Using *Introduction to the Theory of Computation*, Chapter 7
Outline

regular expressions

product, non-deterministic FSAs

regular languages

notes
another way to define languages
In addition to the language accepted by DFSA $L(M)$ and set description $L = \{ \ldots \}$.

Definition: The regular expressions (regexps or REs) over alphabet $\Sigma$ is the smallest set such that:

1. $\emptyset$, $\varepsilon$, and $a$, for every $a \in \Sigma$ are REs over $\Sigma$
2. if $T$ and $S$ are REs over $\Sigma$, then so are:
   - $T + S$ (union) — lowest precedence operator
   - $TS$ (concatenation) — middle precedence operator
   - $T^*$ (star) — highest precedence
regular expression to language:

The \( L(R) \), the language denoted (or described) by \( R \) is defined by structural induction:

**Basis:** If \( R \) is a regular expression by the basis of the definition of regular expressions, then define \( L(R) \):

- \( L(\emptyset) = \emptyset \) (the empty language)
- \( L(\varepsilon) = \{\varepsilon\} \) (the language consisting of just the empty string)
- \( L(a) = \{a\} \) (the language consisting of the one-symbol string)

**Induction step:** If \( R \) is a regular expression by the induction step of the definition, then define \( L(R) \):

- \( L(S + T) = L(S) \cup L(T) \)
- \( L(ST) = L(S)L(T) \)
- \( L(T^*) = L(T)^* \)
regexp examples

- \((L(0 + 1))^*\) = \{0, 1\}

- \(L((0 + 1)^*)\) All binary strings over \{0, 1\}

- \(L((01)^*) = \{\epsilon, 01, 0101, 010101, \ldots\}\)

- \(L(0^*1^*)\) 0 or more 0s followed by 0 or more 1s.

- \(L(0^* + 1^*)\) 0 or more 0s or 0 or more 1s.

- \(L((0 + 1)(0 + 1)^*)\) Non-empty binary strings over \{0, 1\}. 
example

$L = \{ x \in \{0,1\}^* \mid x \text{ begins and ends with a different bit} \}$

\[ R = \left( \frac{0(0+1)^*1}{1(1+0)^*0} \right) + 1(1+0)^*0 \]

prove \[ L = L(R) \]

\[ L \subseteq L(R) \land L(R) \subseteq L \]

Proof (not really): to show \( L(R) \subseteq L \)

assume \( s \) is an arbitrary element of \( L(R) \).

Then, \( \text{WLOG} \) (switch 0s and 1s otherwise)

assume \( s \in L(0(0+1)^*1) \). Then, \( s \) has the form \( tuv \), where \( t \in L(0) \), \( u \in L((0+1)^* \) \)

\( v \in L(1) \)

without loss of generality.
RE identities

some of these follow from set properties...
others require some proof (see 7.2.5 example)

\[ L(R) \cup L(S) \equiv L(S) + L(R) \]

- communitativity of union: \( R + S \equiv S + R \)
- associativity of union: \((R + S) + T \equiv R + (S + T)\)
- associativity of concatenation: \((RS)T \equiv R(ST)\)
- left distributivity: \(R(S + T) \equiv RS + RT\)
- right distributivity: \((S + T)R \equiv SR + TR\)
- identity for union: \(R + \emptyset \equiv R\)
- identity for concatenation: \(R\epsilon \equiv R \equiv \epsilon R\)
- annihilator for concatenation: \(\emptyset R \equiv \emptyset \equiv R\emptyset\)
- idempotence of Kleene star: \((R^\ast)^\ast \equiv R^\ast\)

\[ \mathbb{E}( (R^\ast)^\ast )^\ast \equiv (R^\ast)^\ast \equiv R^\ast \]
product construction

$L$ is the language of binary strings over $\{0, 1\}^*$ with two 1s in a row and an even number of 0s.

idea: $\delta((q_i, q_j), a) = (\delta(q_i, a), \delta(q_j, a))$
non-deterministic FSA (NFSA) example

FSA that accepts $L((010 + 01)^*)$

Construct a corresponding DFSA

Transition function is between sets of accepting states and if string drives our machine to a set that includes an accepting state, said string is accepted.

\[ M = \{ Q, \delta, q_0, F, \Sigma \} \]

\[ \hat{M} = \{ Q, \delta', q_0, F, \Sigma \} \]
they’re equivalent:

\[ L = L(M) \text{ for some DFSA } M \iff L = L(M') \text{ for some NFSA } M' \iff L = L(R(R)) \text{ for some regular expression } R \]