Modeling a Student's Behavior in a Tutorial-*Like* System Using Learning Automata

B. John Oommen, *Fellow, IEEE*, and M. Khaled Hashem

***Abstract*—This paper presents a new philosophy to model the behavior of a student in a tutorial-*like* system using learning automata (LAs). The model of the student in our system is inferred using a higher level LA, referred to as a *meta-LA*, which attempts to characterize the learning model of the students (or student simulators), while the latter use the tutorial-*like* system. The *meta-LA*, in turn, uses LAs as a learning mechanism to try to determine if the student in question is a fast, normal, or slow learner. The ultimate long-term goal of the exercise is the following: if the tutorial-*like* system can understand how the student perceives and processes knowledge, it will be able to customize the way by which it communicates the knowledge to the student to attain an optimal teaching strategy. The proposed *meta-LA* scheme has been tested for numerous environments, including the established benchmarks, and the results obtained are remarkable. Indeed, to the best of our knowledge, this is the first published result that infers the learning model of an LA when it is externally treated as a black box, whose outputs are the only observable quantities. Additionally, our paper represents a new class of multiautomata systems, where the *meta-LA* synchronously communicates with the students, *also modeled* using LAs. The *meta-LA*'s environment "observes" the progress of the student LA, and the response of the latter to the *meta-LA* actions is based on these observations. This paper also discusses the learning system implications of such a *meta-LA*.**

***Index Terms*—Learning automata (LAs), modeling of adaptive systems, student modeling, tutorial-*like* systems.**

I. Introduction

CENTRAL to every learning or adaptive system is an entity that performs the learning and another entity that teaches the latter. Based on real-life analogies, these can informally be referred to as the "student" and the "teacher," respectively. Broadly speaking, this paper deals with the issue of modeling the student in such a learning/adaptive system. In particular, we intend to demonstrate how this modeling can successfully be achieved using learning automata (LAs), when the learning cycle is couched within the framework of a tutorial-*like* system.

The student is the focal point of interest in any tutorial system. The latter is customized to maximize the learning curve for the student. For the tutorial system to achieve this, it needs to treat every student according to his[1] particular skills and abilities. The system's model for the student permits such a customization.

In the context of this paper, a student model is a representation of the student's behavior and status. It is the basis for representing the state of the student. When the student uses the system, the student model records his sequence of actions performed, using which the system should hopefully model the learning paradigm used by the student and should attempt to assess the student's learning progress. Thus, the aim of this paper can be stated as follows: we intend to present a new philosophy to model how a student learns while using a tutorial-*like* system based on the theory of stochastic LA. The stochastic LA will enable the system to determine the learning model of the student, which will consequently enable the tutorial-*like* system to improve the way it "deals" (i.e., presents material, teaches, and evaluates) with the student to customize the learning experience.

Inferring the student's learning model, by what we call as a *meta-LA*, will improve the learning experience for the student. If the teacher knows in advance what the learning model of the student is, he will customize his teaching strategy according to this learning model. For example, if the teacher knows that he is dealing with a smart student, then he will present more difficult problems to the student from the start and be able to present material at a faster rate.

In this paper, we propose a *meta-LA* strategy, which can be used to examine the student model. As the name implies, we have used LAs as a learning tool/mechanism that steers the inference procedure achieved by the *meta-LA*. To be more specific, we have assumed that a student can be modeled as being one of the following types:

1) slow-learner student;
2) normal-learner student;
3) fast-learner student.

Clearly, this subdivision of capabilities can be subject to a more fine resolution, but we believe that such a resolution is sufficient for most "tutorial-*like*" applications and to demonstrate a *prima facie* case for our hypothesis.

For the system to determine the learning model of the student, it monitors the consequent actions of the student and uses the *meta-LA* to unwrap the learning model.

It is conceivable that each student can be represented by other types of learning mechanisms such as neural networks,

Manuscript received February 7, 2008; revised September 29, 2008 and May 22, 2009. First published September 9, 2009; current version published March 17, 2010. The work of B. John Oommen was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). This paper was presented in part at the Twentieth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE '07), Kyoto, Japan, June 2007. This paper was recommended by Associate Editor Z. R. Yang.

B. J. Oommen is with the School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada, and also with the University of Agder, 4876 Grimstad, Norway (e-mail: oommen@scs.carleton.ca).

M. K. Hashem is with MKH Consulting Inc., Ottawa, ON K1C 6S5, Canada (e-mail: k_hashem@yahoo.com).

Digital Object Identifier 10.1109/TSMCB.2009.2027220

[1]For the ease of communication, we request the permission to refer to the entities involved (i.e., the teacher, the student, etc.) in the masculine.

Bayesian or Markovian models, and reinforcement learning models. Although we hypothesize that generalizing our paradigm to other learning models will not be too difficult, we briefly list some of the difficulties that we would encounter if any of these models are used to represent a student.

1) If the student is represented by a neural network, the model will first involve the number of neurons. Second, the topology of the network must be specified, and so, in this case, the topology must also be considered as a "parameter" of the network. Finally, every single edge associated with the topology would have a corresponding weight traditionally learned through the training process. Thus, the entire modeling field would have to incorporate the specifications and instantiation of *all* of these quantities.
2) If the student is represented by a Bayesian or Markovian model, the designer would have to decide on the number of states that the model will contain. Thereafter, he would have to consider the interconnections between the states and their corresponding transition probabilities. Finally, the question of determining the output of the model based on the state will involve a state/output function, which will also have to be determined. *All* these parameters are traditionally done by an inference mechanism and by training.
3) If the student is modeled using a reinforcement learning model, the state vector of the model will not only contain probability distributions but also include the information that is recorded concerning the state of the environment.

In this paper, to keep the discussion focused, we shall concentrate only on the premise that the student is modeled by an LA. The additional problem that we foresee in using other learning models is that of understanding how the students will impart the knowledge to their colleagues[2] and how the categorization can be done with a small set of parameters. In the case of LAs, this can be achieved in a straightforward matter by a simple message-passing mechanism in which *only* the action probability vector is communicated. We add though that reinforcement learning models are probably the next level of generalization to LAs, and we believe that these would be the first avenues by which a researcher can obtain a more finely tuned model of the student.

We venture to add that, on the other hand, one of the most interesting aspects of this endeavor is that we can categorize a learning mechanism (for example, one that utilizes neural networks, Bayesian or Markovian models, or reinforcement learning) as being, for example, *slow*, *normal*, or *fast*, by studying *its* behavior using such a *meta-LA* strategy.

The proposed *meta-LA* scheme has been tested for numerous environments, including the established benchmarks, and the results obtained are remarkable. Our experimental results, which are presented and explained later, show that the *meta-LAs* are successful in determining the learning model of the student, by observing the student's actions and the teaching environment. This would have a direct effect on the effectiveness of the tutorial-*like* system.

Indeed, to the best of our knowledge, this is the first published result that attempts to infer the learning model of an LA when it is externally treated as a black box, whose outputs are the *only* observable quantities.

A. Tutorial-Like Systems: An Overview

Our entire research will be within the context of tutorial-*like* systems [11]. 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 (synonymously referred to also as student simulators) 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 that we study, the teacher has a *stochastic* nature, where he has imprecise knowledge of the material to be imparted. The teacher also does not have 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 know 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 a *multiple-choice* sort or, in the most extreme case, 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.

