Problem Set Four is due in the box up front.
NFAs

• An **NFA** is a
  • **N**ondeterministic
  • **F**inite
  • **A**utomaton

• Can have zero or more transitions defined for each state/symbol pair.

• An NFA $N$ accepts a string $w$ iff there is some possible series of transitions $N$ can follow that ends in an accepting state.
ε-Transitions

- NFAs have a special type of transition called the \textit{ε-transition}.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
\(\varepsilon\)-Transitions

- NFAs have a special type of transition called the **\(\varepsilon\)-transition**.
- An NFA may follow any number of \(\varepsilon\)-transitions at any time without consuming any input.

```
0 0 1 0 0
```
\textbf{ε-Transitions}

- NFAs have a special type of transition called the \textit{ε-transition}.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

• NFAs have a special type of transition called the **ε-transition**.

• An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the \textbf{ε-transition}.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.

- An NFA may follow any number of ε-transitions at any time without consuming any input.

```
0 0 1 0 0
```
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the *ε-transition*.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.

- An NFA may follow any number of ε-transitions at any time without consuming any input.
• NFAs have a special type of transition called the \textbf{ε-transition}.

• An NFA may follow any number of \textit{ε}-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
\(\varepsilon\)-Transitions

- NFAs have a special type of transition called the **\(\varepsilon\)-transition**.
- An NFA may follow any number of \(\varepsilon\)-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
\( \varepsilon \)-Transitions

- NFAs have a special type of transition called the \textbf{\( \varepsilon \)-transition}.
- An NFA may follow any number of \( \varepsilon \)-transitions at any time without consuming any input.
Designing NFAs

• When designing NFAs, *embrace the nondeterminism!*

• Good model: **Guess-and-check**:
  
  • Have the machine *nondeterministically guess* what the right choice is.
  
  • Have the machine *deterministically check* that the choice was correct.
Guess-and-Check

\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$
New Stuff!
NFAs and DFAs

- Any language that can be accepted by a DFA can be accepted by an NFA.
- Why?
  - Just use the same set of transitions as before.
- **Question:** Can any language accepted by an NFA also be accepted by a DFA?
- Surprisingly, the answer is **yes!**
Finite Automata

• NFAs and DFAs are finite automata; there can only be finitely many states in an NFA or DFA.

• An NFA can be in any combination of its states, but there are only finitely many possible combinations.

• **Idea:** Build a DFA where each state of the DFA corresponds to a set of states in the NFA.
Simulating an NFA with a DFA

\[ \begin{align*}
q_0 & \xrightarrow{0} q_1 \\
q_1 & \xrightarrow{1} q_2 \\
q_3 & \xrightarrow{\varepsilon} q_4 \\
q_4 & \xrightarrow{0} q_5
\end{align*} \]
Simulating an NFA with a DFA
Simulating an NFA with a DFA

- Start state: $q_0$
- Transitions:
  - $q_0 \xrightarrow{0} q_1$
  - $q_0 \xrightarrow{\varepsilon} q_3$
  - $q_1 \xrightarrow{1} q_2$
  - $q_3 \xrightarrow{0} q_4$
  - $q_4 \xrightarrow{0} q_5$
  - $q_5 \xrightarrow{1} q_5$
Simulating an NFA with a DFA
Simulating an NFA with a DFA

Start

$q_0$ 0 $q_1$ 1 $q_2$

$\varepsilon$

$q_3$ 0 $q_4$ 0 $q_5$

1

$q_{14}$ 0

Start

$q_{03}$

$q_1$

$q_2$

$q_4$

$q_5$
Simulating an NFA with a DFA

- Start state: $q_0$
- Transitions:
  - $q_0 \xrightarrow{0} q_1$
  - $q_0 \xrightarrow{\varepsilon} q_3$
  - $q_1 \xrightarrow{1} q_2$
  - $q_3 \xrightarrow{0} q_4$
  - $q_4 \xrightarrow{0} q_5$
  - $q_2 \xrightarrow{0} q_2$
  - $q_2 \xrightarrow{1} q_2$

- DFA state transition:
  - $q_{14} \xrightarrow{0} q_{03}$
  - $q_{03} \xrightarrow{1} q$

