Overview

The purpose of this assignment is to further your understanding of the equivalence between *nondeterministic* and *deterministic* finite automata. Recall that a *deterministic finite automaton* (DFA) is defined as a 5-tuple of the form $(Q, \Sigma, F, q_0, \delta)$ where:

- Q is a finite set of states;
- Σ is an input alphabet;
- $F \subseteq Q$ is a set of final states;
- $q_0 \in Q$ is the initial (or start) state; and
- $\delta : Q \times \Sigma \to Q$ is a transition function

while a *nondeterministic finite automaton* (NFA) is a 5-tuple of the form $(Q, \Sigma, F, q_0, \delta)$ where:

- Q is a finite set of states;
- Σ is an input alphabet;
- $F \subseteq Q$ is a set of final states;
- $q_0 \in Q$ is the initial (or start) state; and
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ is a transition function

Also recall both from the reading and lecture, that these two structures are equivalent to each other. In otherwords, for every DFA there is an equivalent NFA and vice versa. An important aspect of the proof of this theorem is its structure – being a *constructive proof*, it in essence outlines an algorithm which given an NFA as its input, will generate the equivalent DFA. Your goal for this assignment is to realize this construction as an actual program.

Specification

Define a program called `NFAConvert` which will accept as its input the name of a file containing a textual representation of a *nondeterministic finite automaton*, and will output in the same format a textual representation of the equivalent *deterministic finite automaton*.

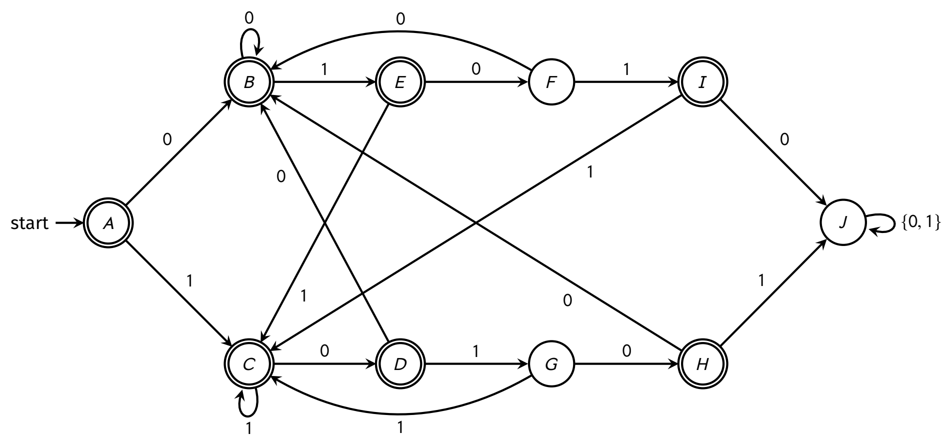
NFA Input File Specification

Because every DFA is by definition also an NFA, the input format will be the same as for your prior project. Consider the NFA, M_0 shown in the figure below:

M_0 will be represented in plain text form as follows[1]:

```
% Q
A
B
C
D
```

[1]Line numbers are included for reference only.



```
E
F
G
H
I
J
% Sigma
0
1
% F
A
B
C
D
E
H
I
% Q0
A
% Delta
A 0 B
A 1 C
B 0 B
B 1 E
C 0 D
C 1 C
D 0 B
D 1 G
E 0 F
E 1 C
F 0 B
F 1 I
G 0 H
G 1 C
H 0 B
```

```
H 1 J
I 0 J
I 1 C
J 0 J
J 1 J
```

Note that in this representation, those lines preceded by a % sign (i.e., lines 1, 12, 15, 25, and 27) are to be treated as single line comments in Java - any content on the line after the % sign is to be ignored and is entirely optional. Also note, that for this assignment you are guaranteed the following about the NFA input file:

- the NFA within it is correctly specified (i.e., is both a a NFA, and in the correct textual form);
- that the order of the NFA's constituent parts are as in the example above (which by design corresponds with how the definition of a DFA is formulated); and
- empty string transitions will designated by a single underscore `'_'` (e.g., the triple J _ A would represent an empty string transition from state J to state A).

Usage

The program `NFAConvert` is intended to be run via the following command-line invocation:

```
$ java NFAConvert input.txt
```

If no input file is specified or if the input file is not found, an appropriate error message must be displayed. For example:

```
$ java NFAConvert
NFAConvert: no input file specified

$ java NFAConvert no-such-file.txt
NFAConvert: the file 'no-such-file.txt' could not be opened
```

If a valid file is specified as input, then the output of the program will be the equivalent DFA in plain text form (using the same format as that input DFA). The DFA is to be obtained using the same algorithm discussed both in lecture and in the reading. As an example, using the file `input.txt` mentioned above, your program should behave as follows[2][3]:

```
$ java NFAConvert input.txt
% Q
{A}
{B}
{C}
{D}
{E}
{F}
{G}
{H}
```

[2]Note that in the example output there is exactly one space between the string and its results. As before, line numbers are included for reference only.

[3]Recall that states of the resulting DFA are *sets of states* of the input NFA. As a consequence, the order of their elements is irrelevant. The same is true of those constituent parts of the DFA that are themselves sets. Furthermore, for simplicity, you may use square brackets instead of braces when converting sets to string form.

```
{I}
{J}
% Sigma
0
1
% F
{A}
{B}
{C}
{D}
{E}
{H}
{I}
% Q0
{A}
% Delta
{A} 0 {B}
{A} 1 {C}
{B} 0 {B}
{B} 1 {E}
{C} 0 {D}
{C} 1 {C}
{D} 0 {B}
{D} 1 {G}
{E} 0 {F}
{E} 1 {C}
{F} 0 {B}
{F} 1 {I}
{G} 0 {H}
{G} 1 {C}
{H} 0 {B}
{H} 1 {J}
{I} 0 {J}
{I} 1 {C}
{J} 0 {J}
{J} 1 {J}
```

Compilation Notes

Your program must compile using the following command-line invocation:

```
$ javac *.java
```

Do not submit any IDE specific files and in the interest of simplicity do not use packages for this assignment. If your program does not compile using the command mentioned above you will receive no credit.