Different "proofs" that the set of regular languages is closed under union

Josh Alman

CS Theory - January 2024

In class, we proved that the set of regular languages is closed under union. The idea behind the proof was that, given two DFAs D_1, D_2, we could make a new DFA D_3 which simultaneously keeps track of which state we're at in each DFA when processing a string. The formal theorem statement is:

Theorem 1. *If L_1, L_2 are regular languages, then $L_1 \cup L_2$ is a regular language.*

In this note, we give four or five different "proofs" of this using the strategy from class. They vary in how much intuition they give and how formal they are. The goal is to highlight some "good" proofs (2 and 4) and some "bad" proofs (1 and 3). When proving something like this on the homework, we expect you to write something between proof 2 and proof 4. Likely you want to choose for yourself how formal to be depending on how much formality helps you to convince yourself that your proof is correct. At a high level, a proof like proof 4 is longer and more tedious but less likely to have mistakes.

1 Proof 1: Too informal

This proof gives the high-level idea but is missing key nontrivial steps. In other words, it is not clear that all the statements here are correct as written, and a student who hasn't seen the proof already probably wouldn't be entirely convinced by this. Even though it's possible to fill in details here to make this a correct proof, it would not receive much credit since it's not convincing on its own. A correct proof needs to give all the nontrivial details.

Proof. We need to show that, given two DFAs D_1, D_2, we can make a DFA D_3 that accepts the strings accepted by either D_1 or D_2. D_3 will have a state for each pair of a state for D_1 and a state for D_2, and the transitions keep track of what's happening in both D_1 and D_2. We accept if either of D_1 or D_2 would accept. Thus, we accept the union of their languages. □

1.1 Proof 1b: Too informal and Wrong!

For comparison, here's another "proof" at the same level of formality as Proof 1 above, which is totally wrong! In other words, the level of formality here wasn't enough to catch a serious bug in the proof approach.

Proof. We need to show that, given two DFAs D_1, D_2, we can make a DFA D_3 that accepts the strings accepted by either D_1 or D_2. D_3 will have a copy of each state of D_1 and a copy of each state of D_2. The transitions are the same as in D_1 and D_2. We accept if either of D_1 or D_2 would accept. Thus, we accept the union of their languages. □

2 Proof 2: Proof with Key Ideas

This proof is more formal, but still not as formal as possible. The goal is to give all the ideas needed to prove the statement, and to explain why they suffice. Many details are alluded to but not fully spelled out, so the reader may need to pause for a moment on some sentences and think about why they're true. On the other hand, we are sure to be precise, and not leave ambiguity about what the DFA D_3 is or what sequences of states we're talking about to prove correctness, and we clearly explain everything needed to show our construction is correct. When writing a proof like this, we're implicitly saying that any omitted steps or formalities are straightforward, and that any student in the class would understand them without much trouble. A correct proof like this would be given full marks, and we would be able to give partial credit to a wrong proof like this since we'd understand the main ideas. That said, be careful about any steps that you're omitting. As an example, if we had made a typo in the definition of δ_3 below, the rest of the proof may not have caught this, since we don't write out details of how δ_3 works every time it is used.

Proof. Since L_1, L_2 are regular languages, we know they have DFAs that recognize them. Call those D_1, D_2, respectively. These can each be written as a 5-tuple:

$$D_1 = (Q_1, \Sigma, q_1, \delta_1, F_1),$$

$$D_2 = (Q_2, \Sigma, q_2, \delta_2, F_2).$$

Let us define a DFA D_3 that recognizes the language $L_1 \cup L_2$. The goal is that states of D_3 will correspond to pairs of a state from D_1 and a state from D_2, and will keep track of which state each of the two DFAs would be at if they were separately processing the input string. Specifically:

- States will be pairs $(q^1, q^2) \in Q_1 \times Q_2$,
- The start state is (q_1, q_2),
- (q^1, q^2) is an accept state if q^1 is an accept state of D_1 or q^2 is an accept state of D_2, and

- The transition at (q^1, q^2) will separately apply δ_1 to q^1 and δ_2 to q^2. More precisely, the transition function δ_3 is defined by

$$\delta_3((q^1, q^2), \sigma) = (\delta_1(q^1, \sigma), \delta_2(q^2, \sigma)).$$

We can see that when a string w is processed by D_3, it will visit states where the first part of the state says where it would be in D_1, and the second part says where it would be in D_2. More precisely,

- if $r_0, r_1, \ldots, r_n$ is the sequence of states (from Q_1) that w traverses in D_1, and
- if $r'_0, r'_1, \ldots, r'_n$ is the sequence of states (from Q_2) that w traverses in D_2,
- then $(r_0, r'_0), (r_1, r'_1), \ldots, (r_n, r'_n)$ is the sequence of states from Q_3 that w traverses in D_3 because of how we defined δ_3.

