Turing 1.0

About the Program

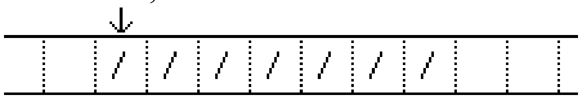
This program was conceived several years ago when I sat through the drudgery of tracing by hand the paths of a Turing machine to check my homework and I realized that this was an ideal job for a program. Therefore, this program is dedicated to and aimed towards higher education computer science curricula.

Introduction to Turing Machines

A Turing machine is a simple model of a general-purpose computer which can read inputs, perform certain operations and write outputs. A Turing machine can perform any computation that any other general-purpose can, and is therefore useful for study in academic computer science curricula.

Turing machines (usually) consist of a tape upon which characters are read and written, and a set of states that define the actions of the machine. A Turing machine may read a character from the tape, write a character to the tape, move the "read/write" head left or right (or leave it where it is) and move to another state.

For example, given a tape with several slashes on it, a space (or several) after the slashes, and the read/write head (represented by the arrow) pointing to the first slash,



we can replace the slashes with asterisks by using the Turing machine state table

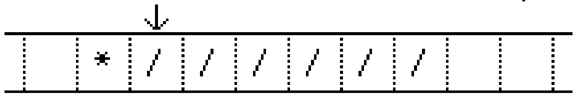
	space	/	*
1		*R	
2			

This table represents the actions of the Turing machine.

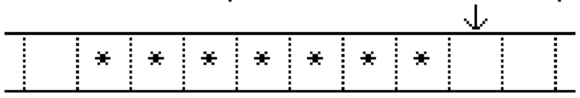
Each cell in the table may consist of up to three parameters, each of which is optional. The first character is what gets written to the tape. The second character is either "L" or "R", to move the tape read/write head left or right. The final parameter is the number of the next state to be executed.

If the current state in this example is state number 1 (following the numbering down the left side of the table), then when a slash is under the read/write head, the active cell is the one containing "*R". Therefore, an asterisk is written (overwriting the slash), and the tape head is moved right.

Since there is no third parameter, the next state to be executed is the current state, number 1. The tape then looks like this:



On the next execution cycle, the current state is number 1 and the character under the tape head is a slash, so the same instruction is executed. This overwrites each slash with an asterisk until the last slash has been overwritten, at which time the tape head is over a space. Since there is no instruction defined for the space character in state 1, the Turing machine halts. At completion, this is the tape:

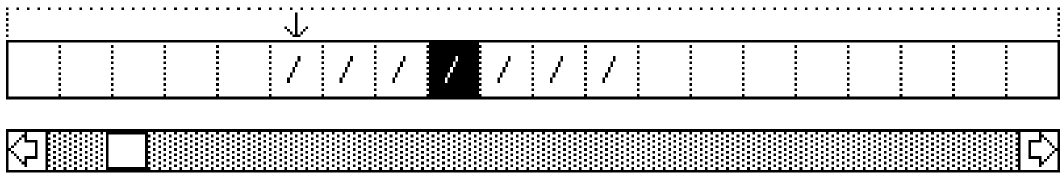


Using Turing 1.0

Turing is fairly straightforward and simple to use. The interface consists mainly of two windows, one containing the Turing machine's tape and the other window containing the state table.

The Tape

The tape window looks something like this:



