NFA/DFA: Closure Properties, Relation to Regular Languages

Lecture 5
Today

NFAs recap: Determinizing an NFA

Closure Properties of class of languages accepted by NFAs/DFAs

Towards proving equivalence of regular languages and languages accepted by NFAs (and hence DFAs)

More closure Properties of regular languages
NFA: Formally

\[ N = (\Sigma, Q, \delta, s, F) \]

\( \Sigma \): alphabet
\( Q \): state space
\( s \): start state
\( F \): set of accepting states

\[ \delta : Q \times (\Sigma \cup \varepsilon) \rightarrow \mathcal{P}(Q) \]

By default, NFA can have \( \varepsilon \)-moves

We say \( q \xrightarrow{w} N p \) if \( \exists a_1, \ldots, a_t \in \Sigma \cup \{ \varepsilon \} \) and \( q_1, \ldots, q_{t+1} \in Q \), such that
\[ w = a_1 \ldots a_t, \quad q_1 = q, \quad q_{t+1} = p, \quad \text{and} \quad \forall i \in [1, t], \quad q_{i+1} \in \delta(q_i, a_i) \]

\[ L(N) = \{ w \mid s \xrightarrow{w} N p \text{ for some } p \in F \} \]

e.g., \( \delta(1, a) = \{ 2 \} \), \( \delta(1, x) = \emptyset \), \( \delta(1, \varepsilon) = \{ 2 \} \).

\( \varepsilon \)-closure \( C\varepsilon(\{ 1 \}) = \{ 1, 2, 3, 0 \} \)
**ε-Moves is Syntactic Sugar**

Can modify any NFA $N$, to get an NFA $N_{\text{new}}$ without ε-moves

$$N_{\text{new}} = (\Sigma, Q, \delta_{\text{new}}, s, F_{\text{new}})$$

$$\delta_{\text{new}}(q, a) = C_\varepsilon(\delta( C_\varepsilon\{\{q\}\}, a))$$

e.g.: $\delta_{\text{new}}(1, 0) = \{0, 2, 3, 4, 5\}$

For $|w| \geq 1$, $q \xrightarrow{w} N p \iff q \xrightarrow{w} N_{\text{new}} p$

$$F_{\text{new}} = \begin{cases} F, & \text{if } C_\varepsilon(\{s\}) \cap F = \emptyset \\ F \cup \{s\}, & \text{otherwise.} \end{cases}$$

**Theorem:** $L(N) = L(N_{\text{new}})$
**ε-Moves is Syntactic Sugar**

Can modify any NFA $N$, to get an NFA $N_{\text{new}}$ without $\varepsilon$-moves

$$N_{\text{new}} = (\Sigma, Q, \delta_{\text{new}}, s, F_{\text{new}})$$

$$\delta_{\text{new}}(q, a) = C_\varepsilon(\delta( C_\varepsilon({q}), a))$$

<table>
<thead>
<tr>
<th>$q$</th>
<th>$C$</th>
<th>$a$</th>
<th>$\delta$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>{ a, b, d}</td>
<td>0</td>
<td>{ a, b, c }</td>
<td>{ a, b, c, d }</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>{ b, d }</td>
<td>{ b, d }</td>
</tr>
<tr>
<td>b</td>
<td>{ b, d }</td>
<td>0</td>
<td>{ b, c }</td>
<td>{ b, c, d }</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>{ d }</td>
<td>{ d }</td>
</tr>
<tr>
<td>c</td>
<td>{ c }</td>
<td>0</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>{ d }</td>
<td>{ d }</td>
</tr>
<tr>
<td>d</td>
<td>{ d }</td>
<td>0</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>{ d }</td>
<td>{ d }</td>
</tr>
</tbody>
</table>
Can modify any NFA $N$, to get an NFA $N_{\text{new}}$ without $\varepsilon$-moves

$$N_{\text{new}} = (\Sigma, Q, \delta_{\text{new}}, s, F_{\text{new}})$$

$$\delta_{\text{new}}(q, a) = C_{\varepsilon}(\delta(C_{\varepsilon}(\{q\}), a))$$

$\varepsilon$-Moves is Syntactic Sugar

$F_{\text{new}} = \begin{cases} F, & \text{if } C_{\varepsilon}(\{s\}) \cap F = \emptyset \\ F \cup \{s\}, & \text{otherwise.} \end{cases}$
NFA to DFA

