State Complexity of Deterministic Finite Automata for Tagged Unions

Evan Leach

August 16, 2025

Abstract

We study generalizability of problems in the toy setting of DFAs, establishing sharp upper and lower bounds on the state complexities of tagged unions of languages. We also give a complete characterization of when the upper bound on state complexity can be beaten.

1 Tagged Unions

Some classes of problems are more easily *generalizable* than others, and we can measure this generalizability by seeing how the size of a minimal program that computes multiple languages relates to the sizes of the minimal programs which compute each language. To make precise the notion of computing multiple languages, we introduce the following definition:

Definition 1. Let Σ denote an alphabet and A denote a finite index set with $\Sigma \cap A = \emptyset$. Suppose that for each $\alpha \in A$, we have a language L_α over Σ. The **tagged union** of these languages is a language over $\Sigma \cup A$ defined by

$$\bigsqcup_{\alpha \in A} L_\alpha = \bigcup_{\alpha \in A} \{\alpha w : w \in L_\alpha\}. \tag{1}$$

Suppose that for each language L_α, the program P_α decides L_α. Then one can decide $L = \bigsqcup_{\alpha \in A} L_\alpha$ by taking an input string αw and running P_α on w. We will call this the **naive algorithm.**

The naive algorithm tells us that the time complexity of L, for instance, relates in a simple way to the time complexity of each L_α: if each L_α has time complexity $\Theta(f_\alpha(n))$, then the time complexity of L is $\Theta(\max_{\alpha \in A} f_\alpha(n))$. For program size, on the other hand, the situation becomes more complicated. The naive algorithm shows us that we can upper-bound the minimal program size of L by the sum of the minimal program sizes of each

L_α plus some constant overhead. However, one can also imagine similar languages whose tagged union can be decided by a smaller program which generalizes these languages.

We will study these phenomena in the toy setting of DFAs, as the *state complexity* of these automata is a useful proxy for program size, and we have good tools for proving lower bounds. This model of computation is, of course, far too weak to make any serious claims, but it will serve to illustrate when programs can or cannot generalize classes of problems.

2 DFAs for Tagged Unions

Deterministic finite automata are remarkably *inefficient* at generalization. This is because, roughly speaking, any DFA deciding a tagged union of languages must contain a DFA for each language in the union. We can make this idea precise in the following theorem:

Theorem 1. *Suppose* $L = \bigsqcup_{\alpha \in A} L_\alpha$ *and the* DFA $M = (Q, \Sigma \cup A, \delta, q_0, F)$ *recognizes* L. *Fix* $\alpha \in A$, *and let* Q_α *denote the set of states in* Q *which are reachable in* M *from* $\delta(q_0, \alpha)$ *by a word in* Σ^*. *Then the* DFA

$$M_\alpha = (Q_\alpha, \Sigma, \delta|_{Q_\alpha \times \Sigma}, \delta(q_0, \alpha), F \cap Q_\alpha) \tag{2}$$

is well-defined and recognizes L_α.

Proof. We first verify well-definedness. If $q \in Q_\alpha$, then q is reachable from $\delta(q_0, \alpha)$ by some $x \in \Sigma^*$ and so $\delta(q, \sigma)$ is reachable by $x\sigma$ for any $\sigma \in \Sigma$. Thus $\delta|_{Q_\alpha \times \Sigma}$ maps into Q_α. Notice also that $\delta(q_0, \alpha)$ is reachable from itself by $\varepsilon \in \Sigma^*$, so $\delta(q_0, \alpha) \in Q_\alpha$.

To see that M_α recognizes L_α, notice that

$$M_\alpha \text{ accepts } w \iff M \text{ accepts } \alpha w \iff \alpha w \in L \iff w \in L_\alpha.$$

□

Lemma 1. *If* L *is nonempty, then we have for each* $\alpha \in A$ *that* $q_0 \notin Q_\alpha$.

