CSE491/596 Monday Sept. 20, 2021: Regular and Non-Regular Languages

Theorem 1: The complement of a regular language is always regular. ☒

I will write the complement of a regular language A as $\widetilde{A}$ or as $\sim A$. The idea is that given a DFA $M = (Q, \Sigma, \delta, s, F)$ such that $L(M) = A$, we can get $M' = (Q', \Sigma, \delta', s', F')$ such that $L(M') = \widetilde{A}$ by taking $Q' = Q$, $s' = s$, $\delta' = \delta$, but $F' = Q \setminus F$. Then for all $x \in \Sigma^*$,

$$x \in \widetilde{A} \iff x \notin A \iff x \notin L(M) \iff \delta^*(s,x) \notin F \iff \delta^*(s,x) \in F' \iff x \in L(M').$$

Thus $L(M') = \widetilde{A}$. Here δ^* is the **extended transition function** from $Q \times \Sigma^*$ to Q such that $\delta^*(q,y) =$ the unique state r such that M can process y from q to r. Note that this is only valid in a DFA. The whole idea of switching accepting and rejecting states does not generally work to complement an NFA (nor a GNFA).

Now suppose we have two DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ (note that Σ is the same). Let $L_1 = L(M_1)$ and $L_2 = L(M_2)$. Then let ***op*** be any binary operation on sets, such as $\cup$ or $\cap$ but note also difference $L_1 \setminus L_2$ and *symmetric difference*

$$L_1 \triangle L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1) = (L_1 \cup L_2) \setminus (L_1 \cap L_2),$$

whose corresponding Boolean operation ***op'*** is XOR, which is sometimes written $\oplus$. Then we have:

$$x \in L_1 \, \boldsymbol{op} \, L_2 \iff (x \in L_1 \, \boldsymbol{op'} \, x \in L_2) \iff \left(\delta_1^*(s_1,x) \in F_1\right) \boldsymbol{op'} \left(\delta_2^*(s_2,x) \in F_2\right)$$

When ***op'*** = AND, this is $\iff (\delta_1^*(s_1,x), \delta_2^*(s_2,x)) \in F_1 \times F_2$.
This means that **if** we define

$M_3 = (Q_3, \Sigma, \delta_3, s_3, F_3)$ with $Q_3 = Q_1 \times Q_2$ and $s_3 = (s_1, s_2)$,

and define $\delta_3((q_1,q_2), c) = (\delta_1(q,c), \delta_2(q,c))$,

and use $F_3 = F_1 \times F_2$,

then $L(M_3) = L(M_1) \cap L(M_2)$.

We can use this **Cartesian product construction** for the other Boolean operations ***op'***. We just have to be more careful about how we define the final states. The general definition is

$$F_3 = \{(r_1, r_2) : r_1 \in F_1 \, \boldsymbol{op'} \, r_2 \in F_2\}.$$

Then $L(M_3) = L(M_1)\ op\ L(M_2)$. Thus we have shown the following theorem.

Theorem 2: The class of regular languages is closed under all Boolean operations.

Actually, we already could have said this right after Theorem 1 about complements. This is because OR is a native regular expression operation. OR and negation ($\neg$) form a complete set of logic operations. For instance, a AND $b \equiv \neg((\neg a)$ OR $(\neg b))$ by DeMorgan's laws.

The Myhill-Nerode Relation

Given a DFA $M = (Q, \Sigma, \delta, s, F)$ and two strings $x, y \in \Sigma^*$, suppose $\delta^*(s, x)$ and $\delta^*(s, y)$ both give the same state q. Then for any further string $z \in \Sigma^*$, the computations on the strings xz and yz go through the same states after q. In particular, they end at the same state r.

- If $r \in F$, then $xz \in L$ and $yz \in L$, where $L = L(M)$.
- If $r \notin F$, then $xz \notin L$ and $yz \notin L$.
- Either way, $L(xz) = L(yz)$, for all z.

Suppose, on the other hand, we have strings x, y for which there exists a string z such that

$$L(xz) \neq L(yz).$$

Then M cannot process x and y to the same state. Moreover, this goes for *any* DFA M such that $L(M) = L$. In particular, every such DFA must at least *have* two states.

Now let us build some definitions around these ideas. Given any language L (not necessarily regular) and strings x, y "over" the alphabet Σ that L is "over", define:

- x and y are *L-equivalent*, written $x \sim_L y$, if for all $z \in \Sigma^*$, $L(xz) = L(yz)$.
- x and y are *distinctive for* L, written $x \nsim_L y$, if there exists $z \in \Sigma^*$ s.t. $L(xz) \neq L(yz)$.

