Lecture 25: Pumping lemma

- Using the pumping lemma (example)
- (time permitting) introduce non-determinism

Applications:

- P.L. has important implications for design of programming language syntax
- Nondeterminism is important for theoretical bounds on computation and for modeling uncertainty (randomness or interaction with peer...
Let $E = \{ L, \emptyset \}$

Let $L$ be a set of strings of balanced parentheses:

- $() \in L$
- $((())) \in L$
- $(X) \in L$
- $(X\in L$

Claim: $L$ is not regular.

**Proof:** By contradiction. Assume $L$ is regular.

Then there exists $A$ on the PL.

Let $x = \sigma \in L, \text{ need } \text{len}(x) \geq n$

Clearly $x \in L$, and $\text{len}(x) = 2n \geq n$.

Now, $A, u, v, w$ as in PL.

$x = \underbrace{((())(())(())(())(())(())( ... (})\in L$

$\underbrace{u}_n \underbrace{v}_n \underbrace{w}_n$

$w^2w = \underbrace{((())(())(())(())(())(())(())(})\in L$

$\underbrace{u}_n \underbrace{v}_n \underbrace{w}_n \underbrace{u}_n \underbrace{v}_n \underbrace{w}_n \in L$.

No contradiction.

Let $x = \underbrace{((())( ... (})) \in L$

Clearly $x \in L$, and $\text{len}(x) = 2n \geq n$.

So $A, u, v, w$ as in PL.

$x = \underbrace{((())( ... (})) \in L$

$\underbrace{u}_n \underbrace{v}_n \underbrace{w}_n$

Since $\text{len}(vw) \leq n$, I know $v$ has only left parentheses.

So $wwv = w^2w$ has more left parentheses than right, so $wwv \notin L$.

But by PL, $wwv \in L$, so we have a contradiction.
Deterministic finite automata

\[ \text{on every input, from any state, there is exactly one transition.} \]

\[ \delta : Q \times \Sigma \to Q \text{ is a function.} \]

Non-deterministic automata

\[ \Rightarrow \text{can have any number of transitions} \]

\[ M = (Q, \Sigma, \delta, q_0, F) \]

\[ \delta(q_0, 0) = \{ q_1, q_2 \} \]

\[ \delta(q_1, 1) = \emptyset \]

\[ 01 \in L(M) \]

\[ 00 \in L(M) \]

\[ 010 \in L(M) \]

\[ 001 \notin L(M) \]

\[ 101 \notin L(M) \]

\[ \text{there is no transition out of start state.} \]

An NFA \( N = (Q, \Sigma, \delta, q_0, A) \) where

- \( Q \) is a finite set of states
- \( \Sigma \) is an alphabet
- \( \delta \) is the transition function giving the set of arrows from \( q \) on input \( a \).

\[ \delta : Q \times \Sigma \to 2^Q \]

- \( q_0 \in Q \) is start state
- \( A \subseteq Q \) is set of accept states
\[ \hat{\delta} \text{ is the extended trans. fn for an NFA } \]
\[ \hat{\delta}(q, x) \text{ gives the set of states reachable from state } q \text{ on input string } x. \]
\[ \hat{\delta} : Q \times \Sigma^* \rightarrow 2^Q \]
\[ \text{is given inductively by:} \]
\[ \hat{\delta}(q, \varepsilon) := \{ q \} \]
\[ \hat{\delta}(q, xa) := \bigcup_{q' \in \hat{\delta}(q, x)} \hat{\delta}(q', a) \]

**DFA**

**NFA**

\[ \hat{\delta}(q_1, xa) := \bigcup_{q' \in \hat{\delta}(q_1, x)} \hat{\delta}(q', a) = \hat{\delta}(q_1, a) \cup \hat{\delta}(q_2, a) \cup \ldots \]

\[ q_1, q_2, q_3 \ldots \]