An Intelligent Tutoring System for Automata Theory: A Proposed Framework

Muneza Khatoon

Department of Computer Science & Information Technology (MANUU)
Maulana Azad National Urdu University, Hyderabad, Telangana
Monirakhatoon90@gmail.com

Abstract: With the advancement in technology and computers superseding humans in almost every territory of their life on this earth. The concept of Intelligent computers trying to replicate the human brain has evolved since the discovery of Artificial computers which work on the same concept as human brain which learn with change in experience over time. Intelligent tutors have been a boon for learners because it is humanely impossible for the human tutor to concentrate on each and every learner in the traditional classroom scenario. This paper summarizes the Intelligent tutoring systems available for the Undergraduate course of Automata Theory for Computer Science students and proposes a new solution based on the shortcomings in the existing systems.

Keyword: Intelligent Tutoring System, Automata Theory, FLAT.

Introduction

Research in education has led to the use of Computers for education which then evolved into delivering instruction online which was an idea that emerged from distance education and made learning possible anytime, anywhere and at whatever pace the learner wants to. The augmentation and advancement in the domains of Education and Artificial Intelligence is the work of researchers who worked in both the domains. Intelligent tutoring systems have been evolved from the traditional computer assisted instruction systems (CAI). When the CAI were developed to behave intelligently they were termed as Intelligent Computer Assisted Instruction (ICAI) which further were modified to be renamed as Intelligent tutoring systems (ITS). [11]

Intelligent tutoring systems are computer-based learning environments that incorporate computational modules on cognitive sciences, learning sciences, computational linguistics, artificial intelligence, mathematics, regular language and other fields that develop intelligent systems that are well-specified computationally. [1]

In [21], researchers presented an initial outline of the ITS requirements, the following are
(a) Knowledge of the domain (Expert model)
(b) knowledge of the learner (Student model)
(c) Knowledge of the teaching strategies (Tutor model)

The outline has not changed over the years, but a new module has been added to it.

The four model architecture [3] preserved the three models but added one more model, and now this architecture has become a standard for intelligent tutoring systems.

The following are the four models:
(a) Knowledge base
(b) Student Model
(c) Pedagogical model
(d) User Interface

Knowledge base: This component is similar to the domain model of the previous architectures. A tutor or coach consolidates declarative, procedural and metacognitive knowledge.

Student model: An internal model which defines the cognitive processes, meta-cognitive strategies and psychological attributes for each learner.
Pedagogical model: This module is similar to its counterpart in the other architectures. It uses a model of the learner’s present comprehension to select an efficient path through its knowledge representation to generate expert behavior by the learner.

User Interface: This module integrates three types of information: knowledge about patterns of interpretations and actions within dialogues; domain knowledge needed for communicating content and knowledge needed for communicating intent.

The Constructivist theory claims that knowledge’ is actively constructed by the student, not passively absorbed from textbooks and lectures. Since the construction builds recursively on knowledge (facts, ideas and beliefs) that the student already has, each student will construct an idiosyncratic version of knowledge. Teaching techniques derived from the theory of constructivism are supposed to be more successful than traditional techniques, because they implicitly address the inevitable process of knowledge construction. [24]

Application of the constructivist theory to the Computer Science education is difficult for the following reasons

- The construction of CS concepts using the constructivist theory is, because sensory data, from class must be integrated into a student’s existing framework that is too superficial.
- Frustration and the perception that computer science is ‘hard’ is due to the fact that models must be self-constructed from the ground up.
- Autodidactic programming experience is not necessarily correlated with success in academic CS studies. These students, like physics students, probably come with firmly held constructions that are not viable for academic studies.
- The reality feedback obtained by working on a computer can be discouraging to students who prefer a more reflective or social style of learning. [24]

The subject of FLAT is included in most of Computer Science Curricula since it discusses the fundamental theories of the discipline and help students understand the mathematical foundation of computation, its power and limitations [20] [10] [19].

Many Computer Science programs teach a course in Theory of Computation (also called Theoretical Computer Science, Formal Languages and Automata Theory(FLAT), etc.) These courses typically require students to design Finite Automata (FAs), Pushdown Automata(PDAs) and Turing Machines (TMs) to recognize languages of the relevant class.