Can modify any NFA $N$, to get an equivalent DFA $M$

To avoid errors, first, remove $\varepsilon$-moves

<table>
<thead>
<tr>
<th>$q$</th>
<th>$a$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>${a,b,c,d}$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>${b,d}$</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>${a,b,c,d}$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>${d}$</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>${d}$</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>${d}$</td>
</tr>
</tbody>
</table>

$\delta$

$T$

$\{a\}$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>${a,b,c,d}$</td>
</tr>
<tr>
<td>1</td>
<td>${b,d}$</td>
</tr>
</tbody>
</table>

$\{a,b,c,d\}$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>${a,b,c,d}$</td>
</tr>
<tr>
<td>1</td>
<td>${b,d}$</td>
</tr>
</tbody>
</table>

$\{b,d\}$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>${b,c,d}$</td>
</tr>
<tr>
<td>1</td>
<td>${d}$</td>
</tr>
</tbody>
</table>

$\{b,c,d\}$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>${b,c,d}$</td>
</tr>
<tr>
<td>1</td>
<td>${d}$</td>
</tr>
</tbody>
</table>

$\{d\}$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>1</td>
<td>${d}$</td>
</tr>
</tbody>
</table>

$\emptyset$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>1</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

Remember to specify final states

0,1
NFA to DFA: Formally

NFA: \( N = (\Sigma, Q, \delta, s, F) \)

\( \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q) \)

DFA: \( M_N = (\Sigma, \mathcal{P}(Q), \delta^+, s^+, F^+) \)

\( \delta^+ : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q) \)

\( \delta^+(T, a) = \bigcup_{q \in T} \delta(q, a) \)

\( s^+ = \{s\}, \quad F^+ = \{ T \mid T \cap F \neq \emptyset \} \)

Theorem: \( L(N) = L(M_N) \)

Proof? Recall definitions of \( L(DFA) \), \( L(NFA) \)
Language Accepted by a DFA

DFA: \( M = (\Sigma, Q_M, \delta_M, s_M, F_M) \)

Two ways to define the state that an input \( w \) leads to starting from a state

\[
q \xrightarrow{w} p \quad \text{if} \quad w = a_1 \ldots a_t \quad \text{and} \quad \exists \, q_1, \ldots, q_{t+1},
\]
such that \( q_1 = q, q_{t+1} = p, \) and

\[
\forall \, i \in [1, t], \quad q_{i+1} = \delta_M(q_i, a_i)
\]

\[\delta^*(q, \varepsilon) = q\]
\[\delta^*(q, au) = \delta^*(\delta_M(q, a), u)\]

Theorem: \( q \xrightarrow{w} p \iff p = \delta^*(q, w) \)

\[
L(M) = \{ \, w \mid \exists \, p \in F_M, \, s_M \xrightarrow{w} p \, \} = \{ \, w \mid \delta^*(s_M, w) \in F_M \, \} = \{ \, w \mid s_{M_{\epsilon}} \uparrow \in F_M \, \}
\]

Prove!
Language Accepted by an NFA

NFA: \( N = (\Sigma, Q_N, \delta_N, s_N, F_N) \)

Two ways to define the set of states that an input \( w \) leads to starting from a set of states

\[
q \overset{w}{\rightarrow} p \\
\begin{align*}
\text{if } \exists a_1 \ldots a_t \text{ and } q_1, \ldots, q_{t+1}, \text{ such that } w = a_1 \ldots a_t, \ q_1 = q, \ q_{t+1} = p, \\
\text{and } \forall \ i \in [1, t], \ q_{i+1} \in \delta_N(q_i, a_i)
\end{align*}
\]

\[
\delta^+(T, a) = \bigcup_{q \in T} \delta_N(q, a)
\]

\[
\delta^+(T, \varepsilon) = T
\]

\[
\delta^+(T, au) = \delta^+(\delta^+(T, a), u)
\]

\[
\begin{align*}
s^+ &= \{s_N\}, \\
F^+ &= \{ T \mid T \cap F_N \neq \emptyset \}
\end{align*}
\]

Theorem: \( q \overset{w}{\rightarrow} p \iff p \in \delta^+({\{q\}}, w) \)

\[
L(N) = \{ w \mid \exists p \in F_N, \ s_N \overset{w}{\rightarrow} p \} = \{ w \mid \delta^+({\{s_N\}}, w) \cap F_N \neq \emptyset \} = \{ w \mid \delta^+(s^+, w) \in F^+ \}
\]
## Side-by-Side

