Linear Programming Model for the Minimum Consistent Finite Automaton Problem

Virginia Nicosia

May 18, 2025

1 Introduction

A Deterministic Finite Automaton (DFA) is a model of computation defined by a set of states, an alphabet (here {0, 1}), a transition function, an initial state, and a set of accepting states. A sequence of symbols is accepted if, starting from the initial state and following the transitions dictated by the sequence symbols, the automaton ends in an accepting state otherwise the sequence is rejected.

The Minimum Consistent Finite Automaton (MCFA) problem asks for a DFA with the minimum possible number of states that is consistent with a given sample of sequences. This sample consists of two disjoint sets: a set A of sequences that the DFA must accept, and a set R of sequences that the DFA must reject. The challenge is to ensure the solution is both minimal and consistent with the sample.

This report presents a Linear Programming (LP) formulation, solved with CPLEX, to construct such a DFA with minimal states.

2 LP Model

I iteratively try increasing numbers of states $n = 1, 2, \ldots,$ up to a maximum (10), and formulate the problem as an LP for each value of n; the first value of n for which the model is feasible corresponds to the minimum number of states.

2.1 Variables

- Transition Variables $trans_{i,b,j} \in \{0,1\}$: for state $i \in \{0,\ldots,n-1\}$, input bit $b \in \{0,1\}$, and next state $j \in \{0,\ldots,n-1\}$. $trans_{i,b,j} = 1$ means the automaton moves from state i to j on input b.
- Accepting State Variables $accS_i \in \{0,1\}$: $accS_i = 1$ if state i is an accepting state.
- Run State Variables $RState_{p,k,s} \in \{0,1\}$: for sequence p, at step k, whether the automaton is in state s.
- Run Accept Variables $RAccept_{p,s} \in \{0,1\}$: $RAccept_{p,s} = 1$ if sequence p ends in state s and s is an accepting state.

2.2 Constraints

The constraints are formulated to ensure that any feasible solution to the LP corresponds to a valid Deterministic Finite Automaton DFA that correctly accepts all sequences in set A and rejects all sequences in set R. This is achieved by modeling the step-by-step processing of each sequence. Let n be the current number of states being tested.

1. Deterministic Transitions: For the automaton to be deterministic, from each state $i \in \{0,\ldots,n-1\}$ and for each input bit $b \in \{0,1\}$, there must be exactly one outgoing transition to a next state $j \in \{0,\ldots,n-1\}$.

$$\sum_{j=0}^{n-1} trans_{i,b,j} = 1 \quad \forall i \in \{0,\ldots,n-1\}, \forall b \in \{0,1\}$$

   This constraint ensures that the $trans_{i,b,j}$ variables correctly define a transition function where each state i has precisely one successor state for input 0 and one for input 1.

2. Simulating Sequence Processing: For each given sequence p of length m_p, we use the $RState_{p,k,s}$ variables to track the state of the automaton at each step $k \in \{0, \ldots, m_p\}$.

   - Initial State: Every sequence p begins in the designated initial state, state 0:

   $$RState_{p,0,0} = 1 \quad \text{and} \quad RState_{p,0,s} = 0 \quad \forall s \in \{1, \ldots, n-1\}$$

   - State Evolution: If at step k the automaton is in state i and reads bit b_k, it transitions to state j if $trans_{i,b_k,j} = 1$: the variables $RState_{p,k+1,j}$ capture this evolution. The following constraints work together to ensure that $RState_{p,k+1,j} = 1$ if the automaton was in some state i at step k ($RState_{p,k,i} = 1$) and the transition $trans_{i,b_k,j} = 1$ is active:
     - For each step k from 0 to $m_p - 1$ (where m_p is the length of sequence p), and for each possible current state i and next state j:

     $$RState_{p,k+1,j} \geq RState_{p,k,i} + trans_{i,b_k,j} - 1$$

     So if $RState_{p,k,i} = 1$ (automaton is in state i at step k) and $trans_{i,b_k,j} = 1$ (the transition from i on bit b_k leads to j), then $RState_{p,k+1,j}$ is forced to be at least 1.
     - Unique State Guarantee: At every step k of processing sequence p , the automaton must be in exactly one state:

     $$\sum_{s=0}^{n-1} RState_{p,k,s} = 1$$

3. Final State Classification (Acceptance/Rejection): After processing all m_p bits of a sequence p, the automaton is in a final state s_{final}, w must then check if this s_{final} is an accepting state, consistent with whether $p \in A$ or $p \in R$.

   - The variable $RState_{p,m_p,s}$ is 1 if sequence p ends in state s, and 0 otherwise.
   - The variable $accS_s$ is 1 if state s is an accepting state, and 0 otherwise.
   - To link these, the auxiliary variable $RAccept_{p,s}$ is defined to be 1 if sequence p ends in state s and state s is an accepting state.

   $$\begin{aligned} RAccept_{p,s} &\leq RState_{p,m_p,s} \\ RAccept_{p,s} &\leq accS_s \\ RAccept_{p,s} &\geq RState_{p,m_p,s} + accS_s - 1 \end{aligned}$$

   - Consistency with Sets A and R: Based on $RAccept_{p,s}$, we enforce the overall acceptance or rejection:
     - If sequence $p \in A$ (it should be accepted), then it must end in some accepting state:

     $$\sum_{s=0}^{n-1} RAccept_{p,s} = 1$$

     - If sequence $p \in R$ (it should be rejected), then it must not end in any accepting state:

     $$\sum_{s=0}^{n-1} RAccept_{p,s} = 0$$

2.3 Objective

Since we're only interested in feasibility for a given n, the objective is:

$$\min 0$$

That means that we're not trying to minimize or maximize any particular value or expression but just to find any solution that satisfies all the constraints defined for the current number of states n.

3 Conclusion

This approach successfully models the MCFA problem using LP and ensures minimality by iterating over increasing values of n until a feasible solution is found. A notable difference in performance was observed between the LP model solved with CPLEX and the CP model implemented with Gecode: within the given time constraints, the LP formulation was able to find optimal solutions for a larger set of instances.