Students often find mastering these foundational FLAT concepts challenging (as they can be very abstract, and thus overwhelming), and even boring (as there is a perception that the field is not only dated, but that it has little current applicability in the real world). [4] [7] [28] [30] [34] [40] This can lead to demotivated and frustrated students [31] who retain little of what they have learned [28]. Problem-solving skills, or rather the lack thereof, have been identified as one of the major contributes to the difficulties associated with building FAs [30]. This poor level of problem-solving ability can lead to the logic errors that were found to be prevalent. Furthermore, students could be using mechanical ‘plug-and-chug’ approaches without understanding the logic of the process. In response to this, several teaching approaches have been proposed to overcome these challenges, as well as to assist in the building of better mental models of FAs [5] [38]

- Constructivist teaching strategies and active learning: This includes the need to help students with immediate feedback at each stage of the process of building FAs so that they can better develop problem-solving skills (something that is very difficult to implement in a distance learning environment)
- Visualization tools: The point of these is to assist students in conceptualizing FAs and how they work by providing an alternative view, allowing them to experiment with the concepts by testing them on various input strings and receiving immediate feedback. Such approaches build on the principle that abstract models can become concrete when students can interact with them. [9]
- Link to current Computer Science applications: This could include using knowledge of programming as motivation for the exploration of FAs as well as the use of its application to real-world problems.
- Intelligent tutoring systems: These provide students with individualized assistance, tailoring the route through the required material based on how the student is progressing.
Students put off taking of the course till their senior semesters in an undergraduate course. Students have meager or scant knowledge about the theoretical and mathematical concepts of computer science. If the teaching material is not motivational enough, the students drown in the terminology [14]. The problem here is if the students do not understand the concepts, they may try rote learning [37].

“The concepts of automata theory have important use in designing and analyzing models of several hardware and software applications. These concepts are abstract in nature and hence used to be taught by a traditional lecture driven style, which is suitable for learners with reflective preferences” [26]

But Computer Engineering students usually tend to have strong active and sensing learning preferences [36].

RELATED WORK:
Following are the few notable examples of E-learning systems that have been used to support the teaching of Formal Languages and Automata Theory.

**FLUTE (Formal Languages and Automata Education)** [42] was introduced in the year 1998, it introduced the concepts of automata theory through an ITS, with examples and dependency graphs between similar topics. However, this system is obsolete and not in current use.

A Multimedia Learning System for smartphone for automata theory [43] was developed. It allows users to simulate the behavior of an automaton, watch lectures, read lecture notes, view presentations, take quiz and also a help section that serves as a manual.

Some approaches have been proposed to address the challenges faced in Online learning and the static acquiring of knowledge in [41] [32] [2] [13]. They have been designed and developed to allow students to build their own knowledge by means of experimentation. This kind of systems are often used in learning activities in which students solve practical problems, so that the knowledge acquisition includes the performance of an adequate number of exercises.

These systems assist students by proposing them exercises, taken from a repository, allowing them to submit their own solutions, correcting and providing them with meaningful feedback, and creating from scratch new exercises based on the student’s evolution. But these systems do not allow students to build their own exercises; they only enable them to work with a fixed set of exercises included in a repository. [18]

When we are motivated to teach, we find the visualization tools, or we create a visualization for students. When students are motivated to learn, they find the way to solutions. Animated storyboards have been proposed for the DFA, NFA and Turing machines [8]. The proposed solution has not been implemented and moreover no explanation is given as to how creating this storyboards is possible.

