Problem Set 1

Answers are to be submitted via Blackboard. Please submit your answers as an archive username.tar.gz containing a PDF file with answers to theoretical questions, and a code directory like the provided one, containing all files required to build your solution.

1 DFA for a small language

In this exercise, we will create a scanner to recognize a minimalistic language for line drawings written as postscript files. It consists of these three types of statements, which are all terminated by a newline character (`'\n'`):
`dx=`*(integer)*
`dy=`*(integer)*
`go`

The character sequences `'dx='`, `'dy='`, and `'go'` are fixed, integers consist of a sequence of digits with an optional `'-'` character for negative values.

1.1

Draw a deterministic automaton (DFA) which accepts all three statement types.

1.2

Write a regular expression corresponding to your automaton.

1.3

Does your automaton use a minimal number of states?
Justify your answer.

2 Implementation

In the file archive `ps1_skeleton.c`, you will find an implementation of this language which tracks a pair of (x,y) coordinates which are initialized to the center of a page, and two values (dx,dy). The assignment statements in our language set the (dx,dy) values respectively, and the 'go' statement alters the (x,y) coordinates by (dx,dy), drawing a line from the previous position to the new one.

The main function already implements the table-based DFA simulation algorithm, but its transition table is empty. When extended with a correct automaton in the table, this program will emit drawing instructions in postscript, which can be converted to a PDF document. The archive also includes a sample file of commands that draw a spiral (`spiral.txt`), which can be used to verify your solution thus:

```
cat spiral.txt | ./scanner | ps2pdf - spiral.pdf
```

2.1

Convert your DFA to table format, and implement it in the `initialize_transition_table` function found in `scanner.c`.
You can dimension the table to match the size of your automaton by modifying the `N_STATES` macro, and assign the accepting state using the `ACCEPT` macro.