NAME

tm − Turing Machine simulator and visualizer

SYNOPSIS

tm −m machine_file −t tape_file [-d] [-s] [-v] [-V]

DESCRIPTION

tm is a Turing Machine simulator and visualizer.

tm has two display modes: visual and non-visual. In visual mode **tm** generates a text curses display of the state machine transition table, with the current state and input entry highlighed. Also, the segment of the tape currently being accessed by the tape head is displayed. In non-visual mode, **tm** simply runs the turing machine, streaming out status information.

In visual mode, the Turing Machine simulator has two execution modes: debug and free-run. In debug mode, the machine will step one instruction each time the user presses a key. In free-run mode, the machine executes without waiting for key presses, but still rapidly displays the machine state and tape. It also possible to toggle the visual updating while still in "visual mode", which yields considerable speed up of the simulation, but with the ability to toggle back into debug mode.

In visual mode, some keyboard keys have special meaning, in both debug and free-run modes: The "Escape" key halts the machine. The "d" key toggles debug mode. This means that you can go between debug and free-run when ever you like. The "v" key toggles the visual updating. When visual updating is turned off, the machine executes much faster, but not as fast as when not in visual mode. The "?" key displays a summary of these commands. In debug mode, other keys simply step the machine forward one instruction.

OPTIONS

−m *machine_file*
: Read the *machine_file* which desribes the state transition table

−t *tape_file*
: Read the *tape_file* which describes the input tape for the Turing Machine.

−d Start the simulation in debug mode. This also implies visual mode.

−s Search for busy beavers.

−v Run the machine in "visual" mode.

−V Run the machine in "verbose" mode, which prints some status information at the end, including the final copy of the tape, the tape size, and the number of shifts executed. The printout of the final tape will print tape frame 0 surrounded by marker strings to make it easier to identify.

If both visual and verbose options are active, then the initial printing of the state transition table, and the initial printing of the tape are suppressed since they would be immediately over written by the visual display, and they both appear in the visual display.

If none of the optional options (d,v,V) are activated, then **tm** does not produce particularly interesting information. Perhaps the most useful application of **tm** without any of the d,v,V options is to see whether a particular Turing Machine ever halts.

BASIC CONCEPTS

A Turing Machine is a very basic computer model introduced by Alan Turing in the 1930s. Turing called these machines "Logical Computing Machines". Such a machine consists of an infinitely long tape, a tape head, a current state, and a set of state transition rules.

The tape is segmented into frames. Each frame can contain a single character. The set of allowed characters is called the **character set**. The tape frames are both readable and writeable.

The tape head can move either left or right on the tape, and performs the reading and writing of the tape.

The set of state transition rules is a table which, given the current state and the current input, provides an instruction in three parts: What to write at the current tape frame, which way to move the tape head, and what state to enter next. This set of state transition rules is described in the *machine_file*

For a cool example of a Turing Machine in science fiction, read Neal Stephenson's book, **The Diamond Age**.

MACHINE FILE FORMAT

The *machine_file* format is a text file which is intended to be human-readable. The file can contain blank lines and comment lines as well as lines which provide information about the state transition table.

The first non-comment line in the *machine_file* should be

charset_max *integer*

where the integer is the maximum value of the character set for the tape. The integer must be positive.

After the **charset_max** line is encountered, the rest of the *machine_file* consists of **'state'** / **'input'** line blocks.

The second non-comment line in the *machine_file* should be

state *comment*

which marks the beginning of a set of rules for a given state. States are numbered sequentially, starting at 0. The **'state'** string can be followed by a comment line. Each time a **'state'** line is encountered, the total number of states in the machine is incremented, and the following **'input'** lines, up until the next **'state'** line, are used to fill the state transition table for the current state.

Between **'state'** lines are **'input'** lines which have the form

input *integer* **write** *integer* **move** *L|R|S* **next** *integer*

The integer following the **'input'** string is the value on the tape which will result in the execution of the rest of this **input** line.

The integer following the **'write'** string is the value written on the tape at the current frame.

Both integer arguments of **'input'** and **'write'** must be within the character set as defined by the **charset_max** line.

The character follwing the **'move'** string is either 'L' for left, 'R' for right, or 'S' for stop, which indicates the direction the tape head is to move. 'S' indicates that the machine should stop.

The integer following the **'next'** string indicates the state to enter into next. The integer must refer to a valid state.

The Turing Machine starts in state 0.

TAPE FILE FORMAT

The *tape_file* is a text file which described the initial value of the tape. The first line of the tape file must be a comment line of the form

# comment

The second line of the tape file must be an integer which is the starting index of the rest of the values in the tape file.

