Nondeterminism and Epsilon Transitions

Mridul Aanjaneya

Stanford University

June 28, 2012
Question

Prove that any square with side length a power of 2, and one square removed, is tileable with \textit{L}'s.
Question

Are such tilings always possible?
**Definition**

A DFA is a 5-tuple \((Q, \Sigma, \delta_D, q_0, F)\) consisting of:

- A finite set of states \(Q\),
- A set of input alphabets \(\Sigma\),
- A transition function \(\delta_D : Q \times \Sigma \rightarrow Q\),
- A start state \(q_0\), and
- A set of accept states \(F \subseteq Q\).

The transition function \(\delta_D\):

- Takes two arguments, a state \(q\) and an alphabet \(a\).
- \(\delta_D(q,a) = \) the state the DFA goes to when it is in state \(q\) and the alphabet \(a\) is received.
Recap: Graph representation of DFA’s

- Nodes correspond to states.
- Arcs represent transition function.
  - Arc from state \( p \) to state \( q \) labeled by all those input symbols that have transitions from \( p \) to \( q \).
- Incoming arrow from outside denotes start state.
- Accept states indicated by double circles.

- Accepts all binary strings without two consecutive 1’s.
Recap: Transition table for DFA's

- A row for each state, a column for each alphabet.
- Accept states are starred.
- Arrow for the start state.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_1$</td>
<td>$q_3$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$q_3$</td>
<td>$q_3$</td>
</tr>
</tbody>
</table>

Diagram:

- $q_1$ to $q_1$ on 0 and 1
- $q_2$ to $q_3$ on 1
- $q_3$ is an accept state
- $q_1$ is the start state
Recap: Regular languages

Definition

A DFA $M = (Q, \Sigma, \delta_D, q_0, F)$ accepts $w$ if there exists a sequence of states $r_0, r_1, \ldots, r_n$ in $Q$ with three conditions:

- $r_0 = q_0$
- $\delta_D(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n - 1$, and
- $r_n \in F$

- Condition 1 says that $M$ starts in the start state $q_0$.
- Condition 2 says that $M$ follows $\delta_D$ between two states.
- Condition 3 says that last state is an accept state.
- We say that $M$ recognizes $L$ if $L = \{w \mid M$ accepts $w\}$. 
Example

Let $L = \{w \mid w \in \{0,1\}^* \text{ and } w, \text{ viewed as a binary integer, is divisible by 5.}\}$
Show that the language of all strings over \{0, 1\} that do not contain a pair of 1’s that are separated by an odd number of 0’s is regular.
Example

Show that the language of all strings over \( \{0, 1\} \) that contain an even number of 0’s and 1’s is regular.
• **Deterministic** finite automata can only be in one state at any point in time.
  - Recall the definition of the transition function $\delta_D$.
• In contrast, **nondeterministic** finite automata (NFA’s) can be in several states at once!
  - The transition function $\delta_N$ is a one-to-many function.

![NFA Diagram]

- This NFA recognizes strings in $\{0,1\}^*$ containing a 1 in the third position from the end.
**Transition table**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>q₁</td>
<td>q₁,q₂</td>
</tr>
<tr>
<td>q₂</td>
<td>q₃</td>
<td>q₃</td>
</tr>
<tr>
<td>q₃</td>
<td>q₄</td>
<td>q₄</td>
</tr>
<tr>
<td>q₄*</td>
<td>q₄</td>
<td>q₄</td>
</tr>
</tbody>
</table>

- **q₁** is a **nondeterministic** state with a **one-many** transition on 1.

Intuitively, the NFA always **guesses right**.
Nondeterministic Finite Automata

Example
An NFA that accepts all strings of the form $0^k$ where $k$ is a multiple of 2 or 3.
An NFA is a 5-tuple \((Q, \Sigma, \delta_N, q_0, F)\) consisting of:

- A finite set of states \(Q\),
- A set of input alphabets \(\Sigma\),
- A transition function \(\delta_N : Q \times \Sigma_{\epsilon} \rightarrow \mathcal{P}(Q)\),
- A start state \(q_0\), and
- A set of accept states \(F \subseteq Q\).

Here, \(\Sigma_{\epsilon}\) denotes the set \(\Sigma \cup \{\epsilon\}\).

