#1 Irregularity

Not all languages are regular. In particular, regular languages are limited by the requirement that finite automata necessarily have a finite quantity states. This prevents many languages, in practice, from being recognized by finite automata. Some examples:

1. chemical equations;
2. (most) programming languages (hence you cannot use regular expressions in your compilers course); and, for example,
3. The language of regular expressions (that is, $R ::= a \mid \epsilon \mid \emptyset \mid R \cup R \mid R \circ R \mid R^*$ for $a \in \Sigma$). *I mean specifically the set L_R of strings that are regular expressions, e.g., if $\Sigma = \{0, 1\}$ then the set of regular expressions over Σ is $\{\epsilon, 1, 0, \emptyset, 1 \cup \epsilon, 0 \cup \epsilon, \epsilon \cup \epsilon, 00, 01, 10, 11, ...\}$.*

Intuitively, languages that permit arbitrary nesting, or that have to "keep track" of depth, cannot be regular.

#1.1 The Pumping Lemma

We have two chiefl tools to determine irregularity: the *Pumping Lemma* and the *Myhill-Nerode Theorem*. We will mostly be using the Pumping Lemma in this class. The *Myhill-Nerode Theorem* is given as an exercise in the text; you do not need anything beyond what you know now to prove it. There is also an additional reading for it linked on the course schedule.

Definition 1 (The Pumping Lemma). *If A is a regular language, then there is a number p (the* pumping length*) where if s is any string in A of length at least p, then s may be divided into three pieces $s = xyz$, satisfying the following conditions:*

1. *For each $i \geq 0$, $xy^iz \in A$,*
2. *$|y| > 0$, and*
3. *$|xy| \leq p$.*

Proof. See Sipser Theorem 1.70. □

Example 1 (Visualizing the Pumping Lemma). Suppose A is regular, with DFA M, and consider $s = xyz$ accepted by M. What does s look like? See Figure 1.72 of Sipser. I will draw this picture in class.

The idea here is that the we have a repetition at q_9, since y loops you back around. It's called the "pumping" lemma because, by condition (1) you can "pump" y however many times you like and remain a valid string in A.

Let's see what each condition of the pumping lemma entails.

Condition 1. When $i = 0$, then xz just "passes through" q_9. Otherwise, we looped around i times.

Condition 2. Even though xz must be in A, The Pumping lemma states $|y| > 0$. So we have to exhibit this pathway y that can repeatedly loop.

Condition 3. We have $|xy| \leq p$, with p the number of states in M. This implies, by the *pidgeon hole principle*, that there is a repetition on this path.

Although the *Pumping Lemma* describes when a *language is regular*, we largely use it instead to show that languages are *irregular*. In particular,

Example 2 (Proving Irregularity with the Pumping Lemma). This is Example 1.73 from Sipser, which I will ask you about on your lab. I give a shorter proof.

Claim 1. *Let* $B = \{0^n 1^n \mid n \geq 0\}$. *Then* B *is not regular.*

Proof. Towards a contradiction, assume that B is regular and p is the pumping length implied by the Pumping Lemma. Choose s to be the string $0^p 1^p$. Because s clearly has length greater than p, the pumping lemma applies, and we may write $s = xyz$ for some x, y, and z. What does y look like? Consider each case.

1. y cannot be empty because condition (2) states $|y| > 0$.
2. y consists only of 0s. But then consider $xyyz$—by condition (1), this too is in B, yet it has more 0s than 1s, kicking it out of B (a contradiction).
3. y consists only of 1s, which is ruled out by the same argument.
4. y is a mix of 0s and 1s. But then $xyyz$ will not be of the form $0^n 1^n$, as we will have some 1s before 0s.

Thus a contradiction is unavoidable if B is regular. ∎