ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #27866

Engaging Students with Multiple Representations of Mathematical Models

Dr. Pradip Peter Dey, National University

Dr. Pradip Peter Dey has more than 20 years of experience in Computer Science research and education. His university teaching and professional experience emphasizes mathematical modeling, information extraction, syntax and semantics of natural language, wireless apps and knowledge representation. He has done an M.S.E. in Computer and Information Science and an interdisciplinary Ph.D. from University of Pennsylvania.

Engaging Students with Multiple Representations of Mathematical Models

Pradip Peter Dey
National University
School of Engineering and Computing,
3678 Aero Court, San Diego, CA 92123, USA

ABSTRACT

Mathematical models of computation such as Turing Machines, Pushdown Automata and Finite Automata can be adequately represented in a tabular form. However, students tend to engage and participate actively when multiple representations including transition graphs, tabular forms and coded strings are used for discussing pertinent issues. These representations can be mutually translated to each other without any loss of formal properties. This paper reports on experiments with multiple representations of Turing Machines for modeling dynamic aspects. Student participation was found to be significantly higher when graphical representations were given using transition graphs in addition to other representations. Most learners liked multiple representations of Turing Machines even when the standard tabular representation was good enough for explaining fundamental aspects of information processing. 87.5% learners preferred multiple representations of a Turing Machine to one representation in a web-based learning environment. The representations of the Turing Machine were used as a supplement to introductory onsite math foundations of computer science course on the following website: http://www.asethome.org/mathfoundations/tmd/. Applications of Turing Machines in other fields were discussed using different representations. Software design tools based on statecharts proposed by David Harel have been very popular for modeling dynamic aspects of software. Statecharts are simplified Turing Machines presented in a graphical notation that is appropriate for explaining software design and development features in an intuitive way.

INTRODUCTION

Except in highly selective colleges, most engineering and computing programs struggle with mathematics readiness problems and offer developmental (or remedial) math courses. Various studies have examined declining mathematics readiness of first-year college students and suggested different types of remedies [2, 10, 40, 42]. Most colleges initially place the underprepared students into some developmental math courses followed by appropriate regular college courses. With effective teaching strategies, developmental math course success rates can be improved considerably diverse groups of students, if students are engaged with mathematical concepts in structurally appropriate motivational context. "Education researchers are ultimately interested in how to structure the educational context to maximize student learning outcomes." [43] Most teachers have a genuine interest in creating an engaged classroom. Visual representations of mathematical concepts help in engaging students with complex ideas about the conceptual issues [1, 2, 42], which are otherwise avoided by students. Turing Machines are mathematical models of computation, which present some special challenges in teaching-learning environments, because in addition to visual representations, some coded representations are also needed for important tasks such as proofs [6, 13, 37]. Some of the multiple representations of Turing Machines are not as intuitive as visual representations and they involve additional work [6, 14]. Engaging students with

multiple representations of Turing Machines can be planned in a way that provides an initial intuitive understanding of the concepts through visual representation, and then more complex ideas can be built with other representations. That is, complex concepts need to be developed in a progressive manner after discussing related easier concepts with intuitive representations.

A Turing machine is an abstract machine proposed by the mathematician Alan Turing in 1936 for defining fundamental aspects of computation [41]. The machine can elegantly present an algorithmic solution to any computable problem. Descriptions of Turing machines are included along with other automata such as Linear Bounded Automata, Pushdown Automata and Finite Automata, in modern automata theory textbooks [6, 11, 14, 16, 19, 21, 22, 24, 32, 36, 37]. Turing machines can be used for modeling dynamic aspects of computation at various levels of abstraction. Dynamic aspects of software can be modeled with statecharts which are simplified Turing machines proposed by David Harel [12, 13]. Turing machines define the most powerful automata class for processing the most complex sets or languages, namely, recursively enumerable sets. The class of recursively enumerable sets properly includes all other computable sets [6, 14, 37]. Figure 1 presents the relationship among automata and the languages (or sets) accepted by them in a simplified way. It is drawn following Cohen [6], where MATHISON is defined as a set of Turing machine codes that are accepted by the corresponding Turing machines they represent.