\(\mathcal{P}(Q)\) denotes the power set of \(Q\).
Transition function of an NFA

- \( \delta_N(q,a) \) is a set of states.
- Extend to strings as follows:
  - **Basis:** \( \delta_N(q,\varepsilon) = q \)
  - **Induction:** \( \delta_N(q,wa) = \) the union over all states of \( \delta_N(p,a) \), where \( p \in \delta_N(q,w) \).
- A string is accepted by an NFA if \( \delta_N(q_0,w) \) contains at least one state \( p \subseteq F \).
- The language of an NFA is the set of strings it accepts.
Every DFA is also an NFA by definition.
- There is simply no nondeterminism.

Surprisingly, for every NFA, there is also an equivalent DFA!
- Two equivalent machines recognize the same language.
- Nonintuitive, as we’d expect NFA’s to be more powerful.
- Useful, as describing an NFA is much simpler.

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.
DFA for recognizing strings with a 1 in the third last position.
Subset construction

- Given an NFA $(Q, \Sigma, \delta_N, q_0, F)$, construct equivalent DFA with:
  - States $\mathcal{P}(Q)$ (set of subsets of $Q$).
  - Inputs $\Sigma$.
  - Start state $\{q_0\}$.
  - Final states = all those with a member of $F$.

Note:
- 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.
• **Example:** We’ll construct the DFA for the following NFA which recognizes strings in \( \{0,1\}^* \) containing a 1 in the second position from the end.
Subset construction
Proof of equivalence: subset construction

- Show by induction on length of $w$ that
  $\delta_N(q_0,w) = \delta_D(\{q_0\},w)$

- **Basis:** $w = \varepsilon$. $\delta_N(q_0,\varepsilon) = \delta_D(\{q_0\},\varepsilon) = \{q_0\}$.

- **Inductive step:** Assume $\text{IH}$ is true for all strings shorter than $w$. Let $w = xa$, then $\text{IH}$ is true for $x$.
  - Let $\delta_N(q_0,x) = \delta_D(\{q_0\},x) = S$.
  - Let $T = \text{the union over all states } p \text{ in } S \text{ of } \delta_N(p,a)$.
  - Then $\delta_N(q_0,w) = \delta_D(\{q_0\},w) = T$ (by definition).
NFA’s with $\varepsilon$-transitions

- State-to-state transitions on $\varepsilon$ input.
- These transitions are *spontaneous*, and do not consider the input string.
• $\text{CL}(q) =$ set of states that can be reached from state $q$ following only arcs labeled $\varepsilon$.

- $\text{CL}(A) = \{A\}$, $\text{CL}(E) = \{B, C, D, E\}$.
- Closure of set of states = union of closure of each state.
Extended transition function

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

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

- Every NFA is an \(\varepsilon\)-NFA.
  - It just has no \(\varepsilon\)-transitions.
- Converse requires us to take an \(\varepsilon\)-NFA and construct an NFA that accepts the same language.
- This is done by combining \(\varepsilon\)-transitions with the next transition on a real input.
- Start with an \(\varepsilon\)-NFA \((Q,\Sigma,q_0,F,\delta_E)\) and construct an ordinary NFA \((Q,\Sigma,q_0,F',\delta_N)\).
Equivalence of NFA, $\varepsilon$-NFA

- Compute $\delta_N(q,a)$ as follows:
  - Let $S = \text{CL}(q)$.
  - $\delta_N(q,a)$ is the union over all $p$ in $S$ of $\delta_E(p,a)$.
- $F' = \text{set of states } q \text{ such that } \text{CL}(q) \text{ contains a state of } F$.
- **Intuition:** $\delta_N$ incorporates $\varepsilon$-transitions before using $a$.
- Proof of equivalence is by induction on $|w|$ that $\text{CL}(\delta_N(q_0,w)) = \delta_E(q_0,w)$.
- **Basis:** $\text{CL}(\delta_N(q_0,\varepsilon)) = \text{CL}(q_0) = \delta_E(q_0,\varepsilon)$.
- **Inductive step:** Assume $\text{IH}$ is true for all $x$ shorter than $w$. Let $w = xa$.
  - Then $\text{CL}(\delta_N(q_0,xa)) = \text{CL}(\delta_E(\text{CL}(\delta_N(q_0,x)),a))$ (by definition).
  - But from $\text{IH}$, $\text{CL}(\delta_N(q_0,x)) = \delta_E(q_0,x)$.
  - Hence, $\text{CL}(\delta_N(q_0,w)) = \text{CL}(\delta_E(\delta_E(q_0,x),a)) = \delta_E(q_0,w)$. 
Example

Diagram of a finite automaton with states A, B, C, D, E, and F, labeled with transitions for symbols 0 and 1, as well as ε transitions.
Summary

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