Nondeterministic Finite-State Automata

CS 301:Theory of Computation

Sumesh Divakaran
Department of Computer Science and Engineering
College of Engineering Trivandrum

14^{th} August 2019

Outline

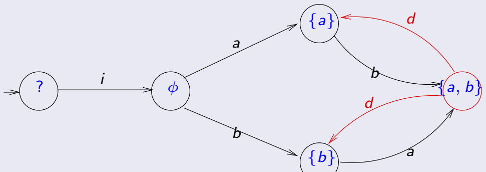
Is the next-state always defined uniquely?

- In a DFA the next-state is uniquely defined for an action/event (represented by an input alphabet) from a state
- There are situations which demand to have multiple next-states from a single state for an alphabet (event or action)
- This kind of a situation can be modeled with an automata which have multiple transitions from a state for a given action (represented by an input alphabet)

Example system demanding multiple next-states

Finite set with init, insert and delete operations

Consider a finite set with init, insert and delete operations. Let i be the alphabet representing the action *init* which initializes the set to ϕ, a be the alphabet representing the action *insert(a)* which inserts the letter a to the set, b be the alphabet representing the action *insert(b)* which inserts the letter b to the set and d be the alphabet representing the action *delete* which deletes an element from a nonempty set.

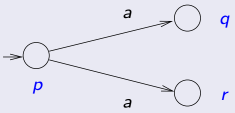


The action d from the state $\{a, b\}$ should take the system to the state $\{a\}$ when it deletes b and to the state $\{b\}$ when it deletes a

Nondeterministic Finite-state Automata (NFA)

- Allows multiple start states.
- Allows more than one transition from a state on a given letter.

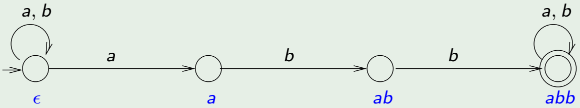
Non-deterministic transitions



- A word is accepted if there is some path on it from a start to a final state.

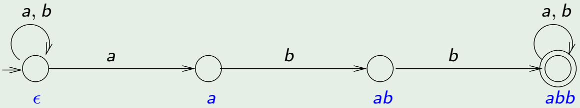
Example NFA

NFA for ?



Example NFA

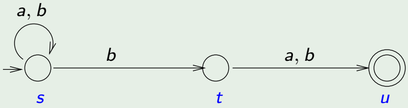
NFA for ?



"contains *abb* as a subword"

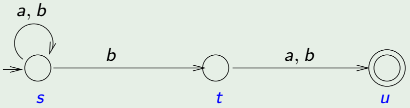
Example NFA

NFA for ?



Example NFA

NFA for ?



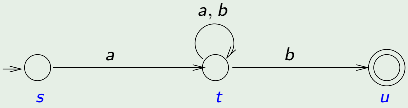
"2^{nd} last symbol is a b"

Example NFA

NFA for "starting with a and ending with b over $\{a, b\}^*$"

Example NFA

NFA for "starting with a and ending with b over $\{a, b\}^*$"

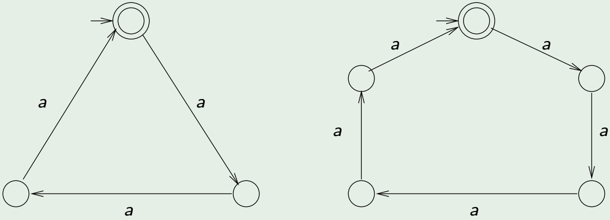


Example NFA

NFA for "$L = \{x \in \{a\}^* \mid \text{length}(x) \text{ is a multiple of 3 or 5}\}$"

Example NFA

NFA for "$L = \{x \in \{a\}^* \mid \text{length}(x) \text{ is a multiple of 3 or 5}\}$"



NFA definition

Mathematical representation of NFA $\mathcal{A}$:

$$
\begin{aligned}
\mathcal{A} =& (Q, S, \Delta, F), \text{ where:} \\
S \subseteq& Q, \text{ the set of start states,} \\
\Delta :& Q \times \Sigma \to 2^Q \\
& \quad \text{where } 2^Q = \{A \mid A \subseteq Q\}
\end{aligned}
$$

$\Delta(p, a)$ gives the set of all states in Q that $\mathcal{A}$ is allowed to move to from the state p on input a. We can write $p \xrightarrow{a} q$ to denote that $q \in \Delta(p, a)$.

Extended transition function for NFAs

The transition function Δ can be extended for strings in Σ^* in a natural way

$$\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$$

according to the rules

$$\begin{aligned} \hat{\Delta}(A, \epsilon) &= A, \\ \hat{\Delta}(A, x \cdot a) &= \Delta(\hat{\Delta}(A, x), a)) \\ &= \bigcup_{q \in \hat{\Delta}(A,x)} (\Delta(q, a)) \end{aligned}$$

$\hat{\Delta}(A, x)$ represents the set of all states reachable under the input string x from some state in A.