Using machine learning in student modeling was the focus of a few previous studies. Legaspi and Sison [19] modeled the tutor in intelligent tutorial systems (ITSs) using reinforcement learning with the temporal difference method as the central learning procedure. Beck [6] used reinforcement learning to learn to associate superior teaching actions with certain states of the student's knowledge. Baffes and Mooney implemented ASSERT [5], which used reinforcement learning in student modeling to capture novel student errors using only correct domain knowledge. Our method is distinct from all the aforementioned, as we shall presently demonstrate.

B. Stochastic LA

LAs have been used in systems that have incomplete knowledge about the environment in which they operate [3], [15],

[2]This is an issue that we shall discuss elsewhere [11], [12].

[21], [25], [27], [45], [56]. The learning mechanism attempts to learn from a *stochastic teacher* that models the environment. In his pioneer work, Tsetlin [57] attempted to use LAs to model biological learning. In general, a random action is selected based on a probability vector, and these action probabilities are updated based on the observation of the environment's response, after which the procedure is repeated.

The term "LAs" was first publicized in the survey paper by Narendra and Thathachar. The goal of LAs is to "determine the optimal action out of a set of allowable actions" [3]. The distinguishing characteristic of automata-based learning is that the search for the optimizing parameter vector is conducted in the space of probability distributions defined over the parameter space, rather than in the parameter space itself [55].

Excellent references that survey the field are the books by Lakshmivarahan [15], Narendra and Thathachar [25], and Najim and Poznyak [21] and a recent special issue of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B [27].

The learning paradigm as modeled by LAs has found applications in systems that possess incomplete knowledge about the environment in which they operate. A variety of applications[3] that use LAs have been reported in the literature. They have been used in game playing [4], [15], [16], pattern recognition [23], [47], object partitioning [36], [37], parameter optimization [7], [25], [45], [46], multiobjective analysis [22], telephony routing [24], [26], and priority assignments in a queuing system [20]. They have also been used in statistical decision making [21], [25], distribution approximation [1], natural language processing, modeling biological learning systems [57], string taxonomy [32], graph partitioning [33], distributed scheduling [51], network protocols (including conflict avoidance [39]) for LANs [40], photonic LANs [42], star networks [41], broadcast communication systems [43], dynamic channel allocation [43], tuning proportional-integral differential controllers [13], assigning capacities in prioritized networks [38], map learning [8], digital filter design [14], controlling client/server systems [44], adaptive signal processing [52], vehicle path control [58], the control of power systems [60], and vehicle suspension systems [10].

In the first LA designs, the transition and the output functions were time invariant, and for this reason, these LAs were considered "fixed-structure" automata (FSSAs). Tsetlin [57] presented notable examples of this type of automata. Later, Vorontsova and Varshavskii introduced a class of stochastic automata known in the literature as variable-structure stochastic automata (VSSAs). In the definition of a 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 [25] operates on a vector (called *the action probability vector*)

$$P(t) = [p_1(t), \ldots, p_r(t)]^T$$

where $p_i(t)(i = 1, \ldots, r)$ is the probability that the automaton will select the action α_i at time "t"

$$p_i(t) = \Pr[\alpha(t) = \alpha_i], \qquad i = 1, \ldots, r$$

and it satisfies

$$\sum_{i=1}^{r} p_i(t) = 1, \qquad \forall t.$$

Note that the algorithm $T : [0,1]^r \times A \times B \to [0,1]^r$ is an updating scheme, where $A = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}, 2 \leq r < \infty$, 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

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

where $P(t)$ is the action probability vector, $\alpha(t)$ is the action chosen at time t, and $\beta(t) \in B$ is the response it has obtained. Typically, B is the set $\{0, 1\}$, where the response "0" corresponds to a reward and "1" corresponds to a penalty.

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. Many families of VSSAs that possess absorbing barriers have been reported [25]. Ergodic VSSAs have also been investigated [25], [30]. These VSSAs converge in distribution, and thus, the asymptotic distribution of the action probability vector has a value that is independent of the corresponding initial vector. Thus, while ergodic VSSAs are suitable for nonstationary environments, automata with absorbing barriers are preferred in stationary environments.

In practice, the relatively slow rate of convergence of these algorithms constituted a limiting factor in their applicability. To increase their speed of convergence, the concept of discretizing the probability space was first introduced in [53] and later extensively studied in [2], [3], [17], [18], [29], [31], and [34]. This concept is implemented by restricting the probability of choosing an action to a finite number of values in the interval [0, 1]. If the values allowed are equally spaced in this interval, the discretization is said to be linear; otherwise, the discretization is called nonlinear. Following the discretization concept, many of the continuous VSSAs have been discretized; indeed, discrete versions of almost all continuous automata have been presented in the literature [2], [3], [17], [18], [29], [31], [34].

Pursuit and estimator-based LAs were introduced to be faster schemes, characterized by the fact that they pursue what can be reckoned to be the *current* optimal action or the set of current optimal actions [2], [3], [17], [18], [30], [49], [54]. 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. While, in nonestimator algorithms, the action probability vector is updated solely on the basis of the environment's response, in a pursuit or estimator-based LA, the update is based on *both*

[3]*The applications listed here (and the references listed in the bibliography) are quite comprehensive but are by no means complete.* Indeed, this relatively new field has been "exploding." It has recently been enhanced by a spectrum of applications in computer science and engineering—from areas as diverse as the design of data structures to the implementation of automatic navigation methods.

the environment's response and the *reward-estimate* vector. Families of pursuit and estimator-based LAs have been shown to be faster than VSSAs [55]. Indeed, even faster discretized versions of these schemes have been reported [2], [3], [17], [18], [30].

C. Contributions of This Paper

This paper presents a novel approach for modeling how the student learns in a tutoring session. We believe that the successful modeling of students will be fundamental to implementing effective tutorial-*like* systems. It will also be central to improving the way by which an environment (perceived as a conceptual "teacher") can enhance its teaching within an arbitrary learning/adaptive system. Thus, the salient contributions of this paper are the following.

1) This paper presents the concept of a *meta-LA* in student modeling, which works with a higher level learning paradigm. The *meta-LA* observes and infers the characteristics of the learning experience of the student in an "efficient" way. In the current instantiation, the *meta-LA* classifies the student to be a *slow*, *normal*, or *fast* learner.
2) To the best of our knowledge, this paper is the first published result that infers the learning model of an LA when it is externally treated as a black box, whose outputs are the only observable quantities.
3) While a traditional tutorial system may not be able to consider the "psychological" perspective of the student, we attempt to consider it by using an inferred model of his cognitive state. After the tutorial-*like* system infers the learning model of the student, it will be able to customize its teaching strategy as per the student's learning ability.
4) As a modeling strategy, we believe that we can categorize a learning mechanism (for example, one which utilizes neural networks, Bayesian or Markovian models, or reinforcement learning) as being, for example, *slow*, *normal*, or *fast*, by studying *its* behavior using such a *meta-LA* strategy. Notice that this mechanism need not be LA based and that it can also be treated as a black box.

II. Intelligent Tutorial and Tutorial-*Like* Systems

Since our research involves tutorial-*like* systems, which are intended to mimic tutorial systems, a brief overview of these follows.