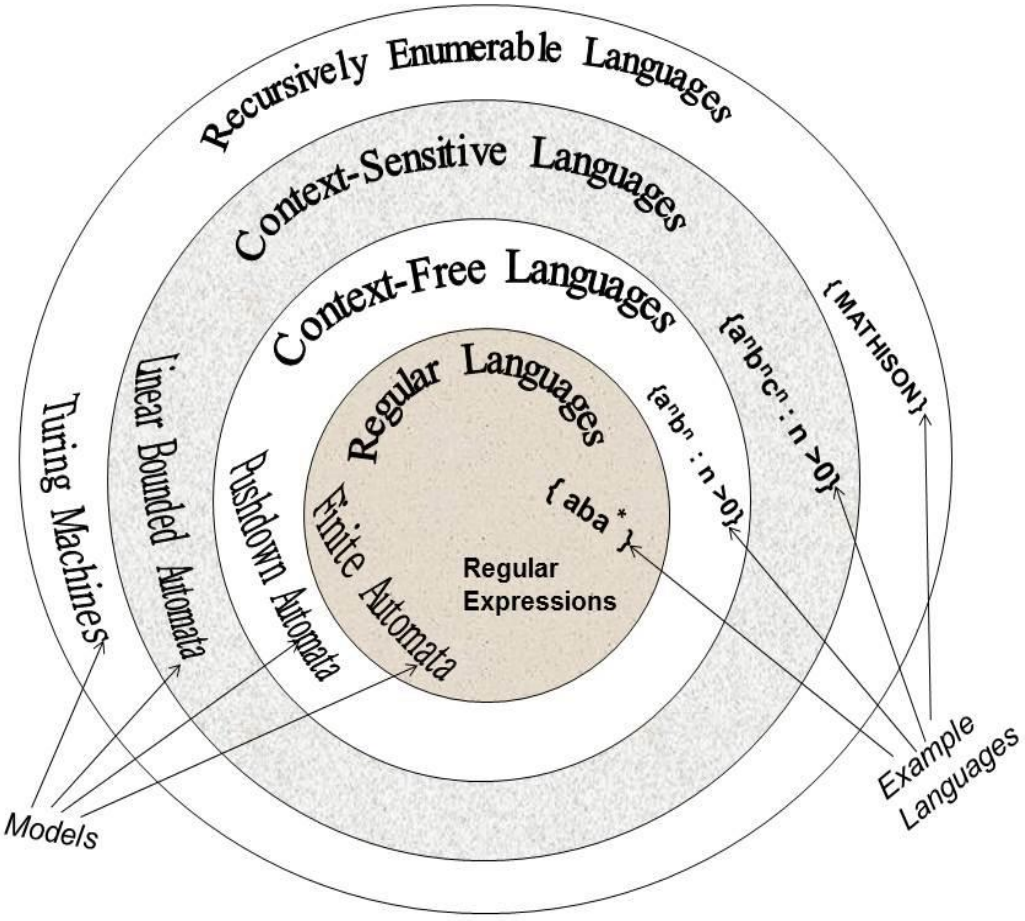

Figure 1: Relationship among automata classes and language classes

In order to prove that MATHISON is recursively enumerable, students need to understand that any Turing machine can be encoded into a string, and some Turing machines accept their own strings and others do not accept their strings. The set of Turing machine codes that are not accepted by their corresponding Turing machines is proven to be not computable. Linear Bounded Automata are Turing machines with some restrictions on the use of memory; they are designed for processing Context-Sensitive languages. Pushdown Automata are defined to have exactly one stack and they are non-deterministic unless otherwise explicitly stated. Pushdown Automata are acceptors of the class of context-free languages. They are less powerful than Turing machines; they cannot accept non-context-free languages. Programming languages are compiled using Context-Free grammars in processing models characterized by Pushdown Automata. Finite Automata accept a proper subset of context-free languages called regular languages denoted by regular expressions. Finite Automata are deterministic unless otherwise explicitly stated. Non-deterministic Finite Automata are equivalent to Finite Automata in the sense that they accept the same sets defined by regular expressions.