Lemma 1. The relation $\sim_L$ is an equivalence relation.

Proof: We need to show that it is

- Reflexive: $x \sim_L x$ is obvious.
- Symmetric: indeed, $y \sim_L x$ immediately means the same as $x \sim_L y$.
- Transitive: Suppose $w \sim_L x$ and $x \sim_L y$. This means:
  - for all $v \in \Sigma^*$, $L(wv) = L(xv)$ and
  - for all $z \in \Sigma^*$, $L(xz) = L(yz)$.

Because v and z range over the same span of strings, it *follows* that

– for all $z \in \Sigma^*$, $L(wz) = L(xz)$ and $L(xz) = L(yz)$.

Hence we get:

– for all $z \in \Sigma^*$, $L(wz) = L(yz)$.

So $w \sim_L y$.

This ends the proof. ☒

Any equivalence relation on a set such as Σ^* partitions that set into disjoint *equivalence classes*. So $x \nsim_L y$ is the same as saying x and y belong to different equivalence classes. [I intended to give an example but skipped it after the initial loss of time: Start with the language E of strings having an even number of 1s. Then the relation $\sim_E$ has exactly two equivalence classes: one for an even number of 1s, the other for odd. Now if you make E_3 be the language where the number of 1s is a multiple of 3, you get 3 equivalence classes. And so on...]

Logic of the Myhill-Nerode Theorem

Now say that a set S of strings is ***Pairwise Distinctive for L*** if all of its strings belong to separate equivalence classes under the relation $\sim_L$. Other names we will use are "distinctive set" and "**PD set**" for L. This is the same as saying:

- for all $x, y \in S, x \neq y$, there exists $z \in \Sigma^*$ such that $L(xz) \neq L(yz)$.

Thus we can re-state something we said above as:

Lemma 2. If L has a PD set S of size 2, then any DFA M such that $L(M) = L$ must process the two strings in S to different states, so M must have at least 2 states.

Note: "L has" does not mean S must be a subset of L, it just means "has by association." Now we can take this logic further:

Lemma k. If L has a PD set S of size k, then any DFA M such that $L(M) = L$ must process the k strings in S to different states, so M must have at least k states.

I've worded this to try to make it as "obvious" as possible, but actually it needs proof: Suppose we have a DFA M with $k-1$ or fewer states such that $L(M) = L$. Then there must be (at least) two strings in S that M processes to the same state. This follows by the **Pigeonhole Principle**. [In this lecture I skipped over the story, but see this recent GLL blog post.]

Then explain why we get the infinite case:

Lemma ∞. If L has a PD set S of size ∞, then any DFA M such that $L(M) = L$ must process the strings in S to different states, so M must have at least ∞ states...but then M is not a *finite* automaton. So L is not accepted by any finite automaton...which means **L is not a regular language**. ☒

Myhill-Nerode Theorem, first half: If L has an infinite PD set, then L is not regular.

Example: $L = \left\{a^n b^n : n \geq 0\right\}$. $\Sigma = \{a, b\}$. $S = \left\{a^n : n \geq 0\right\} = a^*$. Let any $x, y \in S$, $x \neq y$, be given. Then there are different numbers i and j such that $x = a^i$ and $y = a^j$. Take $z = b^i$. Then $xz = a^i b^i \in L$, but $yz = a^j b^i \notin L$, because $i \neq j$. Thus $L(xz) \neq L(yz)$. Thus for all $x, y \in S$ with $x \neq y$, there exists z such that $L(xz) \neq L(yz)$. Thus S is PD for L. Since S is infinite, L is not regular, by MNT. ☒

[I finished by drawing a connection from this to the idea of playing the spears-and-dragons game when you can save any number of spears. In the basic case where you can save at most 1 spear the DFA has 3 states, and these are mandated because $S = \{\epsilon, \$, D\}$ is a PD set of size 3. In particular, even though both $x = \epsilon$ and $y = \$$ are strings **in** the language L_1 of the 1-spear game, they are distinctive **for** L_1 because $z = D$ kills you in the former case (i.e., $xz = \epsilon D = D \notin L_1$) but you stay alive in the latter case (i.e., $yz = \$D \in L_1$). If you can save up to 2 spears, then $\epsilon, \$, \$$ are three distinctive strings (plus D to make a fourth). Well, if you can save unlimited spears, then $S_\infty = \{\epsilon, \$, \$, \$\$, \ldots\}$ becomes an infinite PD set by similar logic to the $\left\{a^n b^n\right\}$ example. So the most liberal form of the game gives no longer a regular language. The next lecture will pick up from here (minus the note at top).]