Nondeterministic Finite Automata

Nondeterminism
Subset Construction

Modified from Jeff Ullman’s slides

Nondeterminism

A nondeterministic finite automaton has the ability to be in several states at once.

Transitions from a state on an input symbol can be to any set of states.

Nondeterminism – (2)

Start in one start state.
Accept if any sequence of choices leads to a final state.

Intuitively: the NFA always “guesses right.”

Example: Moves on a Chessboard

States = squares.
Inputs = r (move to an adjacent red square) and b (move to an adjacent black square); diagonal is adjacent.

Start state, final state are in opposite corners.

Example: Chessboard – (2)

A finite set of states, typically Q.
An input alphabet, typically Σ.
A transition function, typically δ.
A start state in Q, typically q₀.
A set of final states F ⊆ Q.

Formal NFA
Transition Function of an NFA

- $\delta(q, a)$ is a set of states.
- Extend to strings as follows:
  - Basis: $\delta(q, \varepsilon) = \{q\}$
  - Induction: $\delta(q, wa) =$ the union over all states $p$ in $\delta(q, w)$ of $\delta(p, a)$

Language of an NFA

- A string $w$ is accepted by an NFA if $\delta(q_0, w)$ contains at least one final state.
- The language of the NFA is the set of strings it accepts.
- Can NFAs do more than DFAs?

Example: Language of an NFA

- For our chessboard NFA we saw that rbb is accepted.
- If the input consists of only b’s, the set of accessible states alternates between \{5\} and \{1,3,7,9\}, so only even-length, nonempty strings of b’s are accepted.
- What about strings with at least one r?

Equivalence of DFA’s, NFA’s

- A DFA $D$ can be turned into an NFA $N$ that accepts the same language.
- If $\delta_D(q, a) = p$, then let the NFA have $\delta_N(q, a) = \{p\}$.
- Then the NFA is always in a set containing exactly one state – the state the DFA is in after reading the same input.

Equivalence – (2)

- Surprisingly, for any NFA there is a DFA that accepts the same language.
- Proof is the subset construction.
- The number of states of the DFA can be exponential in the number of states of the NFA.
- Thus, NFA’s accept exactly the regular languages.

Subset Construction

- Given an NFA with states $Q$, inputs $\Sigma$, transition function $\delta_N$, state state $q_0$, and final states $F$, construct equivalent DFA with:
  - State set $2^Q$ (Set of subsets of $Q$).
  - Inputs $\Sigma$.
  - Start state $\{q_0\}$.
  - Final states = all those with a member of $F$. 

Critical Point

- The DFA states have *names* that are sets of NFA states.
- But as a DFA state, an expression like \{p,q\} must be read as a single symbol, not as a set.
- Analogy: a class of objects whose values are sets of objects of another class, e.g. the state of a hash table.

Subset Construction – (2)

- The transition function \( \delta_D \) is defined by:
  \[ \delta_D(\{q_1,\ldots,q_k\}, a) = \bigcup_{i = 1,\ldots,k} \delta_N(q_i, a) \]
- Example: We’ll construct the DFA equivalent of our “chessboard” NFA.

Example: Subset Construction

<table>
<thead>
<tr>
<th>( r )</th>
<th>( b )</th>
<th>( {1} )</th>
<th>( {2,4} )</th>
<th>( {5} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,4</td>
<td>5</td>
<td>(2,4)</td>
<td>(5)</td>
</tr>
<tr>
<td>2</td>
<td>4,6</td>
<td>1,3,5</td>
<td>{2,4}</td>
<td>(5)</td>
</tr>
<tr>
<td>3</td>
<td>2,6</td>
<td>5</td>
<td>{5}</td>
<td>{2,4}</td>
</tr>
<tr>
<td>4</td>
<td>2,8</td>
<td>1,5,7</td>
<td>{5}</td>
<td>(2,4)</td>
</tr>
<tr>
<td>5</td>
<td>2,4,6,8</td>
<td>1,3,7,9</td>
<td>{2,4}</td>
<td>(2,4)</td>
</tr>
<tr>
<td>6</td>
<td>2,8</td>
<td>3,5,9</td>
<td>(2,4)</td>
<td>(5)</td>
</tr>
<tr>
<td>7</td>
<td>4,8</td>
<td>5</td>
<td>{5}</td>
<td>{2,4}</td>
</tr>
<tr>
<td>8</td>
<td>4,6</td>
<td>5,7,9</td>
<td>(2,4,6,8)</td>
<td>(2,4,6,8)</td>
</tr>
<tr>
<td>9</td>
<td>6,8</td>
<td>5</td>
<td>(1,3,5,7)</td>
<td>(1,3,5,7)</td>
</tr>
</tbody>
</table>

Alert: What we’re doing here is the *lazy* form of DFA construction, where we only construct a state if we are forced to. (how many?)
Example: Subset Construction

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4,6</td>
<td>1,3,5</td>
</tr>
<tr>
<td>3</td>
<td>2,6</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2,8</td>
<td>1,5,7</td>
</tr>
<tr>
<td>5</td>
<td>2,4,6,8</td>
<td>1,5,7,9</td>
</tr>
<tr>
<td>6</td>
<td>2,8</td>
<td>3,5,9</td>
</tr>
<tr>
<td>7</td>
<td>4,8</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>4,6</td>
<td>5,7,9</td>
</tr>
<tr>
<td>9</td>
<td>6,8</td>
<td>5</td>
</tr>
</tbody>
</table>

Proof of Equivalence: Subset Construction

Show by induction on $|w|$ that $\delta_N(q_0, w) = \delta_D(\{q_0\}, w)$.

