Non-deterministic finite automata with ε transitions

Languages

- Recall.
  - What is a language?
  - What is a class of languages?

Finite Automata

- Consists of
  - A set of states (Q)
  - A start state (q₀)
  - A set of accepting states (F)
  - Read symbols (Σ)
  - Transition function (δ)

- Let’s recap

First there was the DFA

- Deterministic Finite Automata
  - For every state and every alphabet symbol there is exactly one move that the machine can make.
  - δ : Q x Σ → Q
  - δ is a total function: completely defined. I.e. it is defined for all q ∈ Q and a ∈ Σ

Then, the NFA

- Non-determinism
  - When machine is in a given state and reads a symbol, the machine will have a choice of where to move to next.
  - There may be states where, after reading a given symbol, the machine has nowhere to go.
  - Applying the transition function will give, not 1 state, but 0 or more states.

Non-Deterministic Finite Automata (NFA)

- Transition function
  - δ is a function from Q x Σ to 2^Q
  - δ (q, a) = subset of Q (possibly empty)
• And now…
• Introducing…
• The newest in the FA family…
• The Non deterministic finite automata with ε transitions (ε-NFA)

Nondeterministic Finite Automata with ε transitions (ε-NFA)

• For both DFAs and NFAs, you must read a symbol in order for the machine to make a move.
• In Nondeterministic Finite Automata with ε transitions (ε-NFA)
  – Can make move without reading a symbol off the read tape
  – Such a move is called a ε transition

Nondeterministic Finite Automata with ε transitions (ε-NFA)

• Example:
  – Machine to accept decimal numbers

Nondeterministic Finite Automata with ε transitions (ε-NFA)

• How does such a machine accept?
  – A string will be accepted if there is at least one sequence of state transitions on an input (including ε transitions) that leaves the machine in an accepting state.

Nondeterministic Finite Automata with ε transitions (ε-NFA)

• Example:
  – -3.45 is accepted
  – .5678
  – 37 is rejected

Nondeterministic Finite Automata with ε transitions (ε-NFA)

• A Non-Deterministic Finite Automata with ε transitions is a 5-tuple \((Q, \Sigma, q_o, \delta, F)\) where
  – \(Q\) is a finite set (of states)
  – \(\Sigma\) is a finite alphabet of symbols
  – \(q_o \in Q\) is the start state
  – \(F \subseteq Q\) is the set of accepting states
  – \(\delta\) is a function from \(Q \times (\Sigma \cup \{\varepsilon\})\) to \(2^Q\) (transition function)
Nondeterministic Finite Automata with ε transitions (ε-NFA)

- Transition function
  - \( \delta \) is a function from \( Q \times (\Sigma \cup \{\varepsilon\}) \) to \( 2^Q \)
  - \( \delta(q, a) = \) subset of \( Q \) (possibly empty)
  - In our example
    - \( \delta(q_1, 0) = \{q_1, q_4\} \)
    - \( \delta(q_1, .) = \{q_1\} \)
    - \( \delta(q_1, +) = \emptyset \)
    - \( \delta(q_0, \varepsilon) = \{q_1\} \)

ε-Closure

- ε closure
  - Before defining the transition function on a string (\( \delta(q, x) \)), it is useful to first define what is known as the ε closure.
  - Given a set of states \( S \), the ε closure will give the set of states reachable from each state in \( S \) using only ε transitions.

ε-Closure : Recursive definition

- Let \( M = (Q, \Sigma, q_0, \delta, F) \) be an ε-NFA
- Let \( S \) be a subset of \( Q \)
- The ε closure, denotes \( \text{ECLOSE}(S) \) is defined:
  - For each state \( p \in S \), \( p \in \text{ECLOSE}(S) \)
  - For any \( q \in \text{ECLOSE}(S) \), every element of \( \delta(q, \varepsilon) \in \text{ECLOSE}(S) \)
  - No other elements of \( Q \) are in \( \text{ECLOSE}(S) \)

ε-Closure : Algorithm

- Since we know that \( \text{ECLOSE}(S) \) is finite, we can convert the recursive definition to an algorithm.
- To find \( \text{ECLOSE}(S) \) where \( S \) is a subset of \( Q \)
  - Let \( T = S \)
  - While (\( T \) does not change) do
    - Add all elements of \( \delta(q, \varepsilon) \) where \( q \in T \)
    - \( \text{ECLOSE}(S) = T \)

ε-Closure : Example

- Example
**ε-Closure**

- ε closure: Example
  - Find $\text{ECLOSE}\{\{s\}\}$ in our example
  - $T = \{s\}$ initial step
  - $T = \{s, w\}$ add $\delta(s, \varepsilon)$
  - $T = \{s, w, q_0\}$ add $\delta(w, \varepsilon)$
  - $T = \{s, w, q_0, p, t\}$ add $\delta(q_0, \varepsilon)$
  - $\delta(w, \varepsilon) = \delta(w, \varepsilon) = \emptyset$
  - We are done,
    - $\text{ECLOSE}\{\{s\}\} = T = \{s, w, q_0, p, t\}$

