TMD ReadMe

Adam Yedidia

April 8, 2016

This document is intended for users who want to know how to run or compile a TMD directory, without necessarily knowing how to program one themselves. To learn how to create a valid TMD directory, read `tmd_doc.pdf`.

The top-level representation of TMD is a program written in the TMD language, which is a language designed to give the user a simple interface with which to program a multi-tape, 3-symbol Turing Machine with a function stack. Each of the many tapes in the execution of the program is infinite in one direction.

TMD code can be processed in two ways. First, it can be *interpreted*; that is, it can be directly evaluated line-by-line. This is generally done to verify a program's correct behavior, and to correct errors which would result in thrown exceptions in the interpreter but might lead to undefined behavior in the compiled Turing machine (because the compiled Turing machine is optimized for parsimony, whereas the interpreter need not be).

Second, TMD code can be *compiled* down to a description of a single-tape, 2-symbol Turing machine. It is highly recommended, however, to first interpret any piece of TMD code before compiling it, because the interpreter is much better for catching programming errors. The compiler is general-purpose and not restricted to compiling the programs discussed in this thesis. It is optimized to minimize the number of states in the resulting Turing machine, not to make the resulting Turing machine time- or space-efficient.

1 Preparation

Before a TMD directory can be processed in any way, first it must exist in a place where it can be found. Make sure that the target directory is located at:

`parsimony/src/tmd/tmd_dirs/`

(If you created your TMD directory via compilation from Laconic, it will already exist in the right place.)

Then, BEFORE running any of the commands described below, navigate to:

`parsimony/src/tmd/tmd_meta/`

2 Interpretation

To interpret a TMD directory—to run the code as a TMD program, and see what would happen to a multi-tape Turing machine with a stack if the TMD program's commands were run explicitly—run the command:

`python tmd_interpreter.py` [name of TMD directory]

After this command is run, the interpreter will dump a multi-tape TM history to standard output. If you'd prefer that not happen, the `-q` (quiet) flag will cause the interpreter to not output any TM history. The `-s` (max steps) flag, followed by an integer k, will cause the interpreter to only run k TMD commands before stopping. Finally, the `-f` (file output) flag will cause the interpeter to dump the TM history to a history file. The `-f` flag can only be used if the `-s` flag is also enabled. Watch out—if you run the TM for many steps, this can create some truly enormous files! The created TM history file will show up at:

`parsimony/src/tmd/tmd_histories/`[name of TMD directory]`_history.txt`

3 Compilation

To compile a TMD directory—to transform it into a 2-symbol Turing machine—run the command:

`python tmd_compiler_2s_tm_compiler.py` [name of TMD directory]

This will cause a parsimonious 2-symbol Turing machine with equivalent behavior to your TMD program to appear at:

`parsimony/src/tm/tm2/tm2_files/`[name of TMD directory]`.tm2`

If you'd prefer to transform your TMD directory to a 4-symbol Turing machine, run the command:

`python tmd_compiler_4s_tm_compiler.py` [name of TMD directory]

This will cause a 4-symbol Turing machine with equivalent behavior to your TMD program to appear at:

`parsimony/src/tm/tm4/tm4_files/`[name of TMD directory]`.tm4`

Note that unlike the 2-symbol compiler, the 4-symbol compiler is not parsimonious; the 4-symbol Turing machine you'll end up with will have plenty of states. This option is primarily intended for if you want to watch a Turing machine execute which can be thought of as an equivalently-behaved but much-easier-to-read version of the 2-symbol machine. The lack of efficiency in states in the 4-symbol machine comes from the fact that the introspection algorithm is harder to implement with 4 symbols than with 2, and people are generally not interested in results relating to 4-symbol Turing machines in any case.