Properties of Regular Languages

11 February 2020
Nondeterministic lyric transition graph
Assignment 2

Due today

Corrections due Thursday
Relating DFAs and NFAs
What did we see last time?
Any language that can be recognized by a DFA can be recognized by an NFA.

Essentially, every DFA already is an NFA, just one that doesn’t exploit nondeterminism.
Any NFA can be converted into an equivalent DFA.
This method of transforming an NFA into a DFA is called the *subset construction* (or power set construction).

Each state in the DFA is associated with a set of states in the NFA. The start state in the DFA corresponds to the start state of the NFA.

If a state $q$ in the DFA corresponds to a set of states $S$ in the NFA, then the transition from state $q$ on a character $\alpha$ is found as follows:

Let $S'$ be the set of states in the NFA that can be reached by following a transition labeled $\alpha$ from any of the states in $S$.

The state $q$ in the DFA transitions on $\alpha$ to a DFA state corresponding to the set of states $S'$. 
What about NFAs with \( \varepsilon \)-transitions?
\[ \begin{align*}
q_0 & \overset{a,b}{\to} q_2 \\
q_0 & \overset{\varepsilon}{\to} q_3 \\
q_0 & \overset{a}{\to} q_1 \\
q_0 & \overset{b}{\to} q_2 \\
q_1 & \overset{b}{\to} q_2 \\
q_3 & \overset{a,b}{\to} q_4 \\
q_4 & \overset{b}{\to} q_3 \\
\end{align*} \]

\[ \rightarrow \{ q_0, q_3 \} \]
\( \rightarrow \{q_0, q_3\} \)
\[(\{q_0, q_3\}) \rightarrow \{q_0, q_3\}\]
\begin{align*}
\delta(q_0, a, b) &= \{q_0, q_3\} \\
\delta(q_0, \epsilon) &= \emptyset \\
\delta(q_1, a) &= q_1 \\
\delta(q_2, a) &= q_1 \\
\delta(q_2, b) &= q_0 \\
\delta(q_3, a) &= q_4 \\
\delta(q_3, b) &= q_4 \\
\delta(q_4, a) &= q_4 \\
\delta(q_4, b) &= q_4
\end{align*}
\[
q_0 \xrightarrow{a} q_1 \xrightarrow{a, b} q_2 \xrightarrow{b} q_3 \xrightarrow{b} q_4 \xrightarrow{a, b} q_2 \xrightarrow{a, b} q_3 \rightarrow \{q_0, q_3\}
\]
\[ q_0 \xrightarrow{a,b} q_2 \xrightarrow{b} q_1 \]

\[ q_0 \xrightarrow{} q_3 \xrightarrow{a,b} q_4 \]

\[ \rightarrow \{q_0, q_3\} \quad \{q_1, q_4\} \]
\[
\begin{align*}
q_0 \quad &\xrightarrow{a} q_1 \\
&\xrightarrow{\varepsilon} q_3 \\
&\xrightarrow{a, b} q_4 \\
&\xrightarrow{b} q_4 \\
q_2 \quad &\xrightarrow{a, b} \{q_0, q_3\} \\
&\xrightarrow{b} \{q_1, q_4\}
\end{align*}
\]
\[ \begin{align*}
q_0 &\rightarrow \{q_0, q_3\} \\
q_1 &\rightarrow \{q_1, q_4\}
\end{align*} \]
The diagram represents a deterministic finite automaton (DFA). The start state is $q_0$. The transitions are:
- From $q_0$, on input $a$, move to $q_1$, and on input $b$, move to $q_2$.
- From $q_1$, on input $a$, move to $q_3$, and on input $b$, move to $q_4$.
- From $q_2$, on input $b$, move to $q_1$.
- From $q_3$, on input $a$, move to $q_3$, and on input $b$, move to $q_4$.

The final states are $\{q_0, q_3\}$ and $\{q_1, q_4\}$. The language recognized by this DFA includes strings that start with $a$ or $b$ and end with $b$.
\[ \begin{align*}
\text{start} & \rightarrow \{q_0, q_3\} \\
q_0 & \xrightarrow{a, b} q_1 \\
q_0 & \xrightarrow{\varepsilon} q_3 \\
q_3 & \xrightarrow{a, b} q_4 \\
q_2 & \xrightarrow{b} q_1 \\
q_2 & \xrightarrow{b} \{q_1, q_4\}
\end{align*} \]
\[
\begin{align*}
q_0 &\xrightarrow{\epsilon} q_3 \\
q_0 &\xrightarrow{a} q_1 \\
q_0 &\xrightarrow{b} q_2 \\
q_1 &\xrightarrow{a} q_2 \\
q_2 &\xrightarrow{b} q_1 \\
q_3 &\xrightarrow{a, b} q_4 \\
q_4 &\xrightarrow{b} q_3 \\
\end{align*}
\]