Because of this, we observe that D_3 accepts if and only if D_1 or D_2 does. This is because the final state (r_n, r'_n) is an accept state if and only if r_n is an accept state in D_1 (meaning $w \in L_1$) or r'_n is an accept state in D_2 (meaning $w \in L_2$). □

3 Proof 3: Formal Proof with No Intuition

This proof will carefully use the formal definitions of DFA to prove the theorem. This is a correct proof, although you may find it difficult to understand since many symbolic definitions and arguments are given without any intuition for where they're coming from. On the other hand, each step here follows from the previous steps and definitions, so it should be relatively easy to verify that each individual step of this proof is correct. We generally do not recommend writing proofs like this without giving some intuition (in English sentences) for what you're doing and why, and some 'scaffolding' where you explain the structure of your proof at the beginning. A correct proof like this would receive full credit, but an incorrect proof like this may receive little or no partial credit since we may not understand what you were aiming to do.

Proof. Let D_1, D_2 be the DFAs for L_1, L_2, respectively. These can each be written as a 5-tuple:

$$D_1 = (Q_1, \Sigma, q_1, \delta_1, F_1),$$
$$D_2 = (Q_2, \Sigma, q_2, \delta_2, F_2).$$

Define the DFA D_3 as

$$D_3 = (Q_3, \Sigma, q_3, \delta_3, F_3),$$

where $Q_3 = Q_1 \times Q_2$, $q_3 = (q_1, q_2)$, $F_3 = (F_1 \times Q_2) \cup (Q_1 \times F_2)$, and $\delta_3 : Q_3 \times \Sigma \to Q_3$ is defined by (for $q^1 \in Q_1$ and $q^2 \in Q_2$ and $\sigma \in \Sigma$):

$$\delta_3((q^1, q^2), \sigma) = (\delta_1(q^1, \sigma), \delta_2(q^2, \sigma)).$$

First, suppose w is accepted by D_1. Let n be the length of w, and write out $w = w_1 w_2 \cdots w_n$ where each $w_i \in \Sigma$. By definition of the DFA D_1, there is a sequence of states $r_0, r_1, \ldots, r_n$, where each $r_i \in Q_1$, such that:

- $r_0 = q_1$,
- $r_n \in F_1$, and
- for all $i \in \{0, 1, 2, \ldots, n-1\}$ we have $r_{i+1} = \delta_1(r_i, w_i)$.

Define the sequence of states $r'_0, \ldots, r'_n$ where each $r'_i \in Q_2$ recursively as follows:

- $r'_0 = q_2$, and
- for all $i \in \{0, 1, 2, \ldots, n-1\}$ we have $r'_{i+1} = \delta_2(r'_i, w_i)$.

Now, consider the sequence of states $s_0, s_1, \ldots, s_n$, where each $s_i \in Q_3$, defined by $s_i = (r_i, r'_i)$ for all i. We have $\delta_3(s_i, w_i) = (\delta_1(r_i, w_i), \delta_2(r'_i, w_i)) = (r_{i+1}, r'_{i+1}) = s_{i+1}$, and $s_0 = (r_0, r'_0) = (q_1, q_2) = q_3$, and $s_n = (r_n, r'_n) \in F_3$ since $r_n \in F_1$. These are the necessary conditions which show that D_3 accepts w.

Second, suppose w is accepted by D_2. Similar to above, we also have that D_3 accepts w.

Finally, suppose w is accepted by D_3 and again write $w = w_1 w_2 \cdots w_n$. This means there is a sequence of states $s_0, s_1, \ldots, s_n$, where each $s_i \in Q_3$, such that

- $s_0 = q_3$,
- $s_n \in F_3$, and
- for all $i \in \{0, 1, 2, \ldots, n-1\}$ we have $s_{i+1} = \delta_3(s_i, w_i)$.

By definition of Q_3, for each i, we can write $s_i = (r_i, r'_i)$ where $r_i \in Q_1$ and $r'_i \in Q_2$. Since $s_n \in F_3$, we know that $r_n \in F_1$ or $r'_n \in F_2$. Suppose $r_n \in F_1$ is true; the other case is nearly identical.

By definition of δ_3, we have $r_{i+1} = \delta_1(r_i, w_i)$ for all i. We know that $r_0 = q_1$ from definition of s_0 and that $r_n \in F_1$ by supposition. Thus, the sequence $r_0, \ldots, r_n$ shows D_1 accepts w, so $w \in L_1$.

All together, this means D_3 recognizes w if and only if $w \in L_1$ or $w \in L_2$. $\square$

4 Proof 4: Formal Proof

This proof will carefully use the formal definitions of DFA to prove the theorem. With this proof, we should be able to see that each step follows easily from the previous claims we've made, so there should be little confusion about whether it's correct. We also (aim to) do a better job of explaining why we're doing what we're doing, and what the overall structure of the proof is. With a proof like this, we have more confidence that we're not missing a critical detail like specifying the start or accept states appropriately. Furthermore, if you submit a proof like this, if there are small mistakes, we would be able to understand the idea of your argument and give partial credit. On the other hand, this proof is probably longer and more notation-heavy than is really needed to convince a student in the class that the statement is true.

