Whenever I learn a new skill, I concoct elaborate fantasy scenarios where it lets me save the day.

Oh no! The killer must have followed her on vacation!

But to find them, we'd have to search through 200 MB of emails looking for something formatted like an address. It's hopeless!

Everybody stand back.

I know regular expressions.

http://ozeda.com:8890
What is a Regular Expression?

Intuition: A regular expression is a series of languages combined with regular operations

- \((\emptyset \cup 1)\emptyset^*\)
  - \(\emptyset\) and 1 are shorthand for the sets \{\emptyset\} and [1], respectively
  - \((\emptyset \cup 1)\) therefore means (\{\emptyset\} \cup [1]), or, alternatively \{0, 1\}
- \(\emptyset^*\) means \{\emptyset\}*, the language consisting of any number of \(\emptyset\)s.
- Concatenation \(\circ\) is often implied, thus: \((\emptyset \cup 1)\emptyset^*\) is shorthand for \((\emptyset \cup 1)\circ\emptyset^*\)
- Other shorthand: +, |, and ?

Definition: \(R\) is a regular expression if \(R\) is:

1. \(a\) for some \(a\) in the alphabet \(\Sigma\),
2. \(\varepsilon\),
3. \(\emptyset\),
4. \((R_1 \cup R_2)\), where \(R_1\) and \(R_2\) are regular expressions,
5. \((R_1 \circ R_2)\), where \(R_1\) and \(R_2\) are regular expressions, or
6. \((R_1^*)\) where \(R_1\) is a regular expression
What’s the difference?

∅ vs. ε

If $R$ is a regular expression:

$R \cup \emptyset$ and $R \circ \varepsilon$

vs.

$R \cup \varepsilon$ and $R \circ \emptyset$
A Few Practice Expressions

- $R_1 \mid R_2$ is shorthand for $R_1 \cup R_2$
  Commonly called “or”
- $R^+$ is shorthand for $(RR^*)$
- $R?$ means $(R \cup \varepsilon)$
  Sometimes written $[R]$

Precedence rules:
- Star before Concat
  (also $+$ and $?$ before Concat)
- Concat before Union
- Parenthesis when needed

Explain the following:
1. ab|c
2. ab*c
3. ab|cd*
4. a(b|c)*d
5. ab?c
6. $\emptyset$
7. abc$\emptyset$
8. $\emptyset*$
Theorem Time!

“A language is Regular iff some Regular Expression describes it.”

“If and only if” requires proving in two directions!

Lemma: “If a language is described by a regular expression, then it is regular.”

Lemma: “If a language is regular, then it is described by a regular expression.”
Lemma: Regular Expression to NFA

1. \( R = a \), for some \( a \in \Sigma \)

2. \( R = \varepsilon \),

3. \( R = \emptyset \)

4. \((R_1 \cup R_2)\), where \(R_1\) and \(R_2\) are regular expressions

5. \((R_1 \circ R_2)\), where \(R_1\) and \(R_2\) are regular expressions

6. \((R_1^*)\) where \(R_1\) is a regular expression
Example: \((ab \mid a)^*\)

Example from Sipser, p. 68
Example: \((a \mid b)^*aba\)

Atomic units: \(a\) and \(b\)

Union: \(a \mid b\)

Star: \((a \mid b)^*\)

Concat: \(aba\)

Final: \((a \mid b)^*aba\)

Example from Sipser, p. 69
Lemma: NFA to Regular Expression

It's not that easy...