\[\rightarrow \{q_0, q_3\} \quad \{q_1, q_4\}\]
\[
\begin{align*}
q_0 & \rightarrow \{q_0, q_3\} \\
q_1 & \rightarrow \{q_1, q_4\}
\end{align*}
\]
\[
\begin{align*}
q_0 & \xrightarrow{a, b} q_1 \\
q_0 & \xrightarrow{\varepsilon} q_3 \\
q_2 & \xrightarrow{a, b} q_1 \\
q_3 & \xrightarrow{a, b} q_4 \\
q_4 & \xrightarrow{b} q_4 \\
\end{align*}
\]

\[
\begin{array}{c}
a \\
b \\
\rightarrow \{q_0, q_3\} \\
\{q_1, q_4\} \\
\{q_4\} \\
\end{array}
\]
Many steps later…

\[ \{q_0, q_3\} \rightarrow\{q_1, q_4\} \rightarrow \{q_4\} \]
\[
\begin{array}{c|c|c|c}
\text{a} & \{q_1, q_4\} & \{q_4\} \\
\text{b} & \{q_2, q_3\} & \{q_3\} \\
\end{array}
\]
Creating a DFA from an NFA with $\varepsilon$-transitions

1. Compute the $\varepsilon$-closure for each state, i.e., the set of states reachable from that state following only $\varepsilon$-transitions.

2. The start state is the $\varepsilon$-closure of $q_0$, i.e., $E(\{q_0\})$.

3. Define $\delta$ for each $\alpha \in \Sigma$ and each $\varepsilon$-closed set $S$:

   If a state $p \in S$ can reach state $q$ on input $\alpha$ (not $\varepsilon$!), then add a transition on input $\alpha$ from $S$ to $E(q)$.

4. The set of final states for the DFA now includes those sets that contain at least one accepting state of the NFA-$\varepsilon$. 

\( \epsilon \)-closure example
ε-closure example
ε-closure example
\( \varepsilon \)-closure example

Find the set \( E(\{s\}) \):
Find the set $E(\{s\})$:

$E(\{s\}) = \{s\}$
Find the set $E(\{s\})$:

$E(\{s\}) = \{s\}$  
initial step

$E(\{s\}) = \{s, w\}$  
add $\delta(s, \varepsilon)$
Find the set $E(\{s\})$:

- $E(\{s\}) = \{s\}$  
  initial step
- $E(\{s\}) = \{s, w\}$  
  add $\delta(s, \varepsilon)$
- $E(\{s\}) = \{s, w, q_0\}$  
  add $\delta(w, \varepsilon)$

$\varepsilon$-closure example
Find the set $E(\{s\})$:

- $E(\{s\}) = \{s\}$  
  initial step
- $E(\{s\}) = \{s, w\}$  
  add $\delta(s, \varepsilon)$
- $E(\{s\}) = \{s, w, q_0\}$  
  add $\delta(w, \varepsilon)$
- $E(\{s\}) = \{s, w, q_0, p, t\}$  
  add $\delta(q_0, \varepsilon)$
Find the set $E(\{s\})$:

- $E(\{s\}) = \{s\}$
- $E(\{s\}) = \{s, w\}$
- $E(\{s\}) = \{s, w, q_0\}$
- $E(\{s\}) = \{s, w, q_0, p, t\}$

Adding $\delta(s, \varepsilon)$:

- $E(\{s\}) = \{s, w\}$

Adding $\delta(w, \varepsilon)$:

- $E(\{s\}) = \{s, w, q_0\}$

Adding $\delta(q_0, \varepsilon)$:

- $E(\{s\}) = \{s, w, q_0, p, t\}$

$\delta(p, \varepsilon) = \delta(t, \varepsilon) = \emptyset$, so we are done: $E(\{s\}) = \{s, w, q_0, p, t\}$
Example

NFA

\[
\begin{align*}
q & \xrightarrow{0} r \\
r & \xrightarrow{\varepsilon} s
\end{align*}
\]

\[
\begin{align*}
q & \xrightarrow{1} s \\
r & \xrightarrow{0} q \\
r & \xrightarrow{\varepsilon} r
\end{align*}
\]
Example

1. Compute $\varepsilon$-closure for all states:
   
   $E(\{q\}) = \{q\}$
   
   $E(\{r\}) = \{r, s\}$
   
   $E(\{s\}) = \{r, s\}$
Example

1. Compute $\varepsilon$-closure for all states:
   - $E(\{q\}) = \{q\}$
   - $E(\{r\}) = \{r, s\}$
   - $E(\{s\}) = \{r, s\}$

2. Start state = $E(\{q\}) = \{q\}$
Example

1. Compute $\varepsilon$-closure for all states:
   - $E(\{q\}) = \{q\}$
   - $E(\{r\}) = \{r, s\}$
   - $E(\{s\}) = \{r, s\}$

