A generalized non-deterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. $Q$ is a finite set of states
2. $\Sigma$ is a finite alphabet
3. $\delta : (Q - \{q_{\text{accept}}\} \times (Q - \{q_{\text{start}}\}) \rightarrow R$ is the transition function
4. $q_{\text{start}} \in Q$ is the start state
5. $q_{\text{accept}} \in Q$ is the accept state and $q_{\text{start}} \neq q_{\text{accept}}$

A GNFA accepts a string $w \in \Sigma^*$ if there exists a sequence of states $q_0, q_1, \ldots, q_k$ where each $w_i \in \Sigma^*$ and a sequence of states $q_0, q_1, \ldots, q_k$ exists such that

1. $q_0 = q_{\text{start}}$
2. $q_k = q_{\text{accept}}$
3. For each $i$, we have $w_i \in L(R_i)$ where

$$R_i = \delta(q_{i-1}, q_i)$$
Let \( k \) be the number of states of \( G \).

- If \( k = 2 \), \( G \) must consist of a start state, an accept state, and a single arrow connecting them with a regular expression \( R \). Return \( R \).

- If \( k > 2 \), select any state \( q_{\text{rip}} \) different from \( q_{\text{start}} \) and \( q_{\text{accept}} \). Let \( G' \) be the GNFA \((Q', \Sigma, \delta', q_{\text{start}}, q_{\text{accept}})\). + \( Q' = Q - \{q_{\text{rip}}\} \) and for any \( q_i \in Q' - \{q_{\text{accept}}\} \) and \( q_j \in Q' - \{q_{\text{start}}\} \), let \( \delta'(q_i, q_j) = (R_1)(R_2)(R_3) \cup (R_4) \), where \( R_1 = \delta(q_i, q_{\text{rip}}) \), \( R_2 = \delta(q_{\text{rip}}, q_{\text{rip}}) \), \( R_3 = \delta(q_{\text{rip}}, q_j) \), and \( R_4 = \delta(q_i, q_j) \). Compute \( \text{CONVERT}(G') \) and return this value.

Proof:

- Let \( A \) be the DFA for language \( A \).
- Convert \( A \) to a GNFA \( G \) by adding a new start state and a single arrow connecting it to any state different from \( q_{\text{start}} \) and \( q_{\text{accept}} \).
- If \( k = 2 \), \( G \) consists of a start state, an accept state, and a single arrow connecting them with a regular expression \( R \). Let \( k \) be the number of states of \( G \).
• Do $G$ and $G'$ accept the same language?

- Suppose $G$ accepts an input $w$.
  - There is an accepting sequence of states: $q_{\text{start}}$, $q_1$, $q_2$, $q_3$, ..., $q_{\text{accept}}$.
  - If none of them is $q_{\text{rip}}$, then $G'$ will accept $w$.
  - Each of the new regular expression of $G'$ contains the old regular expression as part of a union.
  - If none of them is $q_{\text{rip}}$, then $G'$ will accept $w$.
  - This will be an accepting computation for $G'$.

+ This will be an accepting computation for $G'$.

+ Each of the new regular expression of $G'$ contains the old regular expression.

- If none of them is $q_{\text{rip}}$, then $G'$ will accept $w$.

• If $q_{\text{rip}}$ does appear, remove each run of consecutive $q_{\text{rip}}$ states.

+ Each run of consecutive $q_{\text{rip}}$ states has a new regular expression on the arrow between them.

• Support $G$ accepts an input $w$.

- Suppose $G$ accepts an input $w$.

- Conclude, say $G'$ accepts the same language.

---

### Proof continued

#### Claim 1.65: For any GNFA $G$, CONVERT($G$) is equivalent to $G$.

- Induction on number of states $k$ in GNFA.

- Base case $k = 2$:
  - Let $G$ be the GNFA that converts from $G$. So, $G$ has $k = 1$.
  - Assume true for $k - 1$ states. CONVERT works properly for all GNFA that have $k - 1$ states or less.

- Case $k > 2$:
  - Let $G$ be the GNFA that converts from $G$. So, $G$ has $k - 1$.
  - Assume true for $k - 2$ states.
  - Let $G$ be the GNFA that converts from $G$. So, $G$ has $k - 1$.
  - Assume true for $k - 3$ states.
  - Convert works properly for all GNFA that have $k - 3$ states or less.