We need another tool.
Generalized Nondeterministic Finite Automaton (GNFA)

**Intuition:** Same as NFA, except transition arrows may have any regular expression, rather than being limited to single alphabet characters.
GNFA Restrictions (for convenience)

1. The start state has transition arrows going to every other state but no arrows coming in from any other state.

2. There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.

3. Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.
NFA to GNFA is Trivially Easy

1. New $q_{\text{start}}$ with $\varepsilon$ edge to old $q_0$
2. New $q_{\text{accept}}$ with $\varepsilon$ edges from old $q \in F$
3. Multiple labels (or multiple edges between same 2 nodes) become single edge labeled as union of previous labels
4. Add $\emptyset$ edges between any nodes that do not already have edges.
Formal Definitions for GNFA

- Very similar to NFA except:
  $\delta: (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \rightarrow \mathcal{R}$
- $\mathcal{R}$ is the collection of all regular expressions over the alphabet $\Sigma$
- $q_{\text{start}}$ and $q_{\text{accept}}$ are the start and accept states, respectively
- If $\delta(q_i, q_j) = R$, the arrow from state $q_i$ to state $q_j$ has the regular expression $R$ as its label
- $G = (Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$

A GNFA accepts a string $w$ in $\Sigma^*$ if $w = w_1w_2...w_k$, where each $w_i$ is in $\Sigma^*$ and a sequence of states $q_0, q_1, ..., q_k$ exists such that

1. $q_0 = q_{\text{start}}$ is the start state,
2. $q_k = q_{\text{accept}}$ is the accept state, and
3. For each $i$, we have $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$; in other words, $R_i$ is the expression on the arrow from $q_{i-1}$ to $q_i$. 
GNFA to Regular Expression

Convert any DFA with \( n \) states to a GNFA with \( k \) states, where \( k = n + 2 \).

GNFA has \( k \) states, where \( k \geq 2 \).

If \( k > 2 \), construct an equivalent GNFA with \( k - 1 \) states by removing some \( q_{rip} \).

When \( k = 2 \), the edge from \( q_{start} \) to \( q_{accept} \) will be the equivalent regular expression.

How hard can that be?
Surgically removing $q_{\text{rip}}$

If, in the old GNFA:
1. $q_i$ goes to $q_{\text{rip}}$ with arrow labeled $R_1$,
2. $q_{\text{rip}}$ has a self loop with arrow labeled $R_2$,
3. $q_{\text{rip}}$ goes to $q_j$ with arrow labeled $R_3$, and
4. $q_i$ goes to $q_j$ with arrow labeled $R_4$

In the new GNFA, $q_i$ to $q_j$ is labeled:

$$(R_1)(R_2)^*(R_3) \cup (R_4)$$

Proof on pp. 73-74
What is the expression if:

- $R_1$ is $\emptyset$?
  - $R_4$

- $R_2$ is $\emptyset$?
  - $R_1R_3 \cup R_4$

- $R_3$ is $\emptyset$?
  - $R_4$

- $R_4$ is $\emptyset$?
  - $R_1R_2 \ast R_3$
Formally, GNFA to Regular Expression Using “CONVERT(G)”

CONVERT(\(G\)):
1. Let \(k\) be the number of states of \(G\).
2. If \(k = 2\), return expression \(R\) connecting \(q_{\text{start}}\) and \(q_{\text{accept}}\).
3. If \(k > 2\), select any \(q_{\text{rip}} \in Q - \{q_{\text{start}}, q_{\text{accept}}\}\) and let \(G'\) be the GNFA\((Q', \Sigma, \delta', q_{\text{start}}, q_{\text{accept}})\), where
   \[Q' = Q - \{q_{\text{rip}}\},\]
   and for any \(q_i \in Q' - \{q_{\text{accept}}\}\) and any \(q_j \in Q' - \{q_{\text{start}}\}\), let
   \[\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)\]
   for \(R_1 = \delta(q_j, q_{\text{rip}}), R_2 = \delta(q_{\text{rip}}, q_{\text{rip}}), R_3 = \delta(q_{\text{rip}}, q_j),\) and \(R_4 = \delta(q_i, q_j)\).
4. Return CONVERT(\(G'\)).
Example 1: 2-state DFA to Regular Expression

Does the resulting regular expression match your intuitive understanding of the original DFA?