An Open Architecture to Improve Mathematical Competence in High School

Philippe R. Richard
Département de didactique
Université de Montréal
Canada
philippe.r.richard@umontreal.ca

Simon El-Khoury
Département d'informatique et de recherche opérationnelle
Université de Montréal
Canada
elkhous@iro.umontreal.ca

Esma Aïmeur
Département d'informatique et de recherche opérationnelle
Université de Montréal
Canada
aimeur@iro.umontreal.ca

Josep M. Fortuny
Departament de Didàctica de la Matemàtica i de les Ciències Experimentals
Universitat Autònoma de Barcelona
Espanya
JosepMaria.Fortuny@uab.es

Abstract. This paper presents the development of an intelligent tutoring system called TURING (French acronym of «TUtoRiel INtelligent en Géométrie») in a multidisciplinary project, that joint recent research in didactic of mathematics with the possibilities of computer-based learning environments. From the educational point of view, the system is helpful for the student to improve problem solving aptitudes, mathematical reasoning abilities and communication skills using natural and mathematical language. In addition, the system is helpful to assist the teacher in his responsibility of attending the diversity of the development of mathematical competences in a whole class. From the technical point of view, the TURING architecture is a multi-agent system conceived in a flexible approach so that a teacher can adjust the heuristic and discursive features according to specific practices with real students. In particular, for the discovery of a conjecture or for the realization of a mathematical proof, the system consents to adapt, in the problem solving process, the space of meaningful actions and, in argumentative process, the set of strategic message with pedagogical agents. The TURING is a client-server application that is in its phase of implementation in Java programming language.

Introduction

One of the purposes of research in Didactics of Mathematics is to improve the teaching and learning processes by suggesting new teaching models which lead to significant learning and, as a consequence, to a greater success at school and in society. It is a fact that the increase in the quality of interactions between the different poles taking part in the educational system (student, teacher, knowledge and milieu) has cognitive and social benefits for the students. This increase in the quality of interactions can be achieved by designing, implementing and validating those learning environments that focus teaching on students and that are developed taking into account the latest research on teaching Mathematics and, concretely, the contributions on the use of ICTs and the Internet and the developments in e-learning (elearningeuropa.org).

The use of virtual, interactive and collaborative systems of tutoring allow students to work in a flexible way, either autonomously, in interaction with the virtual tutor, or with the teacher or other students. Moreover, it encourages the great quality of these learning, which supports the acquisition of training attitudes, values and habits that it will have need in the future to enter the world of work (Techno-mathematical Literacies in the Workplace).

Besides, the flexibility of these systems allows to meet the educational and training demands of groups of people whose needs could not have been duly met due to several reasons, namely students with marked cognitive differences, not only those who show deficiencies in their knowledge and attitudes, but also those whose skills are more developed; newly arrived students from other countries who show difficulties in oral communication rather than in written, graphic, virtual and interactive communication and those students who need personalised support outside the classroom due to several reasons.

With the intention of meeting the demands of this situation of diversity and positively stressing its integration into the classroom, the main interests of this project are the following:

- Developing a tutorial system in order to improve students' competence when solving mathematical problems and those teachers' competence needed in order to focus on understanding how their students solve problems in an e-learning environment.
- Analyzing the influence and effects of this sort of interactions on the development of the students' strategic competences when solving problems and characterizing the competence levels generated.

This tutorial system is partly based on what our international team of research and development has carried out and it simulates the complex didactic process by means of a virtual pedagogical agent that interacts with the student. The analysis and characterization of different competence levels in problem solving will allow providing information on the potential of the tutorial system as a learning instrument. The project is innovative by the integration of the theoretical approaches on which it is based.

The development of the project will also include aspects related to teacher training and the teachers' performance as tutors that carry out their task both in the classroom and virtually (eTutor), as well as the dissemination and exploitation of results in the education sector. There exist some other projects with similar goals, for example *Baghera*, *Exploragraph* and *AgentGeom*. Each project has its own characteristics (see section *Related Work*).

