Equivalence of NFAs and DFAs

Mark Greenstreet, CpSc 421, Term 1, 2008/09

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 $\delta : Q \times \Sigma_\epsilon \rightarrow 2^Q$ 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 ϵ.
- N_{ms} accepts s iff there are $y_1, y_2, \ldots y_m \in \Sigma_\epsilon$ and $r_0, r_1, r_2, \ldots r_m \in Q$ such that
  - $s = y_1 \cdot y_2 \cdots y_m$.
  - $r_0 = q_0$.
  - $\forall i \in 1 \ldots m.\ r_i \in \delta(r_{i-1}, y_i)$.
  - $r_m \in F$.
- The language of NFA N_{ms} is the set of all strings that N_{ms} accepts.

$$L(N_{ms}) \quad = \quad \{s \in \Sigma^* \mid N_{ms} \text{ accepts } s\}$$

Exhaustive Enumeration

A direct implementation of Sipser's acceptance condition.

```
boolean accept(Σ* s) { return(accept(q_0, 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

```
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 $\Omega(|Q|^{|s|})$.

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 to 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 $\delta : Q \times \Sigma_\epsilon \to 2^Q$ like Sipser.
- $close_\epsilon(q)$ be the set of all states reachable from q by zero or more ϵ-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 ϵ-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 $N_{mrg} = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
- $close_\epsilon(q)$ be the set of all states reachable from q by zero or more ϵ-moves.
- Extend $close_\epsilon$ to sets.

$$close_\epsilon(G) \quad = \quad \bigcup_{q \in G} close_\epsilon(q)$$

- 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_{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 ϵ-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.

$$\begin{array}{rcll}
step(q,c) & = & close_\epsilon(\{q' \mid q' \in \delta(q,c)\}), & q \in Q, c \in \Sigma \\
step(G,c) & = & \bigcup_{q \in G} step(q,c), & G \subseteq Q, c \in \Sigma \\
\delta(G,\epsilon) & = & G, & G \subseteq Q \\
\delta(G, x \cdot c) & = & step(\delta(G,x), c), & G \subseteq Q, x \in \Sigma^*, c \in \Sigma
\end{array}$$

- 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.
  I'll use $\delta : Q \times \Sigma_\epsilon \to 2^Q$ like Sipser.
- $close_\epsilon(q)$ be the set of all states reachable from q by zero or more ϵ-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 ϵ-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 $N_{mrg} = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
- $close_\epsilon(q)$ be the set of all states reachable from q by zero or more ϵ-moves.
- Extend $close_\epsilon$ to sets.

$$close_\epsilon(G) \quad = \quad \bigcup_{q \in G} close_\epsilon(q)$$

- 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_{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 ϵ-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.

$$
\begin{array}{rclr}
step(q, c) & = & close_\epsilon(\{q' \mid q' \in \delta(q, c)\}), & q \in Q, c \in \Sigma \\
step(G, c) & = & \displaystyle\bigcup_{q \in G} step(q, c), & G \subseteq Q, c \in \Sigma \\
\delta(G, \epsilon) & = & G, & G \subseteq Q \\
\delta(G, x \cdot c) & = & step(\delta(G, x), c), & G \subseteq Q, x \in \Sigma^*, c \in \Sigma
\end{array}
$$

- N_{mrg} accepts s iff

$$\delta(close_\epsilon(\{q_0\}), s) \cap F \neq \emptyset$$

Computing Reachable Sets

A direct implementation of Mark's acceptance condition.

```
Set<Q> eClose(Q q, Set<Q> V) {
    // states reachable from q by ε-moves
    if(q ∈ V) return(V); // already seen q
    V = V ∪ {q};
    for each q′ ∈ δ(q, ε)
        V = V ∪ eClose(q′, V);
    return(V); }

    Set<Q> step(Set<Q> G, Σ c) {
        // states reachable from G by reading symbol c
    }

    Set<Q> δ(Set<Q> G, Σ* s) {
        // states reachable from G by reading string s
    }

    boolean accept(Σ* s) {
        return((δ(close_ε{q_0}, s) ∩ F) ≠ ∅);
    }
```

Computing Reachable Sets

A direct implementation of Mark's acceptance condition.

Set$< Q >$ eClose(Q q, Set$< Q >$ V) $\{$
// states reachable from q by ϵ-moves$\}$

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

Set$< Q >$ δ(Set$< Q >$ G, Σ^* s) $\{$
// states reachable from G by reading string s
$\}$

boolean accept(Σ^* s) $\{$
return($(\delta(close_\epsilon\{q_0\}, s) \cap F) \neq \emptyset$);
$\}$

Computing Reachable Sets

A direct implementation of Mark's acceptance condition.

