Week 10: Finish Equivalence Proof and Limitations

CSC 236:Introduction to the Theory of Computation

Summer 2024

Instructor: Lily

Announcement

- Final Exam Time is out! July 19^{th} in EX 200 (7:00~10:00pm)
  - **BRING YOUR TCARD!** (or other valid forms of ID as listed in the Exam Toolkit). No ID = no entry
  - Students who arrive more than 30 minutes *after* the start of the exam ***will not* be able to take the exam**
  - You **must score** (40 + 10*(# missed midterms))% to pass
  - True or False question marking update: +1 correct answer, -0.5 incorrect answer (used to be -1 incorrect answer)
- Tutorials this week: proving a language is not regular, Turing Machines, and decidability

Consider the following languages

For each of the following languages determine if it regular or not and try to explain why or why not

1. $L_1 = \{0^m 1^n : m, n \in \mathbb{N}\}$.
2. For some $k \in \mathbb{N}$, $L_2 = \{0^n 1^n : n \leq k\}$.
3. $L_3 = \{0^n 1^n : n \in \mathbb{N}\}$.

1. $L_1 = \{0^m 1^n : m, n \in \mathbb{N}\}$.

YES, is regular. To show

1. construct finite automaton or
2. regex $L_1 = \mathcal{L}(0^* 1^*)$

2. For some $k \in \mathbb{N}$, $L_2 = \{0^n 1^n : n \leq k\}$.

YES, is regular. To show

$L_2 = \mathcal{L}(R_2)$ $\quad R_2 = \varepsilon + 01 + 0011 + \cdots + \underbrace{0 \cdots 0}_{k}\underbrace{1 \cdots 1}_{k}$

3. $L_3 = \{0^n 1^n : n \in \mathbb{N}\}$.

No, Not regular.

How do we show this?

Proving Non-Regularity

Pumping Lemma

If $L \subset \Sigma^*$ is regular, then there exists a pumping length $p \in \mathbb{N}$ such that for every $w \in L$ of length *greater than or equal to* p, $\underbrace{w = xyz}$ for $x, y, z \in \Sigma^*$ such that:

READ: string w can be split into 3 pieces.

1. $|xy| \leq p$,
2. $|y| \geq 1$, and
3. $xy^k z \in L$ for every $k \in \mathbb{N}$.

If these are true, then...

can construct infinite set of other strings such that these strings are $\in L$

Note: the Pumping Lemma is only useful for infinite languages (infinite sets of strings). What about finite languages?

If $L \subset \Sigma^*$ is regular, then there exists a pumping length $p \in \mathbb{N}$ such that for every $w \in L$ of length *greater than or equal to* p, $w = xyz$ for $x, y, z \in \Sigma^*$ such that:

1. $|xy| \leq p$,
2. $|y| \geq 1$, and
3. $xy^k z \in L$ for every $k \in \mathbb{N}$.

PHP: if n holes and >n pigeons, then at least one hole has ≥2 pigeons.

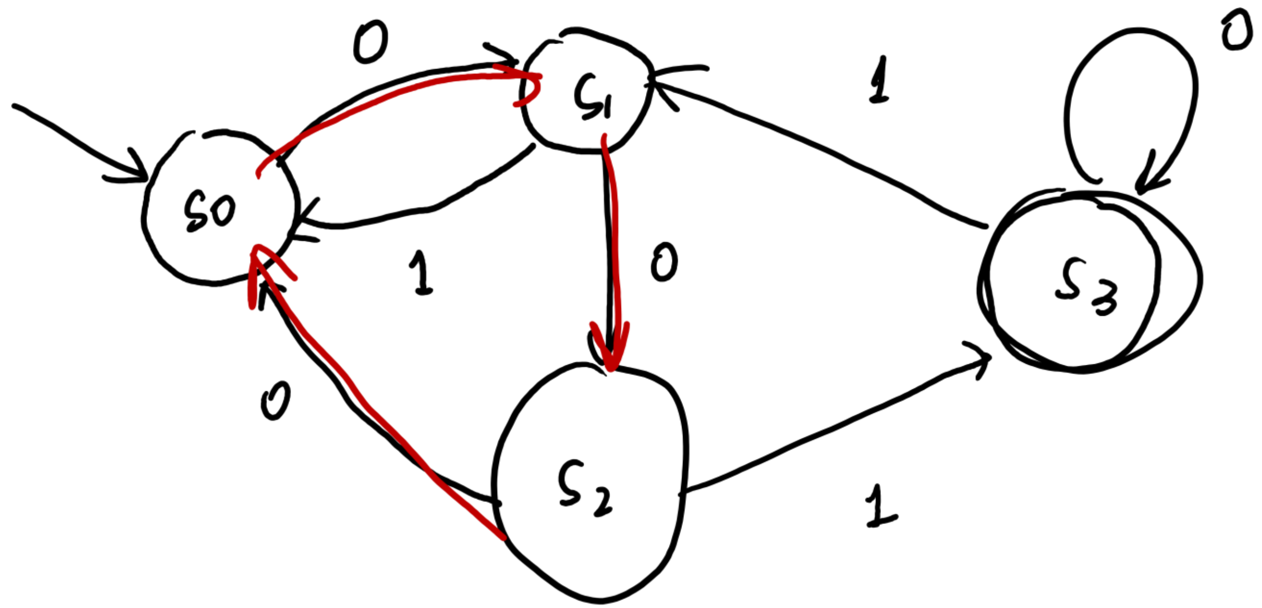
Proof.

