Equivalence of DFAs and NFAs

Two accepters M_1 and M_2 are *equivalent* if:

$$\mathcal{L}(M_1) = \mathcal{L}(M_2)$$

Trivial: Given a DFA M_1, construct an equivalent NFA M_2.

Harder But Solvable: Given a NFA M_1, construct an equivalent DFA M_2.

The idea is to map every subset of states in the NFA M_1 to a single state in the DFA M_2.

Note: The number of DFA states can be exponential in the number of NFA states.

With some computational overhead, we can obtain the benefits of NFA notation and the efficiency of DFA simulation.

Constructing Equivalant DFAs

1. Create a graph G with vertex $\Lambda^*(\{q_0\})$. This will be the DFA initial state.
2. While there are missing edges.
   (a) Pick a missing edge, i.e., vertex u and symbol $a \in \Sigma$, but no a edge from u.
   (b) Compute $v = \Lambda^*(\Delta(u, a))$.
   (c) If v is not in G, put v in G.
   (d) Put an a edge from u to v in G.

---

3. Every vertex in G that contains an NFA final state is a DFA final state.

Why does this algorithm eventually halt?

Proof of Equivalence

Proposition: If R is the set of possible states after the NFA reads w, then the DFA is in state $v = R$ after reading w.

Basis: This is true for $w = \lambda$. Before reading any symbols, the NFA could be in any state in $\Lambda^*(\{q_0\})$. This is the initial state of the DFA.

Induction: Prove that if the proposition is true for w, then it is true for wa, for any $a \in \Sigma$.

Assume: After reading w, R is the set of possible NFA states, and R is the state of the DFA.

Show: After reading wa, if R' is the set of possible NFA states, then R' is the state of the DFA.

Proof: $R' = \Lambda^*(\Delta(R, a))$ is the next DFA state. R' is also the set of possible NFA states because Δ accounts for a transitions, and Λ^* accounts for λ transitions after reading a.