### DFA: $M = (\Sigma, Q_M, \delta_M, s_M, F_M)$

$\delta_M : Q_M \times \Sigma \rightarrow Q_M$

- $\delta^*(q, \varepsilon) = q$
- $\delta^*(q, au) = \delta^*(\delta_M(q, a), u)$

$L(M) = \{ w \mid \delta^*(s_M, w) \in F_M \}$

### NFA: $N = (\Sigma, Q_N, \delta_N, s_N, F_N)$

- $\delta_N : Q_N \times \Sigma \rightarrow \mathcal{P}(Q_N)$
- $\delta^* : \mathcal{P}(Q_N) \times \Sigma \rightarrow \mathcal{P}(Q_N)$
- $\delta^+(T, a) = \bigcup_{q \in T} \delta_N(q, a)$

$L(N) = \{ w \mid \delta^+*(s^+, w) \in F^+ \}$

If $Q_M = \mathcal{P}(Q_N), \delta_M = \delta^+, s_M = s^+, F_M = F^+$, then $L(M) = L(N)$
Closure Properties for NFAs

If $L$ has an NFA, then $\text{op}(L)$ has an NFA where $\text{op}$ can be complement or Kleene star.

If $L_1$ and $L_2$ each has an NFA, then $L_1 \text{ op } L_2$ has an NFA where $\text{op}$ can be a binary set operation (e.g., union, intersection, difference etc.) or concatenation.

Complement and Binary set operations
Consider the equivalent DFA

Union can be seen directly too…
Closure Under Union
Closure Properties for NFAs

If $L$ has an NFA, then $\text{op}(L)$ has an NFA where $\text{op}$ can be complement or Kleene star.

If $L_1$ and $L_2$ each has an NFA, then $L_1 \text{ op } L_2$ has an NFA where $\text{op}$ can be a binary set operation (e.g., union, intersection, difference etc.) or concatenation.

Complement and Binary set operations
Consider the equivalent DFA

(Union can be seen directly too...)

Now: concatenation and Kleene star
Single Final State Form

Can compile a given NFA so that there is only one final state (and there is no transition out of that state)
Closure Under Concatenation
Closure Under Kleene Star

\[ \varepsilon \]
NFAs & Regular Languages

**Theorem**: For any language \( L \), the following are equivalent:

(a) \( L \) is accepted by an NFA
(b) \( L \) is accepted by a DFA
(c) \( L \) is regular

**Saw**: (a) \( \Rightarrow \) (b)

**Later**: (b) \( \Rightarrow \) (c)

**Now**: (c) \( \Rightarrow \) (a)

**Proof** of (c) \( \Rightarrow \) (a) : By induction on the least number of operators in a regular expression for the language
Theorem: $L$ regular $\Rightarrow L$ is accepted by an NFA

Proof: To prove that if $L = L(r)$ for some regex $r$, then $L = L(N)$ for some NFA $N$. By induction on the number of operators in the regex.

Base case: $L$ has a regular expression with 0 operators. Then the regex should be one of $\emptyset, \varepsilon, a \in \Sigma$. In each case, $\exists N$ s.t. $L = L(N)$. ✓

Inductive step: Let $n > 0$. Assume that every language which has a regex with $k$ operators has an NFA, where $0 \leq k < n$.

If $L$ has a regex with $n$ operators, it must be of the form $r_1r_2$, $r_1 + r_2$, or $r_1^*$, and hence $L = L_1L_2$, or $L_1 \cup L_2$ or $(L_1)^*$, where $L_1 = L(r_1)$ and $L_2 = L(r_2)$. Since $r_1$ and $r_2$ must have $< n$ operators, by IH $L_1$, $L_2$ have NFAs. By closure of NFAs under these operations, so does $L$. ✓
NFAs & Regular Languages

Example: $L$ given by regular expression $(10+1)^*$
Closure Properties for Regular Languages

**Theorem**: If \( L_i \) are regular then, so is:

- \( L_1 \cup L_2 \), \( L_1^* \), \( L_1L_2 \)
- \( \overline{L}_1 \)
- \( L_1 \cap L_2 \)
- \( \text{formula}(L_1, L_2, \ldots, L_k) \)
- \( \text{suffix}(L_1) \)
- \( h(L_1) \) and \( h^{-1}(L_1) \), where \( h \) is a homomorphism
- \( \ldots \)

From the definition of regular languages (or from NFA closure properties)

By considering DFAs for the languages and using the complement construction for DFAs

By De Morgan’s Law (or by the cross-product construction for DFAs)