$(000)^2$

$w_1\ w_2\ w_3\ w_4\ w_5\ w_6$

$W = \underbrace{\varepsilon}_{x}\ 0\ 0\ 0\ \underbrace{001}_{z}$

$(000)^k \quad k \in \mathbb{N}$



step n	w_i	state
0	–	S_0
1	$w_1 = 0$	S_1
2	$w_2 = 0$	S_2
3	$w_3 = 0$	S_0
4	$w_4 = 0$	S_1
5	$w_5 = 0$	S_2
6	$w_6 = 1$	S_3

some state repeats

If $L \subset \Sigma^*$ is regular, then there exists a pumping length $p \in \mathbb{N}$ such that for every $w \in L$ of length *greater than or equal to* p, $w = xyz$ for $x, y, z \in \Sigma^*$ such that:

1. $|xy| \leq p$,

2. $|y| \geq 1$, and

3. $xy^k z \in L$ for every $k \in \mathbb{N}$.

Proof.

1. Since L is regular, let M be the DFA which accepts L

$M = (Q, \Sigma, \delta, s, F)$

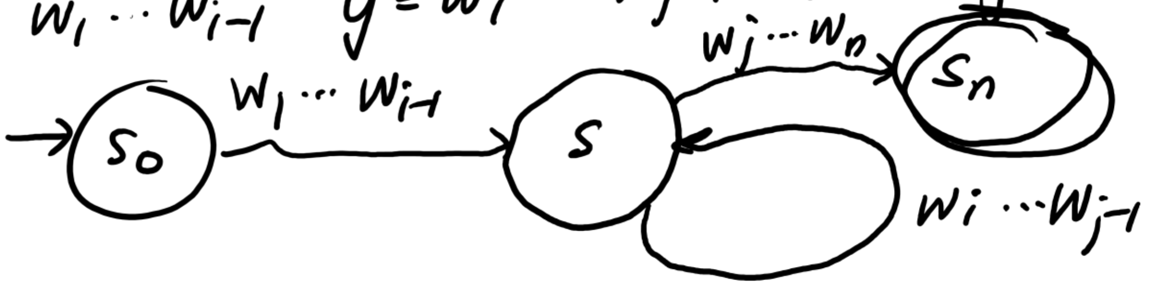
2. let pumping length be $|Q|$, consider $w \in L$, $|w| > p$

↳ $w_1 \cdots w_n$ $n > p$

3. let $s_0, \ldots, s_n$ be the sequence of states in M that we visit when processing w.

4. By PHP, s_i and s_j are the same state with $i < j$

5. $w = xyz$ $x = w_1 \cdots w_{i-1}$ $y = w_i \cdots w_{j-1}$ $z = w_j \cdots w_n$



Example.

Show that $L = \{vv : v \in \{0,1\}^*\}$ is not regular.

1. suppose L is regular. By pumping lemma let p be its pumping length.
2. pick string $w = 0^p 1 0^p 1$. NOTE: $w \in L$
3. WRITE $w = xyz$ $\quad x = 0^i \quad y = 0^j$ $\leftarrow j \geq 1$ $\quad z = 0^{p-i-j} 1 0^p 1$
4. Then $xy^0z \in L$ $\quad xy^0z = 0^i \, \underbrace{\cancel{0^j}}_{\varepsilon} \, 0^{p-i-j} 1 0^p 1$ $= 0^{p-j} 1 0^p 1 \notin L$
5. contradiction $\quad$ L is NOT regular.