Our TURING system shows the application of a model of interactions that is based on research into didactic of mathematics for the creating of an intelligent virtual pedagogical agents in an artificial tutoring system. This system offers to the students a metacognitive support to help them in their development of mathematical competences (e.g. problem solving, mathematical reasoning, communication with natural and mathematical languages). The agent architecture of TURING system is conceived as a hybrid multiagent architecture made up of four distinct agents. Each agent has intelligence to support and help the student, the teacher and the didactic expert using an open architecture and flexible structure.

This paper is organized as follows: the next section discusses other applications similar to the TURING. In section *The Didactic Background of TURING System*, we present the didactic aspects used in it, and in section *The Approach of TURING*, we describe the architecture and the functionality of the system, while gradually bringing closer the didactic aspects with the characteristics of the system. Finally, in the last section, we draw some conclusions and present future work planned for the TURING.

Related Work

Improve teaching and learning process, the main objective of TURING system, is also the goal of many other projects currently under way at several research centres. Like immediate antecedent of the TURING, the *Baghera* project, directed by the researcher Nicolas Balacheff within the Leibniz Laboratory (e.g. Laboratoire Leibniz, 2003). *Baghera* uses algorithmic models in proving geometric problem to check the strategy automatically with a «deductive engine». The *AgentGeom* (Cobo, Fortuny & al., 2005) is a first prototype of an intelligent tutor system that uses the construction of a geometrical figure to generate interactions. Just like *Baghera*, *AgentGeom* is conceived as a multiagent tutorial system of diagnosis that can identify knowledge of the students through their interactions with the system That comprises an adaptation of the system to the cognitive characteristics of the students and to the evolution of their mathematical knowledge. *Exploragraph* (Dufresne & Paquette, 2000) focus on the design of interface for learners taking distance courses and also to integrate in it adaptive functions and advices to support them but they dont use argumentative message in their support. The *Cabri* (e.g. Cabri-géomètre, 2005) is a software aimed to create geometric figures based on the geometric logic. Unlike these projects, the TURING may be distinguished not only by the new integration of theoretical approaches, but especially by the flexible nature of the heuristic strategy and the strategic messages (discursive and argumentative). Jointly with *AgentGeom*, TURING system uses an implementation approach starting from didactic studies with real students for each problem. The implementation can be improved after testing and evaluating it into a socio-cultural reality. The strategy is not related to a pre-existent model of solving process, but it could be adapted so that the student can realize the distinctiveness of each problem; in addition the TURING is a real time system. Our current research can benefit from the messages, the strategy and the algorithms we propose in our context.

The Didactic Background of TURING System

TURING is based on the integration of the social debate simulated in a computer-based environment of human training for the problem solving and mathematical reasoning (Aïmeur, Cobo & al., 2003). The disciplinary contents (concepts, processes and attitudes) come from the compulsory curriculum of mathematics. The theoretical framework combine the theory of didactic situations of Guy Brousseau (1998), the proofs and refutations dialectical of Imre Lakatos (1984), the theory of the functions of the language of Raymond Duval (1995) and the application of an intelligent tutoring system in the development of mathematical competence and acquisition of knowledge by the learner. The system aims to extend and verify what is the field of validity from the preceding framework in a computer-based environment.

The currently existing systems were elaborated starting from a modelling of the human behaviour according to the available computer-based environment (e.g. Guin, 1996; Bunt & Conati, 2002; Webber, Bergia & al., 2002). This natural attitude, is even advised, if one wants to lead rather quickly to concrete achievements, it is still necessary that these achievements are effective assistances for learning. But if one wants moreover to ensure the effectiveness of the environment, it is preferable to model the human behaviour and to design a computer-based environment that considers this modelling. In other words, the methodological problem of the TURING, what constitutes its principal originality, consists in testing the validity of the theoretical framework (above) in a process that authorizes the converse adaptation of the system, especially heuristic and argumentative components.