ITSs are special educational software packages that involve artificial intelligence techniques and methods to represent the knowledge, as well as to conduct the learning interaction [28]. ITSs are characterized by their responsiveness to the learner's need. They adapt according to the knowledge/skill of the users. They also incorporate experts' domain-specific knowledge.

An ITS mainly consists of a number of modules, which is typically three [9] and is sometimes four when a communication module (interface) is added [59]. The former three modules are the domain model (knowledge domain), the student model, and the pedagogical model (which represent the tutor model itself). Self [50] defined these components as the tripartite architecture for an ITS—the *what* (domain model), the *who* (student model), and the *how* (tutoring model). Fig. 1 depicts a common ITS architecture.

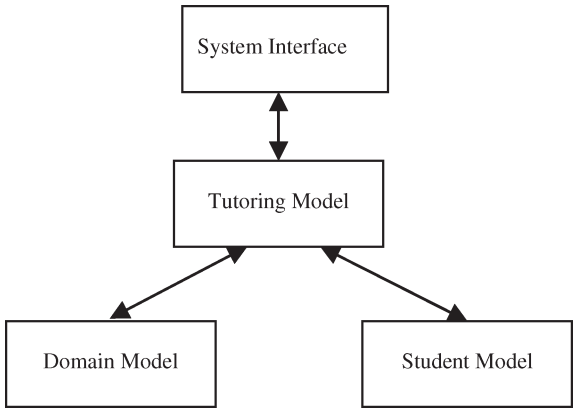

Fig. 1. Common ITS architecture.

A. Tutorial-Like Systems

Tutorial-*like* systems share some similarities with the well-developed field of tutorial systems. Thus, for example, they model the teacher, the student, and the domain knowledge. However, they are different from "traditional" tutorial systems in the characteristics of their models, etc., as will be highlighted below.

1) *Different types of teacher*. In tutorial systems, as they are developed today, the teacher is assumed to have perfect information about the material to be taught. Furthermore, built into the model of the teacher are the knowledge of how the domain material is to be taught and a plan of how it will communicate and interact with the student(s). This teaching strategy may progress and improve over time. The teacher in our tutorial-*like* system possesses different features. First, one fundamental difference is that the teacher is uncertain of the teaching material—he is stochastic. Second, 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 particular student. To achieve this, as mentioned, we assume that the teacher follows the Socratic model of learning by teaching the material using questions that are presented to the students. He then uses the feedback from the students and their corresponding LAs to suggest new teaching material.

   Although removing the "how-to-teach" knowledge from the teacher would take away the "bread-and-butter" premise of the teaching process in a tutorial system, in a tutorial-*like* system, removing this knowledge allows the system to be modeled without excessive complications and renders the modeling of knowledge less burdensome. The success of our proposed methodology would be beneficial to systems in which any domain knowledge pertinent to tutoring teaching material could merely be

plugged into the system without the need to worry about "how to teach" the material.

2) *No real students*. A tutorial system is intended for the use of *real-life* students. Its measure of accomplishment is based on the performance of these students after using the system, and it is often quantified by comparing their progress with other students in a control group, who would use a *real-life* tutor. In our tutorial-*like* system, there are no *real-life* students who use the system. The system could be used by either of the following.
   a) *Student simulators*. Student simulators mimic the behavior and actions of *real-life* students using the system. The latter would themselves simulate how the students improve their knowledge and their interaction with the teacher and with other students. They can also take proactive actions, interacting with the teaching environment by one of the following measures:
      i) asking a question to the teacher;
      ii) asking a question to another student;
      iii) proposing to help another student.
   b) *Artificial entity*. An artificial entity, 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 teaching material presented in our tutorial-*like* system contains material that has some degree of uncertainty. The teaching material contains questions, each of which has a probability that indicates the certainty of whether the answer to the question is in the affirmative.
4) *Testing versus evaluation*. Sanders [48] differentiates between the concepts of "teaching evaluation" and "teaching testing." He defines "teaching evaluation" as an "interpretive process," in which the teacher "values, determines the merit or worth of the students' performance, and their needs." He also defines "teaching testing" as a "data collection process." In a tutorial system, an evaluation is required to measure the performance of the student while using the system and acquiring more knowledge. In our tutorial-*like* system, the student(s) acquires knowledge using a Socratic model, where it gains knowledge from answering questions without having any prior knowledge about the subject material. In our model, the testing will be based on the performance of the set of student simulators.
5) *School of students*. Traditional tutorial systems deal with a teacher who teaches students, but they do not permit the students to interact with each other. 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 or is communicating with a cooperating colleague. Notice that we will now have to consider how each student learns, as well as how the entire *school* learns.

In all brevity, we list below the salient aspects of each of the modules of our paradigm for a tutorial-*like* system.

The student simulator is modeled using an LA paradigm. It uses LAs 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 them to exchange information and learn from each other.

The method by which the model for the student is inferred is the kernel of the study involved here and will be discussed in subsequent sections.

The student–classroom interaction is intended to maximize the learning experience of each student, as well as the collective learning of the students. 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 in which: 1) he always assumes that the knowledge of other students is reliable; 2) he uses the information from his colleagues (partially or totally) only if it is reliable; 3) he is able to unlearn if the information provided by his colleagues is misleading; and 4) he decides to independently learn and does not communicate with other fellow students.

Within our paradigm, the domain knowledge will be presented *via* multiple-choice Socratic-type questions, and thus, for each question, every choice has an associated probability of being correct, implying that 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, and students will not be able to predict the answer for subsequent chapters using prior knowledge.

Similarly, within our paradigm, we model the teacher to be able to teach the Socratic-type domain knowledge, *via* multiple-choice questions. This knowledge is also used to *test* the students of the imparted knowledge. Thus, 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 to be able to learn increasingly complex material. Finally, we build our research on the model that 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 to handle the domain knowledge as the difficulty of the latter increases.

Finally, 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. Thus, for each environment that the student is attempting to learn from, the teacher will define his *own* standards for the specific environment. Thus, 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 *assistance* to each individual student.

III. Network Involving the LA/*Meta-LA*

Our tutorial-*like* system incorporates multiple LAs that are indirectly interconnected with each other. This can be

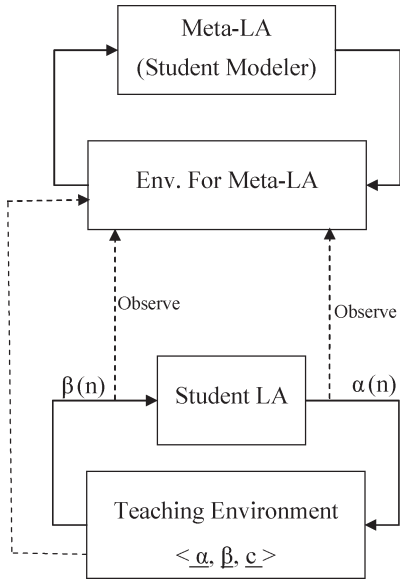

Fig. 2. Modeling using a network of LAs.

perceived as a system of interconnected automata [55]. Since such systems of interconnecting automata may be one of two models, namely, synchronous and sequential, we observe that our system represents a synchronous model. The LAs in our system and their interconnections are illustrated in Fig. 2, in which $\underline{c}$ is the vector of penalty probabilities.

