Learning Automata Based Intelligent Tutorial-*like* System

B. John Oommen[1] and M. Khaled Hashem[2]

[1] *Chancellor's Professor*, *Fellow: IEEE* and *Fellow: IAPR*, School of Computer Science, Carleton University, Ottawa, Canada, *Adjunct Professor* with the University of Agder in Grimstad, Norway
oommen@scs.carleton.ca
[2] School of Computer Science, Carleton University, Ottawa, Canada
k_hashem@yahoo.com

Abstract. The aim of this pioneering research[1] is to study, design, and implement systems that could tutor other sub-systems using techniques that traditional *real-life* Teachers use when they teach *real-life* Students. The research undertaken is a result of merging the fields of Intelligent Tutoring Systems (ITS) and Learning Automata (LA), and leads to a paradigm which we refer to to as "Intelligent Tutorial-*like*" systems. In our proposed novel approach, *every* component incorporates the fundamental principles of LA. Thus, we model the Student (i.e., the learning mechanism) using an LA. Each Student is considered to be a member of a Classroom of Students, each of whom is individually represented by a distinct (and possibly different) LA. We also model the Domain and the Teacher using the LA paradigm.

Our research also works within a new philosophical perspective. We relax the constraint that "traditional" Tutorial systems have, namely the one of assuming that the Teacher is infallible. Rather, we assume that the Teacher is inherently uncertain of the domain knowledge, and is thus of a stochastic nature. However, although he is not absolutely certain about the material being taught, he is also capable of improving his *own* "teaching skills" even while the operation of the system proceeds. Finally, we also attempt to model a realistic learning framework, where the Students can learn not only from the Teacher, but also from other colleague Students in the Classroom.

Keywords: Tutorial-like Systems, Learning Automata, Modeling of Adaptive Systems.

1 Introduction

Central to every learning or adaptive system is an entity which performs the learning, and another entity which teaches the latter. Based on real-life analogies, these

[1] Since this a plenary talk, in the interest of readability, this document is written informally – with *minimal* mathematical formalism. More details of the mathematics, claims, potentials of the results, and the entire research endeavor can be found in the papers cited in the bibliography, and in the Doctoral Thesis of the second author [5].

J.D. Velásquez et al. (Eds.): KES 2009, Part I, LNAI 5711, pp. 360–373, 2009.

can be informally referred to as the "Student" and the "Teacher" respectively. The aim of this research endeavor is to design a Tutorial-*like* system in which *every* component utilizes the fundamental principles of Learning Automata (LA). Indeed, we intend to model the Student (i.e., the learning mechanism) using an LA. We also propose to model a *Classroom* of Students - all of whom are appropriately represented, possibly by *different types* of LA. This, of course, broadens the horizons of both Tutorial systems and the fields of LA because we permit Students to not only learn from the Teacher, but to also learn by interacting with each other. Consequently, this research opens avenues for even more fascinating problems such as:

- How does the imperfect Teacher in such a Tutorial-*like* system present his[2] knowledge?
- How is the Domain knowledge represented?
- More importantly, since the Teacher himself is a component of the system, can we also incorporate learning capabilities in the Teacher, thus permitting him to improve his teaching capabilities as the the Student-Teacher interaction progresses?

This paper argues that LA-based Intelligent Tutorial-*like* systems can be an easy and useful way to model and implement Tutorial systems. We assume that the "Teacher" has an imprecise knowledge of the material to be imparted. While such a model for the Teacher has been studied extensively within the domains of LA, Neural Networks, and reinforcement learning [11,16,22], it is quite new to the field of Tutorial Systems. Thus, we believe that *after* our problem has been satisfactorily solved within a machine learning perspective, it can be, hopefully, ported to the application domain of Intelligent Tutorial systems.

The intention of the research in this paper is *not* to develop a generic system in which the Teacher is uncertain about the teaching material, or a generic model for the strategy by which he would impart the material and test the Students. Such a system would encounter enormous hurdles related to the psychological concepts of cognition, teaching, learning and intelligence, and also the system development aspects. In the research work presented in this paper, we propose that the problem we are studying be couched within the framework of the general machine learning paradigm, and thus we refer to the proposed system as a Tutorial-*like* system. Thus, it is reasonable for us to interchangeably refer to the "Student" as a "Student Simulator", and to the "Teacher" as the "Teacher Simulator", etc.

