MyTuringTable: A Teaching Tool to Accompany Turing’s Original Paper on Computability

Barry Fagin
Department of Computer Science
2354 Fairchild Drive, 6G-139
US Air Force Academy, CO 80840
719-333-7377
barry.fagin@usafa.edu

Dino Schweitzer
Director, Academy Center for Cyberspace Research, 6G-101
US Air Force Academy, CO 80840
719-333-3945
dino.schweitzer@usafa.edu

ABSTRACT
MyTuringTable is a Turing Machine simulator designed to help students read and understand Turing’s seminal 1937 paper on computability. We discuss our reasons for developing the tool, and report on its use in a “Great Ideas in Computer Science” course for the Air Force Academy Cadet Scholars program.

Categories and Subject Descriptors

General Terms
Algorithms, Theory

Keywords
Turing Machines, Educational Software, Computer Science Education, History of Computing

1. INTRODUCTION
Alan Turing’s 1937 paper on the Entscheidungsproblem [9] marked the beginning of computer science as an academic discipline, and ushered in the computing era. As such, we believe it ranks as one of the most significant achievements in the history of ideas, a work that deserves a much larger audience.

Unfortunately, Turing’s paper is dense and difficult to read, particularly for undergraduates. We faced this challenge in developing and teaching a course on “Great Ideas in Computer Science” in the Scholars Program at the US Air Force Academy.

The Scholars Program consists of the top one percent of cadets academically at USAFA, those likely to be competitive for national scholarships such as the Rhodes. These cadets take special versions of the Academy’s core classes together, where they are expected to engage directly with original source materials, to think critically about them, and where possible to encounter and assimilate their important contributions through hands-on practice and implementation. Most Scholars courses are in the humanities and social sciences, but a few are in the sciences and engineering.

Two years ago, the first author sought to create a computer science class for Scholars students, based on the theme of “Great Ideas in Computer Science”. This course would be taught during the freshman year, with no prior knowledge of computing or programming assumed. Consistent with the intent of the Scholars Program, students would read original works by the most important contributors to the field, such as Babbage, von Neumann, and of course Alan Turing. Similarly, they would implement the Method of Finite Differences as described in [6], write programs in assembly language for von Neumann’s IAS machine [1], and learn to write their own Turing Machines (TMs).

However, in the course of preparing the necessary instructional materials, it became quickly apparent that, even for students in an enriched program such as this, learning to write TMs solely from Turing’s writing would be an extremely difficult task. To better acquaint students with the ideas in Turing’s paper and to enable them to more easily write their own TMs, some sort of instructional support was required. We envisioned a TM simulator that was easy to use but at the same time closely tied to Turing’s paper, to emphasize the specific nature of his ideas and originality of his thought.

2. PREVIOUS WORK
A number of interactive instructional tools have been developed to teach formal languages and theoretical models of computation. The idea of using a simulator for computational models is not new. For example, Curtis developed a program that simulated TMs over 40 years ago [3]. In more recent years, several software tools have been developed in an attempt to engage students by adding interactivity and experimentation into a traditional paper and pencil set of topics. The tools are written in different languages and environments with various types of interactivity, some graphical, some tabular. A common example is the TM simulator created at Princeton (http://introcs.cs.princeton.edu/java/74turing/).

Some of these tools focus on a single topic, such as finite automata or TMs, while others attempt to combine several related concepts into a package of tools with a common interface. Chesnevar, et.al., describe their experiences using multiple TM simulators, a state-based version created by Suzanne Skinner, and one with a graphical interface, called VisualTuring (http://sourceforge.net/projects/visualturing/) [2]. The Java Computability Toolkit (JCT) created by Robinson, et.al., provides

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

\textit{ITiCSE '12}, July 3--5, 2012, Haifa, Israel.
Copyright 2012 ACM 978-1-4503-1246-2/12/07...$10.00.
a multiple-window environment for constructing finite automata and TMs graphically [7]. Hannay presents web-based tools with a tabular interface all written in JavaScript [5]. Educators at Duke have extended the idea of a single tool and created a collection of tools for experimenting with automata, grammars, and parsing called JFLAP [8]. Beyond TMs, JFLAP provides tools and demonstrations for several formal concepts in order to support an entire course on formal languages. Work on other software developed for the Academy’s “Great Ideas in Computer Science” course was first presented in [4]. It described the tools required for students to write programs for the original Princeton IAS machine.

3. **MyTuringTable**

MyTuringTable contains a number of features unique to it among TM simulators, based on our desire to link it as closely as possible to the original presentation of the TM in [9]. In particular, while it shows the standard picture of a tape with symbols and a read-write head, the machine itself is table-driven, because Turing originally presented his “computing machines” in that form.

Programming a TM with MyTuringTable involves creating rows of a table with an arbitrarily named state, a symbol under the head, a symbol to write and a direction to move the head. Space is also provided for comments, and an editing column to add or delete rows. Machines can be saved and loaded from disk, and various buttons are provided for single-stepping forward and backward, running a specific number of steps, and resetting the machine. A slider is also available to speed up or slow down the computation.

Because Turing’s paper makes use of a wide variety of symbols, MyTuringTable does as well, supporting any and all single-character symbols from the keyboard that can be drawn by Java. Smart pulldown menus are used whenever possible, for example in selecting the next state and in selecting the symbols that might appear under the head.

One particularly useful feature is the ability to aggregate sets of symbols together on a single row, similar to the shorthand notation occasionally used by Turing. Users can indicate that a transition is to be made for all symbols except a given one, or only for a specific set of symbols, and so on. An example will make this clearer.