Now you try!

Show that the following languages are *not* regular.

1. $L_1 = \{0^n 1^n : n \in \mathbb{N}\}$.
2. $L_2 = \{0^n : n \text{ is a prime}\}$.

Prove that $L_1 = \{0^n 1^n : n \in \mathbb{N}\}$ is not regular.

1. Suppose L_1 is regular. By the pumping lemma, L_1 has some pumping length p

2. pick $w \in L_1$: $w = 0^p 1^p$

3. Determine x, y, z where $w = xyz$ satisfying $|xy| \le p$, $|y| \ge 1$

$x = 0^i$, $y = 0^j$, $z = 0^{p-i-j} 1^p$

4. pumping lemma says $xy^2z \in L_1$, but $xy^2z = 0^i 0^{2j} 0^{p-i-j} 1^p = 0^{p+j} 1^p$

Since xy^2z is NOT of the form $0^n 1^n$ $(n \in \mathbb{N})$, $j \ge 1$ this is a contradiction, L_1 is not regular.

$\Sigma = \{0\}$

Prove that $L_2 = \{0^n : n \text{ is a prime}\}$ is not regular.

1. Assume L_2 is regular and p is its pumping length

2. Pick $w \in L_2$: $w = 0^q$: q smallest prime number larger than p.

3. Determine x, y, z : $x = 0^i$ $y = 0^j$ $(j \geq 1)$ $z = 0^{q-i-j}$

0^0 ← cannot do this.

4. Pump : consider $w' = x y^{(q+1)} z$. (i) By pumping lemma, $w' \in L_2$

(ii) $w' = 0^{i} 0^{j(q+1)} 0^{q-i-j} = 0^{q(j+1)}$ → not prime if $q > 1$.

$jq + \cancel{j} + q - \cancel{j} = q(j+1)$

cancel

Consider $L = \{0^n : n \text{ is even}\}$.

1. assume L is regular, let p be pumping length.

2. pick $w \in L$: $w = 0^{2p}$

3. Determine x, y, z : $x = 0^i$, $y = 0^j$ $(j \geq 1)$, $z = 0^{2p-i-j}$

4. pump : Consider $w' = xy^kz$ for $k \in \mathbb{N}$.
   (i) pumping lemma $w' \in L$
   (ii) $w' = 0^i 0^{jk} 0^{2p-i-j} = 0^{2p+j(k-1)}$

since we cannot pick j, if $j=2$ then the string will always be in L
NOT a contradiction.

General problem format

Given a language L, you will be asked: is L regular?

1. Make a decision: regular or not regular.
2. If regular: proof of regularity.
   - Produce a finite automaton or regular expression which accept the given language
   - Prove correctness of finite automaton (if asked)
3. If not regular: proof of non-regularity.
   - Assume L is regular and apply pumping lemma

$0^n 1^n : \forall n \in \mathbb{N}$

Regular Languages can't count.

NFA to DFA

NOT TRUE : every NFA is a DFA ✗

TRUE : every NFA N has a DFA M such that $\mathcal{L}(N) = \mathcal{L}(M)$ ✓

Given a NFA $M_N = (Q_N, \Sigma, \delta_N, s_N, F_N)$, construct a DFA $M_D = (Q_D, \Sigma, \delta_D, s_D, F_D)$ such that $\mathcal{L}(M_N) = \mathcal{L}(M_D)$.

$\delta_D : Q \times \Sigma \to Q$

$\delta_N : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$

power set of states for S in $\mathcal{P}(Q)$, S is a set of states.

Given M_N, construct M_D: (by defining different terms)

Σ is going to be the same

NFA to DFA

$q_1, q_2, \ldots, q_n$ $I \subseteq [n]$

Given a NFA $M_N = (Q_N, \Sigma, \delta_N, s_N, F_N)$, construct a DFA $M_D = (Q_D, \Sigma, \delta_D, s_D, F_D)$ such that $\mathcal{L}(M_N) = \mathcal{L}(M_D)$.

$Q_D :=$ all subsets of Q_N, i.e. $\mathcal{P}(Q_N)$

$s_D := \mathcal{E}(s_N)$ Read: "epsilon-closure" of s_N
means: start at s_N and repeatedly take ε-transitions

$F_D :=$ all $q_I \in \mathcal{P}(Q_N)$ such that $F_N \cap I \neq \emptyset$
i.e. q_I represents some state which was an accepting state in the NFA.