Language accepted by an NFA

Language accepted by an NFA

The language $\mathcal{L}(\mathcal{A})$ accepted by an NFA $\mathcal{A}$ is defined as:

$$\mathcal{L}(\mathcal{A}) = \{x \in \Sigma^* \mid \hat{\Delta}(S, x) \cap F \neq \phi\}$$

Thus, a string x is accepted when there exists a path from any start state s to any final state f (i.e. when $s \xrightarrow{x} f$)

NFAs more powerful than DFAs for language acceptance?

NFAs more powerful than DFAs for language acceptance?

No. Nondeterminism doesn't increase expressive power

Equivalence of NFAs and DFAs

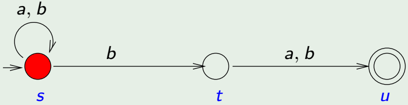
For very NFA M accepting a language $\mathcal{L}(M)$ there exists a DFA N such that $\mathcal{L}(N) = \mathcal{L}(M)$

Corollary

The class of languages accepted by NFA is Regular

How NFA works

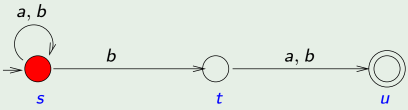
NFA for "2^{nd} last symbol is a b"



Input - *abaaba*

How NFA works

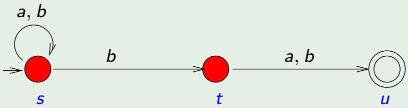
NFA for "2^{nd} last symbol is a b"



Input - *xbaaba*

How NFA works

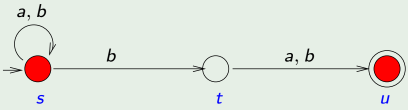
NFA for "2^{nd} last symbol is a b"



Input - *xxaaba*

How NFA works

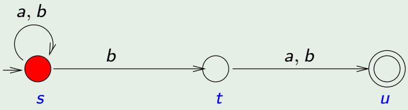
NFA for "2nd last symbol is a b"



Input - *xxxaba*

How NFA works

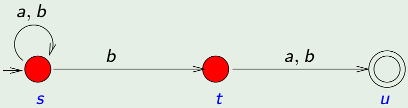
NFA for "2^{nd} last symbol is a b"



Input - *xxxxba*

How NFA works

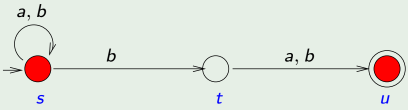
NFA for "2^{nd} last symbol is a b"



Input - $xxxxxa$

How NFA works

NFA for "2^{nd} last symbol is a b"



Input - *xxxxxx*

How NFA works

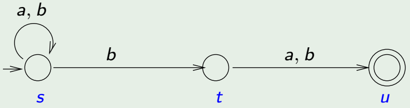
- Initially pebbles should be placed on all the start states
- Let A be the set of states with pebbles and b be the next input symbol. Then, the next set of states to hold the pebbles is:

$$\bigcup_{q \in A} (\Delta(q, b))$$

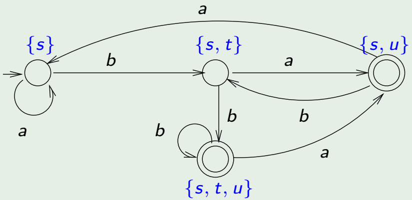
- At any point during the computation, pebbles will be placed in a subset of Q (state set)
- The input string is accepted if one or more final states hold pebbles when the machine finishes reading the input
- Thus, the computation of a given NFA M can be simulated by a DFA N whose states are subsets of states in M

DFA for simulating an NFA

NFA M for "2^{nd} last symbol is a b"



DFA N simulating the NFA M



NFA $\Rightarrow$ DFA

NFA $\Rightarrow$ DFA convertion (subset construction)

Let $M = (Q_M, \Delta_M, S_M, F_M)$ be an NFA over the alphabet set Σ. Then the equivalent DFA (generating same language) $N = (Q_N, \delta_N, s_N, F_N)$ over the same alphabet set Σ can be defined as follows:

$$Q_N = 2^{Q_M}$$

$$s_N = S_M$$

$$\delta_N(A, b) = \bigcup_{q \in A} (\Delta(q, b))$$

$$F_N = \{A \subseteq Q_M \mid A \cap F_M \neq \phi\}$$

NFA $\Rightarrow$ DFA

NFA $\Rightarrow$ DFA convertion (subset construction)