In a traditional system of interconnected automata, the response from the environment associated with one automaton represents the input to another automaton. However, our system has a rather unique LA interaction model, which is used to assist in the process of inferring on the type of the LA itself. While the student LA is affecting the *meta-LA*, there is no direct connection between both of them. The *meta-LA* environment monitors the "performance" of the student LA over a period of time, and depending on that, *its* environment would penalize or reward the action of the *meta-LA*.

This model represents a new structure of interconnection, which can be viewed as being composed of two levels: a higher level automaton, i.e., the *meta-LA*, and a lower level automaton, which is the student LA. The convergence of the higher level automaton is dependent on the behavior of the lower level automaton. A typical scenario would be that the higher level automaton (which infers the type of learning achieved by the lower level LA) converges, while the lower level automaton has yet to converge. If the lower level automaton converges before the higher level automaton, this could mean that the convergence of the higher level automaton is inaccurate.

Observe the uniqueness of such an interaction of modules, as a consequence of the following.

1) The environment for the *meta-LA* is rather atypical but consists of observations, which must implicitly (and not explicitly) be perceived as an environment after doing some processing of the signals observed.
2) The *meta-LA* has access to the environment of the lower level, including the set of the penalty probabilities that are unknown to the lower level LA.
3) The *meta-LA* has access to the actions chosen by the lower level LA.
4) Finally, the *meta-LA* has access to the responses made by the lower level environment.

In conclusion, we observe that items 2–4 collectively constitute the environment for the *meta-LA*. Such a modeling is unknown in the field of LAs.

IV. Modeling a Student

The tutorial-*like* system derives a model for the student by assessing and determining the way he learns. To achieve this, we assume that the system has a finite set of possible learning models for each student. For example, each student can be modeled as being one of the following types:

1) an FSSA model, which represents a slow learner;
2) a VSSA model, which represents a normal learner;
3) a pursuit model LA, which represents a fast learner.

The rationale for the aforementioned categorization and classification is given as follows: When the field of LAs was first pioneered, Tsetlin [25], [57] presented their learning machines, which were capable of learning in an ϵ-optimal manner in certain environments. Although this was significant, it turned out that their machines were remarkably slow, often requiring *tens* of thousands of iterations to converge. Furthermore, the number of states in their machines had to be quite large to attain reasonable convergence accuracy. The reason for the relatively slow convergence was because of the fact that the LAs could change their actions only at their *boundary* states. By proposing the concept of action probability vectors, researchers were able to develop LAs that could potentially choose different actions for subsequent time instants, thus significantly improving the rate of learning. This led to the family of VSSAs such as the L_{RI} scheme, which converged in a few thousands of iterations. Finally, Thathachar and Sastry [49], [54] (and others [3], [17], [18], [35]) proposed a further enhancement of the learning by considering the option of also incorporating *estimates* of the penalty probabilities in the updating rules. This led to the extremely fast convergence obtained by the family of the so-called estimator-based algorithms. Our belief is that these three families of algorithms represent three distinct types of learning paradigms and learning rates sufficient for the present research.

This finite set of learning models represents the different families that characterize the way the student learns. The student could be, for example, a slow learner, a normal leaner, or a fast learner. Of course, a finer learning categorization is also achievable. As mentioned, if the tutorial-*like* system can understand how the student perceives knowledge, it will be able to customize the way by which it communicates the knowledge to the student to aim toward an optimal teaching strategy. For example, if the tutorial-*like* system finds that the student is learning in a way that resembles a pursuit model, it can conclude that the student is "clever" and that it is a fast learner. In this case, the system can adapt its teaching strategy to accommodate a fast-learning student, which, in turn, enables it to improve its teaching strategy.

The student modeler will itself utilize an LA in the *meta-LA* level to represent the different potential learning models for the student. Furthermore, as the student interaction with the tutorial-*like* system increases, *this* LA would hopefully converge to the model that most accurately represents the way the student learns.

A. Formalization of the Student Model

To simplify issues, we assume that the learning model of the student is one of the aforementioned three types.[4] The student model will use three specific potential LAs, each of which serves as an action for a higher level automaton—(i.e., the *meta-LA*).

In our present instantiation, the *meta-LA* that is used to learn the best possible model for the student simulator is a 4-tuple: $\{\underline{\alpha}, \underline{\beta}, P, T\}$, where the variables are described as follows.

1) $\underline{\alpha} = \{\alpha_1, \alpha_2, \alpha_3\}$, in which α_1 is the action corresponding to an FSSA model, which represents a slow learner, α_2 is the action corresponding to a VSSA model, which represents a normal learner, and α_3 is the action corresponding to a pursuit model LA, representing a fast learner.
2) $\underline{\beta} = \{0, 1\}$, in which $\beta = 0$ implies a reward for the present action (i.e., student model) chosen, and $\beta = 1$ implies a penalty for the present action (i.e., student model) chosen.
3) $P = [p_1, p_2, p_3]^T$, where $p_i(n)$ is the current probability of the student simulator being represented by α_i.
4) T is the probability updating rule given as a map as

$$T : (P, \underline{\alpha}, \underline{\beta}) \rightarrow P.$$

Each of these will now be clarified.

1) $\underline{\beta}$ is the input that the *meta-LA* receives (or rather infers). Let us suppose that the current choice of the *meta-LA* is that the student simulator is learning using the model symbolized by α_i. This means that the *meta-LA* will have to infer a reward or a penalty based on how the student simulator is learning and on whether the latter can accurately be represented by α_i or not. This, in turn, means that the *meta-LA* must observe a sequence of decisions made by the student simulator, and based on this sequence, it must deduce whether its current model is accurate or not. We propose to achieve this as follows.

   For a fixed number of queries, which we shall refer to as the "window," the *meta-LA* assumes that the student simulator's model is α_i. Let $\theta(t)$ be a measure of the weakness (MoW) of the lower LA at time "t," measured in terms of the number of penalties received for any action, weighted by the current action probability vector. By inspecting the way by which the student simulator learns within this window, it infers whether this current model should be rewarded or penalized. We propose to achieve this by using two thresholds, which we refer to as $\{\theta_j | 1 \leq j \leq 2\}$ (as shown in Table I). The significance of these thresholds is described by the following.

   a) If the current observed MoW of the student simulator is less than θ_1, we assert that the student simulator demonstrates the phenomenon of a *fast* learner. Now, if the *meta-LA* has chosen α_3, this choice is rewarded. Otherwise, this choice is penalized.
   b) If the current observed MoW of the student simulator is equal to or greater than θ_1 and less than θ_2, we assert that the student simulator demonstrates the phenomenon of a *normal* learner. Thus, if the *meta-LA* has chosen α_2, this choice is rewarded. Otherwise, this choice is penalized.
   c) Finally, if the current observed MoW of the student simulator is equal to or greater than θ_2, we assert that the student simulator demonstrates the learning properties of a *slow* learner. Thus, if the *meta-LA* has chosen α_1, this choice is rewarded. Otherwise, the *meta-LA* is penalized.

   This terminates our discussion on how β is inferred.[5]
2) P is the action probability vector that contains the probabilities that the *meta-LA* assigns to each of *its* actions. At each instant t, the *meta-LA* randomly selects an action $\alpha(t) = \alpha_i$. The probability that the automaton selects action α_i at time t is the action probability $p_i(t) = \Pr[\alpha(t) = \alpha_i]$, where $\sum_{i=1}^{r} p_i(t) = 1 \; \forall \, t, \; i = 1, 2, 3$.