Proof. Suppose for the sake of contradiction that $q_0 \in Q_\alpha$ for some $\alpha \in A$. Then q_0 is reachable from $\delta(q_0, \alpha)$, so there exists a string v such that the computation of M on αv ends on q_0. Since L is nonempty, we can choose some $w \in (\Sigma \cup A)^*$ such that M accepts w. As $w \in L$, we know that the first letter β of w lies in A. Thus $\alpha v w \notin L$, as it contains an element of A past the first position. This contradicts the fact that M accepts $\alpha v w$.

□

3 State Complexity

To make sense of the size of DFAs and the languages they recognize, we now introduce two definitions:

Definition 2. We define the **state complexity** of a DFA $M = (Q, \Sigma, \delta, q_0, F)$ by

$$|M| = |Q|.$$

Definition 3. We define for a language L the **DFA-state complexity** by

$$|L|_{\mathsf{DFA}} = \min \{|M| : M \text{ is a DFA which recognizes } L\}. \tag{3}$$

These definitions lead us to consider *minimal* DFAs for a language L, which have a minimal number of states among all DFAs recognizing L. We can extend Theorem 1 using this concept, but we will first introduce a definition and prove a useful lemma:

Definition 4. Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, we call two states $q_1, q_2 \in Q$ **equivalent** in M if for any string $y \in \Sigma^*$, the computation of M on y starting on q_1 accepts if and only if the computation of M on y starting on q_2 accepts.

Lemma 2. *Let M and M_α be defined as in Theorem 1, and fix $\alpha \in A$. Then any two states $q_1, q_2 \in Q_\alpha$ are equivalent in M_α if and only if they are equivalent in M.*

Proof. Clearly, two states equivalent in M are also equivalent in M_α. To prove the reverse direction, suppose $q_1, q_2 \in Q_\alpha$ are equivalent in M_α. Then we know that for any $y \in \Sigma^*$, the computation of M on y starting on q_1 accepts if and only if the computation of M on y starting on q_2 accepts. We therefore just need to show that this also holds for strings in $(\Sigma \cup A)^* \setminus \Sigma^*$.

To this end, suppose that y contains at least one symbol in A. Since q_1 is reachable from $\delta(q_0, \alpha)$, there exists a string v such that the computation of M on αv ends at q_1. This implies that $\alpha v y \notin L$, since it includes a symbol in A past the first position. The computation of M on y starting at q_1 therefore must reject, and an analogous argument shows that the computation of M on y starting at q_2 also rejects. Thus q_1 and q_2 are equivalent in M.

□

We are now ready to prove the theorem:

Theorem 2. *If the DFA M in the context of Theorem 1 is minimal, then so is each M_α.*

Proof. If L is empty, then $|M| = 1$ and the claim holds trivially. We therefore consider the case where $L \neq \emptyset$.

Suppose for the sake of contradiction that some M_α is not minimal. Then by the Myhill-Nerode theorem, there exist two distinct states $q_1, q_2 \in Q_\alpha$ which are equivalent in M_α. Lemma 2 then tells us that q_1 and q_2 are equivalent in M.

We can therefore create a new DFA M' by removing q_2 (which is not equal to q_0 by Lemma 1) and redirecting any incoming transitions to q_2 so that they instead lead to q_1. As $|M'| < |M|$, this contradicts the minimality of M.

□

4 Upper and Lower Bounds

One natural question is whether we can bound the DFA-state complexity of $\bigsqcup_{\alpha\in A} L_\alpha$ by the DFA-state complexities of each $|L_\alpha|_{\mathsf{DFA}}$, and we indeed can. The upper bound is a direct application of the naive algorithm, and the lower bound is an application of Theorem 1 and Lemma 1.

Theorem 3. *If $L = \bigsqcup_{\alpha\in A} L_\alpha$ is nonempty, then*

$$1 + \max_{\alpha\in A} |L_\alpha|_{\mathsf{DFA}} \leq |L|_{\mathsf{DFA}} \leq 2 + \sum_{\alpha\in A} |L_\alpha|_{\mathsf{DFA}}\,. \tag{4}$$

The assumption of L being nonempty is only necessary for the lower bound. When $L = \emptyset$, we have $\max_{\alpha\in A} |L_\alpha|_{\mathsf{DFA}} = |L|_{\mathsf{DFA}} = 1$.

