RIAO'2000, published by C.I.D. and C.A.S.I.S., Paris, v.1, p.150-159.

[4] Reitz, P. (1992). *Contribution a l'Étude des Environnements d'Apprentissage: Conceptualisation, Spécifications et Prototipage.* Thèse de Doctorat, Université des Sciences et Techniques du Languedoc Montpellier II, France.

[5] Machado J.P., Notare M.R., Costa S., Diverio T. & Menezes P.B. (2001). *Hyper-Automaton System Applied to Geometry Demonstration Environment.* Computer Aided Systems Theory: EUROCAST'2001 - Eigth International Conference, A. Quesada and R. M. Diaz (Eds.), published by IUCTC, Las Palmas de Gran Canaria, Spain, p.136-139 (to appear in LNCS volume 2178).

[6] Meyer B. (1974). *An Introduction to Axiomatic Systems.* Weber & Schmidt, Boston.

[7] Costa E., Lopes M., Ferneda E. (1995). *Mathema: A Learning Environment based on a Multi-Agent Architecture.* 12th Brazilian Symposium on Artificial Intelligence: SBIA'95, J. Wainer and A. B. R. de Carvalho (Eds.), published by Springer-Verlag, Berlin, Lecture Notes in Computer Science 991, p.141-150.

[8] Ferneda E. (1992). *Conception d'un Agent Rationnel et Examen de son Raisonnement en Géométrie.* Thèse de Doctorat, Université des Sciences et Techniques du Languedoc Montpellier II, France.

[9] Luengo V. (1996). *Um Micromundo para la Resolución de Problemas de Demonstración en Geometría.* 22th Conferencia Latinoamericana de Informatica: CLEI'96, edited by R. Cardoso, published by Universidad de Los Andes, Colômbia, p.941-952.

[10] Euclid. (1978). *The Thirteen Books of Euclid's Elements.* (Translation by Heath T.T.) Dover Publications, New York.

[11] Notare M. & Divério T. (2001). *Aprendiz's Learning in Geometry Demonstration.* 7th World Conference on Computer in Education: WCCE'2001, J. Anderson and C. Mohr (Eds.), published by UNI-C, Copenhagen, Denmark. (to appear)