Using machine learning in improving tutoring systems was the study of a few previous researches. Frasson *et al.* [4] designed the main ITS components (student model, domain knowledge, and the tutoring model) in the form of intelligent agents. Lelouche [14] used a collection of interacting agents to represent the original modeling of the tutoring knowledge in an intelligent educational system. Legaspi and Sison [13] modeled the tutor in ITS using reinforcement

[2] For the ease of communication, we request the permission to refer to the entities involved (i.e. the Teacher, Student, etc.) in the masculine.

learning with the temporal difference method as the central learning procedure. Beck [3] used reinforcement learning to learn to associate superior teaching actions with certain states of the student's knowledge. Baffes and Mooney implemented ASSERT [2], which used reinforcement learning in student modeling to capture novel student errors using only correct domain knowledge. Our method is distinct from all the works cited here.

1.1 Tutorial-*like* Systems

Our entire research will be within the context of Tutorial-*like* systems [5]. In these systems, there need not be *real-life* Students, but rather each Student could be replaced by a Student Simulator that mimics a *real-life* Student. Alternatively, it could also be a software entity that attempts to learn. The Teacher, in these systems, attempts to present the teaching material to a *School* of Student Simulators. The Students are also permitted to share information between each other to gain knowledge. Therefore, such a teaching environment allows the Students to gain knowledge not only from the Teacher but also from other fellow Students.

In the Tutorial-*like* systems which we study, the Teacher has a *stochastic* nature, where he has an imprecise knowledge of the material to be imparted. The Teacher also doesn't have a prior knowledge about how to teach the subject material. He "learns" that himself while using the system, and thus, hopefully, improves his skills as a teacher. Observe that, conceptually, the Teacher, in some sense, is also a "student".

On the other hand, the Student Simulators need to learn from the Stochastic Teacher, as well as from each other. Each Student needs to decide when to request assistance from a fellow Student and how to "judge" the quality of information he receives from them. Thus, we require each Student to possess a mechanism whereby it can detect a scenario of procuring inaccurate information from other Students.

In our model of teaching/learning, the teaching material of the Tutorial-*like* system follows a Socratic model, where the domain knowledge is represented in the form of questions, either to be of a *Multiple Choice* sort or, in the most extreme case, of a *Boolean* sort. These questions, in our present paradigm, carry some degree of uncertainty, where each question has a probability that indicates the accuracy for the answer of that question.

1.2 Stochastic Learning Automaton

The stochastic automaton tries to reach a solution to a problem without any information about the optimal action. By interacting with an Environment, a stochastic automaton can be used to learn the optimal action offered by that Environment [1,12,15,16,17,20,24]. A random action is selected based on a probability vector, and then from the observation of the Environment's response, the action probabilities are updated, and the procedure is repeated. A stochastic automaton that behaves in this way to improve its performance is called a Learning Automaton (LA).

In the definition of a Variable Structure Stochastic Automata (VSSA), the LA is completely defined by a set of actions (one of which is the output of the automaton), a set of inputs (which is usually the response of the Environment) and a learning algorithm, T. The learning algorithm [16] operates on a vector (called *the Action Probability vector*)

$\mathrm{P(t)} = [\mathrm{p_1(t)}, \ldots, \mathrm{p}_r\mathrm{(t)}]^{\mathrm{T}}$,

where $\mathrm{p}_i\mathrm{(t)}$ ($\mathrm{i} = 1, \ldots, \mathrm{r}$) is the probability that the automaton will select the action α_i at time 't',

$\mathrm{p}_i\mathrm{(t)} = \Pr[\alpha(\mathrm{t}) = \alpha_i]$, $\mathrm{i} = 1, \ldots, \mathrm{r}$, and it satisfies

$\sum_{i=1}^{r} \mathrm{p}_i\mathrm{(t)} = 1 \; \forall$ t.