Proof. We first prove the upper bound. For each $\alpha \in A$, let $M_\alpha = (Q_\alpha, \Sigma, \delta_\alpha, q_\alpha, F_\alpha)$ be a minimal DFA recognizing L_α. Suppose without loss of generality that the sets Q_α are pairwise disjoint. Create new states q_0 and q_{sink}, and define the sets

$$Q = \{q_0, q_{\text{sink}}\} \cup \bigcup_{\alpha\in A} Q_\alpha \qquad \text{and} \qquad F = \bigcup_{\alpha\in A} F_\alpha.$$

We then create a new DFA $M = (Q, \Sigma \cup A, \delta, q_0, F)$ by setting

$$\delta(q,\sigma) = \begin{cases} q_\sigma & \text{if } q = q_0 \text{ and } \sigma \in A \\ \delta_\alpha(q,\sigma) & \text{if } q \in Q_\alpha \text{ and } \sigma \in \Sigma \\ q_{\text{sink}} & \text{otherwise.} \end{cases}$$

Since M recognizes L and $|Q| = 2 + \sum_{\alpha\in A} |Q_\alpha|$, this proves the upper bound.

To prove the lower bound, suppose that $M = (Q, \Sigma \cup A, \delta, q_0, F)$ is a minimal DFA recognizing L. We know from Theorem 1 that there exists a DFA for each L_α whose state set is Q_α. Since $q_0 \notin Q_\alpha$ by Lemma 1, this means that $|L_\alpha|_{\mathsf{DFA}} + 1 \leq |Q_\alpha| + 1 \leq |Q|$.

□

We now show that these upper and lower bounds are attained for tagged unions of arbitrarily many languages:

Theorem 4. *For any $n \in \mathbb{N}$, there exist languages $L_1, \ldots, L_n$ such that*

$$|L|_{\mathsf{DFA}} = 2 + \sum_{k=1}^{n} |L_k|_{\mathsf{DFA}},$$

where $L = \bigsqcup_{k=1}^{n} L_k$.

Proof. Take $\Sigma = \{\mathtt{a}\}$ and $L_k = \{\mathtt{a}^{jk} : j \in \mathbb{Z}_{\geq 0}\}$. Since $|L_k|_{\mathsf{DFA}} = k$ for each $k = 1, \ldots, n$, we need to show that

$$|L|_{\mathsf{DFA}} \geq 2 + \sum_{k=1}^{n} k.$$

By the Myhill-Nerode theorem, it suffices to find $2 + \sum_{k=1}^{n} k$ distinguishable strings in $(\Sigma \cup A)^*$, and we claim that the strings

$$\begin{aligned}
&1,\\
&2, 2\mathtt{a},\\
&3, 3\mathtt{a}, 3\mathtt{a}^2,\\
&\vdots\\
&n, n\mathtt{a}, \ldots, n\mathtt{a}^{n-1}\\
&\varepsilon, \mathtt{a},
\end{aligned}$$

are such a collection. Clearly, this collection contains $2 + \sum_{k=1}^{n} k$ strings. To prove any two strings w_1, w_2 in this collection are distinguishable, we consider a few cases:

- If $w_1 = \mathtt{a}$, then any y such that $w_2 y \in L$ is a distinguishing extension. An analogous argument holds if $w_2 = \mathtt{a}$.
- If $w_1 = \varepsilon$ or $w_2 = \varepsilon$, then 1 is a distinguishing extension.
- If w_1 and w_2 both start with the same number, then any $\mathtt{a}^j$ such that $w_1 \mathtt{a}^j \in L$ is a distinguishing extension.
- If none of the above three cases hold, we can write $w_1 = k_1 \mathtt{a}^{j_1}$ and $w_2 = k_2 \mathtt{a}^{j_2}$ with $k_1, k_2 \geq 1$ and $j_1, j_2 \geq 0$. Assume without loss of generality that $k_1 < k_2$, and choose $t \in \mathbb{Z}_{\geq 0}$ such that k_1 divides $j_1 + t$. Then $\mathtt{a}^{t+k_1}$ is a distinguishing extension in the case that k_2 divides $j_2 + t$, and $\mathtt{a}^t$ is otherwise.