The first two components of the environment introduce the principle ideas of FSMs. One component presents a short movie-like introduction that welcomes the learners to the topic. The other one is a rich hypertext introduction to the basic concepts. Learners can navigate through the component and learn about the basic concepts. There are simulators provided for concepts of Finite automation and Turing machine and also introduces two games for the learners to apply the concepts they have learned. [25]

Hamada’s e-learning system for automata and theory of computation is composed of several components: an animated (movie-like) welcome component, a hypertext introduction to the theory of computation topics, a finite state machine (FSM) simulator, a Turing machine (TM) simulator, a self-assessment component, a chatting component for supporting online collaborative learning, and other three components showing automata visual examples such as video player, rice cooker, and tennis game. Novice automata learners find it difficult to grasp these comprehensive materials that were designed to meet all kinds of learning preferences. Learners do not know where they should start. [27]

[27] integrates another component to the above system which helps the learner to find their learning style which would help them to make the learning process enjoyable.
SELF-A-Pro, like other ITS, is composed of several modules that fully address the overall system objective. There are three modules that constitute the proposed system: the FLAT problem solver module, the linker module, and the interface module. Each module has been carefully designed to fulfill a set of specific tasks, working exactly with the information needed. The problem solver module is designed to take the exercise presented by the student (i.e. declarative knowledge), analyze the problem or the situation requirements, and then use the knowledge to solve it (i.e. execute procedural operations). After diagnosing a problem, the linker module will suggest the ways for fixing it (i.e. attempts to place conditional knowledge into specific execution). The interface module provides the student with methods to send the exercise and to create the most appropriate view of the system outputs in order to make easier the interaction and experimentation with them, according to the linker module suggestions. [18]

PROBLEM IDENTIFICATION:
All the existing systems, which assist the teaching of formal languages and automata theory assumes that the learner has some basic knowledge of the subject. In fact, none of the systems have been developed to support a student who does not understand the fundamental concepts. The proposed solutions in the existing systems seem to be the same simulators and emulators helping the learners try to implement the concepts and knowledge they have gained over time. Not a single system supports the acquiring of the knowledge of the concepts of automata theory for the novice learners. It is assumed in traditional teaching environment that the learner has already taken a course in Mathematical foundation of Computer Science but owing to the dryness of the concepts and little preference being given to the applicability of the concepts in the real life, students tend to forget these concepts or try rote learning to achieve the required grade for promotion to the next semester or course.
Most of the E-learning systems designed for this purpose use the recorded lectures, scanned textbooks as study materials and scanned handwritten notes which are deemed to be standard for the acquiring of knowledge in the traditional classroom based scenario which assumes that all learners have the same understanding capability and also due to the time limits a teacher is unable to concentrate on each and every learner.
Most of the ITS developed in computer science are independent of the learning module and concentrate on providing personalized solutions according to the evaluation done when learners solve problem questions related to the knowledge gained. But in reality this doesn’t work well for many of the learners, because when a learner has to learn in the traditional manner and just use the system to try out problem, they get annoyed for having to look back to the notes and textbooks for notes and study material.

PROPOSED SOLUTION:
The existing architectures of ITS can be adopted to develop an ITS for Formal Languages and Automata Theory which will evolve from the four model architecture of ITS which is advocated by many researchers in [16] [17] [39]. The four model architecture consist of the following modules

- Knowledge base
- Student Model
- Pedagogical model
- User Interface
The knowledge base of the ITS for automata theory will consist of learning objects which can be mapped to learning preferences and needs that can be deduced by using the multiple intelligences characteristic of the learner. Since the knowledge base consist of domain knowledge, constraints, rules, problems related to the subject, utmost care should be taken when designing the elements. All the other updates when the learner model is changed should be associated to the knowledge base.
The various needs, preferences and strengths of the eight Intelligences can be seen below which is taken from [12]