The Approach of TURING

Contrary to the approaches which introduce an agent tutor to simulate the role of the teacher – with, possibly, an agent companion or a disturbing agent to simulate the attitude of peers –, we formulate the assumption that a virtual pedagogical agent is an autonomous being of nature compared to what should be a human model. In other words, the same virtual pedagogical agent can assume several traditional roles (tutor, companion or disturber, within the meaning of Aïmeur, 1998) while being defined in a distinct way compared to each one of these roles. This assumption is strong, but it makes it possible to ensure jointly the epistemological validity of the collaborative process solving while respecting the solving strategies to which a student would have recourse in the particular context of a given problem. We must underline here that we do not claim to replace regular teaching, but we want to provide a personalized help for the pupils who need some extra help (Fortuny, Giménez & al., 2002). Thus, TURING system needs the flexibility to build virtual pedagogical agents that give the students a cognitive and meta-cognitive support using strategic messages in an argumentative process that jointly considers the meaningful actions of the student, i.e. in our case the meaning of the adopted propositions (discursive and graphic, within the meaning of Richard, 2004a) associated with the discovery of a conjecture and the realization of a proof.

The TURING Architecture

The architecture of TURING system consists of four main agents (Figure 1):

- *Student Agent*. Converts graphic action to discursive action and vice versa. Validates or autocorrect discursive text action. Allow student to solve and write a problem. Return message to the student in some situations.
- *Teacher Agent*. Helps the teacher to modify the system using the messages content. Shows the strategy for a selected problem. Makes easier for the teacher to learn the student's behaviour. Modifies the message any time, even during the problem solving process. Allows teacher to supervise any selected student and shows him witch strategic path the student has used to solve the problem.
- *Didactic Expert Agent*. Allows didactic experts to create their own strategy, modifies and update the strategy according to changes after didactic study and experiment review. This agent assists the expert during the creation of a strategy. It helps to create the strategy automatically using a graph model. The strategy modification can use the data related to a student during the solving process.
- *Tutor Agent*. The Tutor agent is the engine of our TURING system. It is a client server system that has a client side agent and a server side agent, to allow our TURING system to run when it is offline – tutor agent client-side will have the same functionality as the tutor agent of the server-side, but it will be limited to small problems or repeated problem. One of the most important roles of this agent is to allow the communication between both sides.

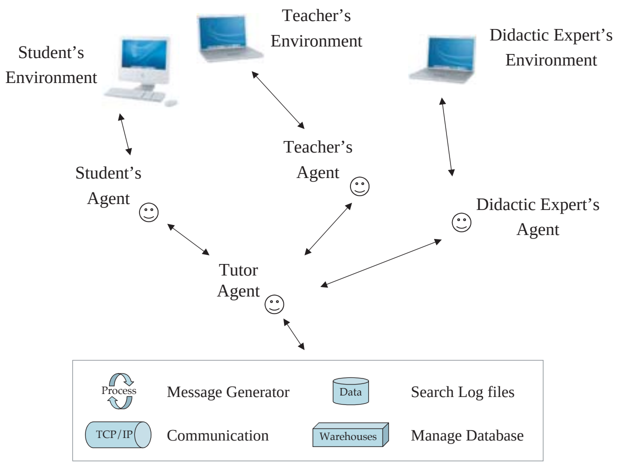
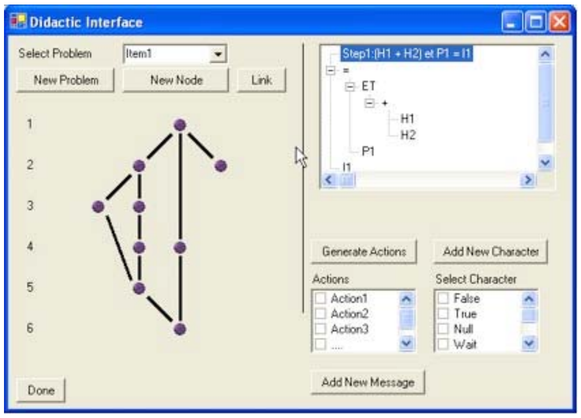


Figure 1. The TURING Architecture

The Tutor Agent takes the action and returns the best matching message to the student. It can offer help to the didactic expert to create a strategy or to show all the available possibilities. It generates messages using an intelligent approach, e.g., collect data (Request, Actions or Inferences) and generate invariants according to some characteristics managed by the didactic expert or the teacher (such as latency of student actions, level of tolerance, etc.).