□

This showcases a clear weakness of DFAs when compared with Turing machines. Because deciding this family of languages just boils down to checking divisibility, the descriptional complexity of a Turing machine deciding $\bigsqcup_{k=1}^{n} L_k$ is uniformly bounded in n. Theorem 4 tells us that for the analogous function

$$f(n) = \left| \bigsqcup_{k=1}^{n} L_k \right|_{\mathsf{DFA}},$$

we have $f(n) \in \Theta(n^2)$.

We now prove the sharpness of the lower bound:

Theorem 5. *For any $n \in \mathbb{N}$, there exist distinct languages $L_1, \ldots, L_n$ such that*

$$|L|_{\mathsf{DFA}} = 1 + \max_{1 \leq k \leq n} |L_k|_{\mathsf{DFA}},$$

where $L = \bigsqcup_{k=1}^{n} L_k$.

Proof. Take $\Sigma = \{\mathtt{a}\}$ and $L_k = \{\mathtt{a}^{k-1}\}$. Since each $|L_k|_{\mathsf{DFA}} = k + 1$ for each $k = 1, \ldots, n$, we need to find a DFA for L with $n + 2$ states.

To this end, we set $Q = \{q_{\text{start}}, q_0, q_1, \ldots, q_n\}$. The DFA

$$M = (Q, \Sigma \cup \{1, \ldots, n\}, \delta, q_{\text{start}}, \{q_{n-1}\}),$$

with

$$\delta(q, \sigma) = \begin{cases} q_{n-k} & \text{if } q = q_{\text{start}} \text{ and } \sigma = k \\ q_{\min(k+1,n)} & \text{if } q = q_k \text{ and } \sigma = \mathtt{a} \\ q_n & \text{otherwise,} \end{cases}$$

recognizes L. Since $|Q| = n + 2$, this proves the claim.

□

5 Conditions for Generalizability

We now have a rather definitive answer to the question of how state complexity relates to tagged unions for DFAs. However, this computational model is far too weak to tell us anything serious about generalizability of problems, since DFAs are so inefficient at this task. We illustrate this point with one last theorem, which gives relatively strict necessary and sufficient conditions for when a family of languages can be generalized (i.e. when the upper bound on $|L|_{\mathsf{DFA}}$ can be beaten):

Definition 5. We define the **Brzozowski derivative** of a language L over Σ with respect

to $v \in \Sigma^*$ by

$$v^{-1}L = \{w \in \Sigma^* : vw \in L\}.$$

Theorem 6. *If $L = \bigsqcup_{\alpha \in A} L_\alpha$ is nonempty, then we have*

$$|L|_{\mathsf{DFA}} < 2 + \sum_{\alpha \in A} |L_\alpha|_{\mathsf{DFA}}$$

if and only if one of the following conditions holds:

1. *There exist $\alpha \in A$ and $w \in \Sigma^*$ such that no string in L_α starts with w.*
2. *There exist distinct indices $\alpha_1, \alpha_2 \in A$ and strings $w_1, w_2 \in \Sigma^*$ such that*

$$w_1^{-1}L_{\alpha_1} = w_2^{-1}L_{\alpha_2}. \tag{5}$$

Proof. Let M denote the DFA for L defined in the upper bound construction in Theorem 3.

We first suppose that $\alpha \in A$ and $w \in \Sigma^*$ are such that no string in L_α starts with w. Then we have $\alpha w y \notin L$ for any $y \in \Sigma^*$. Since we also have for any $y \in (\Sigma \cup A)^*$ containing at least one symbol in A that $\alpha w y \notin L$, this means that no string in L starts with αw.

Let q denote the state reached by M after a computation on αw. Then no accept state in M is reachable from q, so q is equivalent to q_{sink}. By deleting q and redirecting all incoming transitions to q so that they instead lead to q_{sink}, we obtain a DFA for L with $1 + \sum_{\alpha \in A} |L_\alpha|_{\mathsf{DFA}}$ states.