- Final state: $q$
Simulating an NFA with a DFA
Simulating an NFA with a DFA

- **Start State:** $q_0$
- **Final State:** $q_2$
- **Transition:**
  - $q_0 \xrightarrow{0} q_1$
  - $q_0 \xrightarrow{\varepsilon} q_3$
  - $q_3 \xrightarrow{1} q_5$
  - $q_1 \xrightarrow{1} q_2$
  - $q_1 \xrightarrow{0} q_4$
  - $q_4 \xrightarrow{0} q_5$

- **DFA Representation:**
  - **Start State:** $q_{03}$
  - **Final State:** $q_5$
  - **Transition:**
    - $q_{03} \xrightarrow{0} q_{14}$
    - $q_{03} \xrightarrow{1} q$
    - $q \xrightarrow{0} q$
    - $q \xrightarrow{1} q_5$
Simulating an NFA with a DFA
Simulating an NFA with a DFA

- Start state: \( q_0 \)
- \( q_3 \) transitions on \( \varepsilon \) to \( q_4 \)
- \( q_4 \) transitions on 0 to \( q_5 \)
- \( q_5 \) is a final state
- \( q_2 \) transitions on 0 to \( q_1 \)
- \( q_1 \) transitions on 1 to \( q_2 \)
- \( q_0 \) transitions on 1 to \( q_2 \)

- Start state: \( q_{03} \)
- \( q_{14} \) transitions on 0 to \( q_2 \)
- \( q_2 \) transitions on 1 to \( q_{14} \)
- \( q_{14} \) transitions on 0 to \( q_0 \)
- \( q_0 \) transitions on 1 to \( q_{14} \)
- \( q_{14} \) is a final state

- Final states: \( q_2 \) and \( q_5 \)
Simulating an NFA with a DFA
Simulating an NFA with a DFA

Left: NFA
- Start state: $q_0$
- Transitions:
  - $q_0 \xrightarrow{0} q_1$
  - $q_0 \xrightarrow{\varepsilon} q_3$
  - $q_3 \xrightarrow{0} q_4$
  - $q_4 \xrightarrow{0} q_5$
  - $q_1 \xrightarrow{1} q_2$
  - $q_1 \xrightarrow{0} q_0$

Right: DFA
- Start state: $q_{03}$
- Transitions:
  - $q_{03} \xrightarrow{0} q_{14}$
  - $q_{03} \xrightarrow{1} q$
  - $q_{14} \xrightarrow{1} q_2$
  - $q_2 \xrightarrow{0} q_{14}$
  - $q_2 \xrightarrow{1} q$
  - $q \xrightarrow{0} q_0$
  - $q \xrightarrow{1} q_1$
Simulating an NFA with a DFA

start $\xrightarrow[]{\varepsilon} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{1} q_4 \xrightarrow{0} q_5$

$q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_3$

$q_{03} \xrightarrow{0} q_{14} \xrightarrow{1} q_2 \xrightarrow{0} q_1 \xrightarrow{1} q_5$
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA

start
$q_0$ 0 $q_1$ 1 $q_2$
$q_3$ $\varepsilon$ 0 $q_4$ 0 $q_5$

$q_{14}$ 0 1 $q$
$q_2$ 0 $q_1$
$q_03$ 1 $q_5$

$q_3$
$q_5$ 1
Simulating an NFA with a DFA

Start state $q_0$ transitions to $q_1$ on input 0 and to $q_4$ on input 1. $q_1$ transitions to $q_2$ on input 0.

Start state $q_{14}$ transitions to $q_2$ on input 0 and 1. $q_2$ transitions to $q$ on input 0 and 1. $q$ is the only accepting state.

