Lecture Outline

Equivalence of NFAs and DFAs

- Implementing NFAs in software
  - using exhaustive enumeration
  - using sets

- Equivalence of NFAs and DFAs
  - Every DFA is an NFA
  - The powerset construction
  - Every NFA is a DFA

- Implementing NFAs in hardware
Sipser’s acceptance condition for NFAs

- Let $N_{ms} = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
  - Just like the NFA I defined on Friday except that $\delta : Q \times \Sigma_e \rightarrow 2^Q$ is a function.
  - $q' \in \delta(q, c)$ iff $N_{ms}$ can move from state $q$ to state $q'$ when reading $c$. Note that $c$ can be $\epsilon$.
  - $N_{ms}$ accepts $s$ iff there are $y_1, y_2, \ldots y_m \in \Sigma_e$ and $r_0, r_1, r_2, \ldots r_m \in Q$ such that
    - $s = y_1 \cdot y_2 \cdot \ldots \cdot y_m$.
    - $r_0 = q_0$.
    - $\forall i \in 1 \ldots m. \ r_i \in \delta(r_{i-1}, y_i)$.
    - $r_m \in F$.
- The language of NFA $N_{ms}$ is the set of all strings that $N_{ms}$ accepts.
  
  $$L(N_{ms}) = \{ s \in \Sigma^* \mid N_{ms} \text{ accepts } s \}$$
Exhaustive Enumeration

A direct implementation of Sipser's acceptance condition.

```java
boolean accept(Σ* s) { return(accept(q0, s)); }

boolean accept(Q q, Σ* s) {
    // first try ε-moves
    for each q' ∈ δ(q, ε)
        if(accept(q', s)) return(true);

    // if s = ε, we're done
    if(s == ε) return(q ∈ F);

    // now try moves for the first symbol of s
    c = first(s); x = tail(s); // s = c · x
    for each q' ∈ δ(q, c)
        if(accept(q', x)) return(true);
    return(false); // no way to reach an accepting state
}
```

What's wrong with this code?
Eliminating epsilon-loops

```java
boolean eSearch(Q q, Σ* s, Set<Q> V) {
    // V is the set of states we’ve already seen on this search.
    V = V ∪ {q}; // insert ourself into V.
    for each q' ∈ δ(q, ε)
        if(q' ∉ V)
            if(eSearch(q', V))
                return(true);
    return(accept(q, s));
}
```

We can replace the for-loop for ε-moves with the call eSearch(q, s, {q}) and we’ll get code that doesn’t loop forever.

But it will still be slow –
Worst-case run-time $\Omega(|Q|\cdot|s|)$.

Note: for most of this course, we’ll be concerned about computability rather than efficiency. However, a more efficient algorithm for NFA will also show us how to turn a NFA into a DFA.
Mark’s acceptance condition for NFAs

- Let $N_{mrg} = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
  I’ll use $\delta : Q \times \Sigma \epsilon \rightarrow 2^Q$ like Sipser.

- $close_\epsilon(q)$ be the set of all states reachable from $q$ by zero or more $\epsilon$-moves.

- Extend $close_\epsilon$ to sets.

- Let $step(G, c)$ be the set of all states reachable from a state in $G$ when reading symbol $c$. Let $\delta(G, x)$ be the set of all states reachable from a state in $G$ when reading string $x$.

- $N_{mrg}$ accepts $s$ iff

  $$\delta(close_\epsilon(\{q_0\}), s) \cap F \neq \emptyset$$
Mark’s acceptance condition for NFAs

- Let $N_{mrg} = (Q, \Sigma, \delta, q_0, F')$ be an NFA.

- $close_\epsilon(q)$ be the set of all states reachable from $q$ by zero or more $\epsilon$-moves.
  
  $p \in close_\epsilon(q)$ iff
  
  - $p = q$
  - $\exists q' \in close_\epsilon(q). p \in \delta(q', \epsilon)$

- Extend $close_\epsilon$ to sets.

- Let $step(G, c)$ be the set of all states reachable from a state in $G$ when reading symbol $c$. Let $\delta(G, x)$ be the set of all states reachable from a state in $G$ when reading string $x$.

- $N_{mrg}$ accepts $s$ iff
  
  $\delta(close_\epsilon(\{q_0\}), s) \cap F \neq \emptyset$
Mark’s acceptance condition for NFAs

- Let $N_{mrgr} = (Q, \Sigma, \delta, q_0, F')$ be an NFA.

- $close_\varepsilon(q)$ be the set of all states reachable from $q$ by zero or more $\varepsilon$-moves.