Skipped from this course
More Closure Properties

\[ \text{formula}_f(L_1, \ldots, L_k) = \{ w \mid f(b_1, \ldots, b_k) \text{ holds, where } b_i \equiv (w \in L_i) \} \]

e.g., \( f(b_1, b_2, b_3) = \text{majority} (b_1, b_2, b_3) \)

**Theorem:** If \( L_1, \ldots, L_k \) are regular, then for any boolean formula \( f \), \( \text{formula}_f(L_1, \ldots, L_k) \) is regular

**Proof:** Any boolean formula can be written using operators \( \land, \lor \) and \( \neg \) (AND, OR, NOT).

\[
\begin{align*}
\text{formula}_f \land g(L_1, \ldots, L_k) &= \text{formula}_f(L_1, \ldots, L_k) \cap \text{formula}_g(L_1, \ldots, L_k) \\
\text{formula}_f \lor g(L_1, \ldots, L_k) &= \text{formula}_f(L_1, \ldots, L_k) \cup \text{formula}_g(L_1, \ldots, L_k) \\
\text{formula}_{\neg f}(L_1, \ldots, L_k) &= \Sigma^* - \text{formula}_f(L_1, \ldots, L_k)
\end{align*}
\]

Complete the proof by induction on the number of operators in \( f \).
More Closure Properties

\[ \text{suffix}(L) = \{ w \mid w \text{ is a suffix of a string in } L \} = \{ w \mid \exists x \in \Sigma^* \; xw \in L \} \]

**Theorem**: If \( L \) is regular, then suffix\((L)\) is regular

**Proof**: Let \( M \) be a DFA for \( L \).

We shall construct an NFA \( N \) s.t. \( L(N) = \text{suffix}(L(M)) \).

**Idea**: \( N \) will guess the state that \( M \) will be in after seeing a “correct” \( x \) and directly jump to that state. Then starts behaving like \( M \).

Need to ensure that (some thread of) \( N \) accepts \( w \) iff \( w \in \text{suffix}(L) \).

If \( w \in \text{suffix}(L), \exists x, xw \in L. \) Hence \( \exists q \) s.t. \( s \xrightarrow{x}_M q \) and \( q \xrightarrow{w}_M p, p \in F \).

So some thread of \( N \) will jump to \( q \) (\( s \xrightarrow{\epsilon}_N q \)) and accept \( w \) (\( q \xrightarrow{w}_N p \)).

Converse? Trouble if \( N \) jumps to \( q \) and accepts \( w \) from there, but no \( x \) could take \( M \) to \( q \) (i.e., \( q \) unreachable)!
More Closure Properties

suffix(L) = \{ w \mid w \text{ is a suffix of a string in } L \} = \{ w \mid \exists x \in \Sigma^* \ xw \in L \}

**Theorem:** If \( L \) is regular, then suffix(\( L \)) is regular

**Proof:** Let \( M \) be a DFA for \( L \).

We shall construct an NFA \( N \) s.t. \( L(N) = \text{suffix}(L(M)) \).

**Idea:** \( N \) will guess the state that \( M \) will be in after seeing a “correct” \( x \) and directly jump to that state. Then starts behaving like \( M \).

\[
Q_N = Q_M \cup \{s_N\}. \quad F_N = F_M.
\]

\[
\delta_N(q,a) = \{\delta_M(q,a)\} \text{ for } q \in Q_M.
\]

\[
\delta_N(s_N,\epsilon) = \{q \in Q_M \mid q \text{ reachable from } s_M\}
\]

Exercise: Verify “corner cases”: e.g., \( L = \emptyset \), \( \epsilon \notin L \) etc.
More Closure Properties (FYI): Homomorphism/Inverse Homomorphism

Suppose given a mapping \( h : \Sigma \rightarrow \Delta^* \).

Given DFA \( M \) over \( \Sigma \), consider NFA \( N \) over \( \Delta \) (with additional states) s.t. for any two of the original states, \( p, q \), if \( p \xrightarrow{a} M q \) then \( p \xrightarrow{h(a)} N q \) via a path of new states.

Given DFA \( M \) over \( \Delta \), consider DFA \( K \) over \( \Sigma \) and the same set of states, s.t. \( p \xrightarrow{a} K q \) iff \( p \xrightarrow{h(a)} M q \).

\[
\begin{align*}
L(N) &= h(L(M)) \\
e.g., \quad &h(a) = 01
\end{align*}
\]

\[
\begin{align*}
L(K) &= h^{-1}(L(M))
\end{align*}
\]