Intelligence of the System

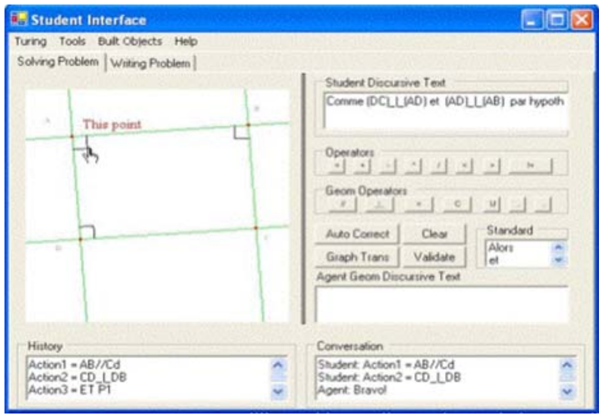
The TURING is based on both studies of human behaviour and system's processes. It simulates the collaborative behaviour in the class with a student (e.g. Kieran, 2001), using strategic messages for a given problem (e.g. Moschkovich, 2004). The system will return a message to the significant action done by the student during the problem solving process. The semantic content of a returned message, when it is necessary, will be similar to the messages of a companion, a teacher or an expert. The flexibility of the warehouses structure and the approach that we used to build this structure assure the intelligence of our system (see subsection *Warehouses*). Each agent that we have seen before has significant intelligence in TURING system's processes, which validates the students' and the teachers' actions using formal models to correct the writing, with some application of fuzzy logic. To give support and help, the system generates messages automatically using some cognitive and heuristic invariants comimg from didactic research, which could be used to simulate human behaviour with cognitive geometry (in the sense of Allen & Trilling, 1997). Analogically, we can say that the TURING approach is close to the Garry Kasparov's attitude when he uses a cognitive strategy against Deep Blue, IBM's chess-playing supercomputer (see IBM Research). Deep Blue strategy is similar to the deductive calculations used in *Baghera*, but still intuitive decision based on smart strategy can save million of transactions – strategy use the short path solution without wasting time in the cases without interest and by finding short cuts.



Step1:(H1 + H2) et P1 = I1
=
ET
+
H1
H2
P1
I1

Action1
Action2
Action3
....

False
True
Null
Wait



Comme (DC)⊥(AD) et (AD)⊥(AB) par hypoth

Alors
et

Action1 = AB//Cd
Action2 = CD_|_DB
Action3 = ET P1

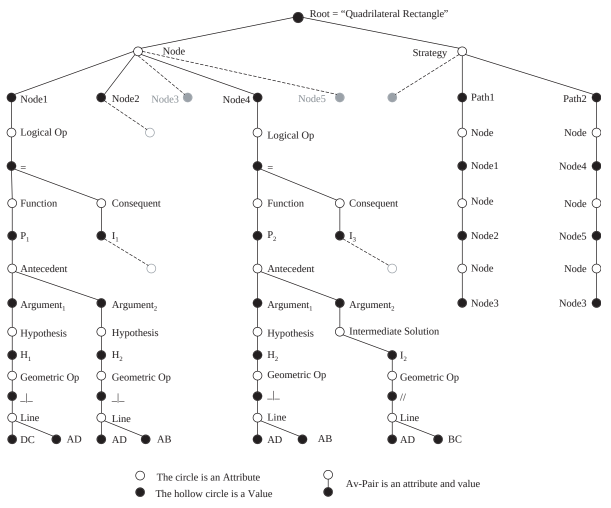
Student: Action1 = AB//Cd
Student: Action2 = CD_|_DB
Agent: Bravo!