   Initially, the *meta-LA* will have an equal probability for each of the three learning models as

$$P(0) = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]^T.$$

   By observing the sequence of $\{\theta(t)\}$, the *meta-LA* updates $P(t)$ as per the function T, as presently described. By virtue of this interaction, one of the action probabilities p_j will hopefully converges to a value as close to unity as we want. This convergence indicates that the *meta-LA* has converged to the choice that α_j is the best learning model, and if T is appropriately chosen, we believe that it will be successful in determining the best learning model appropriate for the student simulator.
3) T is the updating algorithm or the reinforcement scheme used to update the *meta-LA*'s action probability vector and is represented by

$$P(t+1) = T\left[P(t), \alpha(t), \beta(t)\right].$$

   This updating algorithm could follow *any* LA learning schemes. For example, if it obeys the DL_{RI} rule, the action probability vector is updated in a *discretized*

[4]Extending this to consider a wider spectrum of learning models is rather straightforward.

[5]One referee queried the rationale for using LAs when our methodology intricately involved an underlying thresholding scheme. We *gratefully acknowledge his perspective and input in this regard*. This is really a debatable issue, but our experience is that a mere thresholding arrangement is inadequate to infer the model for the learning mechanism. The reason for this is that the environment ("teacher") and the teaching material ("domain") are both stochastic, and thus, the threshold itself would be a random variable. The point here is that we have used this random variable as an implicit input to the *meta-LA* to learn the underlying learning model.

TABLE I
THRESHOLDS USED BY THE *Meta-LA* FOR THE LEARNING MODEL, WHERE $\theta(t)$ IS THE MOW AT TIME "t"

Type Of Student	Boundary Threshold
Fast Learner	$\theta(\mathrm{t}) < \theta_1$
Normal Learer	$\theta_1 \leq \theta(\mathrm{t}) < \theta_2$
Slow Learer	$\theta_2 \leq \theta(\mathrm{t})$

manner when the *meta-LA* is rewarded and unchanged when the *meta-LA* is penalized. In this case, the LA can be described entirely by the following action probability updating equations (where N is the resolution parameter of the discretized scheme):

$$p_j(t+1) = \max\left\{0, p_j(t) - \frac{1}{N}\right\}, \qquad \text{if } \alpha(t)=\alpha_i, \beta(t)=0$$

$$p_i(t+1) = 1 - \sum_{j \neq i} p_j(t+1), \qquad \text{if } \alpha(t)=\alpha_i, \beta(t)=0$$

$$p_k(t+1) = p_k(t), \qquad \forall k, \text{ if } \beta(t)=1.$$

The algorithm starts with the initial action probability vector $P(0) = [(1/3), (1/3), (1/3)]^T$.

V. EXPERIMENTAL RESULTS

This section presents the experimental results obtained by testing the prototype implementation of the *meta-LA*. To obtain these results, we performed numerous simulations to accurately simulate how the student modeler is able to recognize the learning model of the student.

The simulations were performed for many different types of environments. These environments represent the teaching "problem," which contains the uncertain course material of the domain model associated with the tutorial-*like* system. In all the tests performed, an algorithm was considered to have converged if the probability of choosing an action was greater than or equal to a threshold $T(0 < T \leq 1)$. If the automaton converged to the best action (i.e., the one with the highest probability of being rewarded), it was considered to have correctly converged.

As presented in Section III, the *meta-LA* was implemented as a higher level LA, which had access to the learning environment and the actions and penalties of the lower level LA associated with the student simulators, while the latter learned the teaching material. The *meta-LA* was implemented as a discretized LA, i.e., the DL_{RI} scheme, with a resolution parameter $N = 40$.

The student simulator was implemented to mimic three typical types of students as follows:

- Slow learner: For this type of students, the student simulator used an FSSA to mimic the student's behavior. In particular, it used a Tsetlin $\mathrm{L}_{2R,R}$ automaton implementation for this behavior, where it had two states for each action.

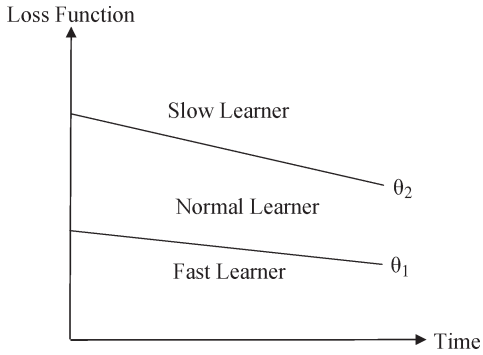

Fig. 3. Loss function displayed as a function of the MoW θ.

- Normal learner: For this category of students, the student simulator used a VSSA to simulate the student's behavior. In particular, in our paper, it was implemented as an L_{RI}, with $\lambda = 0.005$. Observe that the lower level LA updates itself only if it obtained a reward for its action; otherwise, it did nothing.
- Fast learner: To simulate students of this type, the student simulator utilized a pursuit LA to represent the student's behavior. It used the pursuit PL_{RI}, with $\lambda = 0.005$, to simulate fast-learning students. Here too, the LA only updated its probabilities when it obtained a reward for its action.

The rate of learning associated with the student is a quantity that is difficult to quantify. We had to determine the best index in each case. To be able to measure θ, which is the MoW[6] for the student, the *meta-LA* environment calculates the loss function of the student simulator LA. For a "window" of n choice-response feedback cycles of the student simulator, the loss function is defined by

$$\frac{1}{n}\sum n_i c_i$$

where

- n_i number of instances an action α_i is selected inside the "window";
- c_i penalty probability associated with action α_i of the lower level LA.

We consider the MoW for the student to be a function of the loss function and time, as illustrated in Fig. 3. This reflects the fact that the loss function for any learning student typically decreases with time (for all types of learners) as he learns more and as his knowledge improves with time.

As mentioned, the simulation has been performed for the different existing benchmark environments, for which the threshold T was set to be 0.99, and the number of experiments $NE = 75$, over which the results were averaged. The results of these simulations are described below.

[6]Recall that $\theta(t)$ is called the MoW of the lower LA at time "t," and it is measured in terms of the number of penalties received for any action, weighted by the current action probability vector.