This study examines presentation of a Turing machine in various forms in order to understand the effects of presentations for understanding student engagement and student preferences. A Turing machine can be presented in a table, or drawn in a graphical form (transition graph), or coded in a string [6, 14, 37]. These three representations are equivalent in characterizing the Turing machines. Any of these representations can be translated to any other without loss of essential information. An important question is: Is there a presentation that is better in helping learners understand the mathematical model? This study suggests that understanding their essential features and critical thinking about them, and especially student engagement are promoted when multiple representations (including graphical, tabular, and coded representations) are used for discussing the issues. That is, their effectiveness in the teaching learning environments is enhanced by multiple representations. Turing machines are often presented in all three forms; however, the coded form of Turing machines is used primarily for the purpose of various proofs [6, 14, 37]. Following the pioneering work of Rodger in the area of visualization of automata [33-35], we consider the role of presentation forms. In order to engage students, we initially present relatively simple examples of Turing machines in visual forms. After some initial discussion of issues, we use other representations and establish relationships among the representations. We present visualization of a Turing machine in the form of a transition graph for {aba*} along with a tabular representation and an encoded string representation at: http://www.asethome.org/mathfoundations/tmd/ in order to study student engagement and learner preferences. The site was used as a supplement to the class notes.

BACKGROUND

There is a wide variation among humans for understanding engineering issues and scientific problems. Humans may vary widely in the speed and manner with which they acquire new information and ideas and the confidence with which they process and use them [5]. Humans are often categorized based on their preferred styles and attempts have been made to classify learners as verbal learners, visual learners etc. Ormrod [26] suggests that some cognitive styles and dispositions seem to influence how and what students learn and certain "students seem to learn better when information is presented through words (verbal learners), whereas others seem to learn better when it's presented through pictures (visual learners)'' (p. 160). Some of the popular

learning style models include Kolb's Learning Styles Inventory [17-18], Honey and Mumford's (1992) Learning Styles Questionnaire [15] and the Dunn and Dunn learning-styles model [7-9]. These and other models were critically reviewed by Coffield, Moseley, Hall and Ecclestone [5], Kozhevnikov [20], Sternberg, Grigorenko, & Zhang [39], Pashler, McDaniel, Rohrer, and Bjork [28]. Learning is a complex process which is gradually being explored utilizing new research in a number of disciplines including neuroscience, psychology, media, artificial intelligence, cognitive science, sociology and their overlapped areas. We present the results on preferred representations by individual learners without claiming learning attainments. It is assumed that sensory data are more effectively perceived though multiple modalities than just one modality [4, 5, 42].

MULTIPLE REPRESENTATIONS OF TURING MACHINES

Alan Turing provided a mathematical definition of computation in 1936 [41]. In the same year, Emil Post independently developed algorithm machines that have come to be known as Post machines [29]. Turing machines and Post machines are proven to be equivalent and their theory developed in 1930s and 1940s has provided the foundation of the theory of computation. Turing machines are the most popular models for recursively enumerable sets mentioned above. Following Cohen [6], we define Turing machines as follows.

A Turing machine is composed of six components:

1. An alphabet, Σ, which is a finite non-empty set of symbols from which input is built.
2. A READ/WRITE TAPE, divided into a sequence of numbered cells; each of the cells contains one character or a blank, Δ. The input is presented to the machine one letter per cell beginning in the leftmost cell. The rest of the tape is initially filled with blanks (Greek delta symbol).
3. A TAPE HEAD points to the current letter being read from the READ/WRITE TAPE. It can in one step read the contents of the READ/WRITE TAPE, write a symbol on the tape and move left or right one cell.
4. An alphabet, Γ of symbols for the READ/WRITE TAPE. This can include symbols of Σ.
5. A finite set of states including one start state from which execution of the machine begins and some (may be none) HALT states that cause execution to terminate when entered.
6. A set of transitions from state to state where each transition has three elements:

(Read-Letter, Write-Letter, Move)

The first element of the triplet is a letter read by the TAPE HEAD of the machine from the current cell of the tape (as shown in Figure 2). The second element is a letter written on the tape on the same cell where the first element was read from. The third element, Move, tells the TAPE HEAD whether to move one cell right, R, or one cell left, L. The HALT state cannot have any outgoing transition. To terminate execution on certain input successfully, the machine must be led to a HALT state. The input is then said to be accepted by the machine.

