Learning Turing Machines in Asynchronous Mode

INTRODUCTION

Mathematical concepts are learned by reasoning in a cognitive mode that is "effortful and deliberately controlled" unlike the intuitive mode (Kahneman 2002, Stanovich and West 2000). From the explanations of the controlled mathematical mode of reasoning provided by Kahneman in his 2002 Nobel Prize lecture, it is clear that learners of mathematical concepts would benefit from any reasonable extra help that may be provided in the learning environments (Kahneman 2002). Abstract mathematical concepts of automata are among the hardest topics that should benefit from additional help from visualization tools.

The most elegant models of computation are the mathematically defined automata including Turing Machines (TMs), two-stack Pushdown Automata (2PDA), Linear Bounded Automata (LBA), Pushdown Automata (PDA), Finite Automata (FA) and Non-deterministic FA (NFA). These models are usually studied in the fields of theory of computation, automata theory and computability. TMs define the most powerful automata class for processing the most complex sets, namely, recursively enumerable sets. TMs are equivalent to 2PDA as proven by Minsky (1961). Classical forms of PDA are defined to have exactly one stack and they are non-deterministic unless otherwise explicitly stated. PDA are acceptors of the class of Context-Free Languages (CFLs). They are less powerful then TMs; they cannot accept non-CFLs. Programming languages such as C++ and Java are defined best by Context-Free Grammars (CFGs) which are equivalent to PDA. FA define a proper subset of CFLs called regular languages denoted by regular expressions. FA are deterministic unless otherwise explicitly stated. Non-deterministic FA are equivalent to FA in the sense that they accept the same sets defined by regular expressions. The above narrative information with some additional information is usually presented in a tabular form called the Chomsky hierarchy of grammars and languages as shown in Table1 (Cohen, 1997).

The Chomsky Hierarchy of Grammars and Languages		
Type	**Language/Grammar**	**Acceptor**
0	Recursively Enumerable	Turing Machines (TMs) = 2PDA = Post Machines
1	Context-Sensitive	Linear Bounded Automata (LBA) = Turing Machines with bounded tape.
2	Context-Free	Pushdown Automata (PDA)
3	Regular	Finite Automata (FA) = Non-deterministic FA = Transition Graphs

Table 1: The Chomsky Hierarchy

The models in Chomsky hierarchy are generally taught in an automata theory course using standard textbooks (Cohen, 1997; Goddard, 2008; Hopcroft, Motwani, & Ullman 2007;   Kinber & Smith 2001; Kozen, 2006;   Lewis & Papadimitriou 1998; Rich, 2007; Sakarovitch, 2009; Sipser 2006).   This study develops static and dynamic graphics of some of the automata for helping learners. This approach in the teaching learning environments is based on pioneering work of Rodger (Rodger, 2006; Rodger & Finley, 2006, Rodger, Bressler, Finley, Reading, 2006; Rodger, Wiebe, Lee, Morgan, Omar, Su, 2009) and a broad expansion of our earlier work (Dey, Gatton, Amin, Wyne, Romney, Farahani, Datta, Badkoobehi, Belcher, Tigli, & Cruz, 2009).

The class of TMs is the most powerful class of computing machines. Any problem that can be solved computationally can be solved by a TM. Thus, computability means Turing computability. A TM has a finite set of states with one START state and some HALT states and an infinitely long READ/WRITE tape divided into distinct cells. A TM may not have any HALT state at all and sometimes a TM may have multiple HALT states.   The input is initially placed on the tape starting from the leftmost cell. The TAPE HEAD initially points to the leftmost cell. With a given transition, a TM does three things: (1) it reads one symbol from a tape cell, (2) writes one symbol on the same tape cell and (3) moves the TAPE HEAD right or left by one cell.   A TM accepts an input string by starting from a start state, moving through a path of transitions and reaching the HALT state.