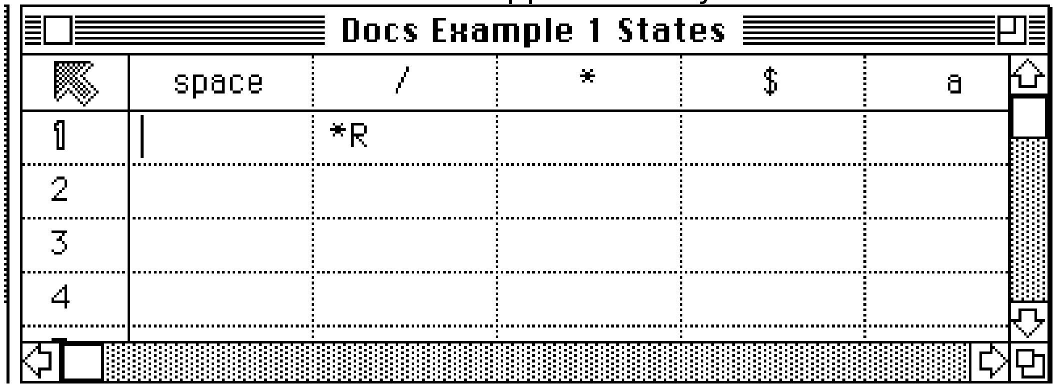
Although some Turing machine models use a two-way infinite tape, the tape model used in Turing 1.0 is finite in both directions. The scroll bar scrolls the tape left or right. The rectangle above the scroll bar contains the tape and is divided into 1-character blocks. The arrow above the tape represents the current position of the read/write head. The inverted character block indicates the current character that will be replaced if something is typed.

Editing the tape

Typing a character replaces the character in the inverted block, and moves the inverted block to the right. In this way tapes can be set up before running a Turing machine. Clicking inside the tape rectangle moves the inverted block to that location. Clicking inside the dotted rectangle moves the read/write head to that location.

The State Table

The state window looks approximately like this:



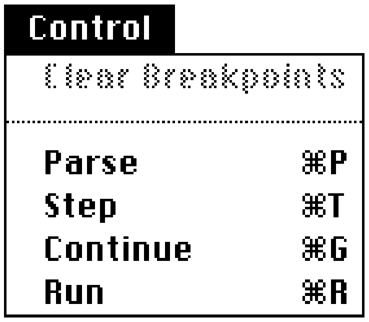
	space	/	*	$	a
1		*R			
2					
3					
4					

The column down the left side of the table numbers the states. The current state's number is in outline form. Across the top of the table is the machine's alphabet. The scroll bars scroll the state cells; the horizontal scroll bar reveals more of the alphabet and the vertical scroll bar reveals more of the state numbers.

Moving Around in and Editing the State Table

Clicking on the large arrow in the upper left of the table puts the first state and the first character in the alphabet into the upper-left of the window; this is the same as using the scrollbars to scroll to the top-left of the table. Transition instructions (such as "*R") can be entered into the table by clicking on the appropriate cell and then typing. Typing the Tab key moves the cursor to the next cell in the row; typing the Return key moves to the next cell in the column. The alphabet for the table may be modified by clicking on the character to be changed, and then typing the new character. Clicking on a state number makes it the current state. A breakpoint may be set for a state by holding down the option key (the cursor changes to a small stop sign) and then clicking on the state number. States that have breakpoints attached to them have their numbers displayed in italics.

The Control Menu



This menu is used to parse and execute the Turing machine. The first item, "Clear Breakpoints", is used to clear any breakpoints (hence the name) that have been set for the machine. "Parse" performs a check on the syntactic correctness of the state transitions. This is automatically done before running. "Step" executes the current transition and then halts. "Continue" causes execution to begin at the current transition and continue until either the mouse button is clicked or a halt state (no transition defined) is entered. "Run" makes state 1 the current state, and begins executing until the mouse is clicked or a halt state is entered.

Afterword

While these instructions detail the parts of the program, more can be learned through the use of Turing. Therefore, several sample machines have been included. "Back and Forth" simply causes the read/write head to seek back and forth between ends of the characters written on the tape. The "AB Sort" file contains a machine which will sort a set of a's and b's written on the tape (with no spaces between them). "Docs Example" contains the example used at the beginning of this document.

Plea

If you use this program, *please* support the shareware ideal and send in your fee. Site licenses are available to institutions of higher education at significant discounts!

Please email any bug reports and enhancement suggestions to one of the addresses below.

Lee Fyock
laf@mitre.org   Internet
Fyock   America Online