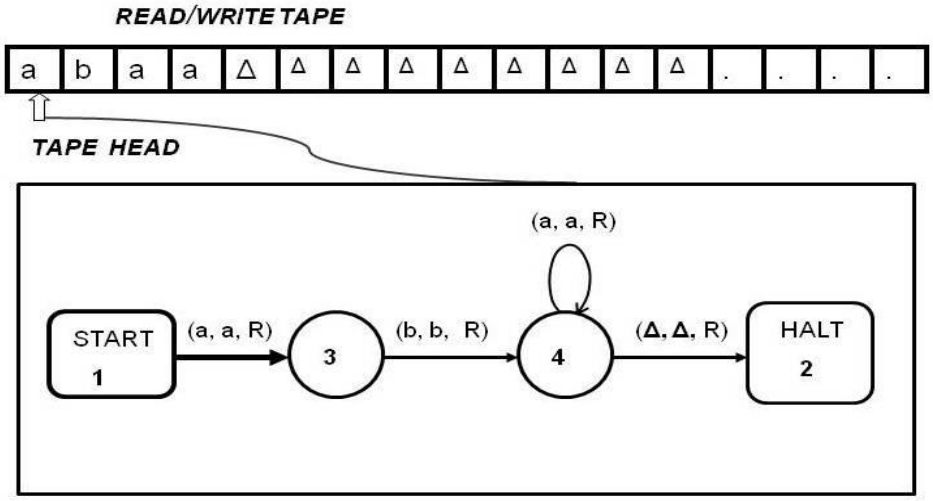

Figure 2: A Turing machine for aba* with input abaa

In order to clarify aspects of the definition, consider the example Turing machine presented in Figure 2 with the input **abaa.** Each letter of the input is presented in one cell starting from the leftmost cell. The rest of the tape is blank. The Turing machine given in Figure 2 is designed to accept every string of the following set: {ab, aba, abaa, abaaa, abaaaa, abaaaaa, **. . .** }. Such a set is called a language. A language is a set of strings. This language is usually abbreviated as **aba*** which is a regular expression. The star in this string, after **a** is known as the Kleene star which means zero or more of **a**'s. The regular expression **aba*** means one **a** followed by one **b** followed by zero or more **a'**s. This language denoted by the regular expression **aba*** can be written as: L_1 = aba* = { ab, aba, abaa, abaaa, abaaaa, abaaaaa, abaaaaaa, **. . .** }

Assume that the Turing machine of Figure 2 has just begun to process the input by starting at the start state (which is assigned number 1 for coding purposes). The Turing machine then takes the first transition from the start state to state 3; this transition is marked by the triplet (a, a, R). According to this triplet, the machine reads the first symbol **a** from the leftmost cell of the tape, writes back **a** on the same cell and moves right. The result is shown in Figure 3, where the machine is scanning the second symbol, **b**, from the tape (after scanning the first symbol).

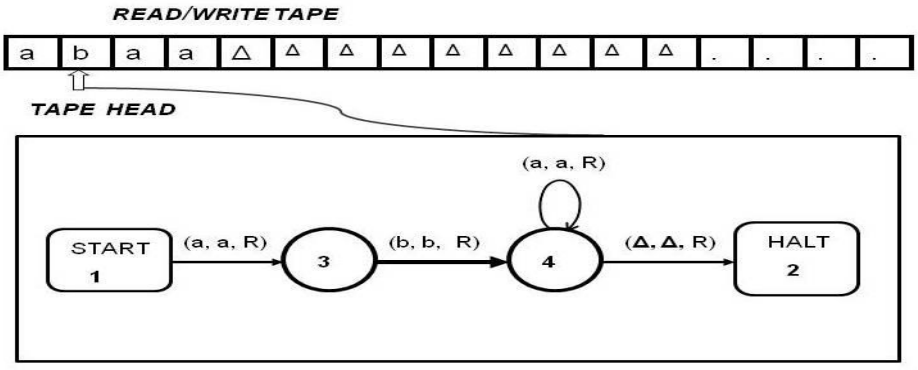

Figure 3: A Turing machine for aba* which has just scanned the first symbol

Next, the machine, from state 3, taking the transition marked by (b, b, R) reads the symbol **b** from the second cell of the tape and writes back **b** on the same cell and moves right on the tape. Taking this transition the machine reaches state 4, as shown in Figure 4.