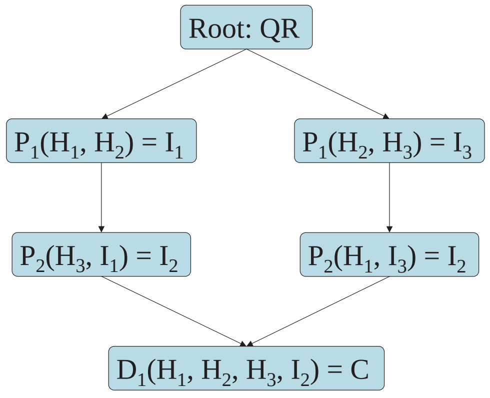
Figure 2 and 3. The TURING didactic expert's interface and student's interface to solve and write a problem

The TURING User Interfaces

TURING is an intelligent tutoring system that is made up of three main constituents: student, didactic and teacher. The *student interface* is used to solve and write a given problem. Figure 3 shows a dynamic Cabri figure. This figure is pre-built in Cabri Geometry II plus or Cabri 3-D. In this interface, the student can validate an action like the discursive writing of «AB = 5 cm», the graphic selection of (AB) // (CD) or a request like a set of actions to make an inference. These actions are the meaningful actions in the context of the problem, i.e. a part of graph. The validate option is a part of the student agent; it takes the correct request and validate it with the available data concerning the problem and the student. The student interface is based on the student agent that also allows the translation between the graphic part and the discursive text. It has the ability to correct the student's text and give help to choose the best expression or request. Actually, the system seeks to check syntax (well formed statement) so that it is usable by the system (standardization in the internal). The student interface offers many other options that help the student to track his activities or to ask a question related to the solving process. In comparison, Figure 2 shows the *didactic expert interface*, which is based on the didactic expert agent; this interface allows the expert to update or create a new problem and its strategy in the form of a graph. The graph is the structure to represent an adapted version of the *basic space of the problem* in the sense of Cobo & Fortuny (2000). The tutor agents use the strategy entered by the didactic expert as the base to compare or process the student's action and return to him suitable messages or others if necessary.

The *teacher interface* is based on the teacher agent so that the real teacher can adjust the system and modify the strategic messages according to the practices of its students. This interface gives the teacher the ability to change some invariants that change and adjust the TURING message strategy; it also allows the teacher to send messages to the student according to what it sees when the process takes place during the class time. The student's path is in red to indicate the state of the solving process of a selected student for a specific problem. This interface allows the teacher watching many students at the same time, being able to consult the history of their activities, a log file that records each move and the conversation activities between the student and the agent.





Figures 4 and 5. Warehouse structure and Graph structure

Warehouses

The warehouses are the central data structure in the TURING. They achieve flexibility and are based on a hierarchy of attributes and values. An attribute is a category in which an object can be classified, for example its «Line». A value is the object classification within that category like «AB». Together, an attribute and its associated value form an attribute-value pair or av-pair. The hierarchy of av-pairs allows the TURING provide a help to the student and teacher to precisely describe the strategy for each problem in different fields it might be used.

While several complex request languages exist in the literature, our approach, based on attributes and values, is particularly simple. We also design the warehouse structure to be independent of the specific language used to perform requests (for example a student can solve the problem in any language he wants, the request will be indexed, and the system will process the request without any dependent with the language of the student), so that it can also be used in the context of other description languages.

The structure of a warehouse is a hierarchical arrangement of av-pairs, such that an av-pair is a descendent of another av-pair in the hierarchy when it is dependent on it (see Figure 4). There can be multiple values per attribute and each value can in turn be refined by multiple attributes. In comparison, the Figure 5 presents the strategy created by the didactic expert represented in a graph structure (mathematical proof in Euclidean geometry). The Graph is a set of nodes or inferences, where each node is a set of actions, linked by logical operators. Thus, in «Function(Argument$_1$, Argument$_2$) = Intermediate Result», Function can be a mathematical Property or Definition (P or D), while argument is an Hypothesis or Intermediate Result (H or I). The graph can be generated automatically using a deductive approach similar to *Baghera*, but the didactic expert with the TURING can create, modify and update it, especially, he can program semantic and discursive inferences for argumentative purposes.

