CS304 - Formal Languages and Automata Theory 4^{th} **Semester**

Lecture 4 : Equivalence of DFA and NFA

Instructor: Prof. Prateek Vishnoi *Indian Institute of Technology, Mandi*

In the last lecture, we defined the NFA, now we try to find out the answer of the question that "How much powerful NFA is as compared to DFAs. More precisely,

Is the class of languages that is accepted by NFA is greater than DFA or not??

Little surprise! NFA's and DFA's, both have same computational power. In other words, if there is a language $\mathcal{L}$ that is accepted by NFA then there exist a DFA that accepts the language $\mathcal{L}$.

How to prove it?

Proof is via construction of DFA for a given NFA. The idea is simple, we can keep track of the multiple transitions on a (state, input) pair by creating a new state. Recall, we maintain the track of two DFAs by creating a state for a state pair of two DFAs(product automaton). Similar idea works here because for every multiple transitions on state, input pair we create a new state. Since number of states are finite in NFA say k thus DFA can have maximum of 2^k states.

Subset Construction (Powerset) Method: NFA to DFA

The subset construction method converts any NFA into an equivalent DFA. The key idea is that a single DFA state represents a **set of NFA states**.

Assume an NFA:

$$N = (Q, \Sigma, \delta, q_0, F)$$

We construct an equivalent DFA:

$$D = (Q', \Sigma, \delta', q_0', F')$$

Step 1: Start State of DFA

The start state of the DFA is the set containing the NFA start state.

$$q_0' = \{q_0\}$$

Step 2: DFA States

Each DFA state is a subset of Q. Hence,

$$Q' \subseteq 2^Q$$

We begin with the start subset and generate new subsets until no new subsets appear.

Step 3: Transition Function

For a DFA state $S \subseteq Q$ and input symbol $a \in \Sigma$,

$$\delta'(S, a) = \bigcup_{q \in S} \delta(q, a)$$

Step 4: Iteration

Whenever a new subset of states is obtained, treat it as a new DFA state and compute its transitions. Repeat until no new subsets are generated.

Step 5: Accepting States

A DFA state is accepting if it contains at least one accepting NFA state.

$$F' = \{\, S \subseteq Q \mid S \cap F \neq \varnothing \,\}$$

Summary

1. Start with $\{q_0\}$.
2. For each subset and input symbol, compute the union of transitions.
3. Add new subsets as DFA states.
4. Repeat until no new subsets appear.
5. Any subset containing an NFA final state becomes a DFA final state.

Intuition

An NFA can be in multiple states simultaneously, whereas a DFA must be in exactly one state. Thus, each DFA state represents all possible NFA states at a given step.

Example: NFA to DFA using Subset Construction

Consider the NFA:

$$Q = \{q_0, q_1\}, \quad \Sigma = \{0, 1\}, \quad q_0 \text{ is start state}, \quad F = \{q_1\}$$

Transition function:

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_1\}$

Step 1: Start State of DFA

$$\{q_0\}$$

Step 2: Compute Transitions

State $\{q_0\}$

$$\delta'(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0, q_1\}$$
$$\delta'(\{q_0\}, 1) = \delta(q_0, 1) = \{q_0\}$$

New state obtained: $\{q_0, q_1\}$

State $\{q_0, q_1\}$

On input 0:

$$\delta'(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$$

On input 1:

$$\delta'(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_1\} = \{q_0, q_1\}$$

No new states appear, so construction stops.

Step 3: DFA States

$$Q' = \{\{q_0\}, \{q_0, q_1\}\}$$

Step 4: Final States

Any subset containing q_1 is final:

$$F' = \{\{q_0, q_1\}\}$$

Final DFA Transition Table

	0	1
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$

Thus, the DFA obtained using the subset construction method is equivalent to the given NFA.

One observation

Number of states in the DFA for any given NFA having $k-$ states is upperbounded by 2^k states. Think about it, why it is true.

But why NFA?

Since we now know that for every NFA accepting a language L there exists an equivalent DFA that accepts the same language L, we can focus on constructing an NFA for a language whenever it is easier to do so. The equivalent DFA can always be obtained later using the subset construction method.

Regular Language

Definition 1. A language is regular if and only if there exist a DFA M such that $L(M) = L$ where $L(M)$ denotes the language accepted by M.

Question

Check whether the language over {0,1} that accepts the language of all binary strings that end with 01 is regular or not.

Idea

A string is accepted if its last two symbols are 01. The NFA can nondeterministically guess the position where the substring 01 begins.

Construction of NFA

Let the NFA be:

$$M = (Q, \Sigma, \delta, q_0, F)$$

where

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{0, 1\}, \quad F = \{q_2\}$$

Transition Function

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
q_2	$\varnothing$	$\varnothing$

Explanation

- State q_0 loops on both inputs to read any prefix.
- On reading 0, the NFA may guess that this 0 is the start of the final substring and move to q_1.
- From q_1, reading 1 moves to accepting state q_2.
- Since q_2 has no outgoing transitions, the string must end after reading 01.

Hence, this NFA accepts exactly the language of strings ending with 01. Since there exist a DFA for every NFA, thus we can imply above language is regular.

Exercises

Solve the exercises given at the end of chapter in [1].

References

[1] Michael Sipser, *Introduction to the Theory of Computation*, 3rd Edition, Cengage Learning, 2012.