ANSWER:--

GIVEN THAT:--

- As for as my knowledge in theory of computations...

  Letâ€™s define both things.

  First of all, a (deterministic) finite state machine MM is a 5-tuple (Q,Î£,Î´,q0,F).

  - Q is a set of states
  - Î£ is the input alphabet
  - Î´ is the transition function, Î´:QÃ—Î£â†’Q. This is what tells you what new state you go to when you are in a particular state and see a particular symbol.
  - q0âˆˆQ is the start state.
  - FâŠ†Q is the set of accepting states.

  Now, the basic way this works is you read the input string left to right in one pass. When you see a symbol, you move to a new state based on the transition function. If when you finish reading the string, youâ€™re in an accepting state, you accept the string, otherwise, you reject it. The set of languages that finite state machines accept is the set of regular languages.

  A (deterministic) Turing machine M is a 6-tuple (Q,Î£,Î´,q0,qaccept,qreject) where

  - Q is the set of states
  - Î£ is the tape alphabet. This includes the input symbols, symbols you can write on the tape, and the blank symbol.
  - Î´:QÃ—Î£â†’QÃ—Î£Ã—{left,right} is the transition function. This means that when youâ€™re in a state and see a symbol, you go to another state, write something on the tape, and go to the left or right on the tape.
  - q0,qaccept,qrejectâˆˆQare the start state, the accepting state and the rejecting state respectively.

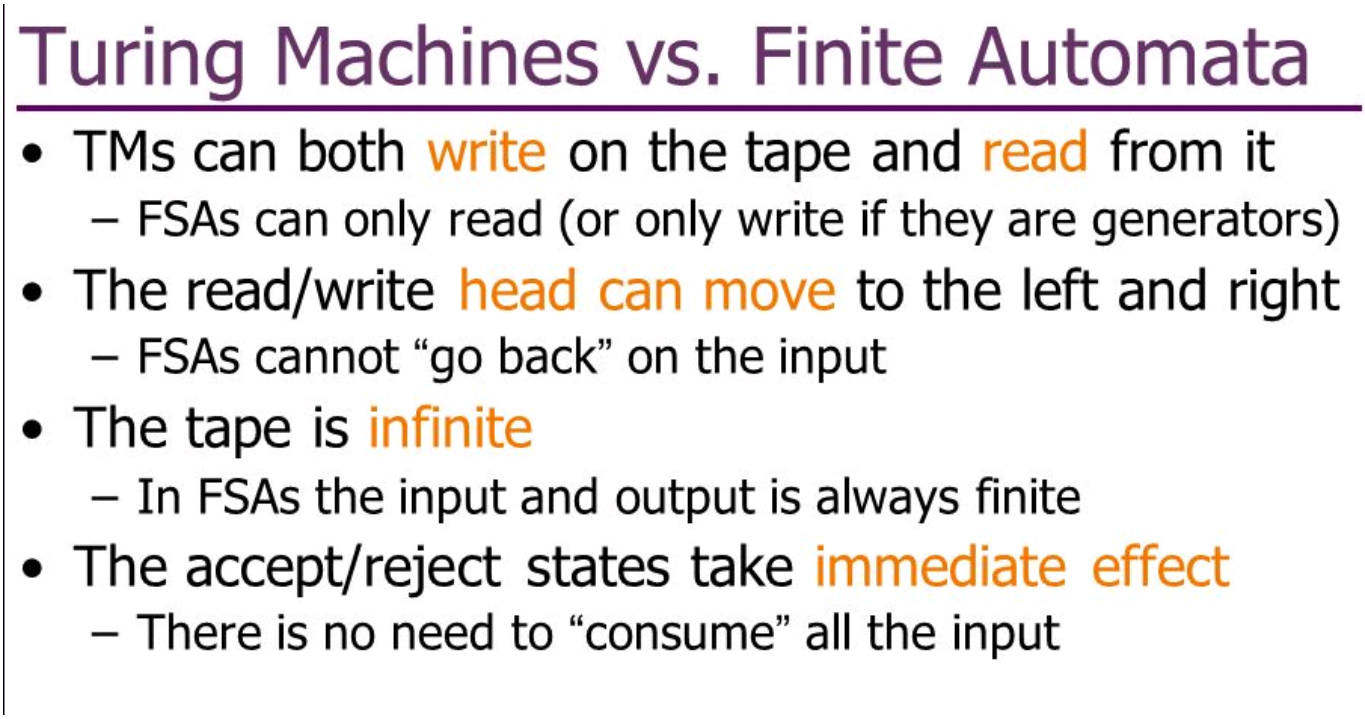
  The way a Turing machine works is you have an infinitely long tape. When you start the machine, it has its input written on a portion of the tape with trailing blanks. The machine can read a symbol and write a new symbol and go back and forth on the tape as often as it wants. We believe that any computable language can be decided by a Turing machine. (N.B. Not every language is computable!).

  The short answer is that Turing machines are more powerful because they can remember more. A 15-state finite automaton, for example, can only remember one of 15 things (â€œIâ€™ve seen 2 câ€™s in a row.â€; â€œI havenâ€™t seen any dâ€™s yet.â€; etc.). A Turing machine can remember an arbitrary number of things, by virtue of its (semi-)infinite tape (for example, â€œIâ€™ve seen *m* aâ€™s and *n* bâ€™s.â€ for every possible value of *m* and *n*). So itâ€™s not surprising that the set of computations that TMs can perform is larger than the set of computations that FAs can

  Actually any practical computer is just a DFA, which is strictly less powerful than a Turing machine is. Though the DFA could be exorbitantly complex and large.

  With finite memory you can represent only finitely many states. One of the basic requirement for Turing machine is an infinite tape, Turing machine with finite tape is strictly equivalent to a DFA.

  Ã…nd every computing machine we encountered is equivalent to or strictly lesser than Turing machines in terms of expressibility



Turing Machines vs. Finite Automata â€¢ TMs can both write on the tape and read from it â€“ FSAs can only read (or only write if they are generators) â€¢ The read/write head can move to the left and right - FSAs cannot â€œgo backâ€ on the input â€¢ The tape is infinite - In FSAs the

input and output is always finite â€¢ The accept/reject states take immediate effect - There is no need to "consumeâ€ all the input