CS 4510 Automata and Complexity January 10th 2024

Lecture 2: Nondeterminism

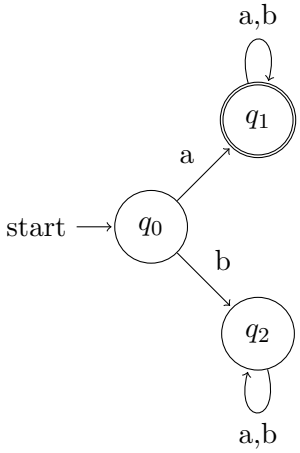
Lecturer: Abrahim Ladha *Scribe(s): Samina Shiraj Mulani*

1 The Generalization of Nondeterminism

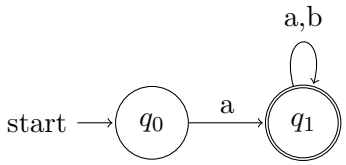
We noted that DFAs are weak. Let's try to extend or generalize them. Recall that a DFA can be represented as $(Q, \Sigma, \delta, q_0, F)$. Given this definition, we wish to extend its power. The only useful thing we can extend is the way in which states interact with each other, i.e., δ. The rest of the device is static. We extend δ in the following three ways:

1.1 Implicit Rejection

We allow transitions to be undefined, and it is understood that undefined transitions implicitly reject. As an example, recall the following DFA which decides the language $\{w \in \Sigma^* \mid w \text{ begins with } a\}$



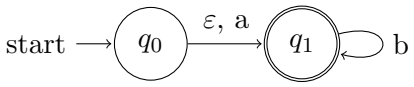
With implicit rejection, we could represent equivalently as



If we are at q_0 and we see b we would reject. This can be helpful for programming. Consider how well-defined a DFA is. Its like it has every edge case covered with all that try-catch nonsense. Implicit rejection allows us to lazily construct only the parts that we care about. Then "undefined behavior" results in immediate rejection. Note that when we perform a complement of the accept states in a DFA that decides a language L, we get the complement of the language. The same does not hold here due to implicit rejection.

1.2 Epsilon Transitions

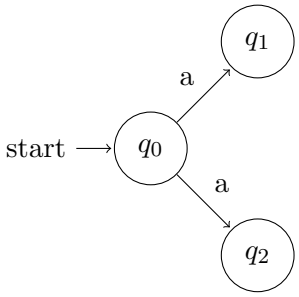
We define "ε-transitions", which can be taken for free. For example



a, ab, abb are some strings which are accepted. But now that we allow ε-transitions, b, bb, ε are also accepted. While normally, each transition "costs" the next letter of the input, an ε-transition costs nothing. You may take it for free. It is important to know that the choice to take it is not forced. A nondeterministic computation may choose not to take it.

1.3 Nondeterministic Transitions

We allow transitions of more than one of the same type. This means that you can have multiple outgoing transitions with the same input. For example



Consider the computation on a word beginning with an a. Which state are you in? q_1? q_2? You are in both!

2 Coping with Nondeterminism

With these three new relaxations, we have defined a new kind of automata, the nondeterministic finite automata (NFA). On input a word, there may be multiple different possible computations, and we say an NFA accepts some string if there exists atleast one computation to an accepting state. It does not matter how many more rejecting computations there are.

Its important to understand nondeterminism and not just have deterministic coping strategies. Nondeterminism isn't real. You could not build a nondeterministic computer, but it doesn't matter. We may still study this unrealizable machine as a purely theoretical device. The following analogies may help in visualizing this power.

1. **Graph Search** BFS or DFS on the NFA until you reach an accept state.
2. **Lucky Coin** During your computation you come to a nondeterministic transition. Imagine you flip a lucky coin that tells you exactly which path to take. Through a purely imaginary way, you have divine information on which path will correctly lead you to an accept state.

3. **Alternate Timelines**[1] For each nondeterministic action, create multiple timelines. Each timeline consists of the what-if for each possible choice. As long as in one timeline you reach an accept state, then the computation is accepting.

3 Formal Definition

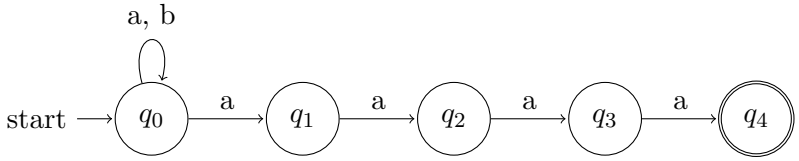
A Nondeterminisitic Finte Automata (NFA) can be represented by a 5-tuple $(\Sigma, Q, q_0, \delta, F)$ where:

1. Q - finite set of states
2. Σ - finite alphabet
3. q_0 - denoted start state
4. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$
   $\mathcal{P}(Q)$ represents the power set of Q, which is the set of all subsets of Q. The power set of a set Q has 2^n elements where n is the number of states in Q. While in a DFA, you may go to exactly one state for every state symbol pair. In an NFA, you may go from one state to a set of states.
5. $F \subseteq Q$ The selection of final or accepting states.