Note that the algorithm $T : [0,1]^r \times \mathrm{A} \times \mathrm{B} \rightarrow [0,1]^r$ is an updating scheme where $\mathrm{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$, $2 \leq \mathrm{r} < \infty$, is the set of output actions of the automaton, and B is the set of responses from the Environment. Thus, the updating is such that

$\mathrm{P(t+1)} = T(\mathrm{P(t)}, \alpha(\mathrm{t}), \beta(\mathrm{t}))$,

where P(t) is the action probability vector, α(t) is the action chosen at time t, and β(t) is the response it has obtained.

If the mapping T is chosen in such a manner that the Markov process has absorbing states, the algorithm is referred to as an absorbing algorithm [16] which are suitable for stationary environments. Ergodic VSSA better suited for non-stationary environments have also been investigated [16,19]. Furthermore, in order to increase their speed of convergence, the concept of discretizing the probability space was introduced [18,19]. This concept is implemented by restricting the probability of choosing an action to a finite number of values in the interval [0,1].

Pursuit and Estimator-based LA were introduced to be faster schemes, since they pursue what can be reckoned to be the *current* optimal action or the set of current optimal schemes [19]. The updating algorithm improves its convergence results by using the history to maintain an estimate of the probability of each action being rewarded, in what is called the *reward-estimate* vector. Families of continues and discretized Pursuit and Estimator-based LA have been shown to be faster than VSSA [23].

LA have been used in systems that have incomplete knowledge about the environment in which they operate. They have been used in telecommunications and telephone routing, image data compression, pattern recognition, graph partitioning, object partitioning, and vehicle path control [1].

1.3 Contributions of This Paper

In this research we propose a new philosophy to design and implement a Tutorial-*like* system in which *every* component utilizes the fundamental principles of LA. We present a novel design of a Tutorial-*like* system in which:

- The Teacher is stochastic, and uncertain about the teaching material.
- The Student is simulated to study the Domain knowledge.
- The Domain knowledge contains uncertain course material.

– The Teacher is dealing with a *School* of Students who learn from him and from each other.
– Since the Teacher himself is a specific component of the system, we also provide him with a mechanism to improve his own teaching abilities as system evolves.

In short, the goal of this research is to investigate how the field of Intelligent Tutorial systems and LA can be merged to produce Tutorial-*like* systems that have capabilities that are unreported in the literature.

2 Intelligent Tutorial/Tutorial-*like* Systems

An ITS consists of a number of modules, which are, typically, the domain model (knowledge domain), the student model, and the pedagogical model (which represent the tutor model itself). Self [21] defined these components as the tripartite architecture for an ITS – the *what* (domain model), the *who* (student model), and the *how* (tutoring model).

As mentioned in the introduction, Tutorial-*like* systems are quite similar (in principle) to traditional Tutorial systems, since they model the Teacher, the Student, and the Domain knowledge. However, there are many features which are distinct in the former, which is what we will briefly highlight below. More details of these distinctions can be found in [5].

1. **Different Type of Teacher.** As opposed to Tutorial systems, the Teacher in our Tutorial-*like* system possesses different features. First of all, one fundamental difference is that the Teacher is uncertain of the teaching material – he is stochastic. Also, the Teacher does not *initially* possess any knowledge about "How to teach" the domain subject. Rather, the Teacher himself is involved in a "learning" process and he "learns" what teaching material has to be presented to the Student.
2. **No Real Students.** A Tutorial system is intended for the use of *real-life* students. Its measure of accomplishment is based on comparing their progress with other students in a control group. Thus, the Tutorial-*like* system could be used by either:
   (a) Students Simulators, that mimic the behavior and actions of *real-life* students using the system.
   (b) An artificial Entity which, in itself, could be another software component that needs to "learn" specific domain knowledge.
3. **Uncertain Course Material.** Unlike the domain knowledge of "traditional" Tutorial systems where the knowledge is, typically, well defined, the domain knowledge for our Tutorial-*like* system contains material that has some degree of uncertainty.
4. *School* **of Students.** Traditional Tutorial Systems deal with a Teacher who teaches Students. A Tutorial-*like* system assumes that the Teacher is dealing with a *School* of Students where each learns from the Teacher on his own, and can also learn from his "colleagues" if he desires.