NFA to DFA

$$J = \varepsilon\left(\bigcup_{i \in I} \delta_N(q_i, a)\right)$$

first transition on a then take any ε-transitions avail.

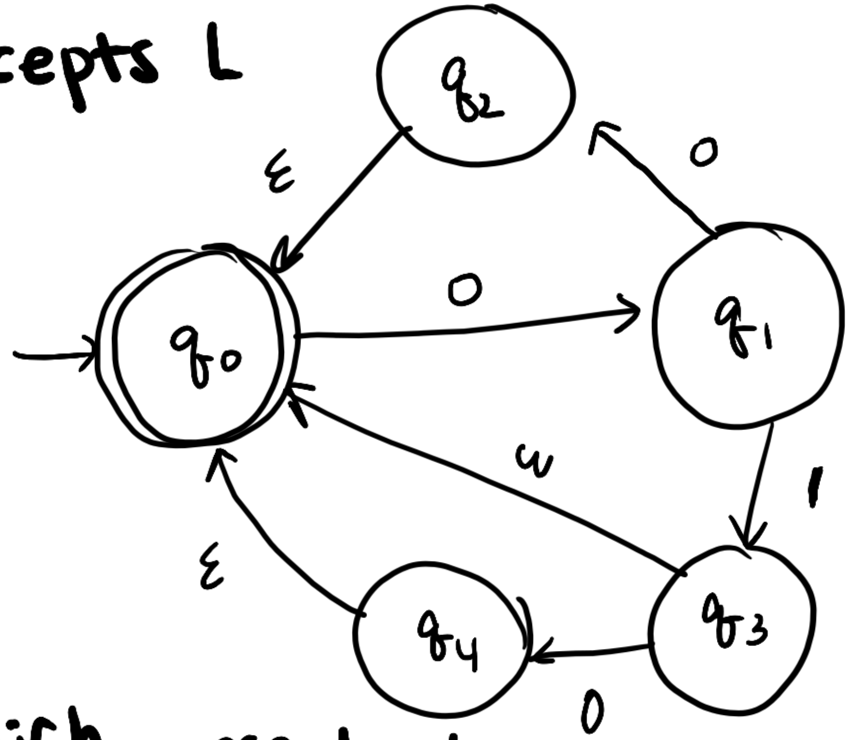
$$\delta_D : \delta_D(q_I, a) = q_J$$

represents the ε-closure of all states in I transitioning on a (i.e, start at any such state, take any ε-transition, a, then any ε-transition