Proof. Since L_1, L_2 are regular languages, we know they have DFAs that recognize them. Call those D_1, D_2, respectively. These can each be written as a 5-tuple:

$$D_1 = (Q_1, \Sigma, q_1, \delta_1, F_1),$$
$$D_2 = (Q_2, \Sigma, q_2, \delta_2, F_2).$$

Let us define a DFA D_3 that recognizes the language $L_1 \cup L_2$. The goal is that states of D_3 will correspond to pairs of a state from D_1 and a state from D_2, and will keep track of which state each of the two DFAs would be at if they were separately processing the input string.

We formally define D_3 as

$$D_3 = (Q_3, \Sigma, q_3, \delta_3, F_3),$$

where $Q_3 = Q_1 \times Q_2$, $q_3 = (q_1, q_2)$ is the pair of start states, $F_3 = (F_1 \times Q_2) \cup (Q_1 \times F_2)$ is the set of pairs where either the first or second part is an accept state, and $\delta_3 : Q_3 \times \Sigma \to Q_3$ is defined by separately applying the transition function for D_1 to the first part and D_2 for the second part, i.e., (for $q^1 \in Q_1$ and $q^2 \in Q_2$ and $\sigma \in \Sigma$):

$$\delta_3((q^1, q^2), \sigma) = (\delta_1(q^1, \sigma), \delta_2(q^2, \sigma)).$$

We will now prove that the language of D_3 is exactly $L_1 \cup L_2$. We will prove it in three steps:

1. If $w \in L_1$, then w is accepted by D_3,
2. If $w \in L_2$, then w is accepted by D_3, and
3. If w is accepted by D_3, then $w \in L_1$ or $w \in L_2$.

These three together imply that D_3 accepts all the strings in $L_1 \cup L_2$ and no other strings.

Step 1: Suppose $w \in L_1$, which means that w is accepted by D_1. Let n be the length of w, and write out $w = w_1 w_2 \cdots w_n$ where each $w_i \in \Sigma$. By definition of the DFA D_1, there is a sequence of states $r_0, r_1, \ldots, r_n$, where each $r_i \in Q_1$, such that:

- $r_0 = q_1$,
- $r_n \in F_1$, and
- for all $i \in \{0, 1, 2, \ldots, n-1\}$ we have $r_{i+1} = \delta_1(r_i, w_i)$.

(This is the sequence of states that w traverses in D_1, and it arrives at an accept state.)

Now, define the sequence of states $r'_0, \ldots, r'_n$ where each $r'_i \in Q_2$ recursively as follows:

- $r'_0 = q_2$, and
- for all $i \in \{0, 1, 2, \ldots, n-1\}$ we have $r'_{i+1} = \delta_2(r'_i, w_i)$.

(This is the sequence of states that w traverses in D_2.)

Now, consider the sequence of states $s_0, s_1, \ldots, s_n$, where each $s_i \in Q_3$, defined by $s_i = (r_i, r'_i)$ for all i. This is the sequence of states that w traverses in D_3 by our definition of δ_3 above. Moreover, $s_0 = (r_0, r'_0) = (q_1, q_2) = q_3$ is the start state of D_3, and $s_n = (r_n, r'_n) \in F_3$ is an accept state because $r_n \in F_1$. Therefore, D_3 accepts w. This concludes step 1.

Step 2: This is nearly identical, just switching the roles of D_1 and D_2 above.

Step 3: Suppose w is accepted by D_3 and again write $w = w_1 w_2 \cdots w_n$. This means there is a sequence of states $s_0, s_1, \ldots, s_n$, where each $s_i \in Q_3$, such that

- $s_0 = q_3$,
- $s_n \in F_3$, and
- for all $i \in \{0, 1, 2, \ldots, n-1\}$ we have $s_{i+1} = \delta_3(s_i, w_i)$.

Since $Q_3 = Q_1 \times Q_2$, for each i, we can write $s_i = (r_i, r'_i)$ where $r_i \in Q_1$ and $r'_i \in Q_2$. Since we have $s_n \in F_3$, we know by definition of F_3 that $r_n \in F_1$ or $r'_n \in F_2$. Suppose $r_n \in F_1$ is true; the other case is nearly identical. Let us show D_1 accepts w.

By definition of δ_3, we know that we have $r_{i+1} = \delta_1(r_i, w_i)$ for all i, i.e., the sequence of states that w traverses in D_1 is $r_0, r_1, \ldots, r_n$. We know that $r_0 = q_1$ is the start state since $s_0 = (q_1, q_2)$ is the start state of D_3. Furthermore, we just assumed $r_n \in F_1$. Thus, D_1 accepts w, so $w \in L_1$ as desired.

We have proved all three parts, and thus concluded the proof. □