A Review of *Turing's World 3.0*

Charlton Wang

May 30, 1995

Abstract

Turing's World 3.0[1] is a useful little program for learning and understanding Turing machines and Finite State Machines (FSMs). The manual to *Turing's World 3.0* is short and easy to read providing both a good introduction and a brief history of Computability Theory as well as sufficient documentation on the *Turing's World 3.0* program itself. The *Turing's World 3.0* program is fairly easy to use and numerous pre-made machines and tapes have been included as instructive aids to the examples given within the text. However, in spite of its suitability for solving most, if not all, of the interesting problems and exercises presented in the book, *Turing's World 3.0*'s adherence to the conventional representation of machines along with several minor user interface shortcomings makes it difficult to use when dealing with more complex machines.

1 *Turing's World 3.0*: The Book

As a basic introduction to computability theory, *Turing's World 3.0* is quite good. A mere 17 pages are used to provide this introduction which presents both a brief and interesting historical overview of computability theory as well as the principal concepts involved (such as Turing machines and FSMs).

Following the introduction is the reference manual to the *Turing's World 3.0* program. The manual describes the program in detail and provides several "walk-through" examples. It is my feeling, however, that one could skip these sections and easily figure out how to use *Turing's World 3.0* in full, perhaps missing only the explanations of features whose behaviors are not intrinsically obvious (such as the Process Tree and the significance of the different uses of fonts).

[1] Barwise, Jon & Etchemendy, John. *Turing's World 3.0: An Introduction to Computability Theory*. Stanford: CSLI Publications, 1993, 123 pages.

The second last chapter describes FSMs in a little more detail and a discussion of nondeterministic machines is provided. The last chapter contains a plethora of exercises that deal with some fun and interesting problems.

Turing's World 3.0 is well written and very readable. Its goal as an introduction to computability theory is successful and is done without the technical jargon that one might expect to find. The fact that it can be read within a few hours makes it even more worthwhile to read—if not from the technical perspective then from the historical one. For more knowledgeable readers, *Turing's World 3.0* has a brief section on further readings with annotated book references.

2 *Turing's World 3.0*: The Program

2.1 Features

The second part of the *Turing's World 3.0* package, which I will describe in more detail (since this is what we're interested in) is the program. Like its hardcopy companion, the *Turing's World 3.0* program is also very simple. The interface is intuitive and easy to use and all the features it possesses can be accessed through the use of menus and buttons in a toolbar (see Figure 1). As an introductory tool or as a tool for drawing and analyzing simple machines (and perhaps some complex ones, too), *Turing's World 3.0* is certainly very suitable.

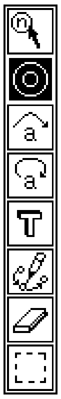

Figure 1: *Turing's World 3.0*'s toolbar. (from top to bottom) Node creation tool, acceptance node tool, arc tool, arc tool (for rounder arcs), text annotation tool, redraw tool, erase tool, rectangular selection tool.

A list of *Turing's World 3.0*'s features includes:

- Basic ability to save, open, and print machines and tape files
- Ability to edit tapes, enter symbolic logic symbols (glyphs), and define custom alphabets
- State diagram drawing features such as placing nodes, drawing arcs, labeling arcs, annotating the state diagram, cutting, pasting etc.
- Execution-related controls such as running a machine at various speeds, pausing, single-stepping, and reverse execution
- Ability to use wildcard character that matches all symbols
- Ability to display Process Tree, Tuples, Tape Description, and Time-Space Count
- Ability to use submachines
- Ability to set machine type (that is, Turing, one-way FSM, two-way FSM, deterministic, nondeterministic)
- Keyboard accelerators

The layout of the *Turing's World 3.0* program is clean and logical. The toolbar resides on the left side of the screen, the tape on the bottom, and the state diagram in the remaining space (with a tuples section for each machine if the option is selected).