The nodes in the graph have pointers to the next node (except the last one), which represents the passage from a cognitive state to another in the strategy of the didactic expert. Using functional notation, each node consists of three parts: antecedent «Argument$_1$, Argument$_2$», consequent «Intermediate Result», and justification (Function itself) and might be divided into multiple nodes with the same structure depending on the complexity of the cognitive state. Each path of the graph (composition of functions) represent a possible strategy in the problem solving.

Processing a Student's Request

Students validate requests to identify their proposition during the process of solving a problem. Requests are built using arrangements of attributes and values, related by relational operators, which we call *av-relations*. A request consists of a simple action or a complex action (set of well formed actions). Requests can be one or more inferences, one or more actions or any text, but separate agents will treat them.

The request-processing algorithm selects the answers that best fit the student's request. This algorithm is at the heart of the tutor agent during the solving problem. It returns the message corresponding to the action's characteristics. The algorithm first starts by searching the warehouses corresponding to the student agent. If the characteristics of the action are found, the process returns the related message; this is the best-case situation. Otherwise, the algorithm sends the action to the tutor agent that runs on the client side until it finds the related answers; this is the intermediate case. If the request was not fulfilled, i.e., the related message is not found than, it sends the actions to the tutor agent on the server side that has the suitable message related to the requested action; this latter situation is the worst case situation. The search starts in the student agent that will communication with the tutor agent if needed. The student agent treats the actions that are out of context or to formalize the request to an understandable formula by the strategy. Only the tutor algorithm searches the didactic graph to choose the best path, which directs the student using a message strategy to follow the most efficient way that helps the student in his reasoning to solve the problem. To anticipate a future work, this algorithm will have the ability to decide what step will go or from where to start. It will be general to all problems and any strategy. It will be tested and the performance must be evaluated to be maximized. The algorithm treats the invariants (cognitive and meta-cognitive), thus decreasing the number of messages associated with the matrices (see below).

Messages During the Solving Process

One of the most important issues in TURING system concerns messages. For each action that the student can produce, a message must be available to improve mathematical competence. The messages are attached to each possible action and all its attributes, like «change node», «action state», «tolerance level», «action wait», «action true», «wait time» and many other parameters related to this action and the problem strategy. To prevent the exponential number of possible messages in a complex problem (e.g. 50 nodes with 20 actions and 10 attributes would require $5{,}12 \times 10^{14}$ messages), we generate a complete set of primitive messages related to the inferential structure of human reasoning (Duval, 1995, Richard, 2004b). In Table 6, we present a matrix that treats the messages associated to a specific case. After the didactic phase of validation, the number of messages can also be simplified by the recognition of functional invariants in the context of a given problem (the same effect of a counterexample, within the meaning of Lakatos, 1984) or a specific didactical contract (in the sense of Brousseau, 1998), which groups some subset of meaningful actions that satisfies the same function. For some cases, the teacher can decide to minimize the messages number by restricting the returned messages for some actions (e.g. action out off point in the logic of the problem can not be sent to the agent tutor).

We see that in the TURING, the messages specific to the solving process differ from the messages of the writing process (see Figure 3). Due to the constraints of space, we limit the exposition to the messages that are likely to appear in the solving process through the case taken on the graph structure of Figure 5. Thus, the set Γ of meaningful actions X in the didactic expert graph is $\{H_1, H_2, H_3, I_1, I_2, I_3, C, P_1\ P_2, D_1\}$; the wrong actions are treated as being out off point – if they were to obey some didactic logic that it is, it will belong already to the graph. In Table 6:

- {{antecedent}} is an «antecedent» subset of Γ;
- {{antecedent}} = 1 represent «an antecedent subset of Γ that is satisfied»;
- Levels represent the latency (e.g. 1, 2 or 5 min) of a level marker. Here, level 0 is a suggestion compared to the components of the didactic relation (milieu, didactic contract, considerations on the role of the artificial agent), level 1 is a mathematic suggestion of descriptive type and level 2 is a suggestion compared to the mathematical contents (concepts and process).
- The functions «state(X)», «nature(X)» and «status(X)» returns respectively to particular values like «written» and «deduced»; «hypothesis», «conclusion», «propriety», «definition» and «I-result»; «antecedent», «justification» and «consequent ».

