For this assignment, a finite state automaton is defined as consisting of a finite set of states, one of which is designated as the state the machine starts in and at least one other which is designated an end state, a finite set of transitions between two, not necessarily distinct, states, and for each transition, a “Write X” operation that writes a single symbol, X. A non-terminal state requires that one of its exit transitions be taken. A finite state automaton is said to “generate a string σ” if σ is a series of symbols that the automaton can produce by going from its start state to one of its end states.

Write a finite state automaton that generates the set of strings of word classes that are illustrated by the sentences in the series in (A). (Understand this to be an infinite series.)

(A)

The man danced.
The man danced on the desk.
The man danced on the desk behind the table.
The man danced on the desk behind the table under the rug.
The man danced on the desk behind the table under the rug beside the cabriolet.

⋮
Write a finite state automaton that generates the set of strings in the infinite series in (C).

\[
\begin{align*}
&\text{The man danced.} \\
&\text{The man on the desk danced.} \\
&\text{The man on the desk behind the table danced.} \\
&\text{The man on the desk behind the table under the rug danced.} \\
&\text{The man on the desk behind the table under the rug beside the cabriolet danced.} \\
&\vdots
\end{align*}
\]

\[
\begin{array}{c}
\text{start} \\
\rightarrow \\
1 \\
\rightarrow D \\
2 \\
\rightarrow N \\
3 \\
\rightarrow V \\
4
\end{array}
\]

Consider now the series that would be produced by combining the two in (A) and (C). So, that might be expressed like this:

\[
\begin{align*}
&\text{The man danced.} \\
&\text{The man danced on the desk.} \\
&\text{The man danced on the desk behind the table.} \\
&\vdots \\
&\text{The man on the desk danced.} \\
&\text{The man on the desk danced on the desk.} \\
&\text{The man on the desk danced on the desk behind the table.} \\
&\vdots \\
&\text{The man on the desk behind the table danced.} \\
&\text{The man on the desk behind the table danced on the desk.} \\
&\text{The man on the desk behind the table danced on the desk behind the table.} \\
&\vdots
\end{align*}
\]

Write a finite state automaton that generates the set of strings in this series.
A way of describing the series in (E), and the automaton that generates it, is as follows:

(1) a. Write a D N string
   b. Write a P D N string \( n \) times, \( 0 \leq n \leq \infty \)
   c. Write V
   d. Write a P D N string \( n \) times, \( 0 \leq n \leq \infty \)
   e. stop

(I want you to understand these instructions to be ordered. So, do (a) before all the others. Do (b) before all the others, except (a). And so on.) Notice that (1b) and (1d) do the same thing. That means that I could also describe the series in (E) this way:

(2) a. Write a D N string
    b. Do subroutine PP
    c. Write V
    d. Do subroutine PP
    e. stop

(3) subroutine PP: Write a P D N string \( n \) times, \( 0 \leq n \leq \infty \)

The subroutine PP is not part of the ordered list in (2); it is an independent process that can be “called.” Both (1) and (2) can describe the series in (E): they generate the same set of strings. Is it possible to express (2) with a finite state automaton? If so: show it to me. If not: explain why.

Finite State Grammars cannot work in the way that (2) describes. One initial problem that stands in the way is that it isn't possible to go out of the machine into a subroutine without writing a symbol, and therefore to exit the list in (2) from either the second or fourth steps would require that it write the same symbol. If that symbol is P, then P isn't in the subroutine. If that symbol is V, as it needs to be when exiting from step two, then we'll wrongly put a V after step four.

We can avoid this part of the problem by admitting into our vocabulary of symbols the empty string, (ε). But that only removes one of the problems that stands in the way of building a finite state grammar that does (2). The other problem is that we can't get back out of the subroutine into the right position in the list in (2) without knowing where we entered the subroutine from. That might be easier to see by looking at the attempt to build a finite state automaton with the form of (2) below.
This generates DNPDN as well as DNVPDNV, and many other strings of that kind. What goes wrong is that when in state 5, after going through the PDN loop, the machine needs to know whether it entered the PDN loop from state 3 or 4 to know which state to exit to. The memory of a finite state automaton is finite. It can “remember” how it got into a loop if there is only one entrance to that loop. That is an ingredient in the information flow in (2) that goes beyond what a finite state automaton can do.

The grammars in (1) and (2) produce the same set of strings. What evidence, if any, would distinguish them?