Let $M = (Q_M, \Delta_M, S_M, F_M)$ be an NFA over the alphabet set Σ. Then the equivalent DFA (generating same language) $N = (Q_N, \delta_N, s_N, F_N)$ over the same alphabet set Σ can be defined as follows:

$$\begin{aligned} Q_N &= 2^{Q_M} \\ s_N &= S_M \\ \delta_N(A, b) &= \bigcup_{q \in A} (\Delta(q, b)) \\ F_N &= \{A \subseteq Q_M \mid A \cap F_M \neq \phi\} \end{aligned}$$

Claim

$\mathcal{L}(N) = \mathcal{L}(M)$

Principle of Mathematical Induction

- $\mathbb{N} = \{0, 1, 2 \ldots\}$
- $P(n)$: A statement P about a natural number n.
- Example:
  - $P_0(n)$ = "n is even."
  - $P_1(n)$ = "Sum of the numbers $1 \ldots n$ equals $n(n+1)/2$."
  - $P_2(n)$ = "For any $x, y \in \Sigma^*$ and $A \subseteq Q$, such that $|y| = n$ $\hat{\Delta}(A, xy) = \hat{\Delta}(\hat{\Delta}(A, x), y)$"

Principle of Induction

If a statement P about natural numbers

- is true for 0 (i.e $P(0)$ is true), and,
- is true for $n+1$ whenever it is true for n (i.e. $P(n) \implies P(n+1)$)

then P is true of all natural numbers (i.e. "For all n, $P(n)$" is true).

Proof of $\mathcal{L}(N) = \mathcal{L}(M)$

Lemma 1

For any $x, y \in \Sigma^*$ and $A \subseteq Q$,

$$\hat{\Delta}(A, xy) = \hat{\Delta}(\hat{\Delta}(A, x), y)$$

Proof.

Induction on the length of y

Basis: $|y| = 0 \implies y = \epsilon$

$$\begin{aligned} \hat{\Delta}(A, x\epsilon) =& \hat{\Delta}(A, x) \text{ by definition of concatenation} \\ =& \hat{\Delta}(\hat{\Delta}(A, x), \epsilon) \text{ by definition of } \hat{\Delta} \end{aligned}$$

$\square$

Proof of $\hat{\Delta}(A, x \cdot y) = \hat{\Delta}(\hat{\Delta}(A, x), y)$

Proof.

Induction step: For $y \in \Sigma^*$ and $a \in \Sigma$,

$$\begin{aligned}
\hat{\Delta}(A, xya) &= \Delta(\hat{\Delta}(A, xy), a) \text{ by the definition of } \hat{\Delta} \\
&= \Delta(\hat{\Delta}(\hat{\Delta}(A, x), y), a) \text{ by the induction hypothesis} \\
&= \hat{\Delta}(\hat{\Delta}(A, x), ya) \text{ by the definition of } \hat{\Delta}
\end{aligned}$$

$\square$

Lemma 2 to prove that $\mathcal{L}(N) = \mathcal{L}(M)$

Lemma 2

For any $A \subseteq Q_M$ and $x \in \Sigma^*$,

$$\hat{\delta}_N(A, x) = \hat{\Delta}_M(A, x)$$

Proof.

Induction on the length of x

Basis: $x = \epsilon$,

we want to show that $\hat{\delta}_N(A, \epsilon) = \hat{\Delta}_M(A, \epsilon)$.

This is done since by definitions of $\hat{\delta}_N$ and $\hat{\Delta}_M$ we have

$\hat{\delta}_N(A, \epsilon) = A = \hat{\Delta}_M(A, \epsilon)$ □

Proof of $\hat{\delta}_N(A, x) = \hat{\Delta}_M(A, x)$

Proof.

Induction step: For $x \in \Sigma^*$ and $a \in \Sigma$,

$$\begin{aligned}
\hat{\delta}_N(A, xa) &= \delta_N(\hat{\delta}_N(A, x), a) && \text{by the definition of } \hat{\delta}_N \\
&= \delta_N(\hat{\Delta}_M(A, x), a) && \text{by induction hypothesis} \\
&= \hat{\Delta}_M(\hat{\Delta}_M(A, x), a) && \text{by the definition of } \hat{\delta}_N \\
&= \hat{\Delta}_M(A, xa) && \text{by Lemma 1}
\end{aligned}$$

$\square$

Proof of Theorem 1 ($\mathcal{L}(N) = \mathcal{L}(M)$)

Proof.

For any $x \in \Sigma^*$,

$$
\begin{aligned}
x \in \mathcal{L}(N) &\iff \hat{\delta}_N(s_N, x) \in F_N \\
&\qquad \text{by the definition of acceptance of } N \\
&\iff \hat{\Delta}_M(S_M, x) \cap F_M \neq \phi \\
&\qquad \text{by definition of } s_N \text{ and } F_N \text{ and Lemma 2} \\
&\iff x \in \mathcal{L}(M) \\
&\qquad \text{by definition of acceptance of } M
\end{aligned}
$$

$\square$