Regular Languages

Summary

January 14, 2025

1 Quick Summary

1.1 DFA

Overview

What it contains

1. Set of states (Q)
2. An alphabet (Σ)
3. Transition function(δ)
4. Starting state (q_0)
5. Set of accepting state(f)

The DFA starts at the initial state and follows the transition function consuming one letter at a time and accepts the string if and only if one of the accepting states is reached.

1.2 NFA

Overview

What it contains

1. Set of states (Q)
2. **An alphabet along with 'ϵ'** ($\Sigma \cup \epsilon$)
3. **Transition Relation** ($\delta \subset Q \times \Sigma \times Q$)
4. Starting state (q_0)
5. Set of accepting state(f)

The NFA starts at the initial state and follows the transition relation consuming at most one letter at a time and accepts the string if and only if at least one of the runs end in an accepting state.
Acceptance: If at least one run terminates on an accepting state.

Languages recognized by NFA

The set of languages recognized by NFA is in fact, the set of regular languages.

1.3 Regular Languages

Closed under Complementation

How

Set all the non-accepting state as accepting states and accepting states as non-accepting states.

Why

Every string that terminated in a previously non-accepting state is accepted and vice versa.

Examples

1. $L = \{a^n b^m \mid n, m \geq 0\}$ over $\{a, b\}$
2. $L = \{a^n b^n \mid n \geq 0\}$ over $\{a, b\}$

Closed under Intersection and Union

Intersection

Do a "cross product" of the two machines (i.e Create a new machine that runs both the machines simultaneously) and accept iff both the machines accept.

Union

Do a cross product of the two machines and accept iff either of the two machines accept.

Closed under Concatenation and Star

Concatenation

Construct a new NFA where the accepting states of the first NFA have an epsilon transition to the start state of the second NFA.

Star

Construct a new NFA where the accepting states of the initial NFA have an epsilon transition to the start state of itself.

2 Tips and Tricks to solve problems

Quick tips and tricks

1. Any computation (decision) requiring finite memory can be performed in a DFA.
2. To show that a language is regular, prefer constructing an NFA over a DFA.
3. Instead, you can also use closure properties on known Regular language examples.
4. The idea of simultaneously running two Automatas can be simulated by taking the cross-product of states. This can be useful in several scenarios.