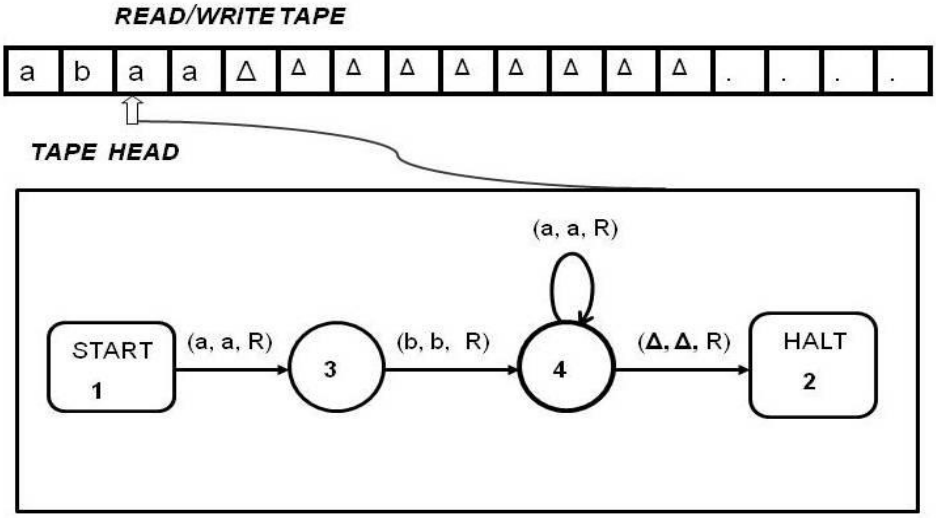

Figure 4: A Turing machine for **aba*** which has just scanned the 2nd symbol from the input

In the next step, the machine, from state 4, taking the loop transition marked by (a, a, R) reads the symbol **a,** from the third cell of the tape, and writes back **a** on the same cell and moves right on the tape. Taking this transition the machine comes back to state 4 as shown in Figure 5.

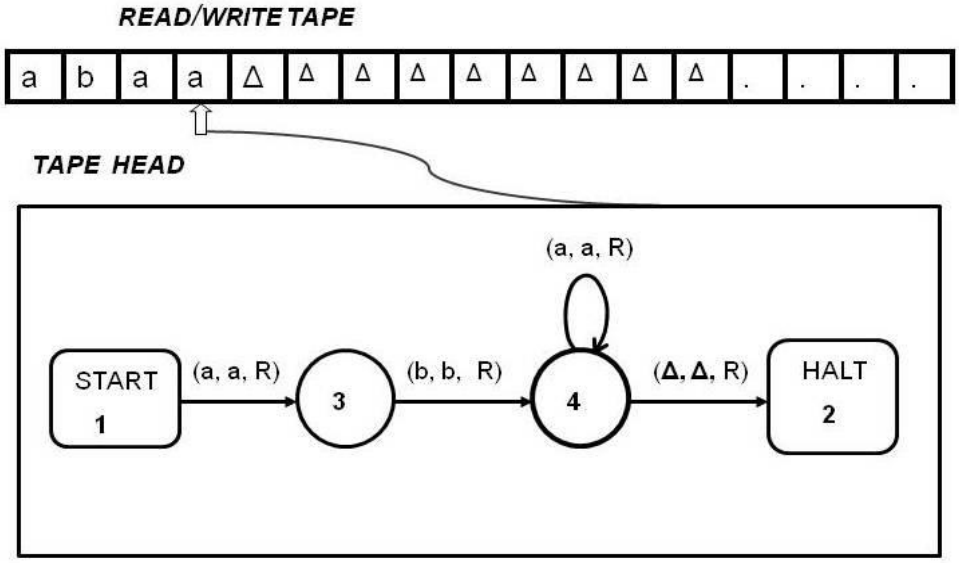

Figure 5. A Turing machine for aba* which has just scanned the 4th symbol from the input

Next, the machine, from state 4, taking the same loop transition marked by (a, a, R) comes back to state 4. In the next step, taking the transition marked by (Δ, Δ, R) from state 4, the machine reads Δ from the fifth cell of the tape, writes back Δ on the same cell and moves right on the

tape. Taking this transition the machine reaches the HALT state as shown in Figure 6. The machine accepts the input, **abaa**, since it reaches the HALT state after traversal of transitions from the start state.