STATIC AND DYNAMIC GRAPHICS

Static and dynamic graphics are often compared for their relative advantages and disadvantages (Tversky, Morrison & Betrancourt, 2002; Lowe, 1999; Lowe, & Schnotz, 2007). Most of these studies suggest that use of static graphics is more effective in educational environments than that of comparable dynamic graphics. Although superior effectiveness of interactive dynamic graphics has been recognized they are not considered to be comparable to static graphics (Tversky, Morrison & Betrancourt, 2002; Wong, Marcus, Ayres, Smith, Cooper, Paas, & Sweller, 2009). We have developed interactive dynamic graphics as well as static graphics that demonstrate some of the automata mentioned above;   these are linked to the following website: http://www.asethome.org/automata/. Static and dynamic graphics are presented in an unbiased way so that learners are able to select the type of visualization they like to use in a given context. A case of static visualization of a TM is presented below. A finite or infinite set of strings is usually called a language. In this sense, a Turing Machine can be viewed as a language processor. We would like to call the following set of strings as Language-1 or L_1.

   L_1 = { ab, aba, abaa, abaaa, abaaaa, abaaaaa, **. . .** }

This language is usually abbreviated as **aba*** which is a regular expression. The star after the second **a** in this string is known as the Kleene star which means zero or more **a**'s.   The regular

expression **aba*** means one **a** followed by one **b** followed by zero or more **a**'s. It is perfectly acceptable to write:

L_1 = aba* = { ab, aba, abaa, abaaa, abaaaa, abaaaaa, ... }

A regular expression such as **aba*** denoting a set of strings usually represent a pattern in the strings. (Note: If we replace **a** by N and b by V in this set then we have a pattern NV, NVN, NVNN, which would be a pattern subset from English where N is a Noun and V is a Verb that matches English sentences like "Boys like eggs", "Boys like eggs, Noodles" ...). The following Turing Machine can process L_1.
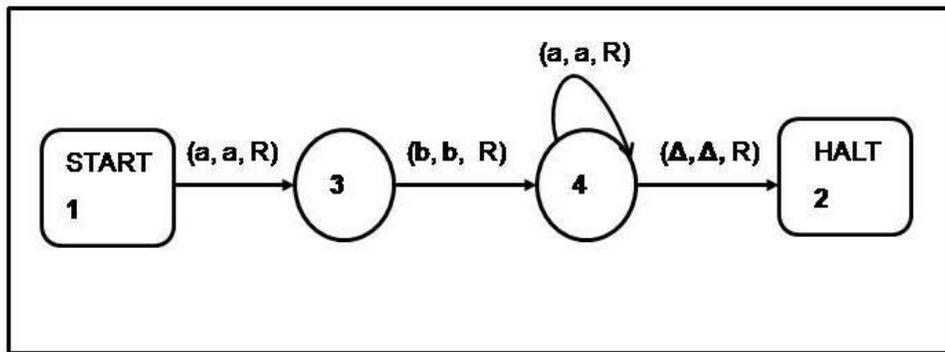

Figure 1: A Turing Machine for **aba***

The tape of this TM is not shown in the diagram; because we do not want to show any specific input on the tape since any string from L_1 is accepted by this machine. The state numbers are used for uniquely identifying the states. A label on a transition has three components separated by commas. Thus, the transition between state 1 and state 3 is labeled by (a, a, R) which means read an **a** from the READ/WRITE tape, write back an **a** on the same cell of the tape and move the TAPE HEAD **R**ight.

Suppose, as an experiment, the given input string is **abaa**, which is a member of the set of strings for which this machine is designed. The machine would look like Figure 2 when the input is placed on the tape.
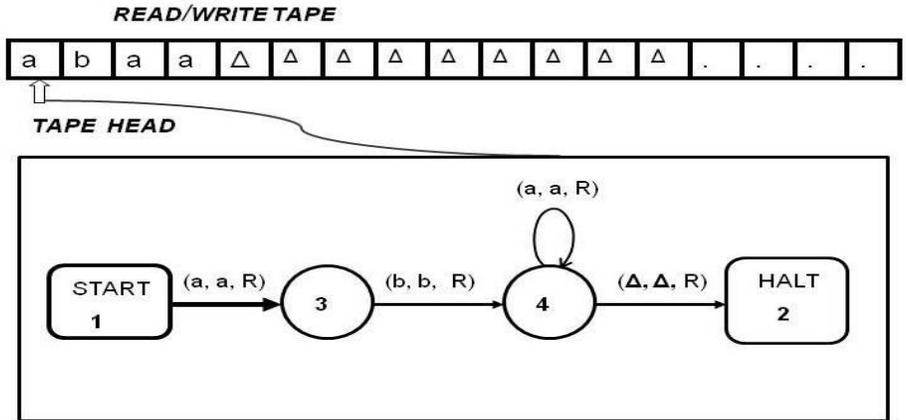
Figure 2: A Turing Machine for **aba*** with input **abaa**