3 Overall Proposed Model

The overall proposed Tutorial-*like* system will be composed of an ensemble of LA modules, where each entity can improve its behavior based on the response it receives from *its corresponding* Environment. Thus, the system also uses LA to model the Student, the learning achieved by the Teacher, the learning achieved by each Student by interacting with the Teacher, and the learning accomplished by the School of Students by interacting between themselves. This paper presents an overall global view of our paradigm, and so we shall briefly (and without extensive mathematical formalism) describe each module in the Tutorial-*like* system.

To approach the general aim of this paper, we shall present the model for the overall Tutorial-*like* system, "piece by piece", where each module uses stochastic LA. As discussed, our aim will be to use LA in every module of the system. However, the overall system is composed of different software modules which are capable of communicating with the model for the Teacher, and with each other. Figure 1 shows the different components of the system and their mutual interactions, described in greater detail, in the subsequent sub-sections.

3.1 The Model for the Student Simulator

In our Tutorial-like system, there is no *real-life* Student. Each Student is represented by a Student Simulator, which tries to mimic his behavior and actions. This is the actual entity that has to be taught the material by the Teacher. This modeling enables the Tutorial-*like* system to function and be tested without the need for *real-life* Students. The crucial features of the model involve the following facets:

- The Student Simulator is modeled using an LA paradigm. It uses LA as the learning mechanism to try to mimic the behavior and the learning of the Student.
- In addition to being able to learn from the Teacher, the Student Simulator is able to communicate with other Student Simulators, to enable it (them) to exchange information and learn from each other.

3.2 The Model for the Student

This model is a *representation* of the behavior and status of the Student (or the Student Simulator). It guides the Tutorial-*like* system to take tailored pedagogical decisions customized to each Student or his simulator. One can think of this as the model which the *system* retains in order to represent the Student, even though the Student may be learning using a completely different philosophy. Thus, for example, while the *real-life* Student may be learning using a neural network strategy, the system may model his learning paradigm using an L_{RI} LA scheme.

Attempting to understand how a learning mechanism within a "black box" learns is by no means trivial. Indeed, as far as we know, this is an unsolved problem. To achieve this, we shall use the philosophy briefly explained below.

- The model of the Student will be inferred using a higher-level of LA, referred to as the *Meta-LA*. While the Students use the Tutorial-*like* system, the