In dealing with nondeterministic machines, *Turing's World 3.0* will display several concurrent copies of the tape with each one corresponding to a particular execution path. When one of the processes (or paths) has terminated unsuccessfully, the corresponding tape disappears. During execution, one cell from each tape will be highlighted. These highlighted cells represent the tape items currently being read by the machine (for a particular node).

A problem with using graphical tapes is that they can become quite long and, thus, may not be displayed in their entirety within the width of one screen. *Turing's World 3.0* has a special feature called the "tape description" that deals with this situation. This description employs an encoding scheme similar to that of the run-length encoding (RLE) scheme in which consecutive occurrences of the same symbol are reduced to number/symbol ordered pairs. An example of *Turing's World 3.0*'s tape description can be found in Figure 2.

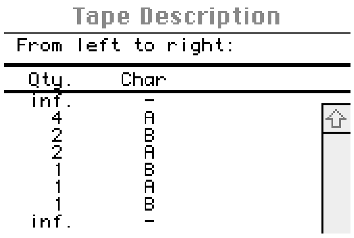

Figure 2: *Turing's World 3.0*'s "Tape Description" feature employing RLE-like encoding scheme.

Turing's World 3.0 also has a method of displaying machines in the conventional textual format which they call "Tuples":

`<State1> <Symbol> <Action> <State2>`

States are represented by node numbers, symbols are represented by characters defined in the alphabet, and actions are represented by either a left arrow, right arrow, or a printable character if the machine is a Turing machine. Figure 3 shows an example of *Turing's World 3.0*'s Tuples.

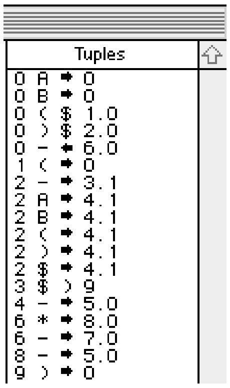

Figure 3: *Turing's World 3.0*'s "tuples" display.

Turing's World 3.0's ability to use submachines is a great organizational tool that helps to "decompose" a large state diagram into a network of logical units. An entire machine can be converted into a submachine and single nodes within a machine can be edited into submachines. An example of the use of submachines can be seen in Figure 4 which is the state diagram for one of the pre-drawn example machines known as the "Parenthesis Checker." This figure also shows *Turing's World 3.0*'s annotations feature in use—a practice strongly preached throughout the manual as a matter of good style.

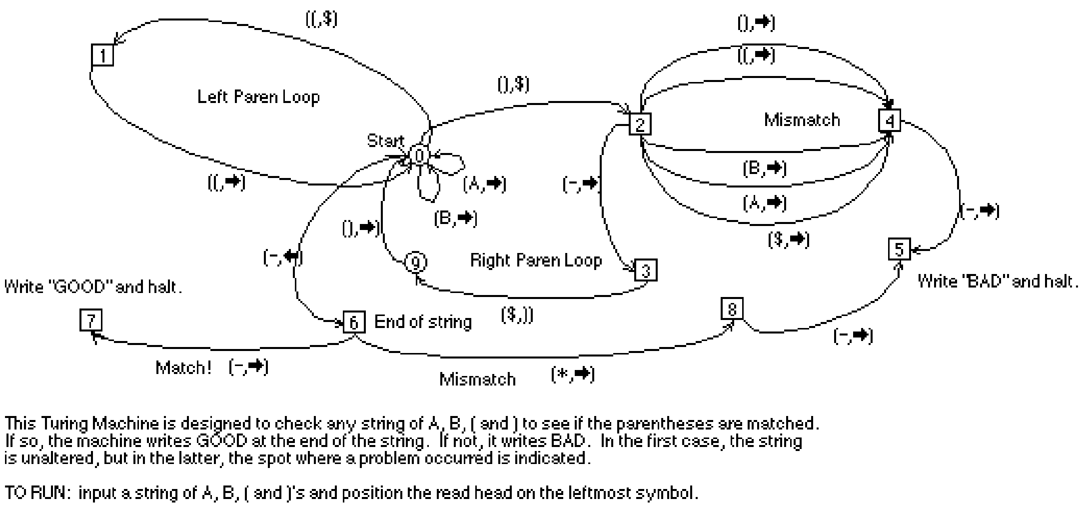

Figure 4: The "Parenthesis Checker" state diagram with submachines (represented by square nodes). Double clicking on a square node will display the submachine in a new window.