- Extend $close_\varepsilon$ to sets.

\[
close_\varepsilon(G) = \bigcup_{q \in G} close_\varepsilon(q)
\]

- Let $step(G, c)$ be the set of all states reachable from a state in $G$ when reading symbol $c$. Let $\delta(G, x)$ be the set of all states reachable from a state in $G$ when reading string $x$.

- $N_{mrgr}$ accepts $s$ iff

\[
\delta(close_\varepsilon(\{q_0\}), s) \cap F \neq \emptyset
\]
Mark’s acceptance condition for NFAs

1. Let $N_{mrg} = (Q, \Sigma, \delta, q_0, F')$ be an NFA.
2. $\text{close}_\epsilon(q)$ be the set of all states reachable from $q$ by zero or more $\epsilon$-moves.
3. Extend $\text{close}_\epsilon$ to sets.
4. Let $\text{step}(G, c)$ be the set of all states reachable from a state in $G$ when reading symbol $c$. Let $\delta(G, x)$ be the set of all states reachable from a state in $G$ when reading string $x$.

\[
\begin{align*}
\text{step}(q, c) &= \text{close}_\epsilon(\{q' \mid q' \in \delta(q, c)\}), \quad q \in Q, c \in \Sigma \\
\text{step}(G, c) &= \bigcup_{q \in G} \text{step}(q, c), \quad G \subseteq Q, c \in \Sigma \\
\delta(G, \epsilon) &= G, \quad G \subseteq Q \\
\delta(G, x \cdot c) &= \text{step}(\delta(G, x), c), \quad G \subseteq Q, x \in \Sigma^*, c \in \Sigma
\end{align*}
\]

$N_{mrg}$ accepts $s$ iff

\[\delta(\text{close}_\epsilon(\{q_0\}), s) \cap F \neq \emptyset\]
Mark’s acceptance condition for NFAs

- Let $N_{mrg} = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
  I’ll use $\delta : Q \times \Sigma_\epsilon \rightarrow 2^Q$ like Sipser.

- $\text{close}_\epsilon(q)$ be the set of all states reachable from $q$ by zero or more $\epsilon$-moves.

- Extend $\text{close}_\epsilon$ to sets.

- Let $\text{step}(G, c)$ be the set of all states reachable from a state in $G$ when reading symbol $c$. Let $\delta(G, x)$ be the set of all states reachable from a state in $G$ when reading string $x$.

- $N_{mrg}$ accepts $s$ iff

  $$\delta(\text{close}_\epsilon\{q_0\}, s) \cap F \neq \emptyset$$
Mark’s acceptance condition for NFAs

- Let $N_{mrg} = (Q, \Sigma, \delta, q_0, F')$ be an NFA.

- $\text{close}_\varepsilon(q)$ be the set of all states reachable from $q$ by zero or more $\varepsilon$-moves. $p \in \text{close}_\varepsilon(q)$ iff

  
  \begin{align*}
  p &= q \\
  \exists q' \in \text{close}_\varepsilon(q). & p \in \delta(q', \varepsilon)
  \end{align*}

- Extend $\text{close}_\varepsilon$ to sets.

- Let $\text{step}(G, c)$ be the set of all states reachable from a state in $G$ when reading symbol $c$. Let $\delta(G, x)$ be the set of all states reachable from a state in $G$ when reading string $x$.

- $N_{mrg}$ accepts $s$ iff

  \[ \delta(\text{close}_\varepsilon(\{q_0\}), s) \cap F' \neq \emptyset \]
Mark’s acceptance condition for NFAs

- Let $N_{mrg} = (Q, \Sigma, \delta, q_0, F)$ be an NFA.

- $\text{close}_\epsilon(q)$ be the set of all states reachable from $q$ by zero or more $\epsilon$-moves.

- Extend $\text{close}_\epsilon$ to sets.

$$
\text{close}_\epsilon(G) = \bigcup_{q \in G} \text{close}_\epsilon(q)
$$

- Let $\text{step}(G, c)$ be the set of all states reachable from a state in $G$ when reading symbol $c$. Let $\delta(G, x)$ be the set of all states reachable from a state in $G$ when reading string $x$.