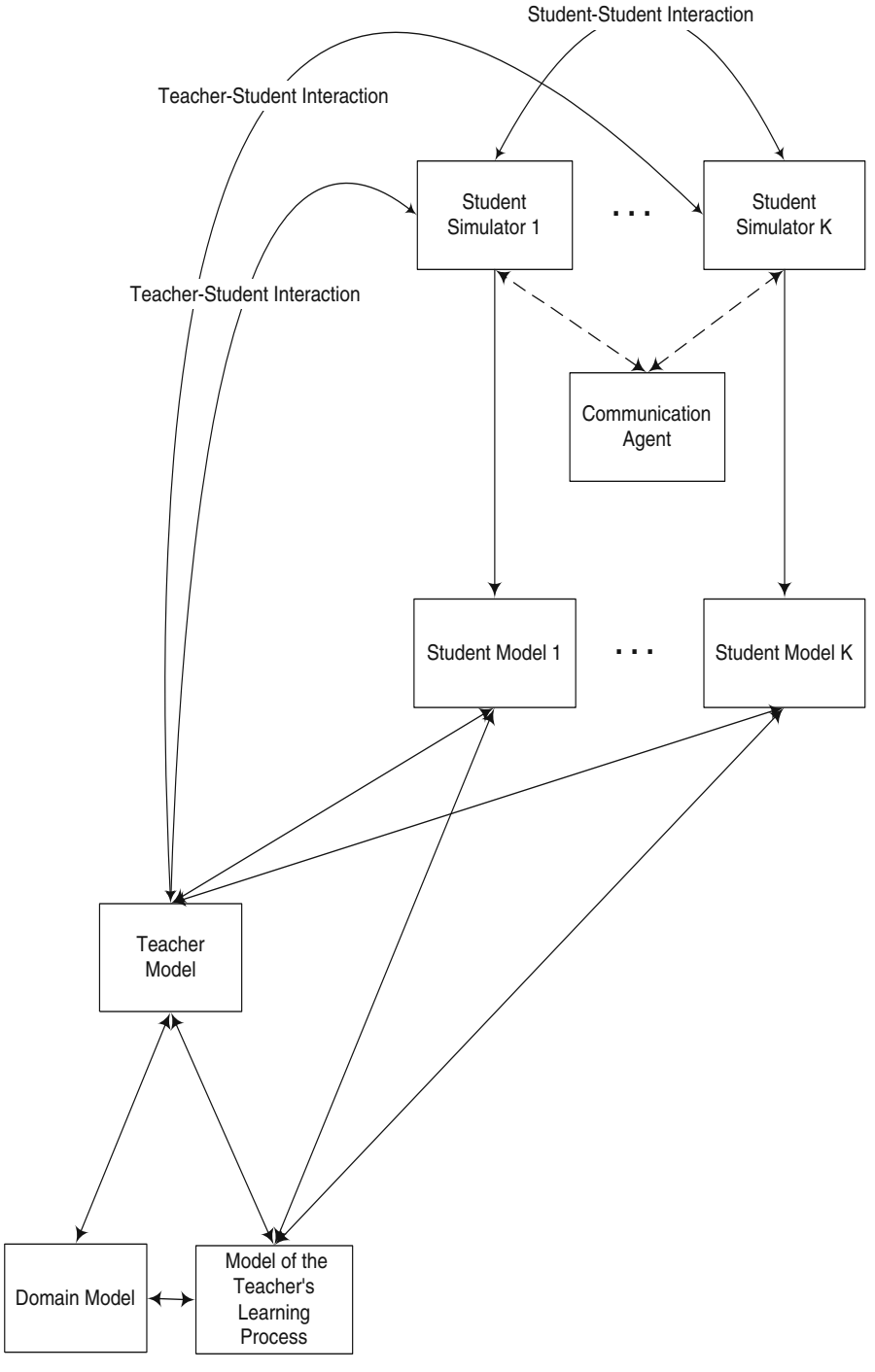

Fig. 1. Components of the LA-based Tutorial-*like* System

Meta-LA attempts to characterize their learning model. The input for the *Meta-LA* is obtained from its Environment, and does its task by observing the input/output characteristics of the Student Simulator.

- The *Meta-LA* will try to determine if the Student in question is one of the flowing types:
  - A Fast Learner.
  - Normal-paced Learner.
  - A Slow Learner.

  Clearly, this sub-division of capabilities can be subject to a more fine resolution, but we believe that such a resolution is sufficient for most tutorial applications.
- The *Meta-LA* will determine the learning model of the Student by:
  - Observing and watching the consequent actions of the Student Simulator and his performance.
  - Examining the Environment's characteristics, as well as observing its response to the Student's actions.
- The *Meta-LA* Environment monitors the "performance" of the Student LA over a period of time. Depending on the sequence of observations, its Environment will penalize or reward the action of the *Meta-LA*.
- The higher-level *Meta-LA* and the lower-level LA are connected as a Network of LA that have a unique interaction model. In this model, the *Meta-LA* is not directly affected by the lower-level (Student) LA, but rather the *Meta-LA*'s Environment monitors the progress of the Student LA. Such a interconnection modeling is novel to the field of LA.

Although the description given here, for the model of the Student, is brief, additional details can be found in [5,6].

3.3 The Model for Student-Classroom Interaction

The Tutorial-*like* system allows the Student to be a member of a *Classroom* of students, where, for the most part, each member of the Classroom is permitted to not only learn from the Teacher(s) but also to "extract" information from any of his colleague students.

The Student-Classroom interaction is intended to maximize the learning experience of each Student as well as the collective learning of the Students. We propose a feasible model for the real-life scenario in which each Student can be provided with multiple sources of knowledge, each of which source is uncertain or inaccurate.

When interacting with each other, a Student, or the Student Simulator, can select an interaction strategy to communicate with other Students. Such a strategy can be one of the following:

1. He always assumes that the knowledge of other Students is reliable.

2. He assesses the knowledge received from other Students. If he considers this knowledge credible, he agrees to utilize this knowledge.
3. He initially accepts to give due consideration to the knowledge from other Students. He then evaluates the knowledge received after a period of probation, and "unlearns" the information gleaned if he infers that this information was misleading.
4. He decides to learn independently, and does not communicate with other fellow Students.