4 Examples

Lets show a few examples

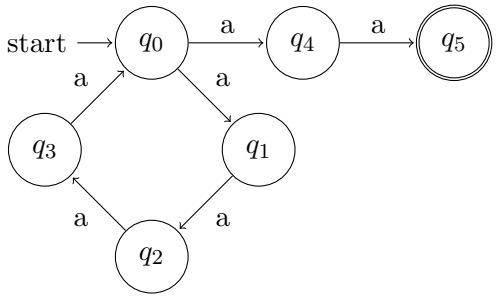
1. $L_1 = \{w \in \Sigma^* \mid w \text{ ends with } aaaa\}$



Consider the computation of this machine on input $aaaaaaaaa$. If you are at q_0 and you read an a, you may choose to either stay at q_0 or move on to q_1. Note that this word is accepted by the NFA because it may correctly guess exactly when it is four a's from the end and then choose to leave q_0. Another way is to consider all possible guesses of when to go to q_1 on seeing an a. If we guess too late, we will terminate on one of q_0, q_1, q_2, q_3 and not accept. If we guess too early, we will reach q_4, but then have more input to read, and must implicitly reject since q_4 has no outgoing transitions. Most of the computations will be rejecting but it doesn't matter, as there is atleast one accepting computation, one correct guess.

[1] Different science fictions have different rules for how time travel works. I am going off of the episode Remedial Chaos Theory from Community.

2. $L_{x,y} = \{a^{xn+y} \mid n \in \mathbb{N}\}$
   Lengths of the strings in this language form an arithmetic progression. We can show that there exists an NFA for every x, y. Note that the loop is of length x while the tail (q_4 to q_5 in the representation) is of length y.



If we ever see a b at any state, we implicitly reject. We nondeterministically choose how many times to go around the loop of length x before we hop off and go to the accept state.

5 Comparison with DFAs

Let $\mathscr{L}(NFA)$ represent the class of languages which are decidable by an NFA. What is its power, relative to a DFA?

Theorem 1. $\mathscr{L}(DFA) \subseteq \mathscr{L}(NFA)$

Proof. Every DFA is an NFA. An NFA has all these super powers, but there is no requirement to use them. Though it may be obvious just from the generalization that is nondeterminism, for exercise, we prove $\mathscr{L}(DFA) \subseteq \mathscr{L}(NFA)$. Let $L \in \mathscr{L}(DFA)$. Then there exists a DFA to decide L. Note that this DFA is also an NFA, so there exists an NFA to decide L. Then $L \in \mathscr{L}(NFA)$. Since this is true for all $L \in \mathscr{L}(DFA)$, we see that $\mathscr{L}(DFA) \subseteq \mathscr{L}(NFA)$. □

Theorem 2. $\mathscr{L}(NFA) \subseteq \mathscr{L}(DFA)$

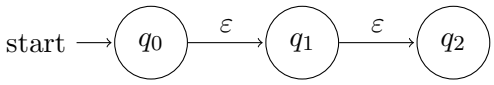
This should surprise you! We gave a normal computation device all this unrealistic unrealizable power. Yet, this power can be simulated using realizable methods. For any NFA, we will show how to simulate it on a DFA. This means that $\mathscr{L}(NFA) \subseteq \mathscr{L}(DFA)$. Combining the aforementioned point, we get $\mathscr{L}(DFA) = \mathscr{L}(NFA)$ We simulate an NFA on a DFA. Although an NFA may be in many states at once, it can only be in finitely many. This is the key idea behind the simulation. To each possible set of state the NFA could be in, we assign one state of our DFA to represent each subset of the NFA. Then the NFA going between subsets of states can be simulated by our DFA going from just one state to another. This is called the powerset construction. Proven by Michael O. Rabin and Dana Scott in 1959, this work earned them the Turing award in 1976. There is also a small comment on economy. NFAs can be smaller. There are languages which have NFAs of n states, but require DFAs of 2^n states. We do not care about the efficienty, rather if these structures exist at all to decide. The simulation of an NFA by a DFA works since 2

to the power of a finite number is still a finite number. There is exponentially more to keep track of, but that is still only a finite amount.

There is also the issue of these epsilon transitions. We define the concept of reach.

$$reach(q_i) = \{q_i \text{ and any state reachable from } q_i \text{ by } \varepsilon\text{-transitions}\}$$

For example



Then $reach(q_0) = \{q_0, q_1, q_2\}$.

Proof. Let N be any NFA with $N = (\Sigma, Q, q_0, \delta, F)$. We construct an equivalent DFA $D = (\Sigma^{'}, Q^{'}, {q_0}', \delta', F')$ so that $L(N) = L(D)$.

