Separating Strings with Automata and SAT

1 Introduction

In this sketch we study how to use SAT to construct a non-deterministic finite automata (NFA) that separates two finite sets of strings. This yields a decision procedure for calculating the metric distance between two regular sets with respect to the (inverse) automata size metric.

2 Preliminaries

An alphabet Σ is a finite set of unique symbols. Let $\Sigma^\star$ denote the set of all finite strings consisting of letters from Σ. A language $\mathcal{L} \subseteq \Sigma^\star$ is then a countable set of strings of Σ. Often Σ is implicitly assumed.

A non-deterministic finite automata (NFA) $\mathcal{A}$ is represented as a tuple $\mathcal{A} = (\Sigma, Q, \Delta, S, F)$ where Σ is a finite alphabet, Q is a finite set of states, $\Delta\colon (Q \times \Sigma) \to 2^Q$ is the transition function, $S \subseteq Q$ is the inital states, and $F \subseteq Q$ is a set of final states.

For some string w we say that $\mathcal{A}$ accepts w if there exists a sequence of transitions on which $\mathcal{A}$ ends in a final state when reading w. We abuse notation and say that $\mathcal{A}$ accepts w if $\mathcal{A}(w) = 1$. Similarly, we say $\mathcal{A}(w) = 0$ if $\mathcal{A}$ rejects w.

Let w_i denote the ith letter of the string w, and ε be the empty string.

3 Separating Sets

The question that this sketch attempts to answer can be formalized as follows:

Question 1. *Given a positive set $P \subseteq \Sigma^\star$ and negative set $N \subseteq \Sigma^\star$ with P and N disjoint, does there eixst an NFA $\mathcal{A} = (\Sigma, Q, \Delta, S, F)$ with $|Q| = n$ such that for all $u \in P$ in the positive set $\mathcal{A}(u) = 1$, but for all $v \in N$ in the negative set $\mathcal{A}(v) = 0$.*

We achieve this by constructing a boolean satisfiability formula that encodes P and N, and is satisfiable if and only if such $\mathcal{A}$ exists. There are several high-level insights that we leverage:

(1) NFAs can be represented as directed multi-edge graphs where each edge is labeled by one letter from Σ. In other words, let $e_{i,j,\sigma}$ be an indicator variable encodes the indicator of a transition from state q_i to state q_j on the letter σ.

(2) For each $u \in P$, we can create a formula that forces a sequence of edge walks resulting in a final state in $\mathcal{A}$. Similarly for each $v \in N$ we can encode a sequence that will force a rejection of v in $\mathcal{A}$.

For any $w \in \Sigma^\star$, consider the following encoding:

$$\pi_w \equiv \bigwedge_{1 \le t < |w|} \left(\bigvee_{i \neq j \neq k} e_{i,j,w_t} \wedge e_{j,k,w_{t+1}} \right)$$

This forces the existence of a path on $\mathcal{A}$ for w. Additionally, to force w to stop on a final state, we may either have every state be a final state, or just set the following formula:

$$\varphi_w \equiv \left[\bigwedge_{i \neq j} (e_{i,j,w_1} \implies s_i) \right] \wedge \left[\bigwedge_{i \neq j} \left(e_{i,j,w_{|w|}} \implies f_j \right) \right]$$

Essentially, this forces each w to begin processing on an initial state as indicated by s_i, and end on a final state as indicated by f_j. Thus, the conjunction $\pi_w \wedge \varphi_w$ will be true if and only if $\mathcal{A}$ accepts w. Hence, the satisfaction of its negation will then force $\mathcal{A}$ to reject w. We leverage this in order to define over P and N:

$$\Phi_{P,N} = \left[\bigwedge_{u \in P} (\pi_u \wedge \varphi_u) \right] \wedge \left[\bigwedge_{v \in N} \neg (\pi_v \wedge \varphi_v) \right]$$

Thus, $\Phi_{P,N}$ defines a formula that will make $\mathcal{A}$ to accept P and reject N.

The variables of the $\Phi_{P,N}$ are the indicators for edges of form $e_{i,j,\sigma}$ and final states of form f_j. Suppose that Σ is known. If $\Phi_{P,N}$ is satisfiable, then $\mathcal{A} = (\Sigma, Q, \Delta, S, F)$ can be extracted as:

$$\begin{aligned} Q &= \{q_1, \ldots, q_n\} \\ \Delta &= \{((q_i, \sigma), q_j) \ : \ e_{i,j,\sigma} = \top\} \\ S &= \{q_i \ : \ s_i = \top\} \\ F &= \{q_j \ : \ f_j = \top\} \end{aligned}$$

Note that Δ is slightly overloaded.