2. Start state = $E(\{q\}) = \{q\}$

3. Compute $\delta$ for input and set of states from Step 1.
   - $\delta(\{q\}, 0) = E(\{s\}) = \{r, s\}$
   - $\delta(\{q\}, 1) = E(\{r\}) = \{r, s\}$
   - $\delta(\{r, s\}, 0) = E(\{q\}) = \{q\}$
   - $\delta(\{r, s\}, 1) = E(\{q\}) = \{q\}$
Example

1. Compute $\varepsilon$-closure for all states:
   
   $\mathit{E}(\{q\}) = \{q\}$
   $\mathit{E}(\{r\}) = \{r, s\}$
   $\mathit{E}(\{s\}) = \{r, s\}$

2. Start state $= \mathit{E}(\{q\}) = \{q\}$

3. Compute $\delta$ for input and set of states from Step 1.
   
   $\delta(\{q\}, 0) = \mathit{E}(\{s\}) = \{r, s\}$
   $\delta(\{q\}, 1) = \mathit{E}(\{r\}) = \{r, s\}$
   $\delta(\{r, s\}, 0) = \mathit{E}(\{q\}) = \{q\}$
   $\delta(\{r, s\}, 1) = \mathit{E}(\{q\}) = \{q\}$

4. Final states $F_D = \{\{r, s\}\}$
Example

1. Compute $\varepsilon$-closure for all states:
   
   $E(\{q\}) = \{q\}$
   $E(\{r\}) = \{r, s\}$
   $E(\{s\}) = \{r, s\}$

2. Start state $= E(\{q\}) = \{q\}$

3. Compute $\delta$ for input and set of states from Step 1.
   
   $\delta(\{q\}, 0) = E(\{s\}) = \{r, s\}$
   $\delta(\{q\}, 1) = E(\{r\}) = \{r, s\}$
   $\delta(\{r, s\}, 0) = E(\{q\}) = \{q\}$
   $\delta(\{r, s\}, 1) = E(\{q\}) = \{q\}$

4. Final states $F_D = \{\{r, s\}\}$
Exercise

Convert this NFA to a DFA.
Step 1

\[
\begin{align*}
E(\{q_0\}) &= \{q_0, q_1, q_6\} \\
E(\{q_1\}) &= \{q_1\} \\
E(\{q_2\}) &= \{q_2, q_3\} \\
E(\{q_3\}) &= \{q_3\} \\
E(\{q_4\}) &= \{q_4, q_5\} \\
E(\{q_5\}) &= \{q_5\} \\
E(\{q_6\}) &= \{q_6\} \\
E(\{q_7\}) &= \{q_7, q_5\}
\end{align*}
\]

Step 2

\[
\begin{align*}
\text{Start} &= E(\{q_0\}) = \{q_0, q_1, q_6\}
\end{align*}
\]

Step 3

\[
\begin{align*}
\delta(\{q_0, q_1, q_6\}, a) &= E(\{q_2, q_7\}) = \{q_2, q_3, q_7, q_5\} \\
\delta(\{q_2, q_3, q_7, q_5\}, a) &= E(\{q_4\}) = \{q_4, q_5\} \\
\delta(\{q_4, q_5\}, a) &= \emptyset
\end{align*}
\]

Step 4

\[
\text{Final} = \{ \{q_2, q_3, q_7, q_5\}, \{q_4, q_5\} \}
\]
This method of transforming an NFA into a DFA is called the **subset construction** (or power set construction).

Each state in the DFA is associated with a set of states in the NFA. The start state in the DFA corresponds to the start state of the NFA, *plus all states reachable via ε-transitions*.

If a state \( q \) in the DFA corresponds to a set of states \( S \) in the NFA, then the transition from state \( q \) on a character \( α \) is found as follows:

Let \( S' \) be the set of states in the NFA that can be reached by following a transition labeled \( α \) from any of the states in \( S \).

*Let \( S'' \) be the set of states in the NFA reachable from some state in \( S' \) by following zero or more ε-transitions.*

The state \( q \) in the DFA transitions on \( α \) to a DFA state corresponding to the set of states \( S'' \).
Wrap-up
A language is called a **regular language** if there exists a DFA $D$ such that $L(D) = L$. 
THEOREM. A language $L$ is regular iff there is some NFA $N$ such that $L(N) = L$.

PROOF SKETCH. If $L$ is regular, there exists some DFA for it, which we can easily convert into an NFA.

If $L$ is accepted by some NFA, we can use the subset construction to convert it into a DFA that accepts the same language, so $L$ is regular.
We now have two perspectives on regular languages:

- Regular languages are languages recognized by DFAs.
- Regular languages are languages recognized by NFAs.