$\varepsilon$ transitions from $q_0$ to $q_3$, and $q_3$ transitions to $q_4$ on input 0 and to $q_5$ on input 1.
Simulating an NFA with a DFA

\[
\begin{align*}
&\text{start} \quad q_0 \quad 0 \rightarrow q_1 \quad 1 \\
&q_3 \quad 0 \rightarrow q_4 \quad 0 \rightarrow q_5 \\
&\varepsilon \quad 1 \\
&\text{q_1} \quad 0 \rightarrow \text{q_2} \\
\end{align*}
\]

\[
\begin{align*}
&\text{start} \quad q_{03} \quad 0 \rightarrow q \quad 1 \\
&q_{14} \quad 0 \rightarrow q_2 \quad 1 \rightarrow q_1 \\
&q_2 \quad 0 \rightarrow q_0 \quad 0 \rightarrow q_5 \\
&q_{14} \quad 0 \rightarrow q_3 \\
&\text{q_1} \quad 0 \rightarrow \text{q_2} \quad 1 \rightarrow \text{q_5} \\
&\text{start} \quad q_0 \quad 0 \rightarrow q_3 \quad 1 \\
\end{align*}
\]
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA

Drawings of two automata: one on the left for the NFA and one on the right for the DFA.
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA
Simulating an NFA with a DFA

0 0 1 0 0
Simulating an NFA with a DFA

---

0 0 1 0 0
Simulating an NFA with a DFA

0 0 1 0 0
Simulating an NFA with a DFA

\[ \begin{array}{c}
q_0 \\
\text{start}
\end{array} \]

\[ \begin{array}{c}
q_3 \\
\varepsilon
\end{array} \]

\[ \begin{array}{c}
q_1 \\
0 \\
1
\end{array} \]

\[ \begin{array}{c}
q_2 \\
0 \\
1
\end{array} \]

\[ \begin{array}{c}
q_4 \\
0 \\
1
\end{array} \]

\[ \begin{array}{c}
q_5 \\
0 \\
1
\end{array} \]

\[ \begin{array}{c}
q_0 \rightarrow 0, 1 \rightarrow q_3
\end{array} \]
Simulating an NFA with a DFA

0 0 1 0 0
Simulating an NFA with a DFA

Left:

- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3, q_4, q_5$
- Transitions:
  - $q_0 \xrightarrow{0} q_1$
  - $q_0 \xrightarrow{\varepsilon} q_3$
  - $q_1 \xrightarrow{0} q_2$
  - $q_1 \xrightarrow{1} q_3$
  - $q_2 \xrightarrow{0} q_4$
  - $q_2 \xrightarrow{1} q_5$
  - $q_3 \xrightarrow{0} q_4$
  - $q_3 \xrightarrow{1} q_5$
  - $q_4 \xrightarrow{0, 1} q_5$

Right:

- Start state: $q_{03}$
- States: $q_0, q_1, q_2, q_3, q_4, q_5$
- Transitions:
  - $q_{03} \xrightarrow{0} q_0$
  - $q_{03} \xrightarrow{1} q_1$
  - $q_0 \xrightarrow{0, 1} q_1$
  - $q_0 \xrightarrow{1} q_2$
  - $q_1 \xrightarrow{0} q_3$
  - $q_1 \xrightarrow{1} q_4$
  - $q_2 \xrightarrow{1} q_5$
  - $q_2 \xrightarrow{0} q_3$
  - $q_2 \xrightarrow{1} q_4$
  - $q_3 \xrightarrow{0} q_5$
  - $q_3 \xrightarrow{1} q_4$
  - $q_4 \xrightarrow{0} q_5$
  - $q_4 \xrightarrow{1} q_3$

Input sequence: $00100$
Simulating an NFA with a DFA

0 0 1 0 0
Simulating an NFA with a DFA

Start state: $q_0$

Transitions:
- $q_0 \xrightarrow{0} q_1$
- $q_0 \xrightarrow{1} q_2$
- $q_3 \xrightarrow{0} q_4$
- $q_3 \xrightarrow{1} q_4$

Input string: $00100$

DFA:
- Start state: $q_{03}$
- Transitions:
  - $q_{03} \xrightarrow{0} q_1$
  - $q_{03} \xrightarrow{1} q_2$
  - $q_{14} \xrightarrow{0} q_2$
  - $q_{14} \xrightarrow{1} q_1$
  - $q_2 \xrightarrow{0,1} q_1$
  - $q_2 \xrightarrow{1} q_1$
  - $q_2 \xrightarrow{0,1} q_1$
  - $q_4 \xrightarrow{0} q_5$
  - $q_4 \xrightarrow{0} q_5$
  - $q_4 \xrightarrow{1} q_5$
  - $q_5 \xrightarrow{0,1} q_3$
  - $q_5 \xrightarrow{1} q_3$