2.2 What's Missing?

Turing's World 3.0's simplicity and easy to use also has its down sides—it's limited in the number of things that it can do. As a result, several key features that one may expect to find are missing.

Turing's World 3.0 adheres to the conventional "node-arc" display method of FSMs, yet anyone who has looked at a slightly more complex state diagram laid out in such fashion knows it is nearly impossible to understand and follow. This is particularly true when many arcs and arrowheads are drawn and the limited resolution of the CRT display results in overlapping graphics (see Figure 5). This is more of a design problem as opposed to a missing feature. A zoom facility to increase the resolution of the selected parts of the machine would be useful.

Another feature that one might have expected to find that is absent is a "snap-to-grid" function or some sort of other alignment facility.

Though *Turing's World 3.0* allows the use of annotations, it does not

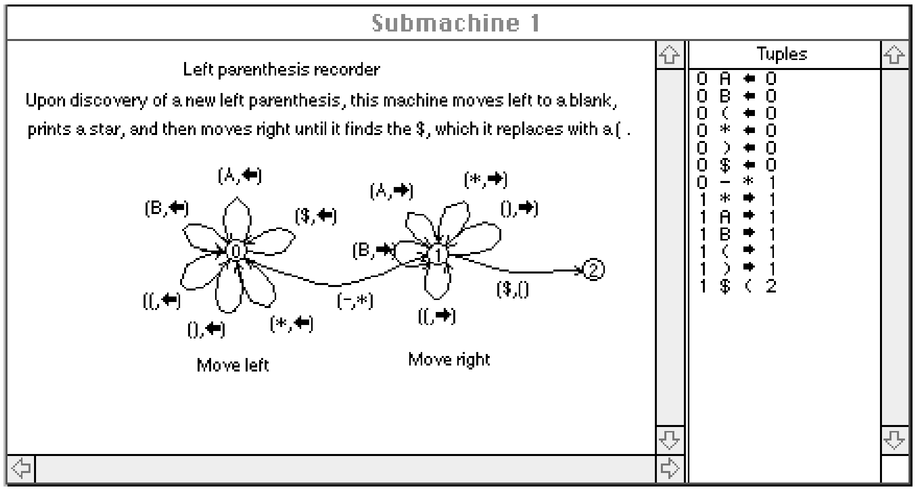

Figure 5: A submachine of the "Parenthesis Checker" with its tuples sub-window. Although this diagram is simple, it is visually convoluted. Try to find the entry point.

allow for labeling directly within the nodes. Thus, any labeling that occurs must be done beside a node—something particularly annoying when there are many arcs.

Turing's World 3.0 can give you a tape description from a tape, but it can not give you a tape from a tape description which is a pity expanding a tape description is clearly much better done by a computer than a human. Thus, you can not create your own tape with a text editor and have *Turing's World 3.0* read the file as a valid tape.

One other aspect of *Turing's World 3.0* that proves to be a little clumsy is the way it handles submachines. In order to use a submachine, you must actually embed the submachine within the main machine. In other words, there is no way for you to say, "This is a submachine that should be found in file xxxx," or, perhaps even better, "This is a submachine that should reference the machine module called xxxx found in the file xxxx." In fact, ideally we want no mention of files at all in a submachine reference; they should have symbolic names within the program, which are then mapped to files by another table (that way we can easily remap all the submachines to different files).

A few more features that might have proven useful include:

- The ability to renumber nodes

- The ability to create annotations in various fonts and rotate them through arbitrary angles
- An arbitrary selection tool that allows one to select any arbitrary region of the state diagram

This list is by no means exhaustive—there are many more features missing that have not been included in this list (I will include them in another document to be written at a later time). Despite these few minor shortcomings, however, we must keep in mind why *Turing's World 3.0* was created. It was created as an introductory learning tool. For that, it certainly does very well and its shortcomings can be ignored. For $26.95, who can complain?

