Enhancing Theory of Computation Teaching Through Integration with other Courses

Mukta Goyal, Shelly Sachdeva
Department of Computer Science
Jaypee Institute of Information Technology University, Noida, India
mukta.goyal@jiit.ac.in,shelly.sachdeva@jiit.ac.in

Abstract:-Teaching and learning, theory of computation is a challenging task from both the perspective since it involves the abstract notions and mathematical background. Moreover, representation of its relevance to other computer science courses is inadequate. Due to the importance of closer interaction between theoretical and applied scientists, motivation among students is required, which may be done by increasing interest and enjoyment. This paper aims towards introducing different approaches for making the course interactive and realistic by integrating it with other courses learnt in previous semesters and current semester of engineering. Consequently the fascination towards teaching increases and students have active participation in the learning of theory of computation (TOC) course.


I. INTRODUCTION

Theory of computation course is core in computer science ACM curriculum. It has not gained much existence in other branches viz. Information systems, Software engineering, Information Technology in ACM Curriculum. Automata Theory is important because it is used for modelling interesting kinds of hardware and software. Hardware designers have understood its importance but it’s an irony that software developers are unaffected with the immense power of this subject. Related books are more inclined towards theoretical concepts rather than practical approach to teach theory of computation. The traditional teaching for theory of computation neither includes the practical applications nor addresses the real-life problems of computer science [10]. Various difficulties in teaching automata are addressed such as material is dated and of little use, requires mathematical background and purely theoretical [2,3].

Engineering students prefer active learning style instead of a passive lecture-driven style [1]. Sigman proposes Moore method [14] to engage students in theory of computation course. Visualization tools are incorporated [3,7] to overcome difficulties. Visualization helps the unseen to be made perceptible, introduce the concept and process into a well organized manner. Paper [2] inculcated a constructivist approach and powerful use of technology to achieve significant learning of course.

However, what remains unaddressed is the relationship between automata and other subject areas learnt during the previous and current semester. The authors have attempted a practical approach to build interest among students while delivering the course. Integrating various courses is a superior means of making computer professionals. We tried to create student satisfaction with its relevance to Computer Science. We facilitated the students to overcome the difficulty of constructing automata and verifying their functionality by a number of exercises, examples and illustrations.

II. TEACHING METHODOLOGY OF COURSE

To bridge the gap between theory and its practical scientific applications, we have proposed different strategies in paper to make theory of computation alive which are as follows:\n1. Vertically downwards and upwards integration of subjects such as Introduction to Computers and Programming, Data Structures, Database Systems, Object-Oriented Programming, NLP and compilers with the automata course.
2. Enriching automata course by inclusion of problems from horizontal subjects such as web application engineering, microprocessors, software engineering.
3. The students were motivated to study a research paper related to the course.
4. Active learning of the students through software tools so that they should have” hands on” component of theory of computation.

The points 1 to 4 are described in the following subsections.

A. Vertical Integration of Subjects

The authors have amalgamated examples from various subjects such as Programming Languages, Data structures, Database Systems, Object Oriented Programming languages, Compilers and Natural Language Processing, some of which are illustrated as follows:

1) Programming Languages: We have introduced the control statements (for, while and do-while) for Programming Languages (PL) through the (DFA-Deterministic Finite Automata) state machines which increases the better understanding among students. For example:

   a) A variable name is of the form [A-Za-z] [A-Za-z0-9]*. Construct an NFA to recognize it automatically.
b) Write a regular expression to match employee code that start with 4 digits and end with two uppercase letters.
c) Draw a finite state automaton which removes /* */ and // style comments.

2) Data Structures: Data structures (DS) problems such as linear search, stack, queue, list and binary search could be integrated with the mealy machine. For example,
a) Design a Push down Automata that changes infix arithmetic expressions to postfix expressions.
b) Draw a Turing machine for rat in the maze problem.
c) The mealy machine for stack is illustrated in the fig 2 as shown below:

3) Database Management systems (DBMS): we correlated the examples from Data Definition Language and Data manipulation Languages. For example,
a) Draw a DFA for the creation of table in SQL.
b) Write a grammar for parsing SQL queries.

4. Object-Oriented Programming and Systems (OOPS): The authors have suggested series of examples helping students to cope with the problems.

5. Natural Language Processing (NLP): This course was integrated in recent year during course evolution as shown in table1.The authors linked the examples such as homophones and oxymoron’s.

B. Enriching Automata Course by Inclusion of Problems from Horizontal Subjects

1. Web Application Engineering (WAE): We have included the examples from JavaScript, HTML, and XML. We can check the validity of the URL’s, E-mail address, form fields like date of birth, account no etc. through the regular expressions. Integrating the regular expression with the web applications makes student to understand better and realize the real utility [11]. For example,
a) Write a grammar that validates email addresses of the form: string0@string1.string2.string3.edu where each string is composed of lowercase letters.
b) Write a regular expression for validating the code of a simple HTML document which displays “Welcome to the web course”.

