Lecture 8
Elementary Questions about Regular Languages

COT 4420

Theory of Computation

Section 4.2

Representation of Languages

- Informal: a logical or prose statement about its strings:
  - $\{0^n 1^n \mid n$ is a nonnegative integer$\}$
  - "The set of strings consisting of some number of 0's followed by the same number of 1's."

- Formal: represent a language by a **Regular Expression**, **NFA**, **DFA** or **Regular Grammars**.

Representation of Languages

When we say: We are given a Regular Language L

We mean: Language L is in a standard representation

Decision Properties

- A *decision property* for a class of languages is an algorithm that takes a formal description of a language and tells whether or not some property holds.
- Example: Is language L empty?

Why Decision Properties?

- Suppose we have a DFA representing a protocol.
- Example: "Is there an upper bound (in terms of transitions) for when the protocol terminates?" = "Is the language finite?"

- Example: "Can the protocol fail?" = "Is the language nonempty?" (when the final state is the "error" state)

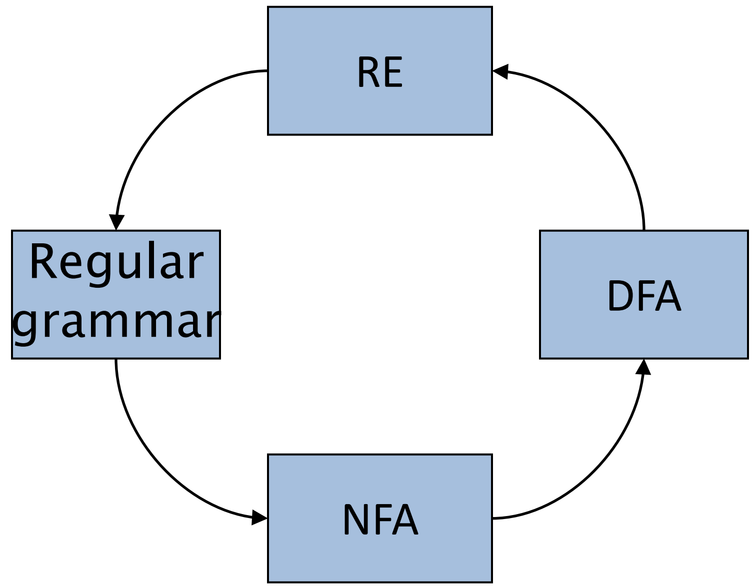
Membership Problem

Question: Is string w in regular language L?

Answer: Take the DFA representing L and check if w is accepted by this DFA.

Membership Problem

- If the regular language is in some other representation, we have an algorithm to convert it to a DFA as we have:



The Emptiness Problem

Question: Given regular language L, does it contain any string at all? Is L empty? (L = ø).

Answer: Take the DFA that accepts L, check if there is any path from the initial state to a final state. Note: need only check strings up to length *n*, where *n* is the number of states of the DFA.

The Infiniteness Problem

Question: Given regular language L, Is it finite?

Answer: Take the DFA that accepts L, check if there is a walk with cycle from the initial state to a final state. If there is a cycle, the language of DFA is infinite, otherwise the language is finite.

The Infiniteness Problem

1. Eliminate states not reachable from the start state.
2. Eliminate states that do not reach a final state.
3. Test if the remaining transition graph has any cycles.
   - Starting at each node N, search forward until you either can reach no more nodes, or you discover you can reach N.
   - If you can reach N, you have a cycle
   - If you exhaust all the nodes as starting points, and you still haven't found a cycle, then there are none.

Equivalence Problem

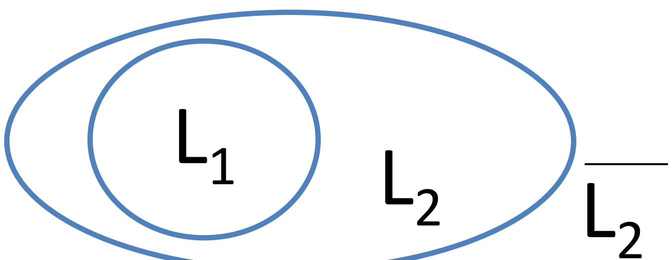
Question: Given regular languages L_1 and L_2, Is $L_1 = L_2$?

Answer:

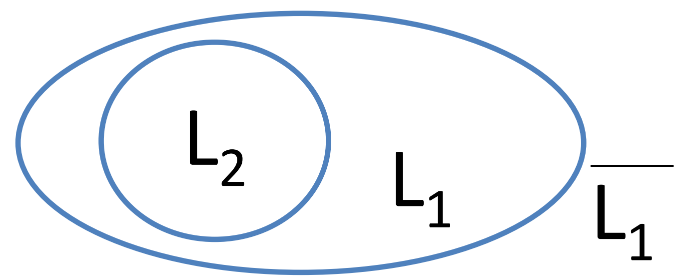
1. Create $L_3 = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$
2. Check if $L_3 = \emptyset$? If L_3 is empty, $L_1 = L_2$

$$L_3 = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$$

$$(L_1 \cap \overline{L_2}) = \varnothing \quad \textbf{and} \quad (\overline{L_1} \cap L_2) = \varnothing$$



$L_1 \subseteq L_2$



$L_2 \subseteq L_1$

$$L_1 = L_2$$