TABLE II
CONVERGENCE OF *Meta-LA* FOR DIFFERENT STUDENT'S LEARNING MODELS IN FOUR-ACTION ENVIRONMENTS

Env.	θ_1	θ_2	PL_{RI}		L_{RI}		Tsetlin	
			# of Iter.	% correct convergence	# of Iter.	% correct convergence	# of Iter.	% correct convergence
$\mathbf{E_A}$	0.50-0.31	0.60-0.35	607	100%	1,037	87%	1,273	91%
$\mathbf{E_B}$	0.28-0.17	0.37-0.24	790	97%	1,306	89%	1,489	85%
Reward probabilities are: $\mathbf{E_A}$: 0.7 0.5 0.3 0.2 $\mathbf{E_B}$: 0.1 0.45 0.84 0.76								

TABLE III
CONVERGENCE OF *Meta-LA* FOR DIFFERENT STUDENT'S LEARNING MODELS IN SIX-ACTION ENVIRONMENTS

Env.	θ_1	θ_2	PL_{RI}		L_{RI}		Tsetlin	
			# of Iter.	% correct convergence	# of Iter.	% correct convergence	# of Iter.	% correct convergence
$\mathbf{E_A}$	0.60-0.38	0.52-0.31	601	96%	1,019	75%	1,207	96%
$\mathbf{E_B}$	0.45-0.32	0.30-0.20	941	92%	1,006	91%	1,125	96%
Reward probabilities are: $\mathbf{E_A}$: 0.7 0.5 0.3 0.2 0.4 0.5 $\mathbf{E_B}$: 0.1 0.45 0.84 0.76 0.2 0.4								

A. Environment With Four Actions

The four-action environment represents a multiple-choice question with four options. The student needs to learn the content of this question and conclude that the answer to this question is the action that possesses the minimum penalty probability. Based on the performance of the student, the *meta-LA* is supposed to "learn" the learning model of the student. The results of this simulation are presented in Table II.

To be conservative, the simulation assumes that when the student starts using the system, the results during a transient learning phase are misleading and inaccurate. Consequently, the first 200 iterations are neglected by the *meta-LA*, where it merely ignores all the interactions, decisions, and responses of the lower level LA.

The two different settings for the two environments are specified by their reward probabilities given in Table II.

The results obtained were quite remarkable. Consider the case of $\mathrm{E_A}$, where the best action was α_1. While an FSSA converged in 1273 iterations,[7] the L_{RI} converged in 1037 iterations, and the pursuit PL_{RI} converged in 607 iterations. By using the values of θ_1 and θ_2 as specified in Table II, the *meta-LA* was able to characterize the true learning capabilities of the student 100% of the time when the lower level LA used a pursuit PL_{RI}. The lowest accuracy of the *meta-LA* for environment $\mathrm{E_A}$ was 87%. The analogous results for $\mathrm{E_B}$ were 97% (at its best accuracy) and 85% for the slow learner. The power of our scheme should be obvious.

[7]The figures include the 200 iterations involved in the transient phase.

B. Environment With Six Actions

For the six-action environment, the student is posed with multiple-choice questions with six options and is required to determine the answer possessing the minimum penalty probability. In turn, the *meta-LA* infers the learning model of the student by examining the performance of the student. The results of this simulation are presented in Table III for the environments whose details are given in the table.

As in the case of the four-action problem, the *meta-LA* ignores the transient response of the first 200 iterations, reckoning them to be misleading and inaccurate.

We now report the results obtained from this simulation. Consider the case of $\mathrm{E_B}$, where the best action was α_3. An FSSA converged in 1125 iterations, where this figures includes the 200 iterations involved in the transient phase. Including this transient phase, the L_{RI} converged in 1006 iterations, and the pursuit PL_{RI} converged in 941 iterations. By using the values of θ_1 and θ_2 as specified in Table III, the *meta-LA* was able to characterize the true learning capabilities of the student 96% of the time when the lower level LA used an FSSA and 92% if it used a pursuit PL_{RI}. The lowest accuracy of the *meta-LA* for environment $\mathrm{E_B}$ was 91%. The analogous results for $\mathrm{E_A}$ were

TABLE IV
CONVERGENCE OF *Meta-LA* FOR DIFFERENT STUDENT'S LEARNING MODELS IN EIGHT-ACTION ENVIRONMENTS

Env.	θ_1	θ_2	PL_{RI}		L_{RI}		Tsetlin	
			# of Iter.	% correct convergence	# of Iter.	% correct convergence	# of Iter.	% correct convergence
$\mathbf{E_A}$	0.63-0.41	0.55-0.31	505	97%	826	91%	1,248	95%
$\mathbf{E_B}$	0.49-0.25	0.41-0.17	435	100%	1,176	86%	1,721	75%
Reward probabilities are:								
$\mathbf{E_A}$:	0.7	0.5	0.3	0.2	0.4	0.5	0.4	0.3
$\mathbf{E_B}$:	0.1	0.45	0.84	0.76	0.2	0.4	0.6	0.7

TABLE V
CONVERGENCE OF *Meta-LA* FOR DIFFERENT STUDENT'S LEARNING MODELS IN TEN-ACTION ENVIRONMENTS

Env.	θ_1	θ_2	PL_{RI}		L_{RI}		Tsetlin	
			# of Iter.	% correct convergence	# of Iter.	% correct convergence	# of Iter.	% correct convergence
$\mathbf{E_A}$	0.55-0.31	0.65-0.40	668	95%	748	93%	1,369	96%
$\mathbf{E_B}$	0.41-0.17	0.51-0.26	616	100%	970	85%	1,140	96%
Reward probabilities are:								
$\mathbf{E_A}$:	0.7	0.5	0.3	0.2	0.4	0.5	0.4	0.3	0.5	0.2
$\mathbf{E_B}$:	0.1	0.45	0.84	0.76	0.2	0.4	0.6	0.7	0.5	0.3

96% (at its best accuracy) and 75% for the normal learner,[8] which are all quite amazing considering that the *meta-LA* views the student as a mere black box.

C. Environment With Eight Actions

A similar simulation was also done for systems that have multiple-choice questions with eight options, whose results (and the corresponding environments) are reported in Table IV.

The results obtained are described as follows: Consider the case of $\mathrm{E_A}$, where the best action was α_1. An FSSA converged in 1248 iterations, where this figures includes the 200 iterations involved in the transient phase. Including this transient phase, L_{RI} converged in 826 iterations, while the pursuit PL_{RI} converged in 505 iterations. The values of θ_1 and θ_2 are specified in Table IV. Using these thresholds, the *meta-LA* was able to characterize the true learning capabilities of the student 97% of the time when the lower level LA used a pursuit PL_{RI} and 95% if it used an FSSA. The lowest accuracy of the *meta-LA* for environment $\mathrm{E_A}$ was 91%, and this was for a normal learner. Parallel results for $\mathrm{E_B}$ are also found in Table IV, and they are quite impressive.

D. Environment With Ten Actions

The ten-action environment represents one with multiple-choice questions with ten options. The respective environments' reward probabilities are given in Table V.

Similar to the environments for four, six, and eight actions, the results obtained for the ten-action environments are very fascinating. For example, consider the case of $\mathrm{E_A}$, where the best action was α_1. While an FSSA converged in 1369 iterations, L_{RI} converged in 748 iterations, and the pursuit PL_{RI} converged in 668 iterations. By using the values of θ_1 and θ_2 as specified in Table V, the *meta-LA* was able to determine the true learning capabilities of the student 95% of the time, when the lower level LA used a pursuit PL_{RI}. The lowest accuracy of the *meta-LA* for environment $\mathrm{E_A}$ was 93%. By comparison, for $\mathrm{E_B}$, the best accuracy was 100% for the fast leaner, and the worst accuracy was 85% for the slow learner.

We believe that the worst-case accuracy of 85% (for the inexact inference in this case) is within a reasonable limit. Indeed, in a typical learning environment, the teacher may incorrectly assume the wrong learning model for the student. Thus, we believe that our *meta-LA* learning paradigm is

[8]Interestingly enough, since the problem is quite difficult, the *meta-LA* confused the normal learner and the slow learner in this case. This is not too surprising considering their relative rates of convergence.

remarkable considering that these are the first reported results for inferring the nature of an LA if it is modeled as a black box.