Final states: $q_1$, $q_2$, $q_3$, $q_4$, $q_5$
Simulating an NFA with a DFA

[Diagram of an NFA and a DFA]

Input: 0 0 1 0 0

- Transition from $q_0$ to $q_1$ on 0
- Transition from $q_1$ to $q_2$ on 1
- Transition from $q_3$ to $q_4$ on $\varepsilon$
- Transition from $q_4$ to $q_5$ on 0

- Transition from $q_{14}$ to $q_2$ on 0
- Transition from $q_{14}$ to $q_1$ on 1
- Transition from $q_2$ to $q_0$ on 1
- Transition from $q_2$ to $q_1$ on 0
- Transition from $q_2$ to $q_4$ on 0, 1
- Transition from $q_4$ to $q_3$ on 0
- Transition from $q_4$ to $q_5$ on 1
- Transition from $q_3$ to $q_5$ on 0

Start state: $q_0$ and $q_{14}$
Accepting state: $q_5$
Simulating an NFA with a DFA
Simulating an NFA with a DFA

0 0 1 0 0
Simulating an NFA with a DFA

0 0 1 0 0
Simulating an NFA with a DFA

- **Left Diagram:**
  - **States:** $q_0, q_1, q_2, q_3, q_4, q_5$
  - **Transition:**
    - $q_0 \xrightarrow{0} q_1$, $q_0 \xrightarrow{1} q_2$
    - $q_3 \xrightarrow{0} q_4$, $q_3 \xrightarrow{1} q_5$
    - $q_5 \xrightarrow{\varepsilon} q_5$
  - **Start State:** $q_0$
  - **Accepting State:** $q_5$

- **Right Diagram:**
  - **States:** $q_0_3, q_1, q_2, q_4, q_5$
  - **Transition:**
    - $q_0_3 \xrightarrow{0} q_1$, $q_0_3 \xrightarrow{1} q_2$
    - $q_2 \xrightarrow{0} q_1$, $q_2 \xrightarrow{1} q_4$
    - $q_4 \xrightarrow{0} q_3$, $q_4 \xrightarrow{1} q_5$
  - **Start State:** $q_0_3$
  - **Accepting State:** $q_5$

- **Input:** 0 0 1 0 0
The Subset Construction

- This construction for transforming an NFA into a DFA is called the **subset construction** (or sometimes the **powerset construction**).
  - States of the new DFA correspond to *sets of states* of the NFA.
  - The initial state is the start state, plus all states reachable from the start state via ε-transitions.
  - Transition on state \( S \) on character \( a \) is found by following all possible transitions on \( a \) for each state in \( S \), then taking the set of states reachable from there by ε-transitions.
  - Accepting states are any set of states where *some* state in the set is an accepting state.

- **Read Sipser for a formal account.**
The Subset Construction

- In converting an NFA to a DFA, the DFA's states correspond to sets of NFA states.
- Fact: $|\mathcal{P}(S)| = 2^{|S|}$ for any finite set $S$.
- In the worst-case, the construction can result in a DFA that is exponentially larger than the original NFA.
- Interesting challenge: Find a language for which this worst-case behavior occurs (there are infinitely many of them!)
A language $L$ is called a **regular language** iff there exists a DFA $D$ such that $\mathcal{L}(D) = L$. 
An Important Result

*Theorem:* A language $L$ is regular iff there is some NFA $N$ such that $\mathcal{L}(N) = L$. 

*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. ■
An Important Result

*Theorem:* A language $L$ is regular iff there is some NFA $N$ such that $\mathcal{L}(N) = L$.

*Proof Sketch:*
An Important Result

**Theorem:** A language $L$ is regular iff there is some NFA $N$ such that $\mathcal{L}(N) = L$.

**Proof Sketch:** If $L$ is regular, there exists some DFA for it, which we can easily convert into an NFA.
An Important Result

Theorem: A language $L$ is regular iff there is some NFA $N$ such that $\mathcal{L}(N) = L$.

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.
An Important Result

*Theorem:* A language $L$ is regular iff there is some NFA $N$ such that $\mathcal{L}(N) = L$.