Case	***Level***	***Message***
{{antecedents}} = 1	0	{I can't see what are you going to accomplish}
	1	{You already chose $\cup$[nature(X)], but which statement would you like to deduce from those?} or {You already chose $\cup$[nature(X)], but which tool would you like to use?} or {You already chose $\cup$[nature(X)], but do you have all what you need?}
	2	{You have $\cup$([state(I_k)] [I_k]) and you already know that $\cup$[X], therefore, which statement would you like to deduce from those?} or {You have $\cup$([state(I_k)] [I_k]) and you already know that $\cup$[X], but which tool would you like to use?} or {You have $\cup$([state(I_k)] [I_k]) and you already know that $\cup$[X], but do you have all what you need?}
{{antecedents}} + {justification} = 1	0	{I don't know how you can do it}
	1	{You already chose $\cup$[nature(X)], but which statement would you like to deduce from those?} or {You already chose $\cup$[nature(X)], but do you have all what you need?}
	2	{You have $\cup$([state(I_k)] [I_k]) and you already know that $\cup$[X], then, according to [P_1], which statement would you like to deduce from those?} or {You have $\cup$([state(I_k)] [I_k]) and you already know that $\cup$[X], therefore, to use [P_1], do u have all what you need?}
{{antecedents}} + {consequent} = 1	0	{I can't see why}
	1	{You already chose $\cup$[nature(X)], but which tool would you like to use?} or {You already chose $\cup$[nature(X)], but do u have all what you need?}
	2	{You have $\cup$([state(I_k)] [I_k]) and you already know that $\cup$[$X_{hypothesis}$], but according to which tool you would like to deduce [$X_{consequent}$]?} or {You have $\cup$([state(I_k)] [I_k]) and you already know that $\cup$[X], but to deduce [$X_{consequent}$] do u have all what you need?}

Table 6. Some invariant messages during the solving process

Conclusion and Future Work

In summary we can highlight, on one hand, the aspects related to the technological characteristics of the TURING, on the other hand, its pedagogical application when helping students in the problem solving process. We believe that our proposition is a suitable approach to enhance and improve Mathematical Competence In Problem Solving trough Interactions with a multi-agent system. We expect to successfully demonstrate the power of our approach.

In the future, there are many aspects that we would like to investigate. In the following we list some of them:

- Use a dictionary to make compatible similar actions (discursive text) or to generalize actions
- Generate and improve the didactic expert strategy automatically. (Use *Baghera*'s approach)
- Generalize our system to be able to work in any e-Learning domain such as medical, physics, etc…
- Define a process to limit the size of stored information (warehouses) by removing unused data according to some priority rule. This involves studying the notion a gradually forgetting about rarely used information.
- Analyzing the influence and effects of using TURING system on the development of the students' strategic competences when solving problems and characterizing the competence levels generated.
- Validate our system using real student environment in a class at high school.

Our implementation is in Java to take advantage of its cross-platform portability. The warehouse structure is developed in XML; access to the warehouse structure is performed in Java using the XML package developed by IBM (Ceponkus & Hoodbhoy, 1999; W3C, 2004).

References

Aïmeur, E. (1998). Application and Assessment of Cognitive Dissonance Theory in The Learning Process. *Journal of Universal Computer Science, 4*(3), 216-247.

Aïmeur, E., Cobo, P., Fortuny, J.M. & Richard, P.R. (2003). Stratégie argumentative et système tutoriel pour l'apprentissage interactif de la géométrie. In *Actes de l'EMF-2003 (Espace mathématique francophone)*, Tozeur.

Allen, R. J., & Trilling, L. (1997). Dynamic Geometry and Declarative Geometric Programming. In J. R. King & D. Schattschneider (Ed.), *Geometry Turned On. Dynamic software in learning, teaching, and research* (pp. 193-197). Washington: MAA Notes Series.

