Try doing a polynomial-time reduction from this problem to SATISFIABILITY:

$$\text{FA-Nonempty} := \{A : A \text{ is a Finite Automation that accepts at least one string}\}.$$

Try doing it in such a way that the logical expression you construct models the execution of the FA.
Use Boolean variables to represent the letters in each position of the input string ...

To perform a polynomial-time reduction from the problem FA-Nonempty (determining if a given finite automaton accepts at least one string) to Satisfiability (SAT), we can construct a logical expression that models the execution of the finite automaton. We'll use Boolean variables to represent the letters in each position of the input string and ensure that a satisfying assignment corresponds to an accepting run of the automaton.

Let's say we have a finite automaton A with Q states, an alphabet Σ, and a transition function δ.

1. Boolean Variables: For each possible configuration of the automaton at each step (state and symbol in the tape), we introduce a Boolean variable. We'll use variables q_i for the state at time i and variables x_i for the symbol in the input tape at time i.

2. Initial Configuration: We set the initial configuration of the automaton as a clause in our SAT expression. This clause would involve setting the appropriate q_0 variable to the initial state of the automaton and the x_0 variable to the first symbol in the input.

3. Transition Constraints: For each time step, we add constraints in the SAT expression that enforce the transition rules of the automaton. These constraints ensure that if the automaton is in state q_i and reads symbol x_i at time step i, it should transition to state q_(i+1) and read symbol x_(i+1) at time step (i+1) as defined by δ.

4. Accepting State: We add a clause that specifies that the automaton must be in an accepting state at some time t. This clause ensures that the automaton accepts at least one string.

5. Formula Composition: We combine all the clauses and constraints into a single logical formula in conjunctive normal form (CNF), which is the standard format for SAT instances.

Now, if the logical formula is satisfiable, it means there exists an assignment of the Boolean variables that models an accepting run of the finite automaton. If the formula is unsatisfiable, it means that no such run exists, indicating that the automaton does not accept any string.

This reduction can be done in polynomial time because it involves a linear number of variables and constraints corresponding to the number of states and time steps in the automaton. If the constructed SAT formula is satisfiable, it demonstrates that the finite automaton has an accepting run, proving the non-emptiness of the language it recognizes.