On the other hand, the Student defines how he is willing to take advantage of the knowledge that his colleagues can offer. Our model proposes two approaches in which the Student can handle this knowledge:

- A complete transfer of information, in which case the Student will acknowledge all the knowledge from the other Student, and forget or erase his learned state completely.
- A partial utilization of information, where the Student will use the knowledge from the other Student only to enhance his knowledge but not to erase the knowledge he has gained otherwise.

In order to facilitate the Student-Classroom interaction, the Student, or the Student Simulator, uses what we refer to as a *Tactic-LA*, to decide about the initiation of interaction steps. The *Tactic-LA* will enable the Student to consider one of the following methods of interaction:

- He is willing to offer his assistance to other Students in the Classroom.
- He is looking for assistance from other Students.
- He is not interested in any of the above options.

The Communication Agent is the component of the Tutorial-*like* system that is assigned to facilitate the interaction between Students. It enables the communication between Students by matching those Students who are willing to offer assistance, with their counterparts who request assistance.

In the context of the Student-Classroom interaction model, the results given in [5,8] can be summarized as follows: A Student-Classroom interaction is beneficial to most Students, especially the weaker ones, when they utilize the information they get from superior colleagues. Such a conclusion is, of course, intuitive, and we believe that our model is the pioneering one in this context.

More extensive details and experimental results for this can be found in [5,8].

3.4 The Model for the Domain

This model will encapsulate the Domain knowledge teaching material that needs to be communicated to the Students. It is the control center that encompasses the entire Domain knowledge, which the Teacher uses to impart knowledge to the Students. This module permits the Tutorial-*like* system to model and implement the Domain knowledge using a Socratic philosophy and *via* multiple-choice questions. It will represent the increasing complexity/difficulty of the material being taught. In essence, the features of this model will be as follows:

- The Domain knowledge will be presented *via* multiple-choice Socratic-type questions. For each question, every choice has an associated probability of being correct. The choice with the highest reward probability is the answer to that question.
- The knowledge imparted is arranged in *Chapters*, each of which will have a set of questions.
- Each *Chapter* represents a level of complexity/difficulty that is harder than the previous one.
- Students will not be able to predict the answer for subsequent *Chapters* using prior knowledge.

The Domain model we propose is capable of increasing the complexity of the Domain knowledge by reducing the range of the penalty probabilities of all actions (i.e., the multiple-choice answers), by incorporating a scaling factor μ. This will result in a clustering of the penalty probabilities, which will then lead to making the questions more complex inasmuch as it will be more difficult for the Student to infer the correct action.

The reader is requested to refer to [5,7] for more details on the model for the Domain.

3.5 The Model for the Teacher

In this module, we model how the Teacher can teach the Socratic-type Domain knowledge, *via* multiple-choice questions. This knowledge is also used to *test* the Students in the imparted knowledge. The aim of this part of the study is to illustrate how the Stochastic Teacher can not only present Socratic-type Domain knowledge to the Students, but also model the way by which he can assist them so as to be able to learn increasingly complex material. Modeling the Teacher involves the following concepts:

- The Teacher retrieves Domain knowledge from the Domain model, so as to present it to the Student.
- The Domain knowledge, as mentioned in Section 3.4, is of the Socratic-type and is stored in the Domain model, via multiple-choice questions.
- The Domain knowledge is modeled with increasing complexity, which is intended to represent the increasing complexity of subsequent *Chapters* that the Student encounters.
- The Students will learn the Domain knowledge from the questions presented to them, and from the feedback obtained by answering these questions.
- The Teacher possesses a formal strategy to improve the ability of the Students to learn more complex material. He does this by assisting them with additional information so as to handle the Domain knowledge as the difficulty of the latter increases.

In order for the Teacher to assist the Students to handle complex Domain knowledge, he provides them with *hints*. These hints will be imparted to the Students in the form of increasing the initial probability for one of the actions in the

Student Simulator's action probability vector. The Teacher has the ability to control the probability that the Student correctly receives the *hint*.

The model for the Teacher is described in greater detail in [5,9].

