Non deterministic finite automata

Deterministic Finite Automata

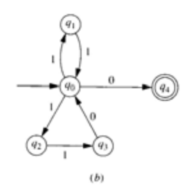
- Automata we've been dealing with have been deterministic
  - For every state and every alphabet symbol there is exactly one move that the machine can make.
  - $\delta : Q \times \Sigma \rightarrow Q$
  - δ is a total function: completely defined. I.e. it is defined for all $q \in Q$ and $a \in \Sigma$

Non-Deterministic Finite Automata (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)

- Example: L corresponds to the regular expression $\{11 \cup 110\}^*0$



Non-Deterministic Finite Automata (NFA)

- How does such a machine accept?
  - A string will be accepted if there is <u>at least one</u> sequence of state transitions on an input that leaves the machine in an accepting state.
  - Such a machine is called a <u>non-deterministic finite automata</u> (NFA)

Non-Deterministic Finite Automata (NFA)

- A Non-Deterministic Finite Automata is a 5-tuple $(Q, \Sigma, \delta, q_o, F)$ where
  - Q is a finite set (of states)
  - Σ is a finite alphabet of symbols
  - $q_o \in Q$ is the start state
  - $F \subseteq Q$ is the set of final states
  - δ is a function from $Q \times \Sigma$ to 2^Q (transition function)

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)
  - In our example
    - $\delta (q_3, 0) = \{q_o\}$
    - $\delta (q_o, 1) = \{q_1, q_2\}$
    - $\delta (q_4, 1) = \varnothing$

Non-Deterministic Finite Automata (NFA)

- Transition function on a string x
  - $\hat{\delta}$ is a function from Q x Σ^* to 2^Q
  - $\hat{\delta}$ (q, x) = subset of Q (possibly empty)
  - Set of all states that the machine can be in, upon following all possible paths on input x.

Non-Deterministic Finite Automata (NFA)

- Recursive definition of $\hat{\delta}$
  1. For any q $\in$ Q $\hat{\delta}$ (q, ε) = {q}
  2. For any y $\in \Sigma^*$, a $\in \Sigma$, q $\in$ Q

$$\hat{\delta}(q, ya) = \bigcup_{p \in \delta^*(q,y)} \delta(p, a)$$

Set of all states obtained by applying δ to all states in $\hat{\delta}$(q,y) and input a.

Non-Deterministic Finite Automata (NFA)

- In our example:
  - $\hat{\delta}$ $(q_0, 110) = \hat{\delta}$ $(q_1, 10) \cup \hat{\delta}(q_2, 10)$
  - $= \hat{\delta}$ $(q_0, 0) \cup \hat{\delta}$ $(q_3, 0)$
  - $= \hat{\delta}$ $(q_4, \varepsilon) \cup \hat{\delta}(q_0, \varepsilon)$
  - $= \{q_4\} \cup \{q_0\}$
  - $= \{q_0, q_4\}$

Non-Deterministic Finite Automata (NFA)

- Definition of accepting
  - A string x is accepted if running the machine on input x, considering all paths, puts the machine into one of the final states
  - Formally:
    - x $\in \Sigma^*$ is accepted by A if
    - $\hat{\delta}$ $(q_0, x) \cap F \neq \varnothing$

Non-Deterministic Finite Automata (NFA)

- Once again, in our example
  - $\hat{\delta}$ $(q_0, 110) = \{q_0, q_4\}$
  - $F = \{q_4\}$
  - $\hat{\delta}$ $(q_0, 110) \cap F = \{q_4\} \neq \varnothing$
  - 110 is accepted by A

Non-Deterministic Finite Automata (NFA)

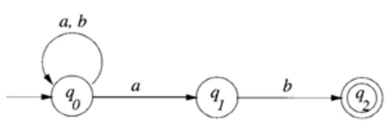
- Language accepted by A
  - The language accepted by A
    - $L(A) = \{ x \in \Sigma^* \mid x \text{ is accepted by } A \}$
- If L is a language over Σ, L is accepted by A iff $L = L(A)$.
  - For all $x \in L$, x is accepted by A.
  - For all $x \notin L$, x is rejected by A.

Non-Deterministic Finite Automata (NFA)

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

Non-Deterministic Finite Automata (NFA)

- Let's try another one:
  - L = set of strings ending in ab



  - Let's see how this fares with JFLAP

Reality Check

- Nondeterministic Finite Automata (NFA)
  - At each state,for each symbol, the machine can move into 0 or more states.
  - δ is a function from $Q \times \Sigma$ to 2^Q
  - A string is accepted if there is at least one sequence of moves on input x placing the machine into an accepting state.
  - Questions?