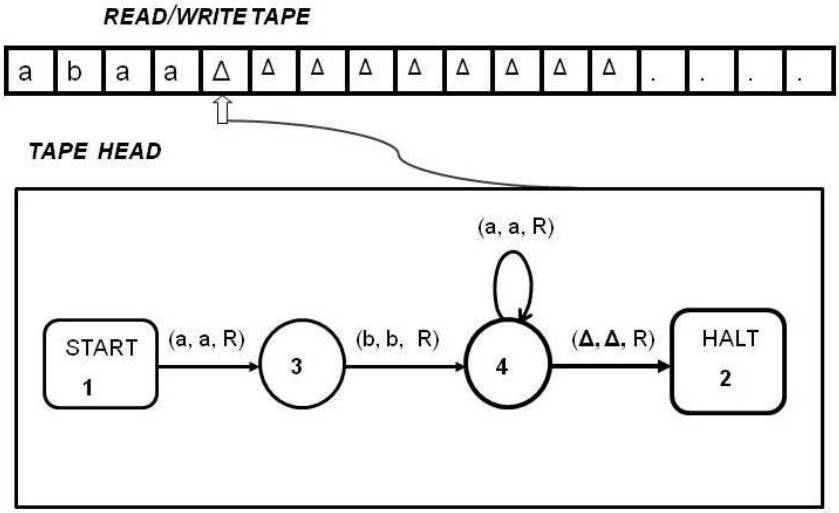

Figure 6. A Turing machine for **aba*** which is scanning the null symbol from the tape

From the Turing machine instances of this section, it is obvious that the machine is designed to accept every string of the set, {ab, aba, abaa, abaaa, abaaaa, abaaaaa, abaaaaaa, **. . .** } = aba* which is a regular language. The above Turing Machine for aba* is represented in Table 1 [6] given below:

FROM	TO	READ	WRITE	MOVE
1	3	a	a	R
3	4	b	b	R
4	4	a	a	R
4	2	Δ	Δ	R

Table 1 The Turing Machine for aba* is represented in a tabular form

An animated dynamic visualization of the Turing machine for aba* is provided on the following website: www.asethome.org/mathfoundations/tmd/. The website is used as a supplement for an introductory onsite course on automata theory where the representations are explained. Towards the end of the course, students were given a brief survey for assessing their attitude about multiple representations of the Turing machine. 21 out of 24 (87.5%) learners answered the following question affirmatively: "Do you prefer multiple representations over one representation?" Three students preferred visual representation to multiple representations. Nobody indicated any special preference for the encoded representation of the Turing machine, abaaabaaaabaaabaaaababababbaaaabaaaabaaaabaaaabaabbabab, developed following Cohen [6]. However, every student participated in the required quizzes and exams where encoding of Turing machines into strings and their utilization in proofs were necessary. In addition, statechart diagrams were practiced for representing dynamic aspects of software following David Harel's

suggestions [12, 13]. The statechart diagrams closely resemble transition graph representation of Turing machines and needed only minor adjustments for abstractions of software development concepts [3, 12, 13, 30, 38]. Statechart diagrams are widely used in software engineering. Students enthusiastically participated in the discussions of the relationships among the various representations and their usefulness in performing various tasks. Most of the anonymous written student comments were positive. Two of the comments are reproduced below:

(1) "The professor was very good at explaining things that were hard to understand. I learned so much information from this teacher."
(2) "Awesome teacher."

In most classes, some computational problems were discussed for which algorithmic solutions need to be developed. It was not a flipped classroom [43]; however, discussions of problem solving were used for promoting student participation. One of the written comments received in the process of peer evaluation was " . . . Within 90 minutes, almost every student came up to the board to try to solve a problem. The students were actively engaged in the Problem Based Learning (PBL) approach. It was very effective. . . This course represented critical thinking at its best. Students were presented with a problem, asked to present a solution, and then defend their solution. If their solution was not correct, the students had the opportunity to correct their solution in a very non-threatening environment. . . . Professor Dey is clearly a content matter expert. This course was certainly one in which he is well-qualified to teach. He delivered the material with a gentle approach that engaged a very diverse group of students. . . . "

In order to serve a diverse group of students, some socio-psychological factors need to be considered and some motivational strategies need to be used [43]. There is a general consensus that students need to feel that they belong to the learning environment and if they try they can be successful [43, 44, 45]. There was no dropout in the class, because steps were taken from the very beginning to help students who were not prepared appropriately for the course and some motivational interventions were initiated in a student-centered environment [43, 44, 45, 47]. Due to active participation of all stakeholders including faculty, National University has achieved impressive results in serving minorities in California. National University provides more master's degrees in all disciplines to African Americans than any other college or university in California [46]. National University also leads California in granting Master's Degrees in all Disciplines to Hispanics [46]. Unless we are dealing with special applications, we would agree with one of the most famous mathematicians, David Hilbert, who stated "Mathematics knows no races or geographic boundaries; for mathematics, the cultural world is one country."