The third and subsequent lines in the tape file are tape frame values or a line of the form

state *integer*

A line beginning with the string "state" is used to indicate the starting state of the machine. If no "state" line is present, then state 0 is assumed to be the starting state. Usually, the user will not supply a starting state because the convention with Turing Machines is that their initial state is the first state in the table. The "state" ine feature is intended to be used only for restarting a Turing Machine simulation from a previous execution. Only one "state" line may be present.

The values of the tape frames are interpretted as sequential from left to right, starting at the index given on the second line of the tape file. Tape values must be one per line.

If a tape value is preceded by the string "head" then the corresponding tape frame will be the starting location of the tape head when the simulation starts. If the tape head is not specified in this way (i.e., if there are no tape values preceded by the "head" string) then the initial head location will be at index 0. If more than one "head" line is present in the tape file, an error occurs.

VISUAL MODE

The "visual" mode displays the state transition table, the segment of the tape in the vicinity of the tape head, and some other information.

The left-most column of the state transition table indicates the state. Each row of the state transition table indicates the instructions to perform for that given state. Each of the columns to the right of the leftmost column are for a given input from the current tape frame. The entries in the state transition table have 3 symbols. The left symbol indicates the value to be written onto the current tape frame, the middle symbol indicates the direction that the tape head will move, and the right symbol indicates the next state to enter. The entry that corresponds to the current state and tape input value is highlighted.

The text window must be tall enough for the state transition table to fit, in order for it to be displayed. If the window is not tall enough, then a message is printed indicating the height needed to display the table. E.g., For a 5-state Turing Machine, the window should be at least 24 lines tall.

The tape display has three parts, verticlly stacked: the tape indices, the tape values, and the tape head position.

The tape display shows the tape indices above the tape values. These indices are not actually used by the Turing Machine, but they are useful to a human trying to keep track of what the Turing Machine is doing. The tape index 0 is the starting point for the tape head.

The tape values are printed with spaces between the values. Remember that, for a true Turing Machine, the tape is infinitely long. However, in this simulation, the tape's stored length is finite (although the tape will "grow" indefinitely, as new portions are accessed) . When the entire stored tape will not fit on the screen, then ellipsis are displayed indicating that there is more of the tape than what is being displayed. Remember that if ellipsis are not visible, it means that the representation of the tape is not currently being stored. It does NOT mean that you are looking at the end of the tape. There is no end of the tape. The tape is infinite.

The tape values will scroll left and right as the tape head moves. The tape head will also move, when scrolling the tape is not appropriate.

Below the tape values portion of the tape display is the tape head location indicator, along with the tape index where the tape head sits. Again, this index is not used by the abstract Turing Machine, but it is useful for humans.

SEE ALSO

curses(3)

BUGS

The tape display assumes that the character set has only 1-digit values. If the charset_max is greater than 9, then the tape display will not work.

Only positive integers are allowed for the character set. In theory, any set of symbols should be allowed, including a set of strings or funny-looking shapes, like Greek letters or card suites. However, since any finite set has ennumerable members, then positive integers will always suffice to simulate a Turing Machine. In fact, only a very small number of symbols is needed to simulate ANY Turing Machine (by creating a Universal Turing Machine). It is just a matter of encoding the symbols with an appropriate scheme.

The tape grows dynamically so this simulator will eventually run out of memory if the Turing Machine runs over a large enough section of tape. If you have access to the source code, it is a trivial matter to change the primitive type of the character set, and of the state, so that you could use a smaller primitive type to conserve memory.

This simulation was designed to be easy to understand, not to run fast. This code would probably not provide a great basis for doing advanced theory of computation. However, the visual display makes debugging state transition tables easier.

The only way to change the maximum number of iterations is to change the source code and recompile. It would be simple to add a command line option to do this instead, which I should do.

The busy beaver rejection algorithms are simple and err on the side of safety, in the sense that no valid busy beavers are rejected, but a lot of unfruitful busy beavers are allowed to run. There are several semi-obvious rejection schemes which could be implemented. For example, Turing machine tables have a kind of symmetry, where more than one machine performs exactly the same. These symmetries are due to the fact that the number associated with a state is not germain (except for state 0 which is the initial state). Also, it does not matter whether a machine moes to the left or right -- which gives a two-fold symmetry. These are not currently considered in the rejection procedures.

Report bugs to turing@mijagourlay.com

WEB SITE

http://www.mijagourlay.com/

AUTHOR

Written and Copyright (C) 1997 by Michael J. Gourlay