- $Q' = \mathcal{P}(Q)$ For each possible subset of the states of the NFA, we creat one state of our DFA.
- $\Sigma' = \Sigma$
- ${q_0}' = reach(q_0)$ If there is an ε-transition from the start state of the NFA, then the computation need not necessarily begin at q_0 if this ε-transition is taken first. Then the start state of our DFA corresponds to the set of possible states in which the computation could begin in the NFA, which is those states reachable from q_0 in the NFA.
- For $S \subseteq Q$ any subset of states of the NFA and $a \in \Sigma$, we define

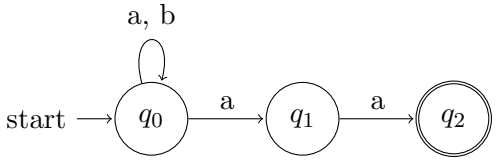
  $$\delta^{'}(S, a) = \bigcup_{q \in S} reach(\delta(q, a))$$

  For S a state of the DFA, its outgoing transitions are defined to be the state corresponding exactly and only to the set of states of the NFA which you can go to on viewing the same symbol.
- $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$ Recall than an NFA accepts if there exists a computation which reaches an accept state. After computation on a word, you may be in several states at once, but if atleast one is accepting, the machine accepts. We set the accepting states of the DFA to be those which contain any accept state of the NFA.

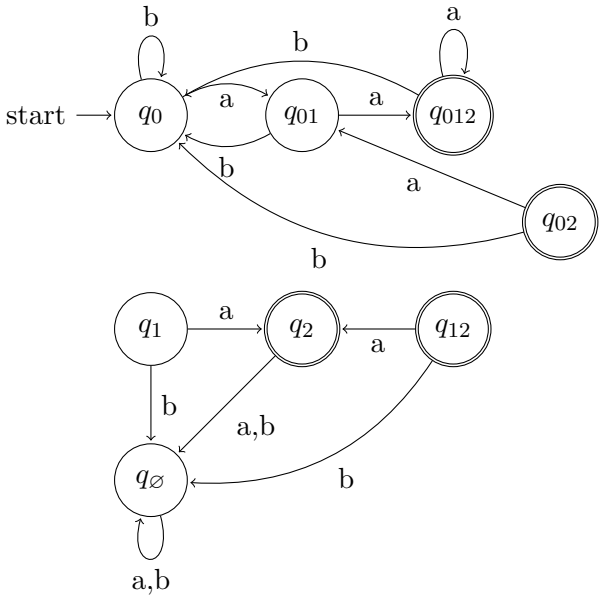
□

5.1 Example

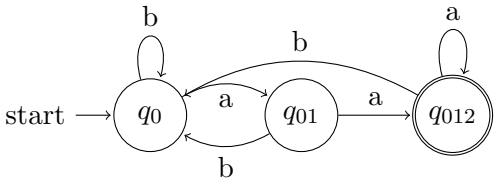
$L_2 = \{w \in \Sigma^* \mid w \text{ ends with } aa\}$



By following the above process, we get the corresponding DFA Not guaranteed minimal DFA but still it is a DFA.

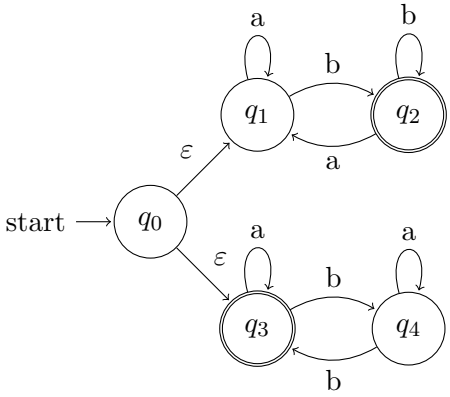


We observe that there are unreachable states like q_{02} and an entire disconnected component. This process does not guarantee to give a minimal DFA, just an equivalent one. On cleaning up these unreachable states, we get the following DFA



Each state represents a superposition of the states in the NFA. A state being unreachable in the DFA could be interpreted to mean that its exact combination of states in the original NFA was unachievable. You cannot be in q_2 in the NFA without also being in q_0 and q_1.

One utility of NFAs is that we can use them to create a more convenient representation of the union of two languages. Consider $L_3 = \{w \in \Sigma^* \mid w \text{ ends with } b\}$ and $L_4 = \{w \in \Sigma^* \mid \#b(w) \text{ is even}\}$ We can construct the following NFA that represents $L_3 \cup L_4$



We can also use this idea to prove that the union of two regular languages is always regular. An alternative way is to follow last lecture's approach using the Cartesian Product, which can be comparatively more cumbersome.

6 The Road Not Taken

By Robert Frost, emphasis mine

Two roads diverged in a yellow wood,
And sorry **I could not travel both**
And **be one traveler,** long I stood
And looked down one as far as I could
To where it bent in the undergrowth;
Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,
And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.
I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

The moral of this poem in the context of our lecture is that Robert Frost is a deterministic actor, one who sees two roads and is forced to choose. If he was nondeterministic, he wouldn't have to choose. He could come to a fork in the road and just take it.