CSC236 Week 09: Languages: The Last Words

Hisbaan Noorani

November 4 – November 17, 2021

Contents

1 Regular languages closure

Regular languages are those that can be denoted by a regular expression or accept by an FSA. In addition:

- L regular $\implies \overline{L}$ regular.
- L regular $\implies \text{Rev}(L)$ regular.

If L has a finite number of strings, then L is regular.

2 Pumping Lemma

If $L \subseteq \Sigma^*$ is a regular language, then there is some $n_L \in \mathbb{N}$ (n_L depends on L such that n_L is the number of states in some FSA that accepts L) such that if $x \in L$ and $|x| \geq n_L$ then:

- $\exists u, v, w \in \Sigma^*, x = uvw$ — x is a sandwich
- $|v| > 0$ — middle of sandwich is non-empty
- $|uv| \leq n_L$ — first two slices no longer than n_L
- $\forall k \in \mathbb{N}, uv^k w \in L$

Idea: if machine $M(L)$ has $|Q| = n_L, x \in L \wedge |x| \geq n_L$, denote $q_i = \delta^*(q_0, x[: i])$, so x "visits" $q_0, q_1, \ldots, q_{n_L - 1}$ with the first n_L prefixes of x (including ε)... so there is at least one state that x "visits" twice (pigeonhole principle, and x has $n_L + 1$ prefixes).

3 Consequences of regularity

How about $L = \{1^n 0^n : n \in \mathbb{N}\}$?

Proof: Assume, for the sake of contradiction, that L is regular. Then, there must be a machine M_L that accepts L. So M_L has $|Q| = m > 0$ states. Consider the string $1^m 0^m$. By the pumping lemma, $x = uvw$, where $|uv| \leq m$ and $|v| > 0$, and $\forall k \in \mathbb{N}, uv^k w \in L$. But, then $uvvw \in L$, so $m + |v|$ 1s followed by just m 0s. *This is a contradiction.* Elements of L must have the same number of 1s as zeros, but $m + |v| > m$. ■

4 Another approach... Myhill-Nerode

Consider how many different states $1^k \in \text{Prefix}(L)$ and end up in... for various k

Scratch work: Could 1, 11, 111 each take the machine to the same state?

Proof: Assume, for the sake of contradiction, that L (previous section) is regular. Then some machine M that accepts L has some number of states $|Q| = m$. Consider the prefixes $1^0, 1^1, \ldots, 1^m$. Since there are $m+1$ such prefixes, at least two drive M to the same state, so there are $0 \leq h < i \leq m$ such that 1^h and 1^i drive M to the same state but then $1^h 0^h$ drive the machine to an accepting state. But so does $1^i 0^h$! But $1^i 0^h$ (since $i \neq h$) sohuld not be accepted. *This is a contradiction.* By assuming that L was regular, we had to conclude there was a mchine that accepted L, which lead to a contradiction. So that assumption is false and L is not regular ■

5 "Real life" consequences

- The proof of irregularity of $L = \{1^n 0^n : n \in \mathbb{N}\}$ suggest a proof of irregularity of $L' = \{x \in \{0,1\}^* : x \text{ has an equal number of 1s and 0s}\}$ (explain... consider $L' \cap L(1^* 0^*)$)
- A similar argument implies irregularity of L'' $\{x \in \Sigma^* : x \text{ has an equal number of } \}$

6 How about $L = \{w \in \Sigma^* : |w| = p \wedge p$ is prime$\}$

Non regular. We can always find a prime that is at least as big as a given machine since there are an infinite amount of primes. There is always some bigger prime. We will prove this by the pumping lemma. /Proof:/ Assume for the sake of contradiction, that L is regular. So there is some machine M with $|Q| = m$ states, that accepts L. Let p be a prime number that is no smaller then m. Such a p exists since there are an infinite number of primes. Then the regular expression 1^p has length $\geq m$. This regular expression $1^p = uvw$ where $|v| > 0$, $|uv| < m$, and $uv^{kw} \in L$ for all natural numbers k. We know that $|uvw| = p$, but $\left|uv^{1+p}w\right| = p + p \cdot |v| = p \cdot (1 + |v|)$, a composite number. *This is a contradiction* since L consists of only strings of prime length. ■

7 A humble admission… (from Prof. Heap)

- At any point in time, my computer, and yours, are DFSAs

  You machine has a finite number of states, and is driven to new states by strings processed by its instruction set…

- Do the arithmetic…

  Prof. Heap's laptop has 66108489728 bits of RAM and 843585945600 bits of disk secondary storage, leading to $2^{66108489728+843585945600}$ possible different states. This is and always will be a finite number.

- However, we could dynamically add/access increasing stores of memory

  He could run down to Spadina/College to get more RAM (or storage) for bigger jobs.

- If we try and calculate this, we will find that it is impossible. We cannot calculate how many states there are in a given machine using that same machine — we will run out of ram.

8 Push Down Automata (PDA)

- DFSA plus an infinite stack with finite set of stack symbols. Each transition depends on the state, (optionally) the input symbol, (optionally) a pop from stack.
- Each transition results in a state, (optional) push onto stack.

Design a PDA that accepts $L = \{1^n 0^n : n \in \mathbb{N}\}$.

Corresponds to a context-free grammar (production rules) that denotes this language

- $S \to 1S0$
- $S \to \varepsilon$

9 Yet more power

- (informally) linear bounded automata: finite states, read/write a tape of memory proportional to input size, tape move are one position L-to-R.

  Many computer scientists think this is an accurate model of contemporary computers.

- (informally) Turing machine: finite states, read/write an infinite tape of memory, tape moves are one position L-to-R.

  This is the “ultimate” model of what computers can do.

Each machine has a corresponding **grammar** (e.g. FSAs $\leftrightarrow$ regexes (right-linear grammar))