Set$<Q>$ eClose(Q q, Set$<Q>$ V) $\{$
    // states reachable from q by ϵ-moves$\}$

    Set$<Q>$ step(Set$<Q>$ G, Σ c) $\{$
        // states reachable from G by reading symbol c
    $\}$

    Set$<Q>$ δ(Set$<Q>$ G, Σ^* s) $\{$
        // states reachable from G by reading string s
        if($s == \epsilon$) return(G);
        $x = head(s)$; $c = last(s)$; // $s = x \cdot c$
        return(eClose(step$(\delta(G, x), c)$));
    $\}$

    boolean accept(Σ^* s) $\{$
        return($(\delta(close_\epsilon\{q_0\}, s) \cap F) \neq \emptyset$);
    $\}$

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 δ.
- Given an NFA, $N = (Q_N, \Sigma, \delta_N, q_{0,N}, F_N)$, we'll construct a DFA, $D = (Q_D, \Sigma, \delta_D, q_{0,D}, F_D)$ 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 δ_D, $q_{0,d}$, and F_D.
  We'll start with δ_D: once we have that, $q_{0,d}$ and F_D are pretty straightforward.

Defining the DFA

The key observation is that the 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.

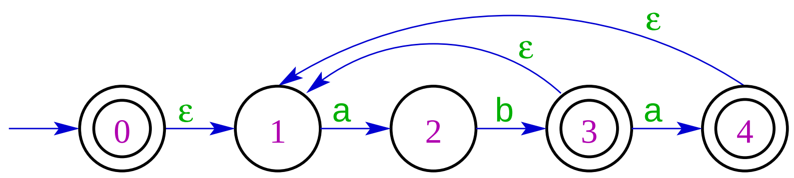
- $Q_D = 2^{Q_N}$: states of D are subsets of Q_N.
- $\delta_D =$ step (the version for sets).
  Note that step : $2^{Q_N} \times \Sigma \rightarrow 2^{Q_N}$ which means that $\delta_D : Q_D \times \Sigma \rightarrow Q_D$ as required.
- $q_{0,D} = close_\epsilon\{q_{0,N}\}$: note that we need the ϵ-closure so we will accept ϵ if there is any accepting state that is reachable from $q_{0,N}$ by zero or more ϵ-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 Σ^*.
- I'll show that $w \in L(D)$ iff $w \in L(N)$.
- Observe that δ_D is exactly the same function as δ_N. See the definitions on slides 7 and 12.
- Proof that $L(D) = L(N)$:

$$
\begin{array}{lll}
 & w \in L(N) & \\
\Leftrightarrow & (\delta_N(close_\epsilon\{q_{0,N}\}, w) \cap F_N) \neq \emptyset, & \text{def. NFA accept, slide 7} \\
\Leftrightarrow & \delta_N(q_{0,D}, w) \in F_D, & \text{def. } q_{0,D} \text{ and } F_D\text{, slide 12} \\
\Leftrightarrow & \delta_D(q_{0,D}, w) \in F_D, & \delta_D = \delta_N\text{, as noted above} \\
\Leftrightarrow & w \in L(D) & \text{def. DFA accept, Sept. 8 lecture notes}
\end{array}
$$

Example: {ab, aba}*



The NFA:

- $Q_N = \{0, 1, 2, 3, 4\}$
- $\Sigma = \{\mathsf{a}, \mathsf{b}\}$;
- $$\begin{array}{lll} \delta_N(0, \epsilon) = \{1\}, & \delta_N(0, \mathsf{a}) = \emptyset, & \delta_N(0, \mathsf{b}) = \emptyset \\ \delta_N(1, \epsilon) = \emptyset, & \delta_N(1, \mathsf{a}) = \{2\}, & \delta_N(1, \mathsf{b}) = \emptyset \\ \delta_N(2, \epsilon) = \emptyset, & \delta_N(2, \mathsf{a}) = \emptyset, & \delta_N(2, \mathsf{b}) = \{3\}, \\ \delta_N(3, \epsilon) = \{1\}, & \delta_N(3, \mathsf{a}) = \{4\}, & \delta_N(3, \mathsf{b}) = \emptyset, \\ \delta_N(4, \epsilon) = \{1\}, & \delta_N(4, \mathsf{a}) = \emptyset, & \delta_N(4, \mathsf{b}) = \emptyset; \end{array}$$
- $q_{0,N} = 0$;
- $F_N = \{0, 3, 4\}$.

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): Equivalance 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 throud 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