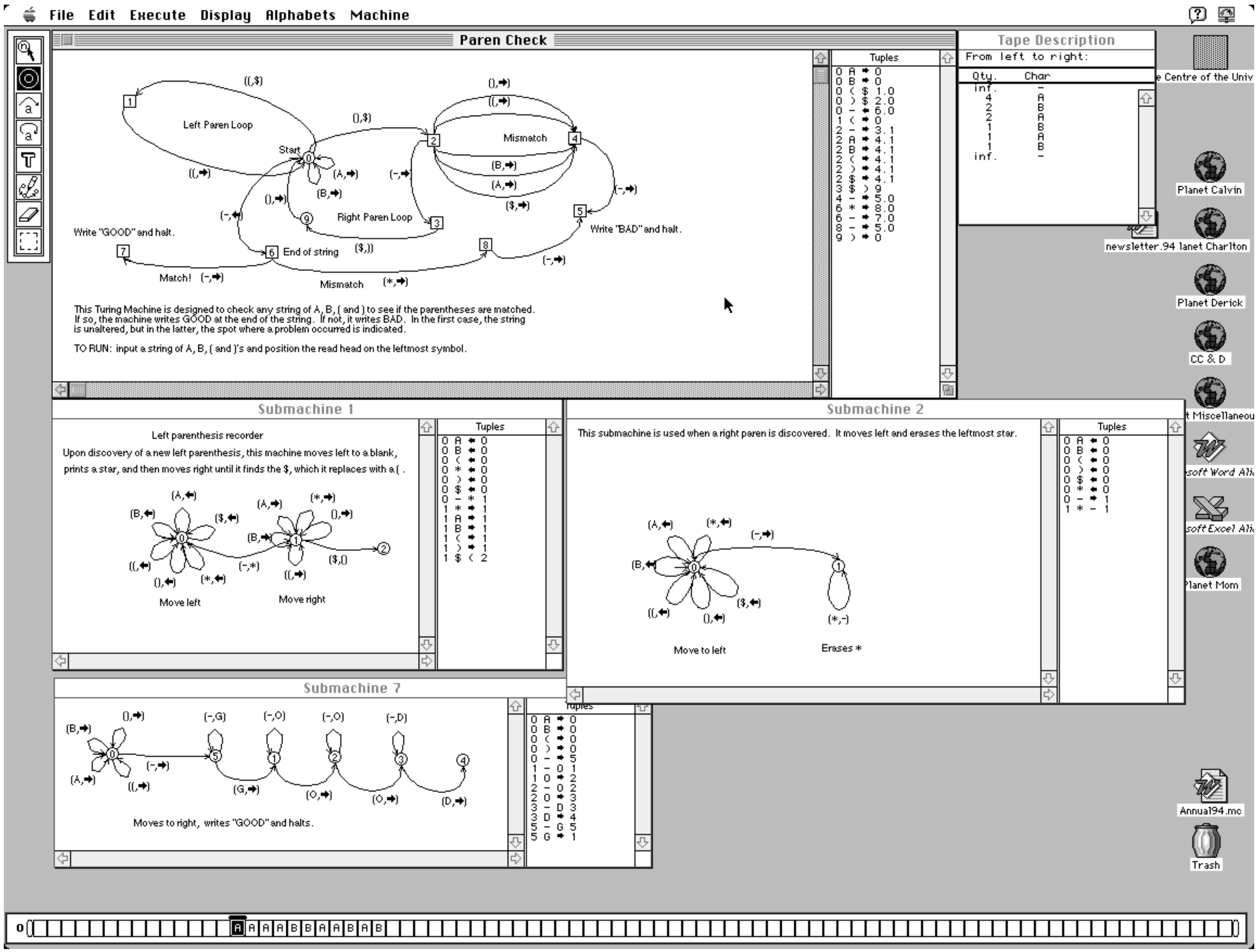

Figure 6: An overall view of *Turing's World 3.0* components at work.