We can now reason about regular languages in two different ways, and we can use whichever model is more convenient.
Closure properties of regular languages
Recall: We previously showed that regular languages were closed under complement.

Every regular language is recognized by a DFA, and we can build a DFA to recognize the complement by swapping the accept and reject states.
Union
Union of two languages

If $L_1$ and $L_2$ are languages over the alphabet $\Sigma$, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.

If $L_1$ and $L_2$ are regular languages, is $L_1 \cup L_2$?
Show that if $A$ and $B$ are regular, then so is $A \cup B$.

**Proof by construction:**

Given the automata

$$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \text{ recognizing } A$$

$$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \text{ recognizing } B$$

construct $M = (Q, \Sigma, \delta, q_0, F)$ recognizing $A \cup B$.

For simplicity, let the alphabets be the same.

...
You might think: Run the string through $M_1$, see whether $M_1$ accepts it, then run the string through $M_2$ and see whether $M_2$ accepts it.
You might think: Run the string through $M_1$, see whether $M_1$ accepts it, then run the string through $M_2$ and see whether $M_2$ accepts it.

But you can’t try something on the whole input string, and try another thing on the whole input string; *you only get one pass!*
The new machine guesses non-deterministically which of the two machines accepts the input.
\[
L(N) = L_1 \cup L_2
\]
This construction proves the class of regular languages is closed under the union operation.
Intersection
The intersection of two languages

If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

If $L_1$ and $L_2$ are both regular, is $L_1 \cap L_2$ regular?
The intersection of two languages

If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

If $L_1$ and $L_2$ are both regular, is $L_1 \cap L_2$ regular?
Intersection of two languages

If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

If $L_1$ and $L_2$ are both regular, is $L_1 \cap L_2$ regular?
Intersection of two languages

If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

If $L_1$ and $L_2$ are both regular, is $L_1 \cap L_2$ regular?
Intersection of two languages

If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

If $L_1$ and $L_2$ are both regular, is $L_1 \cap L_2$ regular?

De Morgan’s laws!
Concatenation
Concatenation of strings

Recall: If \( w \in \Sigma^* \) and \( x \in \Sigma^* \), the concatenation of \( w \) and \( x \), denoted \( w \circ x \) or just \( wx \), is the string formed by tacking all characters in \( x \) onto the end of \( w \).

E.g., if \( w = \text{quo} \) and \( x = \text{kka} \), the concatenation \( wx = \text{quokka} \).
Concatenation of strings

*Recall*: If $w \in \Sigma^*$ and $x \in \Sigma^*$, the concatenation of $w$ and $x$, denoted $w \circ x$ or just $wx$, is the string formed by tacking all characters in $x$ onto the end of $w$.

E.g., if $w = \text{quo}$ and $x = \text{kka}$, the concatenation $wx = \text{quokka}$

A quokka, happy just to be mentioned
Concatenation of languages

The concatenation of two languages $L_1$ and $L_2$ over the alphabet $\Sigma$ is the language

$$L_1 L_2 = \{wx \mid w \in L_1 \land x \in L_2\}$$

E.g., consider the languages

$Noun = \{\text{Puppy, Rainbow, Whale, }\ldots\}$

$Verb = \{\text{Hugs, Juggles, Loves, }\ldots\}$

$Det = \{\text{A, The}\}$

The language $Det Noun Verb Det Noun$ is

$$\{APuppyHugsTheWhale, TheRainbowJugglesTheRainbow,$$
$$\text{TheWhaleLovesAPuppy, }\ldots\}$$
Concatenation of languages

Two views of $L_1L_2$:

- The set of all strings that can be made by concatenating a string in $L_1$ with a string in $L_2$.
- The set of strings that can be split into two pieces: a piece from $L_1$ followed by a piece from $L_2$.

Conceptually it’s similar to the Cartesian product of two sets, only with strings.
Concatenation of languages

If $L_1$ and $L_2$ are regular languages, is $L_1L_2$?

How could we know where the first string ends and the second begins?

There isn’t a straightforward way to do this with a DFA; our model makes it too hard to keep track of the possibilities.

With NFAs, it’s easy!
Given a string $w$, run a finite automaton for $L_1$ on $w$.

Whenever it reaches an accepting state, optionally hand the rest of $w$ to the finite automaton for $L_2$.

If the automaton for $L_2$ accepts the rest, $w \in L_1L_2$.

If the automaton for $L_2$ rejects the remainder, either $w \notin L_1L_2$ or the split was incorrect.
The new machine guesses non-deterministically where to split the input in order to have a first part accepted by $N_1$ and a second part accepted by $N_2$. 
\[ L(N) = L_1 L_2 \]
This construction proves the class of regular languages is closed under concatenation.
Acknowledgments

This lecture incorporates material from:

Nancy Ide
Keith Schwarz
Michael Sipser
Jeffrey Ullman