3.6 The Model for Improving the Teacher's Skills

Our Tutorial-*like* system allows the Teacher to "learn" and improve his "teaching skills" while he is using the system. This is accomplished by providing a higher-level LA, referred to as the *Meta-Teacher*, which will infer the required customization that the *Teacher* needs for the particular Student. The salient features of this model are as follows:

- The Teacher will utilize the Student model. As explained in Section 3.2, this will be done by using the *Meta-LA* to infer the learning model of the Student and his learning capabilities.
- The *Meta-Teacher*, as a higher-level learning concept, will try to infer the required customization that the *Teacher* needs for the specific Student.
- For each Environment that the Student is attempting to learn from, the Teacher will define his *own* standards for the specific Environment.
- The *Meta-Teacher* will make its inferences based on observing the progress of the Students at an intermediate stage of the learning.
- Based on the knowledge inferred from the *Meta-Teacher* and the Student model, the Teacher will be able to customize the teaching material presented, and provide the appropriate *hints* to each individual Student.
- This customization demonstrates the adaptability of the Teacher to the particular needs and skills of each Student.

More detailed mathematical and experimental information concerning the model for improving the Teacher's skills are found in [5,10], and omitted here in the interest of brevity.

3.7 The Overall Prototype of the Tutorial-*like* System

In all brevity, the prototype attempts to incorporate all the features of the different models, described in the above sub-sections. The goal of the prototype is to provide a researcher with the software necessary to simulate the teaching/learning experience within the generalized framework proposed by the tenets of this research.

Typically, a researcher who uses the system, proceeds along the following steps to test the prototype:

- Specify the configuration parameters for the Students and the Classroom. These parameters serve to define the characteristics of the Student/Classroom who interact with the Teacher. This includes the identity of each Student, the type of Student that the Student Simulator is trying to mimic (either Fast, Normal, or Below-Normal), the interaction strategy used by the Student, the rate of learning for the Student, etc.

- Define the configuration of the Domain knowledge.
- Define the learning Environment, its characteristics and how its complexity increases with the different *Chapters*.
- Provide a mechanism by which the Teacher will provide *hints* to assist the Students.
- Define how the Teacher himself will improve his teaching skills and abilities.

At the end of the simulations, the researcher who uses the prototype will be able to graphically observe the progress of each Student. How each student has learned, and the effect of the interaction between the Students will be depicted in these graphs.

4 Conclusion

In this paper, we have pioneered a new class of systems, which merged the fields of Intelligent Tutorial Systems (ITS) and Learning Automata (LA). We refer to these systems as being "Intelligent Tutorial-*like*" systems. Our research has succeeded in developing such Tutorial-*like* systems where every module was composed using LA, and where *every* entity improved its behavior based on the response it received from *its* corresponding Environment. Although the details of the various modules is omitted in the interest of brevity, they are found in the Doctoral Thesis of the second author [5] and in a sequence of publications [6,8,7,9,10] that have concentrated on the individual modules themselves.

Our Tutorial-*like* system has incorporated different concepts that are novel and unique. In our system, every facet of the interaction involved a model (or a non *real-life* Student), and in which the design and implementation followed an established learning paradigm, where every module of the system utilized the fundamental principles of LA. First of all, the Student (i.e., the learning mechanism) was modeled using a LA. The *imperfect* Teacher was modeled using an LA Environment. The Classroom of Students was also modeled using LA, with each of them being represented by distinct, and possibly *different types* of LA.

Throughout the paper, we have argued that LA-based Intelligent Tutorial-*like* systems can serve to be an easy and useful way to model and implement system-based (as opposed to *real-life* Student-based) Tutorial systems. It is our belief that after our problem has been satisfactorily solved within a machine learning perspective, it can be, hopefully, ported to the application domain of Intelligent Tutorial systems.

The paper also presented a philosophic view of such Tutorial-*like* systems in which we relaxed the infallible constraint that the "Teacher is 'perfect' ". By modeling the Teacher as a stochastic "Entity", we were able to perceive it as an Environment or Medium in which the Student is learning.

We conclude this paper by stating that we do not claim to have solved all the problems associated with the new field of Tutorial-*like* systems. However, we do believe that we have taken a few small but significant and pioneering steps in the direction of laying the foundational groundwork for the future.

