CS 374: Algorithms & Models of Computation

Chandra Chekuri

University of Illinois, Urbana-Champaign

Spring 2017
NFAs continued, Closure
Properties of Regular Languages

Lecture 5
January 31, 2017
Theorem

Languages accepted by DFAs, NFAs, and regular expressions are the same.
Theorem

Languages accepted by DFAs, NFAs, and regular expressions are the same.

- DFAs are special cases of NFAs (trivial)
- NFAs accept regular expressions (we saw already)
- DFAs accept languages accepted by NFAs (today)
- Regular expressions for languages accepted by DFAs (later in the course)
Equivalence of NFAs and DFAs

**Theorem**

For every NFA $N$ there is a DFA $M$ such that $L(M) = L(N)$. 
A non-deterministic finite automata (NFA) $N = (Q, \Sigma, \delta, s, A)$ is a five tuple where

- $Q$ is a finite set whose elements are called states,
- $\Sigma$ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow \mathcal{P}(Q)$ is the transition function (here $\mathcal{P}(Q)$ is the power set of $Q$),
- $s \in Q$ is the start state,
- $A \subseteq Q$ is the set of accepting/final states.

$\delta(q, a)$ for $a \in \Sigma \cup \{\epsilon\}$ is a subset of $Q$ — a set of states.
Extending the transition function to strings

**Definition**
For NFA $N = (Q, \Sigma, \delta, s, A)$ and $q \in Q$ the $\epsilon\text{reach}(q)$ is the set of all states that $q$ can reach using only $\epsilon$-transitions.

**Definition**
Inductive definition of $\delta^* : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$:
- if $w = \epsilon$, $\delta^*(q, w) = \epsilon\text{reach}(q)$
- if $w = a$ where $a \in \Sigma$
  \[\delta^*(q, a) = \bigcup_{p \in \epsilon\text{reach}(q)} \left( \bigcup_{r \in \delta(p, a)} \epsilon\text{reach}(r) \right)\]
- if $w = xa$,
  \[\delta^*(q, w) = \bigcup_{p \in \delta^*(q, x)} \left( \bigcup_{r \in \delta(p, a)} \epsilon\text{reach}(r) \right)\]
Formal definition of language accepted by $N$

**Definition**
A string $w$ is accepted by NFA $N$ if $\delta^*_N(s, w) \cap A \neq \emptyset$.

**Definition**
The language $L(N)$ accepted by a NFA $N = (Q, \Sigma, \delta, s, A)$ is

$$\{ w \in \Sigma^* \mid \delta^*(s, w) \cap A \neq \emptyset \}.$$
Simulating an NFA by a DFA

- Think of a program with fixed memory that needs to simulate NFA $N$ on input $w$.
- What does it need to store after seeing a prefix $x$ of $w$?
Think of a program with fixed memory that needs to simulate NFA $N$ on input $w$. What does it need to store after seeing a prefix $x$ of $w$? It needs to know at least $\delta^*(s, x)$, the set of states that $N$ could be in after reading $x$. Is it sufficient?
Simulating an NFA by a DFA

- Think of a program with fixed memory that needs to simulate NFA \( \mathcal{N} \) on input \( w \).
- What does it need to store after seeing a prefix \( x \) of \( w \)?
- It needs to know at least \( \delta^*(s, x) \), the set of states that \( \mathcal{N} \) could be in after reading \( x \).
- Is it sufficient? Yes, if it can compute \( \delta^*(s, xa) \) after seeing another symbol \( a \) in the input.
- When should the program accept a string \( w \)?
Simulating an NFA by a DFA

- Think of a program with fixed memory that needs to simulate NFA $N$ on input $w$.
- What does it need to store after seeing a prefix $x$ of $w$?
- It needs to know at least $\delta^*(s, x)$, the set of states that $N$ could be in after reading $x$.
- Is it sufficient? Yes, if it can compute $\delta^*(s, xa)$ after seeing another symbol $a$ in the input.
- When should the program accept a string $w$? If $\delta^*(s, w) \cap A \neq \emptyset$.