- suppose $G$ accepts a single arrow from state $s$ to accept state $t$.

- Induction on number of states $k$ in GNFA.

- Base case $k = 2$:
  - Let $G$ be the GNFA that converts from $G$. So, $G$ has $k = 1$.
  - Assume true for $k - 1$ states. CONVERT works properly for all GNFA that have $k - 1$.

- Case $k > 2$:
  - Let $G$ be the GNFA that converts from $G$. So, $G$ has $k - 1$.
  - Assume true for $k - 2$ states.
  - Let $G$ be the GNFA that converts from $G$. So, $G$ has $k - 1$.
  - Assume true for $k - 3$ states.
  - Convert works properly for all GNFA that have $k - 3$ states or less.

- Induction on number of states $k$ in GNFA.
Examples

\{ w \mid w \text{ has an equal number of 0s and 1s as substrings} \}

\{ w \mid w \text{ has an equal number of 0s and 1s} \}

Consider

- In the second class, we showed that \( a^n b^n \) is not regular.

What languages are not regular?

- You must understand their limitations.

To understand the power of finite automata,

\[
\text{Regular Expressions (cont) → Non-regular Languages}
\]
Pumping Lemma

If \( A \) is a regular language, then there is a number \( p \) (the pumping length) where, if \( s \) is any string in \( A \) of length at least \( p \), then \( s \) may be divided into three parts, \( s = xyz \), satisfying the following conditions:

1. for each \( i \geq 0 \), \( xy^i z \in A \)
2. \( |y| > 0 \)
3. \( |xy| \leq p \)

\( \forall z \in \mathbb{Z}, \forall x \in \mathbb{Z} \) for each \( i \geq 0 \), \( xy^i z \in A \), satisfying the following:

If \( A \) is a regular language, then there is a number \( d \) (the pumping length) where, if \( s \) is any string in \( A \) of length at least \( d \), then \( s \) may be divided into three parts, \( s = xyz \), satisfying the following conditions:

1. for each \( i \geq 0 \), \( xy^i z \in A \)
2. \( |y| > 0 \)
3. \( |xy| \leq d \)

\( \forall z \in \mathbb{Z}, \forall x \in \mathbb{Z} \) for each \( i \geq 0 \), \( xy^i z \in A \), satisfying the following:

\( d \geq |fx| \)

<table>
<thead>
<tr>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt;</td>
</tr>
</tbody>
</table>

How can we prove a language is not regular?

1. Regular language is recognizable by a DFA
2. Say DFA has states
3. DFA can only be in one of those states, \( d \) states
4. So, if input has more than \( d \) states, DFA must repeat a state
5. Everything that is new between the two instances of that state
6. If string that is pumped up or pumped down is not part of language, then language must not be regular.
Example:

\{0^{|x|}\}

To use pumping lemma, assume that language is regular, and show there is a contradiction.

Assume it is regular.

- Let \( p \) be the pumping length for the language.
- Choose \( s \) to be \( 0^p1^p \).
- By the pumping lemma, there must be a \( x, y, z \) in which \(|xy| \leq p\) and \(|y| \neq 0\) for some \( j \geq 0\) such that \( |x|, j + y = 0 \) for some \( k > 0 \), and \( |z| \geq 0 \).
- By the pumping lemma, there must be a \( x, y, z \) in which \(|xy| \leq p\) and \(|y| \neq 0\).

Proof of Pumping Lemma:

- Let \( M \) be a DFA that recognizes \( A \) and \( p \) be number of states.
- Use a DFA to prove this, much easier than using a NFA!

- Case 1: There is no string in \( A \) of length at least \( d \).
- Case 2: Let \( s \in A \) and have length \( u \).
- Let \( M \) be a DFA that recognizes \( A \) and be number of states.

\( d \geq u \)
Example: \{ \{ 0, 1 \} \mid 0, 1 \}

- Assume it is regular, so must be a
- But what string should we choose?
- How about \( d_010 \) \\
- Won't work, as we can pump \( 00 \)
- How about \( d_001 \) \\
- But then string should we choose?
- How about \( d_010 \) \\
- This forces us to pump in the first set of 0's

Example: \{ w \mid w \text{ has an equal number of 0s and 1s} \}

- Assume it is regular, so must be a
- But what string should we choose?
- If we pick \((01) \)
- We cannot find contradiction