Daedalus Documentation

1 Syntax

Currently, the syntax allows for 6 instructions. Line labels are supported and required for some instructions

1.1 Instructions

The current list of instructions are:

- `Accept`
- `Reject`
- `If` *`symbol`* `Goto` *`label`*
- `Write` *`symbol`*
- `Goto` *`label`*
- `Move` *`direction`*

1.1.1 Accept

When the program reaches this state, the Turing Machine will halt in the `Accepted` state.

1.1.2 Reject

When the program reaches this state, the Turing Machine will halt in the `Rejected` state.

1.1.3 If-Goto

When the program reaches this state, the program will jump to *`label`* if the symbol currently being read by the Turing Machine is *`symbol`*, otherwise the program will continue to the next instruction.

1.1.4 Write

When the program reaches this state, the Turing Machine will write *`symbol`* to the current position of the Turing Machine's head on the tape.

1.1.5 Goto

When the program reaches this state, the program will jump to *`label`* unconditionally.

1.1.6 Move

When the program reaches this state, the Turing Machine will move the given direction (*`r`* for right or *`l`* for left).

1.2 Labels

Labels must be identified by being placed on their own line and ended by a colon. They cannot contain spaces, however they can include and begin with numbers, underscores, and any other special symbol. A label in a Daedalus program might look like

```
this_is_a_label:
write 1
move r
goto this_is_a_label
```

1.3 Multiple Files

1.3.1 In Project

In a single *Daedalus* directory, there can be multiple *.ddls* files. There must be a single file titled *main*. This is considered the entry point for the Daedalus program. In addition, all files in the directory will be appended to this file for compilation. To reference the code in these other files, a standard notation is to put a label at the beginning of the file labelled *`call_`*(file name) and end each path of execution with *`return_`*(file name). It should be considered bad practice to *`accept`* inside of a function as it should be deferred to the main function to *`accept`*. However, a function still may *`reject`* if it was not called with proper syntax. If a file is not included but called with *`goto`*, it will throw an error as the label will not be defined, Conversely, if you include a file but don't reference it, its return call will not be defined and will throw an error. Importantly, the files in the directory must all have the same alphabet.

1.3.2 Standard Library

In addition to being able to include multiple files in the directory, there are certain files included with the distribution of *Daedalus*. These files are included in a different way than the files in a single directory. Firstly, they do not need to be included in the same directory as the project. To include them, it's only needed to *`goto`* their call label which has the format *`call_standard_`*(operation). Secondly, the standard library functions can be used with any *Daedalus* project so long as the alphabets have the same number of symbols. Keep in mind that the order of symbols in the alphabet will be maintained.

2 Daedalus Files

A Daedalus file uses the file extension *.ddls*, its intermediate language uses the extension *.wb1*, and the final Turing Machine file uses *.tm*. A *Daedalus* file must begin with the symbols used by the Turing Machine, each separated by a space on the same line. All symbols must be a single character. The first symbol designated will be interpreted as the blank symbol.

3 Example Code

The following is an example of a program for unary addition.

```
0 1
start:
if 1 goto found_num
move r
goto start
found_num:
move r
if 0 goto concat
goto found_num
concat:
write 1
move r
if 1 goto seek_end
reject
seek_end:
move r
if 0 goto zero_end
goto seek_end
zero_end:
move l
write 0
accept
```