Objectives

- Demonstrate the properties of regular languages.
- Demonstrate the correspondence between a Deterministic Finite Automata and a Right Linear Grammar.
- Identify languages that can be recognized by regular languages.
- Build an automata for a given regular language.

State Machines

Consider the following state machine:

![State Machine Diagram]

**Problem 1)** Trace the following strings as inputs to the above state machine. Which strings are part of the language recognized by the state machine?

<table>
<thead>
<tr>
<th>Input</th>
<th>Recognized?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110</td>
<td>Y/N</td>
</tr>
<tr>
<td>00011</td>
<td>Y/N</td>
</tr>
<tr>
<td>0101</td>
<td>Y/N</td>
</tr>
<tr>
<td>11100010</td>
<td>Y/N</td>
</tr>
</tbody>
</table>

**Problem 2)** In English\(^1\), describe the language this automata accepts.

**Problem 3)** Is the language finite or infinite?

**Problem 4)** Is the amount of computation required to recognize or reject a string finite or infinite?

\(^1\)or what ever language you prefer
Correspondence to Right Linear Grammars

Here is that automata again, with an equivalent grammar.

\[
\begin{align*}
S_0 & \rightarrow 1S_10S_2 \\
S_1 & \rightarrow 1S_00S_3 \\
S_2 & \rightarrow 1S_30S_0 \\
S_3 & \rightarrow 1S_20S_1\epsilon
\end{align*}
\]

**Problem 5)** What do the \( S_n \) represent?

**Problem 6)** How is a transition modeled in the grammar?

**Problem 7)** The grammar is right linear, because there is at most one non-terminal symbol on the right hand side of any production. Suppose we added a rule like this one: \( S_0 \rightarrow 1S_10S_2 \). Could you still come up with a deterministic finite automata that matches the new grammar? Why or why not?
Categorization

**Problem 8)** Describe in English the following regular expressions

- `[a-zA-Z][a-zA-Z0-9]+`
- `[a-z]*([es|ed|ing])`
- `<[a-z0-9]+@[a-z0-9]+(\.[a-z0-9]+)+>`

**Problem 9)** Which of the following can be described by regular expressions?

- All the words in the English language
- All the Fibonacci numbers
- "All Your Base Are Belong To Us" video
- Numbers that are multiples of 4 (assume $\geq 2$ digits)
- Words that have exactly as many $a$ as they have $b$s
- Palindromes

**Demo: Using grep and sed**
Building an Automata

We can build an automata that recognizes integers that are multiples of 7!

**Problem 10)** To get started, fill out this table. The first two rows are done for you.

<table>
<thead>
<tr>
<th>$n \mod 7$</th>
<th>$10n \mod 7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

**Problem 11)** Now build your automata. If you are not sure how to get started, then ask yourself “how many states will I need?” and “what does a transition indicate?”.