CONCLUSION

Turing machines are useful for modeling the dynamic aspects of computable problems. Software development relies on modeling the software in various levels of abstraction, including the design level. Design tools based on statecharts [12, 13] have been very useful for modeling dynamic aspects of software. Statecharts are simplified Turing machines presented in a notation that is appropriate for software features in an intuitive way, and they are also good for demonstrating practical use of Turing machines in various application domains [3, 12, 13, 30, 38]. After reviewing the current study, a more sophisticated research can be suggested for engaging students in the learning process. It is reasonable to assume that more specific data can be collected in the future for providing definitive evidence about enhanced student learning from engaging students with multiple representations.

Learning problems with other mathematical models also can be investigated in the future. The multiple representation method advocated in this paper may not always work in every learning environment; it worked for a diverse group of students in National University, because the learning environment was designed to be inclusive. Students should feel that they belong to the environment. Students should quickly believe that their teachers care about them and their learning.

ACKNOWLEDGEMENT: The author is thankful to the anonymous 2019 ASEE-SW conference reviewers for their comments and suggestions made during the review process. Thanks also to John Cicero, Jodi Reeves, Ronald P. Uhling, Bhaskar Raj Sinha, Hassan Badkoobehi, Laith Al Any, Jay Dey, and many others for their suggestions, encouragements, cooperation, and/or help during the preparation of this paper.

REFERENCES

[1] Ainsworth, S., (2006). DeFT: A conceptual framework for considering learning with multiple representations. Learning and Instruction, 16, 183-198.
[2] Bailey, T., Jenkins, D., & Leinbach, T. (2005). Community college low-income and minority student completion study: Descriptive statistics from the 1992 high school cohort. New York: Columbia University, Teachers College, Community College Research Center.
[3] Braude, E., Bernstein, M. *Software Engineering: Modern Approaches*, (2nd Edition), John Wiley & Sons, 2011.
[4] Brenner, M.E., Brar, T., Duran, R., Mayer, R. E., Moseley, B., Smith, B. R., Webb, D. The Role of Multiple Representations in Learning Algebra, International Study Group for the Psychology of Mathematics Education, Columbus, OH, October, 1995, http://files.eric.ed.gov/fulltext/ED391659.pdf, retrieved July 12, 2017,
[5] Coffield, F., Moseley, D., Hall, E., Ecclestone, K. *Learning styles and pedagogy in post-16 learning: A systematic and critical review.* London: Learning and Skills Research Centre, 2004.
[6] Cohen, D. *Introduction to Computer Theory*, (2nd ed.), John Wiley & Sons, 1997.
[7] Dunn, R. Rita Dunn answers questions on learning styles. *Educational Leadership*, 48, 15–19, 1990.
[8] Dunn, R., Dunn, K. *Teaching secondary students through their individual learning styles.* Needham Heights, MA: Allyn and Bacon, 1992.
[9] Dunn, R., Griggs, S. *Synthesis of the Dunn and Dunn learning styles model research: who, what, when, where and so what – the Dunn and Dunn learning styles model and its theoretical cornerstone.* New York: St John's University, 2003.
[10] Francis Atuahene, F. and Russell, T. Mathematics Readiness of First-Year University Students, 2016, retrieved on November 16, 2018 from https://files.eric.ed.gov/fulltext/EJ1130188.pdf
[11] Goddard, W. *Introducing the Theory of Computation,* Jones & Bartlett, 2008.
[12] Harel, D. Statecharts: A visual formalism for complex systems, *Science of Computer Programming,* v.8 n.3, p.231-274, 1987.
[13] Harel, D. & Politi, M. *Modeling Reactive Systems with Statecharts: The Statemate Approach*, McGraw-Hill, 1998.
[14] Hopcroft, E., Motwani, R., Ullman, D. *Introduction to Automata Theory, Languages and Computation,* Pearson Education, 2007.
[15] Honey, P., Mumford A. *The manual of learning styles*. Maidenhead: Peter Honey Publications, 1992.
[16] Kinber, E., Smith, C. *Theory of Computing: A Gentle Introduction,* , Prentice Hall, 2006.
[17] Kolb, D. *Experiential learning*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[18] Kolb, D. *Learning style inventory*. Boston: McBer, 1985.
[19] Kozen, D. *Theory of Computation*, Springer, 2006.
[20] Kozhevnikov, M. Cognitive styles in the context of modern psychology: Toward an integrated framework of cognitive style, *Psychological Bulletin*, 133, 2007.
[21] Kulkarni, V. *Theory of Computation,* Oxford University Press, 2013.