**Key Observation:** A DFA $M$ that simulates $N$ should keep in its memory/state the set of states of $N$.

Thus the state space of the DFA should be $\mathcal{P}(Q)$. 
NFA $N = (Q, \Sigma, s, \delta, A)$. We create a DFA $M = (Q', \Sigma, \delta', s', A')$ as follows:

- $Q' = \mathcal{P}(Q)$
Subset Construction

NFA $N = (Q, \Sigma, s, \delta, A)$. We create a DFA $M = (Q', \Sigma, \delta', s', A')$ as follows:

- $Q' = \mathcal{P}(Q)$
- $s' = \epsilon\text{reach}(s) = \delta^*(s, \epsilon)$
Subset Construction

NFA \( N = (Q, \Sigma, s, \delta, A) \). We create a DFA \( M = (Q', \Sigma, \delta', s', A') \) as follows:

- \( Q' = \mathcal{P}(Q) \)
- \( s' = \epsilon\text{reach}(s) = \delta^*(s, \epsilon) \)
- \( A' = \{ X \subseteq Q \mid X \cap A \neq \emptyset \} \)
Subset Construction

NFA $N = (Q, \Sigma, s, \delta, A)$. We create a DFA $M = (Q', \Sigma, \delta', s', A')$ as follows:

- $Q' = \mathcal{P}(Q)$
- $s' = \epsilon\text{reach}(s) = \delta^*(s, \epsilon)$
- $A' = \{X \subseteq Q \mid X \cap A \neq \emptyset\}$
- $\delta'(X, a) = \bigcup_{q \in X} \delta^*(q, a)$ for each $X \subseteq Q, a \in \Sigma$. 

\[
\{ v_1, v_2, \ldots, v_k \} \quad X \in Q
\]
Example

No \(\epsilon\)-transitions

\[
\begin{array}{c}
q_0 \quad 1 \quad q_1 \\
0, 1 & 0, 1
\end{array}
\]

Thus, to simulate the NFA, the DFA only needs to maintain the current set of states of the NFA. The formal construction based on the above idea is as follows. Consider an NFA \(N = (Q, \Sigma, \delta, s, A)\). Define the DFA \(\text{det}(N) = (Q_0, \Sigma, 0, s_0, A_0)\) as follows.

- \(Q_0 = P(Q)\)
- \(s_0 = \overset{*}{\Sigma}(s, \varepsilon)\)
- \(A_0 = \{X \in Q | X \notin A\} = \{\overset{*}{\Sigma}(q, a) | q \in Q, a \in \Sigma\}\)

An example NFA is shown in Figure 4 along with the DFA \(\text{det}(N)\) in Figure 5.

We will now prove that the DFA defined above is correct. That is

Lemma 4.

\[\L(N) = \L(\text{det}(N))\]

Proof. Need to show \(\forall w \in \Sigma^* \cdot \text{det}(N)\) accepts \(w\) if and only if \(N\) accepts \(w\). For the induction proof to go through we need to strengthen the claim as follows. Again for the induction proof to go through we need to strengthen the claim as follows.

In other words, this says that the state of the DFA after reading some string is exactly the set of states the NFA could be in after reading the same string.

The proof of the strengthened statement is by induction on \(|w|\).

Base Case: If \(|w| = 0\) then \(w = \varepsilon\). Now \(\overset{*}{\Sigma}(s_0, \varepsilon) = s_0 = \overset{*}{\Sigma}(s, \varepsilon)\) by the definition of \(\overset{*}{\Sigma}\) and definition of \(s_0\).
Example

No $\epsilon$-transitions

An example NFA is shown in Figure 4 along with the DFA $\text{det}(N)$ in Figure 5.

We will now prove that the DFA defined above is correct. That is

**Lemma 4.**

$L(N) = L(\text{det}(N))$

**Proof.**

Need to show $\forall w \in \Sigma^*$. $\text{det}(N)$ accepts $w \iff N$ accepts $w$.