Brousseau, G. (1998). *Théorie des situations didactiques*. Grenoble: La Pensée Sauvage.

Bunt, A. & Conati, C. (2002). Assessing Effective Exploration in Open Learning Environments Using Bayesian Networks. In *Proceedings of the International Conference on Intelligent Tutoring Systems 2002*, Biarritz. Retreived at http://www.cs.ubc.ca/%7Econati/my-papers/its2002BuntConati.pdf.

Cabri-géomètre (2005). *Proceedings of the CabriWorld 2004*, Rome.

Ceponkus, A. & Hoodbhoy, F. (1999). *Applied XML: A toolkit for programmers*. Wiley.

Cobo, P. & Fortuny, J.M. (2000). Social interactions and cognitive effects in contexts of area-comparison problem solving. *Educational Studies in Mathematics, 42*, 115-140.

Cobo, P., Fortuny, J.M., Puertas, E. & Richard, P.R. (2005). AgentGeom: a Multiagent System for Pedagogical Support in a Geometric Proof Problem. *International Journal of Computers for Mathematical Learning*. Manuscript in evaluation.

Dufresne, A., & Paquette, G. (2000). *ExploraGraph: A Flexible and Adaptive Interface to Support Distance Learning*. World Conference on Educational Multimedia, Hypermedia and Telecommunications 2000(1), 304-309.

Duval, R. (1995). *Sémiosis et pensée humaine*. Berne: Peter Lang.

Fortuny, J.M., Giménez, J. & Richard, P.R. (2002). Distance Education at Secondary Levels: Contexts and Norms for the Learning of Mathematics. *Proceedings of the E-Learn 2002 World Conference of Association for the Advancement of Computing in Education*.

Guin, D. (1996). A Cognitive Analysis of Geometry Proof Focused on Intelligent Tutoring Systems. In J.M. Laborde (Ed.), Intelligent Learning Environments: the Case of Geometry (pp. 82-93). Berlin: Springer Verlag.

IBM Research (n.d.). *Kasparov vs Deep Blue, the Rematch*. Retrieved March 18, 2005, from http:// www.research.ibm.com/deepblue/.

Kieran, C. (2001). The Mathematical Discourse of 13-year-old Partnered Problem Solving and Its Relation to the Mathematics that Emerges. *Educational Studies in Mathematics, 42*, 115-140.

Laboratoire Leibniz (2003). Baghera Assessment Project: Designing an hybrid and emergent educational society. In Soury-Lavergne S. (Ed.), *Rapport pour la commission européenne, Programme IST, Les Cahiers du Laboratoire Leibniz nº 81*, Grenoble.

Lakatos, I. (1984). *Preuves et réfutations. Essai sur la logique de la découverte mathématique*. Paris: Hermann.

Moschkovich, J. N. (2004). Appropriating mathematical practices: a case study of learning to use and explore functions through interaction with a tutor, *Educational Studies in Mathematics, 55*, 49-80.

Richard, P.R. (2004a). L'inférence figurale: un pas de raisonnement discursivo-graphique. *Educational Studies in Mathematics, 57*, 229–263.

Richard, P.R. (2004b). *Raisonnement et stratégies de preuve dans l'enseignement des mathématiques*. Berne: Peter Lang.

W3C (2004, August 27) *Extensible Markup Language (XML)*. Retrieved March 18, 2005, from http://www.w3.org/ XML/.

Webber, C., Bergia, L., Pesty, S. & Balacheff, N. (2002). The Baghera project: a multi-agent architecture for human learning. In *Proceedings of the Workshop Multi-Agent Architectures for Distributed Learning Environments (AIED2001)*, 12-17, San Antonio.

Acknowledgement

Dr. Philippe R. Richard could contribute to the development of this research thanks to a subvention granted by the *Fonds québécois de la recherche sur la société et la culture* (FQRSC 2005-AI-97435, Gouvernement du Québec) and the logistic support of the *Laboratoire Turing* (http://turing.scedu.umontreal.ca).