4. **USING MYTURINGTABLE TO IMPLEMENT THE FIRST TURING MACHINE**

Turing’s first machine, taken directly from [9], is shown below:

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>m-config</td>
<td>symbol</td>
</tr>
<tr>
<td>b</td>
<td>None</td>
</tr>
<tr>
<td>c</td>
<td>None</td>
</tr>
<tr>
<td>e</td>
<td>None</td>
</tr>
<tr>
<td>f</td>
<td>None</td>
</tr>
</tbody>
</table>

Right away, we see the potential for confusion. m-config stands for machine configuration, what we would normally call a state. The phrase “final m-config” in the last column really means “next state”. The operations column is clearer: “P0,R” means print a zero under the head and move the head right. This machine computes the infinite sequence “0 1 0 1…”, alternating 0’s and 1’s between blank squares on the tape.

The initial screen of MyTuringTable is shown in Figure 2:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference’10, Month 1–2, 2010, City, State, Country. Copyright 2010 ACM 1-58113-000-0/00/0010…$10.00.
This brings up a text entry box, in which we type the symbol to be printed. Closing the box enters the new symbol in the table and creates a new row in the table in anticipation of further state entries:

Continuing to the “Direction” entry, we select either left or right from a pulldown as appropriate:

To create the “Next State” entry (highlighted in yellow), we click and choose from another pulldown. Initially, MyTuringTable knows about the special states Start and Halt, a default numbering scheme S1, S2, etc, and an option “new” if the user wishes to type in a custom state name.
Continuing the process a few more steps, we come to the entry where the machine must write a “1”. Clicking in the “Head” column, we see the symbol “0” has been added to the pulldown.

This is not the symbol we need, so once again we select “new” and type in a “1”. Continuing the machine in this way gives the complete table below:

Pressing “Step” causes the first row of the table to be executed. A 0 is printed under the head, it moves to the right, the next state becomes S1, and the new current state is highlighted:

The tape moves in the appropriate direction to keep the head on the screen. We chose to move the head as opposed to moving the tape to be as close as possible to the original description of TMs in [9].

5. A MORE COMPLEX EXAMPLE

Turing’s second example illustrates considerably more features of the TM, and therefore of MyTuringTable. This example uses input, makes decisions based on the symbols under the head, and makes extensive use of shorthand notation to combine states. The table for this machine is shown below:
It produces the sequence “0010110111…” on alternate squares, with the squares in between used for bookkeeping purposes.

The first row in the table is shorthand for the states necessary to write the symbols “0 0” on the tape and move the head to the first “0”. While MyTuringTable does not as yet support this kind of notation, it does permit the user to edit tape cells directly (right click) and explicitly position the head (left click). Providing the designated input for the machine in Figure N produces the following:

Figure 14, Setting Up the Input

Proceeding through the states from the top of the table down, we come to the entry where the transition should be triggered with either a 0 or a 1. Separate entries could of course be written for both, but in support of Turing’s notation and for similar reasons of convenience, MyTuringTable allows the user to specify a collection of symbols by selecting “Set” under the pulldown for symbols under the head:

Figure 15, the Symbol Set Dialog

When this option is selected, all symbols currently in the machine are brought up in a dialog box. Individual symbols can be checked, and inclusion or exclusion selected. The use of a set for symbols under the head is indicated in the table with the word setN, which can be clicked on at any time to reveal its contents and to permit revisions as necessary. Sets can be reused as necessary.

The complete machine and its progress after several steps are shown below:

Figure 16, Turing’s Second Machine After Several Steps
Note the presence of the “x” symbols used by the machine for bookkeeping. State “F” makes use of a second set to mean “any symbols except a blank”. Turing’s paper uses “None” to indicate a blank, which is somewhat unclear. We permit the use of “None” in columns for consistency with Turing, but encourage the use of “blank” by students for clarity whenever possible.

6. OTHER FEATURES
To firmly establish the connection between TMs, computable numbers, and enumerability, Turing introduces the notion of a standard form for computing machines, in which all states are labeled as q, along with some other simple conventions. Tables in standard form have a corresponding character string standard description, which in turn yields a unique description number. Since description numbers can be placed in one-to-one correspondence with the integers, this shows that the class of computable numbers is enumerable.

MyTuringTable includes commands to convert tables to standard form, to generate their standard description, and to calculate their description number.

7. ASSESSMENT
We now have two years’ worth of data on the effectiveness of MyTuringTable, corresponding to two offerings of “Great Ideas in Computer Science”. Students receive an extensive problem set requiring them to write simple TMs, ranging in complexity from the machine in Figure 1 up to a binary incrementer. We note that these students, while exceptionally motivated and talented academically, in the vast majority of cases will not be computer science majors and have never programmed before. They also have only four class sessions (about a week and a half) in which to read Turing’s paper, become familiar with the concept of a TM, and complete the assignment.

Results for 44 students are shown below:

8. FUTURE WORK
From the point of view of the history of computing, the most important aspect of the TM is its universality: The fact that TM can be constructed to read the description of any TM and perform its computations. To construct his universal machine, Turing makes extensive use of function tables, the rough equivalent of macro definitions. Function tables can be composed to produce successively more complex machines. In fact, it is effectively impossible to produce any TMs of reasonable utility without them, as their size and notation become too cumbersome.

In addition to minor tweaks and improvements in functionality, we hope to add function tables to MyTuringTable in a future release. This would significantly increase the complexity and utility of the TMs students can create. It is even conceivable that, eventually, MyTuringTable could be used to create a mathematically correct implementation of the Universal TM.

9. ACKNOWLEDGMENTS
The authors gratefully acknowledge support from the Air Force Academy Department of Computer Science in developing the Great Ideas course for the Scholars Program. The authors are grateful to the Academy Center for Cyberspace Research for its support for educational software development. MyTuringTable is publicly available and can be downloaded from www.dinoonline.net.

10. REFERENCES