Now suppose that the indices $\alpha_1, \alpha_2 \in A$ and strings $w_1, w_2 \in \Sigma^*$ are such that (5) holds. Let q_1 denote the state reached by M after a computation on $\alpha_1 w_1$, and q_2 denote the state M reaches after computing $\alpha_2 w_2$. We have for any string $y \in \Sigma^*$ that

$$\begin{aligned} \alpha_1 w_1 y \in L \iff w_1 y \in L_{\alpha_1} &\iff y \in w_1^{-1}L_{\alpha_1} \\ &\iff y \in w_2^{-1}L_{\alpha_2} \iff w_2 y \in L_{\alpha_2} \iff \alpha_2 w_2 y \in L, \end{aligned}$$

and for any string $y \in (\Sigma \cup A)^*$ containing at least one symbol in A, we have

$$\alpha_1 w_1 y, \alpha_2 w_2 y \notin L.$$

Thus the states q_1 and q_2 are equivalent in M, so we can obtain a DFA for L with $1 + \sum_{\alpha \in A} |L_\alpha|_{\mathsf{DFA}}$ states by deleting q_2 and redirecting all incoming transitions to q_2 so that they instead lead to q_1.

To prove the reverse direction, suppose that M is a minimal DFA for L with

$$|M| < 2 + \sum_{\alpha \in A} |L_\alpha|_{\mathsf{DFA}}.$$

Define the sets Q_α as in Theorem 1. By Theorem 2, we have $|Q_\alpha| = |L_\alpha|_{\mathsf{DFA}}$ for each $\alpha \in A$.

Since L is nonempty, we know that there exists some $\alpha \in A$. Let q_{sink} denote the state that M reaches after computing $\alpha\alpha$. As no string in L starts with $\alpha\alpha$, we know that no accept state in M is reachable from q_{sink}. Notice also that $q_0 \neq q_{\text{sink}}$ as L is nonempty.

Because

$$|\{q_0\}| + |\{q_{\text{sink}}\}| + \sum_{\alpha \in A} |Q_\alpha| = 2 + \sum_{\alpha \in A} |L_\alpha|_{\mathsf{DFA}} > |M|,$$

the pigeonhole principle tells us that the sets $\{q_0\}$, $\{q_{\text{sink}}\}$, and each Q_α are not all pairwise disjoint. The sets Q_α do not contain q_0 by Lemma 1, so we must either have that $q_{\text{sink}} \in Q_\alpha$ for some $\alpha \in A$ or that two distinct sets Q_{α_1} and Q_{α_2} have nonempty intersection.

If $q_{\text{sink}} \in Q_\alpha$ for some $\alpha \in A$, then there exists a word $w \in \Sigma^*$ such that the computation of M_α on w ends on q_{sink}. Thus no string in L_α starts with w, and the first condition holds.

If there exist distinct sets Q_{α_1} and Q_{α_2} with some $q \in Q_{\alpha_1} \cap Q_{\alpha_2}$, then there are strings $w_1, w_2 \in \Sigma^*$ such that the computation of M_{α_1} on w_1 and the computation of M_{α_2} on w_2 both end on q. Thus for any $y \in \Sigma^*$, the DFA M accepts $\alpha_1 w_1 y$ if and only if it accepts $\alpha_2 w_2 y$. Since

$$\begin{aligned} y \in w_1^{-1} L_{\alpha_1} \iff w_1 y \in L_{\alpha_1} &\iff \alpha_1 w_1 y \in L \\ &\iff \alpha_2 w_2 y \in L \iff w_2 y \in L_{\alpha_2} \iff y \in w_2^{-1} L_{\alpha_2}, \end{aligned}$$

the second condition is satisfied.

□

Note that if $|L|_{\mathsf{DFA}} \leq \sum_{\alpha \in A} |L_\alpha|_{\mathsf{DFA}}$ (i.e. the upper bound is beaten by more than 1 state), then some Q_{α_1} and Q_{α_2} must overlap. The above argument then tells us that the second condition must hold. In other words, the second condition is the *only* way to beat the upper bound by more than 1 state. The only way a DFA can substantially generalize a family of languages is if many of the languages are, in some sense, eventually the same.