The machine, starting from the start state, taking the first transition to state 3, reads the first symbol **a** from the leftmost cell of the tape writes back **a** on the same cell and moves right. This is shown in Figure 3.

Figure 3: A Turing Machine for **aba*** which has just scanned the first symbol from the input.

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. The machine configuration is shown in Figure 4.

READ/WRITE TAPE

| a | b | a | a | Δ | Δ | Δ | Δ | Δ | Δ | Δ | Δ | Δ | . | . | . | . |

TAPE HEAD

(a, a, R)

START 1 —(a, a, R)→ 3 —(b, b, R)→ 4 —(Δ, Δ, R)→ HALT 2

Figure 4:   A Turing Machine for **aba*** which has just scanned 2nd symbol from the input
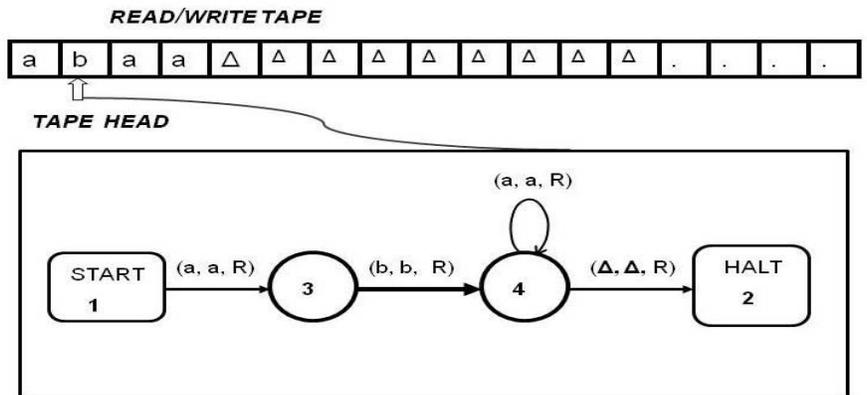
Next, the machine, from state 4, taking the transition marked by (a, a, R) reads the symbol **a** from the third cell of the tape and writes back **a** on the same cell and moves right on the tape. Taking this transition the machine comes back to state 4. The machine configuration is shown in Figure 5.

READ/WRITE TAPE

| a | b | a | a | Δ | Δ | Δ | Δ | Δ | Δ | Δ | Δ | Δ | . | . | . | . |

TAPE HEAD

(a, a, R)

START 1 —(a, a, R)→ 3 —(b, b, R)→ 4 —(Δ, Δ, R)→ HALT 2

Figure 5. A Turing Machine for aba* which has just scanned the 3rd symbol from the input

Next, the machine, from state 4, taking the transition marked by (a, a, R) reads the symbol **a** from the fourth 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. The machine configuration is shown in Figure 6.

5

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

In the next step, taking the transition marked by (Δ, Δ, R) from state 4, the machine reads Δ from the fourth 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. The machine accepts the input, **abaa**, since it reaches the HALT state after traversal of transitions from the start state. The machine configuration is shown in Figure 7.

Figure 7. A Turing Machine for **aba*** which has reached the Halt state and accepted the input

In other words, this machine is designed to accept every string of the set aba* = { ab, aba, abaa, abaaa, abaaaa, abaaaaa, ... }. The static graphics of the type shown in Figures 2-7 are presented with dynamic graphics or animations linked to the web page (http://www.asethome.org/automata/) which can be considered for teaching computer science courses.

Following Cohen (1997), we define TMs as follows.

A TM, composed of six components, is defined as follows.

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.
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 is a letter read by the TAPE HEAD of the machine from the current cell. 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.