<table>
<thead>
<tr>
<th>Intelligence Area</th>
<th>Strengths</th>
<th>Preferences</th>
<th>Needs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbal / Linguistic</td>
<td>Writing, reading, memorizing dates, thinking in words, telling stories</td>
<td>Write, read, tell stories, talk, memorize, work at solving puzzles</td>
<td>Books, tapes, paper diaries, writing tools, dialogue, discussion, debated, stories, etc.</td>
</tr>
<tr>
<td>Mathematical / Logical</td>
<td>Math, logic, problem-solving, reasoning, patterns</td>
<td>Question, work with numbers, experiment, solve problems</td>
<td>Things to think about and explore, science materials, manipulative,</td>
</tr>
<tr>
<td>Visual / Spatial</td>
<td>Maps, reading charts, drawing, mazes, puzzles, imagining things, visualization</td>
<td>Draw, build, design, create, daydream, look at pictures</td>
<td>video, movies, slides, art, imagination games, mazes, puzzles, illustrated book</td>
</tr>
<tr>
<td>Bodily / Kinesthetic</td>
<td>Athletics, dancing, crafts, using tools, acting</td>
<td>Move around, touch and talk, body language</td>
<td>Things to build, movement, tactile experiences, hands-on learning, etc.</td>
</tr>
<tr>
<td>Musical</td>
<td>Picking up sounds, remembering melodies, rhythms, singing</td>
<td>Sing, play an instrument, listen to music, hum</td>
<td>Sing-along time, musical instruments, etc.</td>
</tr>
<tr>
<td>Interpersonal</td>
<td>Leading, organizing, understanding people, communicating, resolving conflicts, selling</td>
<td>Talk to people, have friends, join groups</td>
<td>group games,</td>
</tr>
<tr>
<td>Intrapersonal</td>
<td>Recognizing strengths and weaknesses, setting goals, understanding self</td>
<td>Work alone, reflect pursue interests</td>
<td>self-paced projects, choices</td>
</tr>
<tr>
<td>Naturalistic</td>
<td>Understanding nature, making distinctions, identifying flora and fauna</td>
<td>Be involved with nature, make distinctions</td>
<td>Order, same/different, connections to real life and science issues, patterns</td>
</tr>
</tbody>
</table>

Table: Strengths, preferences and needs of the Eight Intelligences

Although some of these intelligences cannot be utilized in the teaching of automata theory, many of them can still be applied. Future work in this area would be identifying learning objects for each type of Intelligence in automata theory concepts.

**Student Model:**
The student model is the internal model, that will maintain the following details about the learner

- Current emotional state, that can be deduced by his actions while using the system as in [35]
- Current knowledge level of the subject by asking questions about the preliminary topics in the subject. Although, the initial assumption would be similar to that of a blank slate, which is different from the existing systems.
- Type of intelligence, which is deduced by using a questionnaire based upon the Gardner’s theory of multiple intelligence [15] which will help to create the learning paths accordingly.
• Retention and recall capabilities which will help the knowledge base update the learner model accordingly and generate expert decisions.

• Performance and status of course completion on a scale of 0-100 where a zero indicates none of the bloom’s learning objectives [22] have been achieved and a 100 indicates the fulfillment of principles of the constructivist theory [6] where in the learner constructs new knowledge.

**Pedagogical model:**
This model will use the learner profile and matches it to the many learning paths available and finds the most efficient path, so that the learner progresses in such a manner that generates expert behavior.

**User interface:**
The user interface will integrate the three models and communicate the knowledge and intent to the learner in such a manner that it will not overwhelm the learner with excess information and at the same time help to maintain the learning pace and provides redirection to other links using adaptive hypermedia [33].

**Evaluation module:**
The ITS would also contain an evaluation module which will test the learner knowledge at regular intervals to check the learning outcomes achievement and these can be treated as milestones for the learner. The most basic types of evaluations would include quiz based multiple choice questions to the intermediate ones of using simulators and emulators to solve problems. The advanced evaluation would be creating, solving games and puzzles by using the concepts learned. It would also include applying the concepts to some real life situations thereby increasing the understandability.

Here the knowledge base will help the students to solve problems by identifying the wrong solutions and giving hints of the possible correct solutions similar to a human tutor.

Depending upon the results of the evaluation module, the student model will update the learners’ profiles at regular intervals and also update the available learning paths.

**Conclusion and future work:**
In this paper, problems have been identified in the existing intelligent tutoring systems for automata theory and a solution is proposed based upon the findings which tries to solve the existing problems.

Future work will include the design and development of such a system and report the results of evaluation by comparing its use with the traditional classroom approach. If the results are satisfactory, this system design and behavior can be generalized to support other theoretical subjects such as Mathematical Foundation of computer science, digital logic, compiler designs etc.

**References**