- $N_{mrg}$ accepts $s$ iff

$$
\delta(\text{close}_\epsilon\{q_0\}, s) \cap F \neq \emptyset
$$
Mark’s acceptance condition for NFAs

- Let $N_{mrg} = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
- $\text{close}_\epsilon(q)$ be the set of all states reachable from $q$ by zero or more $\epsilon$-moves.
- Extend $\text{close}_\epsilon$ to sets.
- Let $\text{step}(G, c)$ be the set of all states reachable from a state in $G$ when reading symbol $c$. Let $\delta(G, x)$ be the set of all states reachable from a state in $G$ when reading string $x$.

\[
\begin{align*}
\text{step}(q, c) &= \text{close}_\epsilon(\{q' \mid q' \in \delta(q, c)\}), & q \in Q, c \in \Sigma \\
\text{step}(G, c) &= \bigcup_{q \in G} \text{step}(q, c), & G \subseteq Q, c \in \Sigma \\
\delta(G, \epsilon) &= G, & G \subseteq Q \\
\delta(G, x \cdot c) &= \text{step}(\delta(G, x), c), & G \subseteq Q, x \in \Sigma^*, c \in \Sigma
\end{align*}
\]

- $N_{mrg}$ accepts $s$ iff

\[
\delta(\text{close}_\epsilon(\{q_0\}), s) \cap F \neq \emptyset
\]
Computing Reachable Sets

A direct implementation of Mark’s acceptance condition.

Set\(<Q>\) eClose\(Q, q, Set<Q> V\) \{ 

// states reachable from \(q\) by \(\epsilon\)-moves
if\((q \in V)\) return\(V\); // already seen \(q\)
\(V = V \cup \{q\}\);
for each \(q' \in \delta(q, \epsilon)\)
\(V = V \cup eClose(q', V)\);
return\(V\); \}

Set\( <Q> \) step\(Set<Q> G, \Sigma c\) \{ 

// states reachable from \(G\) by reading symbol \(c\)
\}

Set\( <Q> \) \(\delta(Set<Q> G, \Sigma^* s)\) \{ 

// states reachable from \(G\) by reading string \(s\)
\}

boolean accept\(\Sigma^* s\) \{ 

return\(((\delta(close_\epsilon\{q_0\}; s) \cap F) \neq \emptyset)\);
\}

15 September 2008 – p.8/15
Computing Reachable Sets

A direct implementation of Mark's acceptance condition.

\[
\text{Set}< Q > \, \text{eClose}(Q, q, \text{Set}< Q > V) \\{ \\
\quad \text{// states reachable from } q \text{ by } \epsilon-\text{moves} \}
\]

\[
\text{Set}< Q > \, \text{step}(\text{Set}< Q > G, \Sigma c) \\{ \\
\quad \text{// states reachable from } G \text{ by reading symbol } c \\
\quad V = \emptyset; \\
\quad \text{for each } q \in G \\
\quad \quad V = \text{eClose}(\delta(q, c), V) \\
\quad \text{return}(V); \\
\}
\]

\[
\text{Set}< Q > \, \delta(\text{Set}< Q > G, \Sigma^* s) \\{ \\
\quad \text{// states reachable from } G \text{ by reading string } s \\
\}
\]

\[
\text{boolean accept}(\Sigma^* s) \\{ \\
\quad \text{return}((\delta(close_{\epsilon}\{q_0\}, s) \cap F) \neq \emptyset); \\
\}
\]
Computing Reachable Sets

A direct implementation of Mark’s acceptance condition.

\[
\text{Set}\langle Q \rangle \text{ eClose}(Q, q, \text{Set}\langle Q \rangle \text{ V}) \{ \\
// \text{states reachable from } q \text{ by } \epsilon\text{-moves} \\
\}
\]

\[
\text{Set}\langle Q \rangle \text{ step}(\text{Set}\langle Q \rangle \text{ G}, \Sigma \text{ c}) \{ \\
// \text{states reachable from } G \text{ by reading symbol } c \\
\}
\]

\[
\text{Set}\langle Q \rangle \delta(\text{Set}\langle Q \rangle \text{ G}, \Sigma^* \text{ s}) \{ \\
// \text{states reachable from } G \text{ by reading string } s \\
\text{if}(s == \epsilon) \text{return}(G); \\
x = \text{head}(s); c = \text{last}(s); // s = x \cdot c \\
\text{return}(\text{eClose}(\text{step}(\delta(G, x), c))); \\
\}
\]

\[
\text{boolean accept}(\Sigma^* \text{ s}) \{ \\
\text{return}(\delta(\text{close}_{\epsilon}\{q_0\}, s) \cap F \neq \emptyset); \\
\}
\]
Time-Complexity for Reachability

- Processing each symbol can involve considering up to $|Q|$ states, each of which can have up to $|Q|$ successor states.
- eClose takes at most $|Q|$ time.
- Thus, each symbol of $s$ can be processed in $O(|Q|^2)$ time.
- The total times is $O(|s| \cdot |Q|^2)$.
- This is much better than the exponential time for the earlier approach.
Equivalence of NFAs and DFAs

We want to show that the sets of languages recognized by NFAs and the set recognized by DFAs are the same.

Showing that every language recognized by a DFA is also recognized by an NFA is easy: every DFA is an NFA.