**IH(n):** for all $w$ of length $n$,

$\delta_N(q_0, w) = \delta_D(\{q_0\}, w)$

**Basis:** Show IH(0). Only string of length 0 is $\epsilon$, and

$\delta_N(q_0, \epsilon) = \{q_0\}$

$\delta_D(\{q_0\}, \epsilon) = \{q_0\}$

**Induction Step (1)**

Assume $n \geq 0$ and IH(n) to show IH(n+1)

Let $w$ be an arbitrary string of length $n+1$

Need to show $\delta_N(q_0, w) = \delta_D(\{q_0\}, w)$

Let $w = xa$, so $|x| = n$, (use IH(n) on $x$)

Let $S = \delta_N(q_0, x)$;

By IH(N), $S = \delta_D(\{q_0\}, x)$

$\delta_N(q_0, w) = \delta_N(q_0, xa) = \delta_N(p, a)$ the union over all states $p$ in $S$

by def. extension of $\delta_N$

**Induction Step (2)**

Assume $n \geq 0$ and IH(n) to show IH(n+1)

$w = xa; |x| = n; \delta_N(q_0, x) = S = \delta_D(q_0, x)$

$\delta_N(q_0, w) = \delta_N(p, a)$

$\delta_D(\{q_0\}, w) = \delta_D(\{q_0\}, xa)$

$\delta_D(S, a) = \delta_D(xa)$

So $\delta_N(q_0, w) = \delta_D(\{q_0\}, w)$
NFA’s With $\varepsilon$-Transitions

- We can allow state-to-state transitions on $\varepsilon$ input.
- These transitions are done spontaneously, without looking at the input string.
- A convenience at times, but still only regular languages are accepted.

Example: $\varepsilon$-NFA

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(E)</td>
<td>(B)</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>B</td>
<td>$\emptyset$</td>
<td>(C)</td>
<td>(D)</td>
</tr>
<tr>
<td>C</td>
<td>$\emptyset$</td>
<td>(D)</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>D</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>E</td>
<td>${F}$</td>
<td>$\emptyset$</td>
<td>(B, C)</td>
</tr>
<tr>
<td>F</td>
<td>${D}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

Closure of States

- $\text{CL}(q) =$ set of states you can reach from state $q$ following only arcs labeled $\varepsilon$.
- Example: $\text{CL}(A) = \{A\}$; $\text{CL}(E) = \{B, C, D, E\}$.
- Closure of a set of states = union of the closure of each state.

Example: Extended Delta

- $\delta(A, \varepsilon) = \text{CL}(A) = \{A\}$.
- $\delta(A, 0) = \text{CL}(\{E\}) = \{B, C, D, E\}$.
- $\delta(A, 01) = \text{CL}(\{C, D\}) = \{C, D\}$.
- Language of an $\varepsilon$-NFA is the set of strings $w$ such that $\delta(q_0, w)$ contains a final state.

Extended Delta

- Basis: $\delta(q, \varepsilon) = \text{CL}(q)$.
- Induction: $\delta(q, xa)$ is computed as follows:
  1. Start with $\delta(q, x) = S$.
  2. Take the union of $\text{CL}(\delta(p, a))$ for all $p \in S$.
- Intuition: $\delta(q, w)$ is the set of states you can reach from $q$ following a path labeled $w$.

And notice that $\delta(q, a)$ is not that set of states, for symbol $a$.

Equivalence of NFA, $\varepsilon$-NFA

- Every NFA is an $\varepsilon$-NFA.
  - It just has no transitions on $\varepsilon$.
- Converse requires us to take an $\varepsilon$-NFA and construct an NFA that accepts the same language.
  - We do so by combining $\varepsilon$-transitions with the next transition on a real input.

Warning: This treatment is a bit different from that in the text.
Equivalence – (2)

- Start with an $\epsilon$-NFA with states $Q$, inputs $\Sigma$, start state $q_0$, final states $F$, and transition function $\delta_E$.
- Construct an “ordinary” NFA with states $Q$, inputs $\Sigma$, start state $q_0$, final states $F'$, and transition function $\delta_N$.

Equivalence – (3)

- Compute $\delta_N(q, a)$ as follows:
  1. Let $S = CL(q)$.
  2. $\delta_N(q, a)$ is the union over all $p$ in $S$ of $\delta_E(p, a)$.
- $F'$ is the set of states $q$ such that $CL(q)$ contains a state of $F$.
- Intuition: $\delta_N$ incorporates $\epsilon$-transitions before using $a$ but not after.

Equivalence – (4)

- Prove by induction on $|w|$ that
  \[ CL(\delta_N(q_0, w)) = \delta_E(q_0, w). \]
- Thus, the $\epsilon$-NFA accepts $w$ if and only if the “ordinary” NFA does.
Interesting closures: $CL(B) = \{B, D\}$; $CL(E) = \{B, C, D, E\}$

Example: $\varepsilon$-NFA-to-NFA

Since closure of $E$ includes $B$ and $C$, which have transitions on 1 to $C$ and $D$.

Since closure of $E$ includes $B$ and $C$, which have transitions on 1 to $C$ and $D$.

Since closure of $E$ includes $B$ and $C$, which have transitions on 1 to $C$ and $D$.

Summary

- DFA’s, NFA’s, and $\varepsilon$-NFA’s all accept exactly the same set of languages: the regular languages.
- The NFA types are easier to design and may have exponentially fewer states than a DFA.
- But only a DFA can be implemented!