VI. Conclusion and Future Work

This paper has presented a novel strategy for a tutorial-*like* system inferring the model of a student (student simulator). The student modeler itself uses an LA as an internal mechanism to determine the learning model of the student, so that it could be used in the tutorial-*like* system to customize the learning experience for each student. To achieve this, the student modeler uses a higher level automaton, the so-called *meta-LA*, which observes the student simulator LA's actions and the teaching environment and attempts to characterize the learning model of the student (or student simulator), while the latter uses the tutorial-*like* system. The *meta-LA* tries to determine if the student in question is a *fast*, *normal*, or *slow* learner.

From a conceptual perspective, this paper has presented a new approach for using interconnecting automata, where the behavioral sequence of one automaton affects the other automaton. The numerous other differences of such an interconnection are also listed in this paper.

From the simulation results, we can conclude that this approach is a feasible and valid mechanism applicable for implementing a student modeling process. The results shows that the student modeler was successful in determining the student's learning model in a high percentage of the cases—sometimes with an accuracy of 100%.

Our proposed approach has some advantages: it demonstrates ease of design, implementation, and testing. For future work, we believe that our approach can also be ported for use in real-life tutorial systems. By finding a way to measure the rate of learning for *real-life* students, the student modeler should be able to "approximate" the learning model for these students. This would decrease the complexity of implementing student models in traditional tutorial systems too. Our results can also be useful in enhancing traditional adaptive/learning systems—even though they may not be, strictly speaking, LA based.

The incorporation of modeling the domain, the teacher, the classroom interactions, and the teacher's learning process to implement a tutorial-*like* system is a research task for which some results are currently available [11].

References

[1] N. Abe and M. Warmuth, "On the computational complexity of approximating distributions by probabilistic automata," *Mach. Learn.*, vol. 9, pp. 205–260, 1998.

[2] M. Agache, "Families of estimator-based stochastic learning algorithms," M.S. thesis, School Comput. Sci., Carleton Univ., Ottawa, ON, Canada, 2000.

[3] M. Agache and B. J. Oommen, "Generalized pursuit learning schemes: New families of continuous and discretized learning automata," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 738–749, Dec. 2002.

[4] N. Baba, T. Soeda, and Y. Mogami, "An application of stochastic automata to the investment game," *Int. J. Syst. Sci.*, vol. 11, no. 12, pp. 1447–1457, Dec. 1980.

[5] P. Baffes and R. Mooney, "Refinement-based student modeling and automated bug library construction," *J. Artif. Intell. Educ.*, vol. 7, no. 1, pp. 75–116, 1996.

[6] J. Beck, "Learning to teach with a reinforcement learning agent," in *Proc. 15th Nat. Conf. AI/IAAI*, Madison, WI, 1998, p. 1185.

[7] H. Beigy and M. R. Meybodi, "Adaptation of parameters of BP algorithm using learning automata," in *Proc. VI Brazilian Symp. Neural Netw. (SBRN)*, 2000, pp. 24–31.

[8] T. Dean, D. Angluin, K. Basye, S. Engelson, L. Kaelbling, E. Kokkevis, and O. Maron, "Inferring finite automata with stochastic output functions and an application to map learning," *Mach. Learn.*, vol. 18, no. 1, pp. 81–108, Jan. 1995.

[9] E. Fischetti and A. Gisolfi, "From computer-aided instruction to intelligent tutoring systems," *Educ. Technol.*, vol. 30, no. 8, pp. 7–17, Aug. 1990.

[10] T. Gordon, C. Marsh, and Q. H. Wu, "Stochastic optimal control of vehicle suspension systems using learning automata," *Proc. Inst. Mech. Eng., Pt. I. J. Syst. Control Eng.*, vol. 207, no. 13, pp. 143–152, 1993.

[11] M. K. Hashem, "Learning automata based intelligent tutorial-like systems," Ph.D. dissertation, School Comput. Sci., Carleton Univ., Ottawa, ON, Canada, 2007.

[12] M. K. Hashem and B. J. Oommen, "Using learning automata to model a student-classroom interaction in a tutorial-*like* system," in *Proc. IEEE Int. Conf. SMC*, Montreal, QC, Canada, Oct. 2007, pp. 1177–1182.

[13] M. N. Howell and M. C. Best, "On-line PID tuning for engine idle-speed control using continuous action reinforcement learning automata," *Control Eng. Pract.*, vol. 8, no. 2, pp. 147–154, Feb. 2000.

[14] M. N. Howell and T. J. Gordon, "Continuous learning automata and adaptive digital filter design," in *Proc. UKACC Int. Conf. CONTROL, IEEE Conf. Publication No. 455*, 1998, vol. 1, pp. 100–105.

[15] S. Lakshmivarahan, *Learning Algorithms Theory and Applications*. New York: Springer-Verlag, 1981.

[16] S. Lakshmivarahan, "Two person decentralized team with incomplete information," *Appl. Math. Comput.*, vol. 8, pp. 51–78, 1981.

[17] J. K. Lanctôt, "Discrete estimator algorithms: A mathematical model of computer learning," M.S. thesis, Dept. Math. Statist., Carleton Univ., Ottawa, ON, Canada, 1989.

[18] J. K. Lanctôt and B. J. Oommen, "Discretized estimator learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1473–1483, Nov./Dec. 1992.

[19] R. S. Legaspi and R. C. Sison, "Modeling the tutor using reinforcement learning," in *Proc. PCSC*, 2000, pp. 194–196.

[20] M. R. Meybodi, "Learning automata and its application to priority assignment in a queueing system with unknown characteristics," Ph.D. dissertation, School Elect. Eng. Comput. Sci., Univ. Oklahoma, Norman, OK, 1983.

[21] K. Najim and A. S. Poznyak, *Learning Automata: Theory and Applications*. Oxford, U.K.: Pergamon, 1994.

[22] K. S. Narendra and K. Parthasarathy, "Learning automata approach to hierarchical multiobjective analysis," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 1, pp. 263–273, Jan./Feb. 1991.

[23] K. S. Narendra and M. A. L. Thathachar, "Learning automata: A survey," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-4, no. 4, pp. 323–334, Jul. 1974.

[24] K. S. Narendra and M. A. L. Thathachar, "On the behavior of a learning automata in a changing environment with routing applications," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, no. 5, pp. 262–269, May 1980.

[25] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[26] K. S. Narendra, E. Wright, and L. G. Mason, "Applications of learning automata to telephone traffic routing," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, no. 11, pp. 785–792, Nov. 1977.

[27] M. S. Obaidat, G. I. Papadimitrious, and A. S. Pomportsis, "Learning automata: Theory, paradigms, and applications," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 706–709, Dec. 2002.

[28] N. Omar and A. S. Leite, "The learning process mediated by intelligent tutoring systems and conceptual learning," in *Proc. Int. Conf. Eng. Educ.*, Rio de Janeiro, Brazil, 1998, p. 20.

[29] B. J. Oommen, "Absorbing and ergodic discretized two-action learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 2, pp. 282–293, Mar./Apr. 1986.

[30] B. J. Oommen and M. Agache, "Continuous and discretized pursuit learning schemes: Various algorithms and their comparison," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 3, pp. 277–287, Jun. 2001.