$\forall w \in \Sigma^*$. $\text{det}(N)(s_0, w) \in A_0 \iff \text{det}(N)(s_0, w) = \Sigma^*$.

Again for the induction proof to go through we need to strengthen the claim as follows.

$\forall w \in \Sigma^*$. $\text{det}(N)(s_0, w) = \Sigma^*$.

In other words, this says that the state of the DFA after reading some string is exactly the set of states the NFA could be in after reading the same string.

The proof of the strengthened statement is by induction on $|w|$. 

**Base Case**

If $|w| = 0$ then $w = \varepsilon$. Now $\text{det}(N)(s_0, \varepsilon) = s_0 = N(s, \varepsilon)$ by the defn. of $\text{det}(N)$ and defn. of $s_0$. 

**Induction Step**

Assume that for some $k$, the statement holds for all strings of length less than $k$. We need to show that it also holds for strings of length $k+1$. 

Let $w = \varepsilon a$ for some symbol $a$. By the inductive hypothesis, we know that $\text{det}(N)(s_0, \varepsilon) = N(s, \varepsilon)$. Now we need to show that $\text{det}(N)(s_0, \varepsilon a) = N(s, \varepsilon a)$.

By the defn. of $\text{det}(N)$, we have $\text{det}(N)(s_0, \varepsilon a) = \Sigma^*$. By the inductive hypothesis, we know that $N(s, \varepsilon a) = \Sigma^*$. Therefore, $\text{det}(N)(s_0, \varepsilon a) = N(s, \varepsilon a)$.

Thus, the statement holds for all strings of length $k+1$. 

By the principle of mathematical induction, the statement holds for all strings $w \in \Sigma^*$. 

In conclusion, we have shown that $L(N) = L(\text{det}(N))$. 

Chandra Chekuri  (UIUC)  CS374  12  Spring 2017  12 / 19
Incremental construction

Only build states reachable from \( s' = \varepsilon \text{reach}(s) \) the start state of \( M \)

\[
\delta'(X, a) = \bigcup_{q \in X} \delta^*(q, a)
\]
Incremental algorithm

- Build $M$ beginning with start state $s' == \epsilon\text{reach}(s)$
- For each existing state $X \subseteq Q$ consider each $a \in \Sigma$ and calculate the state $Y = \delta'(X, a) = \bigcup_{q \in X} \delta^*(q, a)$ and add a transition.
- If $Y$ is a new state add it to reachable states that need to explored.

To compute $\delta^*(q, a)$ - set of all states reached from $q$ on string $a$

- Compute $X = \epsilon\text{reach}(q)$
- Compute $Y = \bigcup_{p \in X} \delta(p, a)$
- Compute $Z = \epsilon\text{reach}(Y) = \bigcup_{r \in Y} \epsilon\text{reach}(r)$
Proof of Correctness

**Theorem**

Let $N = (Q, \Sigma, s, \delta, A)$ be a NFA and let $M = (Q', \Sigma, \delta', s', A')$ be a DFA constructed from $N$ via the subset construction. Then $L(N) = L(M)$. 

**Stronger claim:**

**Lemma**

For every string $w$, $\delta^*_{N}(s, w) = \delta^*_{M}(s', w)$.

Proof by induction on $|w|$.

**Base case:** $w = \epsilon$.

$\delta^*_N(s, \epsilon) = \epsilon$ reach $(s)$.

$\delta^*_M(s', \epsilon) = s' = \epsilon$ reach $(s)$ by definition of $s'$. 

Chandra Chekuri (UIUC)
Proof of Correctness

Theorem
Let $N = (Q, \Sigma, s, \delta, A)$ be a NFA and let $M = (Q', \Sigma, \delta', s', A')$ be a DFA constructed from $N$ via the subset construction. Then $L(N) = L(M)$.

Stronger claim:

Lemma
For every string $w$, $\delta^*_N(s, w) = \delta^*_M(s', w)$.

Proof by induction on $|w|$.