[22] Lewis, H., Papadimitriou, C. *Elements of the Theory of Computation*, Prentice Hall, 1998.
[23] Lowe, R. K. & Schnotz, W. (Eds.) *Learning with animation*, New York: Cambridge University Press, 2007.
[24] Martin, J. *Introduction to Languages and the Theory of computation*, 4th edition, McGraw-Hill, 2010.
[25] Minsky, M. L. Recursive insolvability of Post's problem of 'Tag' and other topics in Theory of Turing Machines, *Annals of Mathematics*, 437-55, 1961.
[26] Ormrod, J. E. *Educational psychology: Developing learners* (6th ed.). Upper Saddle River, NJ: Pearson, 2007.
[27] Maricopa Community Colleges, Maricopa County Community College District 2016 Monitoring Report, retrieved November 16, 2018, from
[28] Pashler, H., McDaniel, M., Rohrer, D., Bjork, R. Learning Styles: Concepts and Evidence. *Psychological Science in the Public Interest*, 9(3), 105–119, 2008.
[29] Post, **A.** Finite Combinatory Processes - Formulation 1, *Journal of Symbolic Logic*, 1: 103-105, 1936.
[30] Pressman, R. Maxim, B. *Software Engineering: A Practitioner's Approach.* 8th Ed., McGraw-Hill, 2014.
[31] Rau, M. A., Matthews, P. G. How to Make "More" Better? Principles for Effective use of Multiple Representations to Enhance Students' Learning about Fractions, *International Journal on Mathematics Education*, v49 n4 p531-544, Aug 2017.
[32] Rich, E. *Automata, Computability and Complexity: Theory and Applications*, Prentice Hall, 2007.
[33] Rodger, S. H. & Finley, T. W. *JFLAP: An Interactive Formal Languages and Automata Package*. Jones & Bartlett Publishers, 2006.
[34] Rodger, S.H., Bressler, B., Finley, T. & Reading, S. Turning automata theory into a hands-on course, SIGCSE 2006: 379-383, 2006.
[35] Rodger, S. H., Wiebe, E., Lee, K. M., Morgan, M., Omar, K. & Su, J. Increasing engagement in automata theory with JFLAP, SIGCSE 2009: 403-407, 2009.
[36] Sakarovitch, J. *Elements of Automata Theory*, Cambridge University Press, 2009.
[37] Sipser, M. *Introduction to the Theory of Computation*, (3rd Ed.) Cengage Learning, 2012.
[38] Sommerville, I. *Software Engineering*, 10th Ed., New York, Addison-Wesley, 2015.
[39] Sternberg, R.J., Grigorenko, E.L., Zhang, L. Styles of learning and thinking matter in instruction and assessment. Perspectives on Psychological Science, 3, 486–506, 2008.
[40] Trusty, J., & Niles, S. G. High-school math courses and completion of the bachelor's degree. Professional School Counseling, 7(2), 99-107, 2003.
[41] Turing, A.M. On Computable Numbers, with an Application to the Entscheidungs problem, *Proceedings of the London Mathematical Society*, 2. 42: 230–65, 1936.
[42] Wong, A., Marcus, N., Ayres, P., Smith, L., Cooper, G., Paas, F. & Sweller, J. Instructional animations can be superior to statics when learning human motor skills, *Computers in Human Behavior*, Volume 25, Issue 2, 2009.
[43] Lazowski, R. A., & Hulleman, C. S. (2016). Motivation interventions in education: A meta-analysis. Review of Educational Research, 86(2), 602-640.
[44] Hulleman, C. S., Kosovich, J. J., Barron, K. E., & Daniel, D. (2017). Making connections: Replicating and extending the utility value intervention in the classroom. Journal of Educational Psychology, 109(3), 387-404.
[45] Cook D. A., Artino A. R. (2016). Motivation to learn: an overview of contemporary theories. Med. Educ. 50, 997–1014.
[46] National University, Author at National University, Retrieved on 12/12/2018 from: https://www.nu.edu/author/nueduauthor/page/42/
[47] Wright, G. B. (2011). "Student-Centered Learning in Higher Education" (PDF). International Journal of Teaching and Learning in Higher Education. 23 (3): 93–94