2) Microprocessors: Examples such as Traffic Light Controller, Microwave Oven, washing Machine, and vending machine were discussed which trained the student to put in sustained and disciplined work over the entire period of course.

3) Software engineering (SE): The authors directed students to make automata for those assignments for which they have to make UML diagrams. It really worked in the classrooms as we lectured and discussed the topics.

4) Many real life examples such as context-aware mobile phone, digital audio player (fig 4), sprite, water level controller, and counter strike problem were included.

© 2009 ACEEE
DOI: 01.IJRTET.01.02.151
In fig 4. TA denotes Touch pad activated, TO (Touch pad timeout), FF (Fast forward), NT (Next track), and PT (Previous track).

C. Motivation to Study a Research Paper
There was a tremendous increase in the growth of learning TOC for average students after reading the research papers related to TOC course. Also, the time lap which they were taking to draw different models (such as finite automata, pushdown automata, Turing machine) for a given problem was decreased. It enhances the problem based learning of the student. Problem based learning method develop the ability in students to think what they are studying, why they are studying and how they are applying to the real life applications [12]. This strategy helped students to apply fundamental principles to solve problems. Students take what they have learned, and under the facilitation of an instructor, define, design, and develop a solution to a problem in their interest areas.

D. Active Learning through Software Tools
Various formal language and automata theory tools were analyzed through practical experience and recommended to integrate into an established curriculum [5,7]. Increasing visualization and interaction in the course is enhanced through the popular software tools [3,6,8]. JFLAP is a powerful learning tool that enables students to design and simulate various automata [8]. The authors experimented for five years with JFLAP during classroom teaching to subsequently increase access to rigorous learning of the course. Integrating the lessons with the simulators provide students effective, interactive and enjoyable environment. Various examples such as homophones, stacks, washing machine etc were validated through JFLAP by our students.

III. EXPERIMENTAL RESULTS
The authors have incorporated previous knowledge, current knowledge and experimented with the students taking TOC course to enhance active indulging of the students in theory of computation course. The course was evolved by various iterations during subsequent years as shown in table 1. The course is taught in the odd semester of 2008 at Jaypee Institute of Information Technology University. We divided the total strength of students (210) into two lecture sections viz, LS1 and LS2 taken by two different instructors. In LS1 the first author emphasized to teach the course through integration of other courses as shown in table 1 for 2008 whereas in LS2 the other instructor followed the traditional approach (without integration). There has been approximately 70% enhancement in the performance evaluation of LS1 students as shown in the table 2. The performance evaluation was done through open book exams and challenging home assignments related to various domains. Students learn to teach themselves and their success builds confidence in their abilities. They, working through the problems (from other courses) discovered the material central to the course. Thus, they assimilate an attitude to work differently from students doing traditional course.

TABLE 1. COURSE EVOLUTION BY INTEGRATING

<table>
<thead>
<tr>
<th>YEAR</th>
<th>REAL LIFE EXAM PLES</th>
<th>TOOL (JFLAP)</th>
<th>MICRO PROCESSOR</th>
<th>RESEARCH PAPER</th>
<th>DSP</th>
<th>OPERATING SYSTEMS</th>
<th>WEB Services</th>
<th>DATABASES</th>
<th>NSLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DIFFERENT COURSES DURING VARIOUS YEARS

TABLE 2. RESULTS OF EVALUATION

<table>
<thead>
<tr>
<th>APPROACH</th>
<th>AVERAGE</th>
<th>STANDARD DEVIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGRATED</td>
<td>36</td>
<td>15</td>
</tr>
<tr>
<td>TRADITIONAL</td>
<td>21</td>
<td>14.9</td>
</tr>
</tbody>
</table>
IV. SUMMARY AND CONCLUSIONS

There is a lot of buzz among computer science teachers to teach theory of computation using integration with other courses. But other questions are still there. Can we really teach the important automata concepts in a course focused on problems from other courses? What teaching and development tools are available? Will it be successful in classrooms? Teaching through integration of subjects had shown how students matured in their understanding of various concepts during their learning on the course. Significant improvements were achieved in student’s perception of learning automata, especially those who were applying theoretical concepts mechanically rather than mounting significant learning. Problem based learning method with continuous assessment provide students with better understanding of subject and the information he learnt which is directly in use to the real life and, remain active for longer period. Students appreciated the relevance of theory of computation as new knowledge was added to an existing knowledge structure.

ACKNOWLEDGMENT

The authors wish to thank Prof. Sanjay Goel from Jaypee Institute for his valuable guidance during the subsequent years.

REFERENCES