NFA diagrams

• At each step, several choices for next state
  – if possible to reach accept, then input accepted

Formal description of NFA operation

NFA  \( M = (Q, \Sigma, \delta, q_0, F) \)
accepts a string \( w = w_1w_2w_3\ldots w_n \in \Sigma^* \)
if \( w \) can be written (by inserting \( \varepsilon \)'s) as:
\[
y = \gamma_1\gamma_2\gamma_3\ldots\gamma_m \in (\Sigma \cup \{\varepsilon\})^*
\]
and \( 3 \) sequence \( r_0, r_1, \ldots, r_m \) of states for which
\[
- r_0 = q_0
- r_i \in 0(i, y_{i+1}) \text{ for } i = 0, 1, 2, \ldots, m-1
- r_m \in F
\]
Closure under concatenation

\[ C = (A \circ B) = \{ xy : x \in A \text{ and } y \in B \} \]

Closure under star

\[ C = A^* = \{ x_1 x_2 x_3 \ldots : k \geq 0 \text{ and each } x_i \in A \} \]

NFA, FA equivalence

**Theorem:** A language \( L \) is recognized by a FA if and only if \( L \) is recognized by a NFA.

Must prove two directions:

1. \( \Rightarrow \) \( L \) is recognized by a FA implies \( L \) is recognized by a NFA.
2. \( \Leftarrow \) \( L \) is recognized by a NFA implies \( L \) is recognized by a FA.

(usually one is easy, the other more difficult)

NFA, FA equivalence

\( \Rightarrow \) \( L \) is recognized by a FA implies \( L \) is recognized by a NFA

**Proof:** A finite automaton is a nondeterministic finite automaton that happens to have no \( \epsilon \)-transitions, and for which each state has exactly one outgoing transition for each symbol.

NFA, FA equivalence

\( \Leftarrow \) \( L \) is recognized by a NFA implies \( L \) is recognized by a FA.

**Proof:** We will build a FA that simulates the NFA (and thus recognizes the same language).

- Alphabet will be the same
- What are the states of the FA?
Given NFA $M = (Q, \Sigma, \delta, q_0, F)$, construct FA $M' = (Q', \Sigma', \delta', q'_0, F')$.

Helpful defn: $E(S) = \{q \in Q : q$ reachable from $S$ by traveling along 0 or more $\epsilon$-transitions $\}$.

New transition fn: $\delta'(R, a) = \cup_{r \in R} E(\delta(r, a))$.

New start state: $q'_0 = E(q_0)$.

New accept states: $F' = \{R \in Q' : R$ contains an accept state of $M\}$.

So far...

**Theorem**: the set of languages recognized by NFA is closed under union, concatenation, and star.

**Theorem**: a language $L$ is recognized by a FA if and only if $L$ is recognized by a NFA.

Next...

- Describe the set of languages that can be built up from:
  - unions
  - concatenations
  - star operations
- Called "patterns" or regular expressions
- **Theorem**: a language $L$ is recognized by a FA if and only if $L$ is described by a regular expression.

Regular expressions

- $R$ is a regular expression if $R$ is:
  - $a$, for some $a \in \Sigma$
  - $\epsilon$, the empty string
  - $\emptyset$, the empty set
  - $(R_1 \cup R_2)$, where $R_1$ and $R_2$ are reg. exprs.
  - $(R_1 \cdot R_2)$, where $R_1$ and $R_2$ are reg. exprs.
  - $(R_1^*)$, where $R_1$ is a regular expression
- A reg. expression $R$ describes the language $L(R)$. 

Note: The diagrams represent NFA and FA equivalence with state transitions and transitions labeled with $\delta$ and $\delta'$ respectively.
Regular expressions

- example: \( R = (0 \cup 1) \)  
  - if \( \Sigma = \{0,1\} \) then use “\( \Sigma \)” as shorthand for \( R \)

- example: \( R = 0 \circ \Sigma^* \)  
  - shorthand: omit “\( \circ \)”
  - \( R = 0 \Sigma^* \)
  - precedence: \( \ast \), then \( \circ \), then \( \cup \), unless override by parenthesis
  - in example \( R = 0(\Sigma^*) \), not \( R = (0 \Sigma)^* \)

Some examples

- \( \{ w : w \text{ has at least one } 1 \} \)
  \( = \Sigma^*1\Sigma^* \)

- \( \{ w : w \text{ starts and ends with same symbol} \} \)
  \( = 0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 \)

- \( \{ w : |w| \leq 5 \} \)
  \( = (\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma) \)

- \( \{ w : \text{every } 3\text{rd position of } w \text{ is } 1 \} \)
  \( = (1\Sigma^*)^*(\varepsilon \cup 1 \cup 1\Sigma^*) \)

Manipulating regular expressions

- The empty set and the empty string:
  - \( R \cup \emptyset = R \)
  - \( R\varepsilon = \varepsilon R = R \)
  - \( R\emptyset = \emptyset R = \emptyset \)
  - \( u \) and \( + \) behave like \( + \), \( \ast \)
  - \( \emptyset \), \( \epsilon \) behave like 0,1

- additional identities:
  - \( R \cup R = R \)
  - \( (R \ast R)\ast R = (R \cup R)\ast R \)
  - \( R(R\ast R)\ast R = (R\ast R)\ast R \)

Regular expressions and FA

- **Theorem**: a language \( L \) is recognized by a FA if and only if \( L \) is described by a regular expression.

  Must prove two directions:
  \( \left( \Rightarrow \right) \) \( L \) is recognized by a FA implies \( L \) is described by a regular expression
  \( \left( \Leftarrow \right) \) \( L \) is described by a regular expression implies \( L \) is recognized by a FA.

- Proof: given regular expression \( R \) we will build a NFA that recognizes \( L(R) \).
  then NFA, FA equivalence implies a FA for \( L(R) \).

Regular expressions and FA

- \( R \) is a regular expression if \( R \) is
  \( - a \), for some \( a \in \Sigma \)
  \( - \varepsilon \), the empty string
  \( - \emptyset \), the empty set
Regular expressions and FA

- \((R_1 \cup R_2)\), where \(R_1\) and \(R_2\) are reg. exprs.

- \((R_1 \circ R_2)\), where \(R_1\) and \(R_2\) are reg. exprs.

- \((R_1^*)\), where \(R_1\) is a regular expression