References

1. Agache, M., Oommen, B.J.: Generalized pursuit learning schemes: New families of continuous and discretized learning automata. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32(6), 738–749 (2002)
2. Baffes, P., Mooney, R.: Refinement-based student modeling and automated bug library construction. Journal of AI in Education 7(1), 75–116 (1996)
3. Beck, J.: Learning to teach with a reinforcement learning agent. In: Proceedings of The Fifteenth National Conference on AI/IAAI, Madison, WI, p. 1185 (1998)
4. Frasson, C., Mengelle, T., Aïmeur, E., Gouardères, G.: An actor-based architecture for intelligent tutoring systems. In: Lesgold, A.M., Frasson, C., Gauthier, G. (eds.) ITS 1996. LNCS, vol. 1086, pp. 57–65. Springer, Heidelberg (1996)
5. Hashem, M.K.: Learning Automata Based Intelligent Tutorial-like Systems. PhD thesis, School of Computer Science, Carleton University, Ottawa, Canada (2007)
6. Hashem, M.K., Oommen, B.J.: On using learning automata to model a student's behavior in a tutorial-*like* system. In: Okuno, H.G., Ali, M. (eds.) IEA/AIE 2007. LNCS (LNAI), vol. 4570, pp. 813–822. Springer, Heidelberg (2007)
7. Hashem, M.K., Oommen, B.J.: Using learning automata to model a domain in a tutorial-*like* system. In: Proceedings of ICMLC 2007, the 2007 International Conference of Machine Learning and Cybernetics, Hong Kong, August 2007, pp. 112–118 (2007)
8. Hashem, M.K., Oommen, B.J.: Using learning automata to model a student-classroom interaction in a tutorial-*like* system. In: Proceedings of IEEE-SMC 2007, the 2007 IEEE International Conference on Systems, Man and Cybernetics, Montreal, October 2007, pp. 1177–1182 (2007)
9. Hashem, M.K., Oommen, B.J.: Using learning automata to model the behavior of a teacher in a tutorial-*like* system. In: Proceedings of IEEE-SMC 2007, the 2007 IEEE International Conference on Systems, Man and Cybernetics, Montreal, October 2007, pp. 76–82 (2007)
10. Hashem, M.K., Oommen, B.J.: Using learning automata to model the "learning process" of the teacher in a tutorial-*like* system. In: Proceedings of ISCIS 2007, the 2007 International Symposium on Computer and Information Sciences, Paper No. 1.3.C-012, Ankara, Turkey (November 2007)
11. Haykin, S.: Neural Networks: A Comprehensive Foundation. IEEE Press/ Macmillan College Publishing Company, New York (1994)
12. Lakshmivarahan, S.: Learning Algorithms Theory and Applications. Springer, Heidelberg (1981)
13. Legaspi, R.S., Sison, R.C.: Modeling the tutor using reinforcement learning. In: Proceedings of the Philippine Computer Science Congress (PCSC), pp. 194–196 (2000)
14. Lelouche, R.: A collection of pedagogical agents for intelligent educational systems. In: Gauthier, G., VanLehn, K., Frasson, C. (eds.) ITS 2000. LNCS, vol. 1839, pp. 143–152. Springer, Heidelberg (2000)
15. Najim, K., Poznyak, A.S.: Learning Automata: Theory and Applications. Pergamon Press, Oxford (1994)
16. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice-Hall, New Jersey (1989)
17. Obaidat, M.S., Papadimitrious, G.I., Pomportsis, A.S.: Learning automata: Theory, paradigms, and applications. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32(6), 706–709 (2002)

18. Oommen, B.J.: Absorbing and ergodic discretized two-action learning automata. IEEE Transactions on Systems, Man, and Cybernetics SMC-16, 282–293 (1986)
19. Oommen, B.J., Agache, M.: Continuous and discretized pursuit learning schemes: Various algorithms and their comparison. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 31, 277–287 (2001)
20. Poznyak, A.S., Najim, K.: Learning Automata and Stochastic Optimization. Springer, Berlin (1997)
21. Self, J.: The defining characteristics of intelligent tutoring systems research: ITSs care, precisely. International Journal of AI in Education 10, 350–364 (1999)
22. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
23. Thathachar, M.A.L., Sastry, P.S.: Varieties of learning automata: An overview. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32(6), 711–722 (2002)
24. Thathachar, M.A.L., Sastry, P.S.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic, Boston (2003)