Base case: $w = \epsilon$.
$\delta^*_N(s, \epsilon) = \epsilon\text{reach}(s)$.
$\delta^*_M(s', \epsilon) = s' = \epsilon\text{reach}(s)$ by definition of $s'$.
Lemma

For every string $w$, $\delta^*_N(s, w) = \delta^*_M(s', w)$.

Inductive step: $w = xa$  
(Note: suffix definition of strings) 
$\delta^*_N(s, xa) = \bigcup_{p \in \delta^*_N(s, x)} \delta^*_N(p, a)$ by inductive defn of $\delta^*_N$
Lemma

For every string \( w \), \( \delta^*_N(s, w) = \delta^*_M(s', w) \).

**Inductive step:** \( w = xa \) (Note: suffix definition of strings)

\[
\delta^*_N(s, xa) = \bigcup_{p \in \delta^*_N(s, x)} \delta^*_N(p, a)
\]

by inductive defn of \( \delta^*_N \)

\[
\delta^*_M(s', xa) = \delta_M(\delta^*_M(s, x), a)
\]

by inductive defn of \( \delta^*_M \)

Thus \( \delta^*_N(s, xa) = \delta^*_M(s', xa) \) which is what we need.
Lemma

For every string $w$, $\delta_N^*(s, w) = \delta_M^*(s', w)$.

Inductive step: $w = xa$ (Note: suffix definition of strings)

$\delta_N^*(s, xa) = \bigcup_{p \in \delta_N^*(s, x)} \delta_N^*(p, a)$ by inductive defn of $\delta_N^*$

$\delta_M^*(s', xa) = \delta_M(\delta_M^*(s, x), a)$ by inductive defn of $\delta_M^*$

By inductive hypothesis: $Y = \delta_N^*(s, x) = \delta_M^*(s, x)$
Lemma

For every string $w$, $\delta^*_N(s, w) = \delta^*_M(s', w)$.

Inductive step: $w = xa$ (Note: suffix definition of strings)

$\delta^*_N(s, xa) = \bigcup_{p \in \delta^*_N(s, x)} \delta^*_N(p, a)$ by inductive defn of $\delta^*_N$

$\delta^*_M(s', xa) = \delta_M(\delta^*_M(s, x), a)$ by inductive defn of $\delta^*_M$

By inductive hypothesis: $Y = \delta^*_N(s, x) = \delta^*_M(s, x)$

Thus $\delta^*_N(s, xa) = \bigcup_{p \in Y} \delta^*_N(p, a) = \delta_M(Y, a)$ by definition of $\delta_M$. 

*Proof continued*
Proof continued

**Lemma**

For every string $w$, $\delta^*_N(s, w) = \delta^*_M(s', w)$.

**Inductive step:** $w = xa$ (Note: suffix definition of strings)

$\delta^*_N(s, xa) = \bigcup_{p \in \delta^*_N(s, x)} \delta^*_N(p, a)$ by inductive defn of $\delta^*_N$

$\delta^*_M(s', xa) = \delta_M(\delta^*_M(s, x), a)$ by inductive defn of $\delta^*_M$

By inductive hypothesis: $Y = \delta^*_N(s, x) = \delta^*_M(s, x)$

Thus $\delta^*_N(s, xa) = \bigcup_{p \in Y} \delta^*_N(p, a) = \delta_M(Y, a)$ by definition of $\delta_M$.

Therefore,

$\delta^*_N(s, xa) = \delta_M(Y, a) = \delta_M(\delta^*_M(s, x), a) = \delta^*_M(s', xa)$

which is what we need.
Part II

Closure Properties of Regular Languages
Regular Languages

Regular languages have three different characterizations

- Inductive definition via base cases and closure under union, concatenation and Kleene star
- Languages accepted by DFAs
- Languages accepted by NFAs

Regular language closed under many operations:
- Union, concatenation, Kleene star via inductive definition or NFAs
- Complement, union, intersection via DFAs
- Homomorphism, inverse homomorphism, reverse, ...