**Nondeterministic Finite Automata with ε transitions (ε-NFA)**

- Now let's define $\delta$
  1. For any $q \in Q$, $\delta(q, \varepsilon) = \text{ECLOSE}\{\{q\}\}$
  2. For any $y \in \Sigma$, $a \in \Sigma$, $q \in Q$

$$\hat{\delta}(q, y) = \text{ECLOSE}\left( \bigcup_{p \in \text{ECLOSE}(q, y)} \delta(p, a) \right)$$

Set of all states obtained by applying $\delta$ to all states in $\delta^*(q, y)$ and input $a$ and taking the ε closure of the result

**Nondeterministic Finite Automata with ε transitions (ε-NFA)**

- Accepting a string
  - A string $x$ is accepted if running the machine on input $x$, considering all paths, including the use of ε transitions, puts the machine into one of the accepting states
  - Formally:
    - $x \in \Sigma^*$ is accepted by $M$ if
    - $\hat{\delta}(q_0, x) \cap F \neq \emptyset$

**Nondeterministic Finite Automata with ε transitions (ε-NFA)**

- I bet that you’re asking…
  - Can JFLAP handle ε-NFA?
  - Well, let’s check and see!

**Nondeterministic Finite Automata with ε transitions (ε-NFA)**

- Language accepted by $M$
  - The language accepted by $M$
    - $L(M) = \{ x \in \Sigma^* | x$ is accepted by $M \}$

- If $L$ is a language over $\Sigma$, $L$ is accepted by $M$ iff $L = L(M)$.
  - For all $x \in L$, $x$ is accepted by $M$.
  - For all $x \notin L$, $x$ is rejected by $M$. 
Nondeterministic Finite Automata with ε transitions (ε-NFA)

- Why they’re a good idea
  - Given a regular expression, it is far easier to create an ε-NFA for the language described by the expression than it is to create a plain old DFA.
  - It will also be essential when showing the FAs accept the class of Regular Languages.
  - Questions?

DFA / NFA / ε-NFA Equivalence

- Surprisingly enough
  - ε transitions to our NDFA does NOT give it any additional language accepting power.
  - DFAs and NFAs and ε-NFA are all equivalent
    - Every language that can be accepted by a ε-NFA can also be accepted by an DFA which can also be accepted by a NFA.
  - Let’s show this

ε-NFA -> DFA

- Given ε-NFA find DFA
  - Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be a ε-NFA then
    - There exists a DFA, $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$
    - Such that $L(E) = L(D)$

ε-NFA -> DFA

- Formal definition
  - $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be a ε-NFA
  - We define DFA, $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$
    - $Q_D = 2^{Q_E}$
    - $q_D = E\text{CLOSE}(q_0)$
    - $F_D = \text{sets containing at least one state from } F_E$
\[
\begin{array}{c}
\text{ε-NFA} \rightarrow \text{DFA} \\
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{State} & \text{ε closure} \\
\hline
q_0 & \{q_0, q_1\} \\
q_1 & \{q_1\} \\
q_2 & \{q_2\} \\
q_3 & \{q_3, q_5\} \\
q_4 & \{q_4\} \\
q_5 & \{q_5\} \\
\hline
\end{array}
\]

\[
\begin{array}{c}
\text{ε-NFA} \rightarrow \text{DFA} \\
\end{array}
\]

- Now we must show that D accepts the same language as E
  - Can be shown (using induction) that for all \( x \in \Sigma^* \)
    - \( \delta_D(q, x) = \delta_E(q, x) \)
  - See Theorem 2.22

\[
\begin{array}{c}
\text{ε-NFA} \rightarrow \text{DFA} \\
\end{array}
\]

- The other direction is fairly straightforward.
  - For any DFA, there is an equivalent NFA
  - An NFA is nothing more than a ε-NFA with no ε transitions. Thus
    - \( \delta(q, \varepsilon) \) for all states \( q = \emptyset \)

\[
\begin{array}{c}
\text{DFA} \rightarrow \text{ε-NFA} \\
\end{array}
\]

- Show that D and E recognize the same language
  - \( x \) is accepted by E iff \( \delta_E(q_0, x) \cap F_E \neq \emptyset \)
  - \( x \) is accepted by D iff \( \delta_D(q_0, x) \cap F_D \neq \emptyset \)
  - Thus,
    - \( x \) is accepted by D iff \( x \) is accepted by E

- Questions?
What have we shown

• For every DFA, there is an NFA that accepts the same language and visa versa
• For every DFA, there is a $\varepsilon$-NFA that accepts the same language, and visa versa
• Thus, for every NFA there is a $\varepsilon$-NFA that accepts the same language, and visa versa
• DFAs, NFAs, and $\varepsilon$-NFAs are equivalent!

Questions?

• Let’s take a break.