CONCLUDING REMARKS

Automata are demonstrated using static and animated graphics for the benefit of students and teachers. Students are free to use the ones they like. Future work includes adding more animated automata, upgrading current animations with better features and interactions and analyzing feedbacks from learners.

References:

Cohen, D. (1997). *Introduction to Computer Theory* (2nd ed.), New York: John Wiley & Sons.

Dey, P., Gatton, T., Amin, M., Wyne, M., Romney, G., Farahani, A., Datta, A., Badkoobehi, H., Belcher, R., Tigli, O., Cruz, A. (2009). Agile Problem Driven Teaching in Engineering, Science and Technology. *In the Proceedings of the American Society for Engineering Education-Pacific Southwest 2009 conference* ASEE-PSW 2009, San Diego, California, U.S.A., March 19-20, 2009.

Goddard, W. (2009) *Introducing the Theory of Computation*, Jones & Bartlett Publishers.

Hegarty, M. (2005) Dynamic visualizations and learning: getting to the difficult questions. Learning and Instruction. v14. 343-351.

Hopcroft, J. E., Motwani, R. & Ullman, J. (2007). *Introduction to Automata Theory, Languages, and Computation*. Pearson Education.

Kahneman, D. (2002). Maps of bounded rationality: A perspective on intuitive judgment and choice (Nobel Prize Lecture). In Tore Frängsmyr, (ed.). *Les Prix Nobel. The Nobel Prizes 2002*, 416-499, Retrieved March 12, 2008 from http://nobelprize.org/nobel_prizes/economics/laureates/2002/kahneman-lecture.html

Kozen, D. (2006). *Theory of Computation*. Springer .

Lewis, H. R. and C. H. Papadimitriou (1998). *Elements of the Theory of Computation*. Prentice Hall

Lowe, R. K. (1999). Extracting information from an animation during complex visual learning. *European Journal of Psychology of Education*, 14, 225–244.

Lowe, R. K. & Schnotz, W. (Eds) (2007) *Learning with animation*, New York: Cambridge University Press.

Minsky, M. L. (1961) Recursive insolvability of Post's problem of 'Tag' and other topics in Theory of Turing Machines, Annals of Mathematics, 437-55, 1961.

Kinber, E. and Smith, C. (2001). *Theory of Computing: A Gentle Introduction*. Prentice Hall.

Rich, E. (2007) Automata, Computability and Complexity: Theory and Applications

Rodger, S.H. (2006) Learning automata and formal languages interactively with JFLAP. ITiCSE 2006: 360

Rodger, S. H. & Finley, T. W. (2006). *JFLAP: An Interactive Formal Languages and Automata Package*. Jones & Bartlett Publishers.

Rodger, S.H., Bressler, B., Finley, T. & Reading, S. (2006). Turning automata theory into a hands-on course. *SIGCSE 2006*: 379-383

Rodger, S.H., Eric Wiebe, Kyung Min Lee, Chris Morgan, Kareem Omar, Jonathan Su. (2009). Increasing engagement in automata theory with JFLAP. *SIGCSE 2009*: 403-407

Sakarovitch, J. (2009). *Elements of Automata Theory*, Cambridge University Press. Translated by Reuben Thomas.

Sipser, M. (2006) *Introduction to the Theory of Computation*, PWS Publishing.
R. Gregory Taylor (1998) *Models of Computation and Formal Languages,* Oxford University Press.

Wong, A., Marcus, N., Ayres, P., Smith, L., Cooper, G., Paas, F. & Sweller, J. (2009). Instructional animations can be superior to statics when learning human motor skills, *Computers in Human Behavior,* Volume 25, Issue 2, March 2009, Pages 339-34

Stanovich, K.E. and West, R.F. (2000). Individual differences in reasoning: Implications for the rationality debate, *Behavioral and Brain Sciences*, 23, 645-665.

Tsay, Y., Chen, Y., Tsai, M., Wu, K. & Chan, W. (2007). GOAL: *A Graphical Tool for Manipulating Büchi Automata and Temporal Formulae, in Tools and Algorithms for the Construction and Analysis of Systems*. Springer.

Tversky, B., Morrison, J. B., & Betrancourt, M. (2002). Animation: can it facilitate?. *International Journal of Human–Computer Studies*, 57, 247–262.