*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. ■
Announcements!
Set Five Out

Set Five goes out today. This problem set focuses on finite automata, regular languages, and their properties, and will be your first foray into automata theory. We hope that you enjoy it!

Handouts

00: Course Information
01: Syllabus
02: Problem Set Policies
03: Honor Code
04: Set Theory Definitions
07: Guide to Proofs
14: Practice Midterm 1
14S: Practice Midterm 1 Solns
15: Practice Midterm 2

Resources

Course Reader
Lecture Videos
Theorem and Definition Reference
Office Hours Schedule
Grades
DFA/NFA Designer
Regex Designer
Problem Set Five

● Problem Set Five released, due on **Monday, November 4**.

● Note the due date is *Monday* rather than *Friday*.

● Late periods now carry over to *Wednesday* rather than *Monday*.

● No checkpoint problem.

● Explore finite automata, regular languages, and their properties!
Midterm Logistics

• Midterm is next Tuesday, October 29 from 7PM – 10PM.
• Covers material up through and including DFAs.
• Review handout on exam policies and procedures for open-note and limited-computer policies.

**Alternate exams:** Contact Keith ASAP if you haven't heard back about alternate exams.

**Review session:** 2:15PM – 4:15PM on Saturday in room 370-370.

• Have questions for the review session: ask them on Google Moderator!
Your Questions
“I am having trouble being confident in my first order logic translations. Are there ways to self check the translation? Also, is it possible to release some more English-to-first-order-logic translation problems as practice?”
“Diagonalization is a really cool and powerful proof technique, but are there other ways to show that infinite sets have different cardinalities? What happens if the problem does not easily lend itself to diagonal arguments?”
Back to Automata...
Why This Matters

● We now have two perspectives on regular languages:
  ● Regular languages are languages accepted by DFAs.
  ● Regular languages are languages accepted by NFAs.
● We can now reason about the regular languages in two different ways.
The 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 $L_1 \cup L_2$?
The 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 $L_1 \cup L_2$?
The 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 $L_1 \cup L_2$ regular?
The 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 $L_1 \cup L_2$?
The 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 $L_1 \cup L_2$ 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 $L_2$.
- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
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 $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
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 $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
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 $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
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 $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
Concatenation

- The **concatenation** of two languages $L_1$ and $L_2$ over the alphabet $\Sigma$ is the language
  
  $L_1L_2 = \{ \text{wx} \in \Sigma^* \mid \text{w} \in L_1 \land \text{x} \in L_2 \}$

- The set of strings that can be split into two pieces: a piece from $L_1$ and a piece from $L_2$.

- Conceptually similar to the Cartesian product of two sets, only with strings.
Concatenation Example

Let $\Sigma = \{ a, b, ..., z, A, B, ..., Z \}$ and consider these languages over $\Sigma$:

- **Noun** = \{ Puppy, Rainbow, Whale, ... \}
- **Verb** = \{ Hugs, Juggles, Loves, ... \}
- **The** = \{ The \}

The language $\text{TheNounVerbTheNoun}$ is

\{ ThePuppyHugsTheWhale, TheWhaleLovesTheRainbow, TheRainbowJugglesTheRainbow, ... \}
If $L_1$ and $L_2$ are regular languages, is $L_1L_2$?

Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?
Concatenating Regular Languages

- If $L_1$ and $L_2$ are regular languages, is $L_1L_2$?
- Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

Machine for $L_1$  

Machine for $L_2$
Concatenating Regular Languages

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

- Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

Machine for $L_1$

Machine for $L_2$

`bookkeeper`
Concatenating Regular Languages

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

• Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

Machine for $L_1$

Machine for $L_2$
Concatenating Regular Languages

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

- Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

Machine for $L_1$

Machine for $L_2$
Concatenating Regular Languages

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

● Intuition – can we split a string $w$ into two strings $xy$ such that $x \in L_1$ and $y \in L_2$?

