Closure of Regular (HALF)
Half(L) = \{ x \mid \exists y, |x| = |y| \text{ and } xy \in L \} \\

- First thought might be that we can do this by Regular Expressions \\
- A little thought to that makes it clear that there is no way to figure out the lengths of unbounded parts being split up in a regular expression \\
- In fact, we cannot even directly use a Regular Expression for Prefix \\
- Ah, then why not a variant of our technique we applied for Quotient, Prefix, Suffix? \\
- But, once again we have no way to express constrained unbounded lengths \\
- We seem to need to go back to a function-based approach -- FSAs
\[ \text{Half}(L) = \{ x \mid \exists y, |x| = |y| \text{ and } xy \in L \} \]

- Let \( L \) be a Regular language over the finite alphabet \( \Sigma \). \( L \) is recognized by some DFA.
- Let that DFA be \( A_1 = (Q, \Sigma, \delta_1, q_1, F) \).
- Clearly, when this DFA gets half the way through any string it is in some fixed state since the DFA is deterministic.
- Let’s call the state at the halfway point, \( h \).
- Then \( \delta_1(q_1, x) = h \)
- Since \( \exists y, |x| = |y| \) where \( xy \in L \), then for any such \( y \), \( \delta_1(h, y) \in F \)
- We want a parallel approach that looks at \( x \) and \( y \) simultaneously
Half(L) = \{ x \mid \exists y, |x| = |y| \text{ and } xy \in L \} \\

• Our parallel algorithm will use non-determinism since we have no idea what y is, but we know its length is the same as x.
• We will also be non-deterministic as we will guess the value of h.
• The automaton we need to have a three-part state, <q, p, h>. q is the state that A_1 deterministically computes while reading x. p is an educated guess of where A_1 would be if it was reading y and had traversed as many characters of y as it has of x. h is the guessed state we would enter at the halfway point.
• We would like to start at any state <q_1, h, h>, where h \in Q but we are only allowed one start state, so we invent a new one q_0 and have it go to all these desired start states on \lambda.
\[
\text{Half}(L) = \{ x \mid \exists y, |x| = |y| \text{ and } xy \in L \}
\]

- Formally we create the NDA below.

- \( A_2 = (\langle Q \times 2^q \times 2^q \rangle \cup \{q_0\}, \Sigma, \delta_2, q_0, F') \), where
  \[
  \delta_2(q_0, \lambda) = \text{union}(h \in Q) \{<q_1, h, h>\} \text{ and }
  \]
  \[
  \delta_2(<q, r, h>, a) = \text{union}(b \in \Sigma) \{<\delta_1(q, a), \delta_1(r, b), h>\}, q, r, h \in Q
  \]
  \[
  F' = \text{union}(q \in \Sigma) \{<h, f, h>\}, f \in F
  \]

- Why this works:
  The first part of a state \(<q, r, h>\) tracks \(A_1\).
  The second part of a state \(<q, r, h>\) tracks \(A_1\) for precisely all possible strings that are the same length as what \(A_1\) is reading in parallel. This component starts with a guess as to what state \(A_1\) might end up in.
  The third part of a state \(<q, r, h>\) remembers the initial guess, \(h\).
  Thus, \(\delta_2(<q_1, h, h>, x) = \{\delta_1(q_0, x), \delta_1(q, y), h>\}\) for arbitrary \(y\), \(|x| = |y|\)

- If our guess \(h\) is right, then \(\delta_1(q_0, x) = h\) and so the first and third components are the same.

- We accept if the initial guess was right and the second component is final, meaning \(xy\) is in \(L\).