Showing that every language recognized by an NFA is also recognized by a DFA is more work. That’s what we’ll take on in the next few slides.
From an NFA to an Equivalent DFA

- Basic strategy: we noted that the definitions of NFAs and DFAs are quite similar – the main difference is the definition of $\delta$.

- Given an NFA, $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$, we’ll construct a DFA, $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ such that $L(D) = L(N)$.

- Our strategy is based on thinking about how we defined the acceptance condition for an NFA – we wrote a function that keeps track of the set of possible states that the NFA can be in after reading each symbol of the input.

- If $Q_N$ is finite, then the set $2^Q_N$ is finite as well (even though it may be very big). We’ll let $Q_D = 2^Q_N$: now each state of the DFA describes the set of states that the NFA could be in at that point.

- Now, we need to define $\delta_D$, $q_0$, and $F_D$. We’ll start with $\delta_D$: once we have that, $q_0$ and $F_D$ are pretty straightforward.
Defining the DFA

The key observation is that the \textit{step} function as defined on slide 6 provides the next-state function that we need for the DFA whose states are subsets of $Q_N$.

- $Q_D = 2^{Q_N}$: states of $D$ are subsets of $Q_N$.
- $\delta_D = \text{step}$ (the version for sets).
  Note that $\text{step} : 2^{Q_N} \times \Sigma \rightarrow 2^{Q_N}$ which means that $\delta_D : Q_D \times \Sigma \rightarrow Q_D$ as required.
- $q_{0,D} = \text{close}_\epsilon\{q_{0,N}\}$: note that we need the $\epsilon$-closure so we will accept $\epsilon$ if there is any accepting state that is reachable from $q_{0,N}$ by zero or more $\epsilon$-moves.
- $F_D = \{ B \subseteq Q_N \mid B \cap F_N \neq \emptyset \}$: The accepting states of $D$ are all states that contain at least one accepting state of $N$. 
Proof that $L(D) = L(N)$

- Let $w$ be a string in $\Sigma^*$.
- I'll show that $w \in L(D)$ iff $w \in L(N)$.
- Observe that $\delta_D$ is exactly the same function as $\delta_N$. See the definitions on slides 7 and 12.
- Proof that $L(D) = L(N)$:

  \[
  \begin{align*}
  w \in L(N) &\iff (\delta_N(\text{close}_\epsilon\{q_0, N\}, w) \cap F_N) \neq \emptyset, \\
  &\iff \delta_N(q_{0,D}, w) \in F_D, \\
  &\iff \delta_D(q_{0,D}, w) \in F_D, \\
  &\iff w \in L(D)
  \end{align*}
  \]

  $\delta_D = \delta_N$, as noted above  
  def. DFA accept, Sept. 8 lecture notes  
  def. NFA accept, slide 7  
  def. $q_{0,D}$ and $F_D$, slide 12
Example: \( \{ab, aba\}^* \)

The NFA:

- \( Q_N = \{0, 1, 2, 3, 4\} \)
- \( \Sigma = \{a, b\} \)
- \( \delta_N(0, \epsilon) = \{1\} \), \( \delta_N(0, a) = \emptyset \), \( \delta_N(0, b) = \emptyset \)
- \( \delta_N(1, \epsilon) = \emptyset \), \( \delta_N(1, a) = \{2\} \), \( \delta_N(1, b) = \emptyset \)
- \( \delta_N(2, \epsilon) = \emptyset \), \( \delta_N(2, a) = \emptyset \), \( \delta_N(2, b) = \{3\} \)
- \( \delta_N(3, \epsilon) = \{1\} \), \( \delta_N(3, a) = \{4\} \), \( \delta_N(3, b) = \emptyset \)
- \( \delta_N(4, \epsilon) = \{1\} \), \( \delta_N(4, a) = \emptyset \), \( \delta_N(4, b) = \emptyset \)

- \( q_{0,N} = 0 \)
- \( F_N = \{0, 3, 4\} \)
This week

Reading: Note: this is different than the schedule in the Sept. 3 notes – we’re one lecture ahead of schedule.

   September 15 (Today): Equivalence of NFAs and DFAs
       The rest of Sipser 1.2. (i.e. pages 53-63).

   September 17 (Wednesday): Regular Expressions
       Read Sipser 1.3. Lecture will cover through example 1.58 (i.e. pages 63-69).

   September 19 (Friday): Equivalence of DFAs and Regular Expressions
       The rest of Sipser 1.3 (i.e. pages 69–76).

Homework:

   September 19 (next Friday): Homework 1 due. Homework 2 goes out (due Sept. 26).

Midterm: Oct. 8