DFA / NFA Equivalence

- Surprisingly enough
  - Adding nondeterminism to our DFA does NOT give it any additional language accepting power.
  - DFAs and NFAs are equivalent
    - Every language that can be accepted by an NFA can also be accepted by a DFA and visa-versa

DFA / NFA Equivalence

- How we will show this
  1. Given an NFA that accepts L, create an DFA that also accepts L
  2. Given an DFA that accepts L, create an NFA that also accepts L

Are we ready?

NFA->DFA

- Given N FA find DFA
  - Let N = (Q_N, Σ, δ_N, q_0, F_N) be a NFA then
    - There exists a DFA, D = (Q_D, Σ, δ_D ,q_D, F_D)
    - Such that L(N) = L(D)

NFA -> DFA

- Basic idea
  - Recall that for a NFA, δ: Q x $\Sigma \rightarrow 2^Q$
  - Use the states of D to represent subsets of Q.
  - If there is one state of D for every subset of Q, then the non-determinism of N can be eliminated.
  - This technique, called <u>subset construction</u>, is a primary means for removing non-determinism from an NFA.

NFA -> DFA

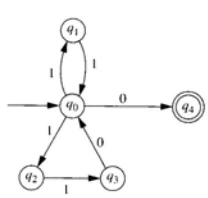
- Formal definition
  - N = (Q_N, Σ, δ_N, q_0, F_N) be a NFA
  - We define DFA, D = (Q_D, Σ, $\delta_{D,}$ q_D, F_D)
    - $Q_D = 2^Q$
    - $q_D = \{q_0\}$
    - For $q \in Q_D$ and $a \in \Sigma$,
    - $\delta_D(q,a) = \bigcup_{p \in q} \delta_N(p,a)$
    - $F_D = \{q \in Q_D \mid q \cap F_N \neq \varnothing\}$
  - Note that we need only include states on D (subsets of Q) if the state is reachable.

NFA -> DFA

- Algorithm for building D
  - Add $\{q_0\}$ to Q_D
  - While there are states of Q_D whose transitions are yet to be defined
    - Let $q \in Q_D$
    - For each $a \in \Sigma$, determine the set of states, P, in N that are reachable from q on input a
    - If there is no state in Q_D corresponding to P, add one.
    - Define δ_D(q, a) = state in Q_D corresponding to P
  - Define F_D as any state in Q_D that corresponds to a subset containing any of the final states of N

NFA -> DFA

- Example



(b)

NFA -> DFA

- Now we must show that D accepts the same language as N
  - It can be shown (by induction) that for all x $\in \Sigma^*$
    $\hat{\delta}_D$ (q_D, x) $\hat{\delta}_N$ (q_0, x)
    - Note that both of these are Sets of states from N
    - See Theorem 2.11 in Text

NFA -> DFA

- Show that D and N recognize the same language
  - x is accepted by D iff $\hat{\delta}_D(q_D, x) \in F_D$
  - F_D contains sets that contain any state in F_N
  - Thus
    - $\hat{\delta}_D(q_D, x) \in F_D$ iff $\hat{\delta}_N(q_N, x) \in F_N$
    - x is accepted by D iff x is accepted by N

What have we shown

- In Step 1 we've shown:
  - Given a NFA
    - There exists an DFA that accepts the same language
    - Non-determinism can be removed from an NFA by using a subset construction algorithm.
  - Questions?

Step 2: Given DFA find NFA

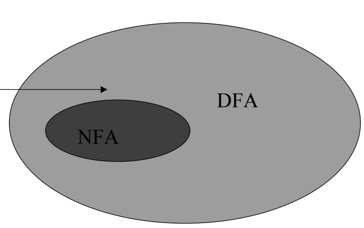
- Observe that a DFA can easily be converted to an equivalent NFA:
  - DFAs – all transitions lead to exactly one state
  - Define the transitions of the NFA to consists of sets of only 1 element.

What have we shown

- In Step 2 we've shown:
  - Given a DFA
    - There exists an NFA that accepts the same language

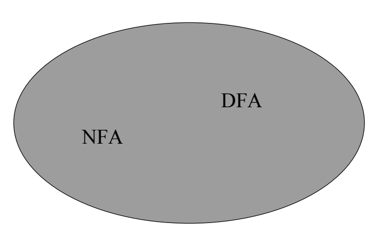
What have we shown

Is there something in here?



If L $\in$ NFA then L $\in$ DFA

Equivalence



If L $\in$ DFA then L $\in$ NFA

Summary

- Non-deterministic finite automata (NFA)
  - Machine now can “choose” it’s path.
  - Each transition takes you from a state to a set of states.

  - Equivalent in language recognition power to DFA.

  - Questions?