[31] B. J. Oommen and J. P. R. Christensen, "ϵ-optimal discretized reward-penalty learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 3, pp. 451–457, May/Jun. 1988.

[32] B. J. Oommen and E. V. de St. Croix, "String taxonomy using learning automata," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 27, no. 2, pp. 354–365, Apr. 1997.

[33] B. J. Oommen and E. V. de St. Croix, "Graph partitioning using learning automata," *IEEE Trans. Comput.*, vol. 45, no. 2, pp. 195–208, Feb. 1996.
[34] B. J. Oommen and E. R. Hansen, "The asymptotic optimality of discretized linear reward-inaction learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-14, no. 3, pp. 542–545, May/Jun. 1984.
[35] B. J. Oommen and J. K. Lanctôt, "Discretized pursuit learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 4, pp. 931–938, Jul./Aug. 1990.
[36] B. J. Oommen and D. C. Y. Ma, "Deterministic learning automata solutions to the equi-partitioning problem," *IEEE Trans. Comput.*, vol. 37, no. 1, pp. 2–13, Jan. 1988.
[37] B. J. Oommen and D. C. Y. Ma, "Stochastic automata solutions to the object partitioning problem," *Comput. J.*, vol. 35, pp. A105–A120, 1992.
[38] B. J. Oommen and T. D. Roberts, "Continuous learning automata solutions to the capacity assignment problem," *IEEE Trans. Comput.*, vol. 49, no. 6, pp. 608–620, Jun. 2000.
[39] G. I. Papadimitriou and D. G. Maritsas, "Learning automata-based receiver conflict avoidance algorithms for WDM broadcast-and-select star networks," *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 407–412, Jun. 1996.
[40] G. I. Papadimitriou, P. Nicopolitidis, and A. S. Pomportsis, "Self-adaptive polling protocols for wireless LANs: A learning-automata-based approach," in *Proc. IEEE ICECS*, St. Julian's, Malta, Sep. 2001, pp. 1309–1312.
[41] G. I. Papadimitriou and A. S. Pomportsis, "Self-adaptive TDMA protocols for WDM star networks: A learning-automata approach," *IEEE Photon. Technol. Lett.*, vol. 11, no. 10, pp. 1322–1324, Oct. 1999.
[42] G. I. Papadimitriou and A. S. Pomportsis, "Learning-automata-based MAC protocols for photonic LANs," in *Proc. IEEE ICON*, Singapore, Sep. 5–8, 2000, p. 481.
[43] G. I. Papadimitriou and A. S. Pomportsis, "On the use of stochastic estimator learning automata for dynamic channel allocation in broadcast networks," in *Proc. IEEE/CEC*, San Diego, CA, Jul. 2000, pp. 112–116.
[44] G. I. Papadimitriou, A. L. Vakali, and A. S. Pomportsis, "Designing a learning-automata-based controller for client/server systems: A methodology," in *Proc. 12th IEEE ICTAI*, Vancouver, BC, Canada, Nov. 2000, pp. 422–425.
[45] A. S. Poznyak and K. Najim, *Learning Automata and Stochastic Optimization*. Berlin, Germany: Springer-Verlag, 1997.
[46] A. S. Poznyak, K. Najim, and M. Chtourou, "Learning automata with continuous inputs and their application for multimodal functions optimization," *Int. J. Syst. Sci.*, vol. 27, no. 1, pp. 87–95, Jan. 1996.
[47] R. Ramesh, "Learning automata in pattern classification," M.E. thesis, Indian Inst. Sci., Bangalore, India, 1983.
[48] J. R. Sanders, presented at the 24th Annu. Meeting Joint Committee Stand. Educ. Eval., Oct. 1998. [Online]. Available: http://www.wmich.edu/evalctr/jc/Minutes/JCMinutes98.htm
[49] P. S. Sastry, "Systems of learning automata: Estimator algorithms applications," Ph.D. dissertation, Dept. Elect. Eng., Indian Inst. Sci., Bangalore, India, Jun. 1985.
[50] J. Self, "The defining characteristics of intelligent tutoring systems research: ITSs care, precisely," *Int. J. Artif. Intell. Educ.*, vol. 10, pp. 350–364, 1999.
[51] F. Seredynski, "Distributed scheduling using simple learning machines," *Eur. J. Oper. Res.*, vol. 107, no. 2, pp. 401–413, Jun. 1998.
[52] C. K. K. Tang and P. Mars, "Games of stochastic learning automata and adaptive signal processing," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 851–856, May/Jun. 1993.
[53] M. A. L. Thathachar and B. J. Oommen, "Discretized reward-inaction learning automata," *J. Cybern. Inf. Sci.*, vol. 2, no. 1, pp. 24–29, Spring 1979.
[54] M. A. L. Thathachar and P. S. Sastry, "A new approach to designing reinforcement schemes for learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 168–175, Jan./Feb. 1985.
[55] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: An overview," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 711–722, Dec. 2002.
[56] M. A. L. Thathachar and P. S. Sastry, *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. Boston, MA: Kluwer, 2003.
[57] M. L. Tsetlin, *Automaton Theory and the Modeling of Biological Systems*. New York: Academic, 1973.
[58] C. Ünsal, P. Kachroo, and J. S. Bay, "Multiple stochastic learning automata for vehicle path control in an automated highway system," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 29, no. 1, pp. 120–128, Jan. 1999.
[59] R. Winkels and J. Breuker, "What's in an ITS? a functional decomposition," in *New Directions for Intelligent Tutoring Systems*, E. Costa, Ed. Berlin, Germany: Springer-Verlag, 1990.
[60] Q. H. Wu, "Learning coordinated control of power systems using interconnected learning automata," *Int. J. Elect. Power Energy Syst.*, vol. 17, no. 2, pp. 91–99, Apr. 1995.

B. John Oommen (F'03) was born in Coonoor, India, on September 9, 1953. He received the B.Tech. degree from the Indian Institute of Technology, Madras, India, in 1975, the M.E. degree from the Indian Institute of Science, Bangalore, India, in 1977, and the M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, in 1979 and 1982, respectively.

In 1981, he joined the School of Computer Science, Carleton University, Ottawa, ON, Canada, where he is currently a Full Professor. Since July 2006, he has been awarded the honorary rank of *Chancellor's Professor*, which is a lifetime award from Carleton University. He is also an Adjunct Professor with the University of Agder, Grimstad, Norway. He is the author of more than 310 refereed journal and conference publications. He has been on the editorial board of *Pattern Recognition*. His research interests include automata learning, adaptive data structures, statistical and syntactic pattern recognition, stochastic algorithms, and partitioning algorithms.

Dr. Oommen is a Fellow of the International Association for Pattern Recognition. He has been on the editorial board of the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS.

M. Khaled Hashem received the B.Eng. and M.Eng. degrees from Ain Shams University, Cairo, Egypt, in 1986 and 1992, respectively, and the M.A. degree in public administration and the Ph.D. degree in computer science from Carleton University, Ottawa, ON, Canada, in 1993 and 2007, respectively.

He worked in the software development industry for 15 years. Currently, he is a Technical Architect Consultant for web and search technologies with MKH Consulting Inc., Ottawa. His current research interests include learning machines and automata, modeling of adaptive systems, and tutorial systems.