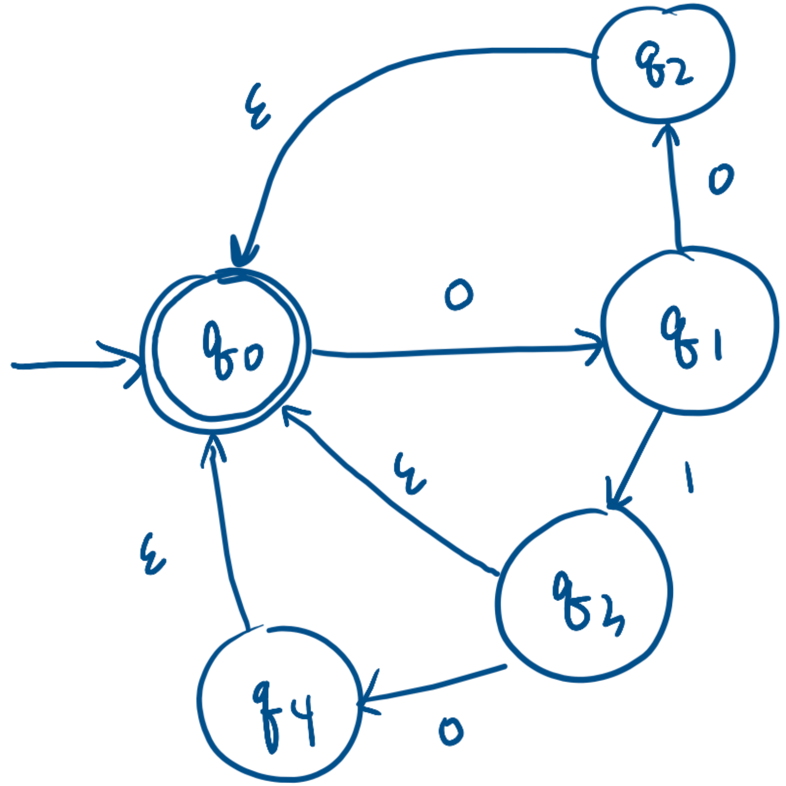
NFA to DFA Example

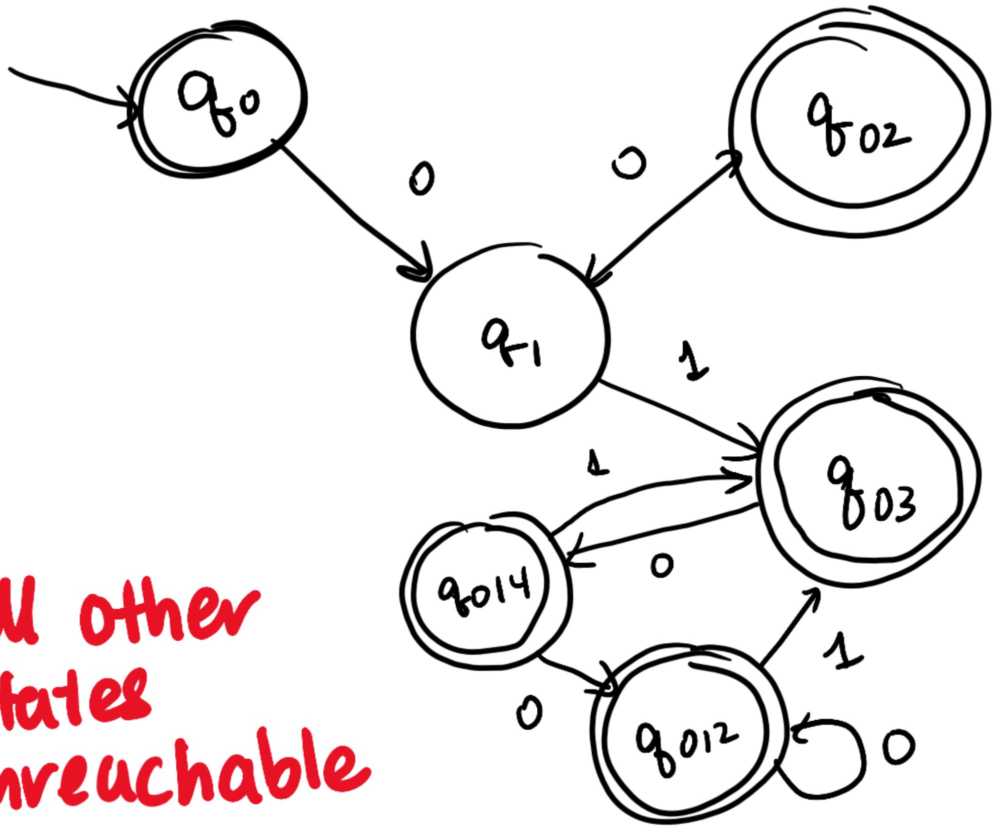
Consider $L = \mathcal{L}\left(\left(0(0+10+1)\right)^*\right)$ and its corresponding NFA.

NFA which accepts L



make DFA which accepts L

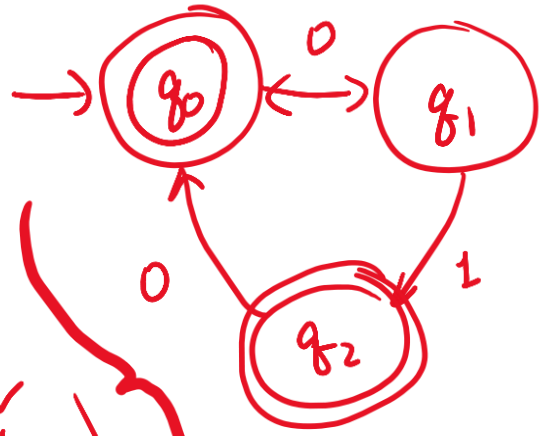




all other states unreachable

1. one state for each subset of {0,1,2,3,4}
2. accepting state if contains state representing 0
3. start state is any state reachable using ε-transitions.

DFA output might not be minimal



another DFA which accepts L (has fewer states)

Recap

- DFAs, NFAs, and regular expressions have the same expressive power (we showed the first two are equivalent, you will show the last is equivalent to the first two)
- Pumping lemma: used to show that a language is not regular
- Regular languages cannot count

Next time… review!