Different representations allow for flexibility in proofs
Regular Languages

Regular languages have three different characterizations

- Inductive definition via base cases and closure under union, concatenation and Kleene star
- Languages accepted by DFAs
- Languages accepted by NFAs

Regular language closed under many operations:

- union, concatenation, Kleene star via inductive definition or NFAs
- complement, union, intersection via DFAs
- homomorphism, inverse homomorphism, reverse, . . .

Different representations allow for flexibility in proofs
Example: PREFIX

Let $L$ be a language over $\Sigma$.

**Definition**

$\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$
Example: PREFIX

Let $L$ be a language over $\Sigma$.

**Definition**

$$\text{PREFIX}(L) = \{ w \mid wx \in L, x \in \Sigma^* \}$$

**Theorem**

*If $L$ is regular then $\text{PREFIX}(L)$ is regular.*
Example: PREFIX

Let $L$ be a language over $\Sigma$.

**Definition**

$\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$

**Theorem**

*If $L$ is regular then $\text{PREFIX}(L)$ is regular.*

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes $L$. Create new DFA $M' = (Q, \Sigma, \delta, s, Z)$ where $X = \{q \in Q \mid s \text{ can reach } q \text{ in } M\}$ and $Y = \{q \in Q \mid q \text{ can reach some state in } A\}$. Then $Z = X \cap Y$ and $L(M') = \text{PREFIX}(L)$.
Example: PREFIX

Let \( L \) be a language over \( \Sigma \).

**Definition**

\[
\text{PREFIX}(L) = \{ w \mid wx \in L, x \in \Sigma^* \}
\]

**Theorem**

*If \( L \) is regular then \( \text{PREFIX}(L) \) is regular.*

Let \( M = (Q, \Sigma, \delta, s, A) \) be a DFA that recognizes \( L \)

\[
X = \{ q \in Q \mid s \text{ can reach } q \text{ in } M \}
\]

Create new DFA \( M' = (Q, \Sigma, \delta, s, Z) \)

Claim: \( L(M) = \text{PREFIX}(L) \).
Example: PREFIX

Let $L$ be a language over $\Sigma$.

**Definition**

$\text{PREFIX}(L) = \{w | wx \in L, x \in \Sigma^*\}$

**Theorem**

*If $L$ is regular then $\text{PREFIX}(L)$ is regular.*

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes $L$

$X = \{q \in Q | s \text{ can reach } q \text{ in } M\}$

$Y = \{q \in Q | q \text{ can reach some state in } A\}$
Example: PREFIX

Let $L$ be a language over $\Sigma$.

**Definition**

$\text{PREFIX}(L) = \{ w | wx \in L, x \in \Sigma^* \}$

**Theorem**

*If $L$ is regular then $\text{PREFIX}(L)$ is regular.*

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes $L$.

$X = \{ q \in Q | s \text{ can reach } q \text{ in } M \}$

$Y = \{ q \in Q | q \text{ can reach some state in } A \}$

$Z = X \cap Y$

Create new DFA $M' = (Q, \Sigma, \delta, s, Z)$
Example: PREFIX

Let $L$ be a language over $\Sigma$.

**Definition**

$\text{PREFIX}(L) = \{ w \mid wx \in L, x \in \Sigma^* \}$

**Theorem**

If $L$ is regular then $\text{PREFIX}(L)$ is regular.

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes $L$

$X = \{ q \in Q \mid s \text{ can reach } q \text{ in } M \}$

$Y = \{ q \in Q \mid q \text{ can reach some state in } A \}$

$Z = X \cap Y$

Create new DFA $M' = (Q, \Sigma, \delta, s, Z)$

**Claim:** $L(M) = \text{PREFIX}(L)$. 
Exercise: SUFFIX

Let $L$ be a language over $\Sigma$.

**Definition**

$\text{SUFFIX}(L) = \{ w \mid xw \in L, x \in \Sigma^* \}$

Prove the following:

**Theorem**

*If $L$ is regular then $\text{PREFIX}(L)$ is regular.*