● **Idea:** Run the automaton for $L_1$ on $w$, and whenever $L_1$ reaches an accepting state, optionally hand the rest off $w$ to $L_2$.
   - If $L_2$ accepts the remainder, then $L_1$ accepted the first part and the string is in $L_1L_2$.
   - If $L_2$ rejects the remainder, then the split was incorrect.
Concatenating Regular Languages
Concatenating Regular Languages

Machine for $L_1$
Concatenating Regular Languages

Machine for $L_1$

Machine for $L_2$
Concatenating Regular Languages

Machine for \( L_1 \)

\begin{align*}
\varepsilon \\
\varepsilon \\
\varepsilon
\end{align*}

Machine for \( L_2 \)
Concatenating Regular Languages

Machine for $L_1$

Machine for $L_2$
Concatenating Regular Languages

Machine for $L_1$

Machine for $L_1L_2$

Machine for $L_2$
Lots and Lots of Concatenation

- Consider the language $L = \{ \text{aa, b} \}$
- $LL$ is the set of strings formed by concatenating pairs of strings in $L$.
  $$\{ \text{aaaa, aab, baa, bb} \}$$
- $LLL$ is the set of strings formed by concatenating triples of strings in $L$.
  $$\{ \text{aaaaaa, aaaaab, aabaa, aabb, baaaa, baab, bbaa, bbb} \}$$
- $LLLL$ is the set of strings formed by concatenating quadruples of strings in $L$.
  $$\{ \text{aaaaaaaa, aaaaaaab, aaaaabaa, aaaaabb, aabaaaa, aabaab, aabbaa, aabbb, baaaaaa, baaaaab, baabaa, baabb, bbaaaa, bbaaab, bbbaa, bbbb} \}$$
We can define what it means to “exponentiate” a language as follows:

$L^0 = \{ \varepsilon \}$

- The set containing just the empty string.
- Idea: Any string formed by concatenating zero strings together is the empty string.

$L^{n+1} = LL^n$

- Idea: Concatenating $(n+1)$ strings together works by concatenating $n$ strings, then concatenating one more.
The Kleene Closure

• An important operation on languages is the **Kleene Closure**, which is defined as

\[ L^* = \bigcup_{i=0}^{\infty} L^i \]

• Mathematically:

\[ w \in L^* \iff \exists n \in \mathbb{N}. w \in L^n \]

• Intuitively, all possible ways of concatenating any number of copies of strings in \( L \) together.
The Kleene Closure

If $L = \{ \text{a, bb} \}$, then $L^* = \{ \epsilon, \text{a, bb, }$

$$\text{aa, aabb, abba, abbbb, bbba, bbabb, bbbba, bbbbbbb, }$$

$$\text{...} \}$$
Reasoning about Infinity

• If $L$ is regular, is $L^*$ necessarily regular?

**A Bad Line of Reasoning:**

• $L^0 = \{ \varepsilon \}$ is regular.
• $L^1 = L$ is regular.
• $L^2 = LL$ is regular
• $L^3 = L(LL)$ is regular
• ...

• Regular languages are closed under union.
• So the union of all these languages is regular.
Reasoning about Infinity
Reasoning about Infinity
Reasoning about Infinity
Reasoning about Infinity
Reasoning about Infinity
Reasoning about Infinity

\[ x \]
Reasoning about Infinity
Reasoning About the Infinite

• If a series of finite objects all have some property, their infinite union does not necessarily have that property!
  • No matter how many times we zigzag that line, it's never straight.
  • Concluding that it must be equal “in the limit” is not mathematically valid (nor is it correct!)
    • (This is why calculus is interesting).

**Better idea:** Can we convert an NFA for the language $L$ to an NFA for the language $L^*$?
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$
The Kleene Star

Machine for $L$

Machine for $L^*$
The Kleene Star

Question: Why add the new state out front? Why not just make the old start state accepting?
Summary

- NFAs are a powerful type of automaton that allows for **nondeterministic** choices.
- NFAs can also have **ε-transitions** that move from state to state without consuming any input.
- The **subset construction** shows that NFAs are not more powerful than DFAs, because any NFA can be converted into a DFA that accepts the same language.
- The union, intersection, complement, concatenation, and Kleene closure of regular languages are all regular languages.
Next Time

• **Regular Expressions**
  • Building up the regular languages, one piece at a time.

• **Intuiting Regular Languages**
  • What exactly is a regular language?
  • When would you use them?