Regular Languages III

Non-determinism

Logistics

- Did every receive a test e-mail from me yesterday?

Homework #2

- After the lecture
  - Help session for Homework #2

Before We Start

- Any questions?

Plan for today

- Begin proof of Kleene Theorem

Languages

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

Regular Languages

- Regular languages
  - Means of defining: Regular Expressions
  - Machine for accepting: Finite Automata

Finite Automata

- A finite automaton (finite-state machine) is a 5-tuple $(Q, \Sigma, q_o, \delta, A)$ where
  - Q is a finite set (of states)
  - Σ is a finite alphabet of symbols
  - $q_o \in Q$ is the start state
  - $A \subseteq Q$ is the set of accepting states
  - δ is a function from Q x Σ to Q (transition function)

Computation Hall of Fame

- Steven Cole Kleene
  - 1909-1994
  - b. Hartford, Conn.
  - PhD – Princeton (1934)
  - Prof at U of Wisc at Madison (1935 – 1979)
  - Introduced Kleene Star op
  - Defined regular expressions

Kleene Theorem

- A language L over Σ is regular iff there exists an FA that accepts L.
  1. If L is regular there exists an FA M such that L = L(M)
  2. For any FA, M, L(M) is regular
     L(M), the language accepted by the FA can be expressed as a regular expression.

Proving Kleene Theorem

- Approach
  - Define 2 variants of the Finite Automata
    - Nondeterministic Finite Automata (NDFA)
    - Nondeterministic Finite Automata with Λ transitions (NDFA-Λ)
  - Prove that FA, NDFA, and NDFA-Λ are equivalent w.r.t. the languages they accept
  - For a regular expression, build a NDFA-Λ that accepts the same language
  - For an FA build a regular expression that describes the language accepted by the FA.

Tonight

- We will define these two FA variants:
  - Nondeterministic Finite Automata (NDFA)
  - Nondeterministic Finite Automata with Λ transitions (NDFA-Λ)

Deterministic Finite Automata

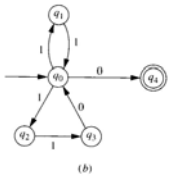
- Automata we've been dealing with have been deterministic
  - For every state and every alphabet symbol there is exactly one move that the machine can make.
  - $\delta : Q \times \Sigma \rightarrow Q$
  - δ is a total function: completely defined. I.e. it is defined for all $q \in Q$ and $a \in \Sigma$

Non-Deterministic Finite Automata (NDFA)

- 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 (NDFA)

- Example: L corresponds to the regular expression $(11 + 110)^*0$



Non-Deterministic Finite Automata (NDFA)

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

Non-Deterministic Finite Automata (NDFA)

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

Non-Deterministic Finite Automata (NDFA)

- Transition function
  - δ is a function from $Q \times \Sigma$ 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 (NDFA)

- Transition function on a string x
  - δ^* is a function from Q x Σ^* to 2^Q
  - δ^* (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 (NDFA)

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

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

Set of all states obtained by applying δ to all states in δ^* (q,y) and input a.

Non-Deterministic Finite Automata (NDFA)

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

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

Set of all states obtained by applying δ^* to all states in δ (q,a) and input y.

Non-Deterministic Finite Automata (NDFA)

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

Non-Deterministic Finite Automata (NDFA)

- 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 accepting states
  - Formally:
    - $x \in \Sigma^*$ is accepted by M if
    - $\delta^*(q_0, x) \cap A \neq \varnothing$

Non-Deterministic Finite Automata (NDFA)

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

Non-Deterministic Finite Automata (NDFA)

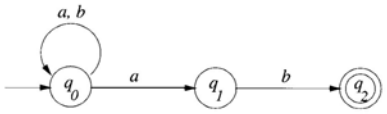
- Language accepted by M
  - The language accepted by M
    - L(M) = { x ∈ Σ^* | x is accepted by M }
- If L is a language over Σ, L is accepted by M iff L = L(M).
  - For all x ∈ L, x is accepted by M.
  - For all x ∉ L, x is rejected by M.

Non-Deterministic Finite Automata (NDFA)

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

Non-Deterministic Finite Automata (NDFA)

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



  - Let's see how this faires with JFLAP

Reality Check

- Nondeterministic Finite Automata (NDFA)
  - At each state,for each symbol, the machine can move into 0 or more states.
  - δ is a function from Q x Σ 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?

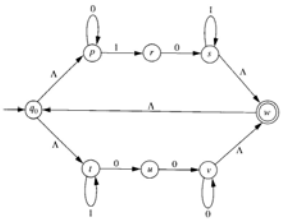
- Any questions
- Next:
  - Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- For both FAs and NDFAs, you must read a symbol in order for the machine to make a move.
- In Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)
  - Can make move without reading a symbol off the read tape
  - Such a move is called a Λ transition

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Example:
  - L corresponds to $(0^*101^* + 1^*000^*)^+$

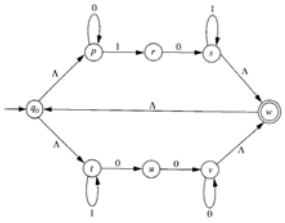


Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- 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 Λ transitions) that leaves the machine in an accepting state.

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Example:
  - 1100 is accepted
  - 1101 is rejected



Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

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

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Transition function
  - δ is a function from Q x $(\Sigma \cup \{\Lambda\})$ to 2^Q
  - δ (q, a) = subset of Q (possibly empty)
  - In our example
    - δ (p, 0) = {p}
    - δ (p, 1) = {r}
    - δ (r, 1) = ∅
    - δ (q_0, Λ) = {p, t}

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Transition function on a string
  - δ^* is a function from Q x Σ^* to 2^Q
  - δ^* (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.
  - We'll need to consider all paths that include the use of Λ transitions

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

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

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Λ closure: Recursive definition
  - Let M = (Q, Σ, q_o, δ, A) be a NDFA- Λ
  - Let S be a subset of Q
  - The Λ closure, denotes Λ(S) is defined:
    - For each state p ∈ S, p ∈ Λ(S)
    - For any q ∈ Λ(S), every element of δ(q, Λ) ∈ Λ(S)
    - No other elements of Q are in Λ(S)

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Λ closure: Algorithm
  - Since we know that Λ(S) is finite, we can convert the recursive definition to an algorithm.
  - To find Λ(S) where S is a subset of Q
  - Let T = S
  - While (T does not change) do
    - Add all elements of δ(q, Λ) where q ∈ T
  - Λ(S) = T

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Λ closure: Example
  - Find Λ({s}) in our example
  - T = {s} initial step
  - T = {s, w} add δ(s, Λ)
  - T = {s, w, q_0} add δ(w, Λ)
  - T = {s, w, q_0, p,t} add δ(q_0, Λ)
  - δ(w, Λ) = δ(w, Λ) =∅
  - We are done,
    - Λ({s}) = T = {s, w, q_0, p,t}

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Now lets define δ^*
  1. For any q ∈Q, δ^* (q, Λ) = Λ({q})
  2. For any y ∈ Σ^*, a ∈ Σ, q ∈Q

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

Set of all states obtained by applying δ to all states in δ^* (q,y) and input a and taking the Λ closure of the result

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Example: find δ^* (q_0, 101) $\delta^*(q, ya) = \Lambda\left(\bigcup_{p \in \delta^*(q,y)} \delta(p,a)\right)$
  - δ^* (q_0, 1) = δ^* (q_0, Λ 1) δ^*(q_0 Λ) = Λ(q_0)
  - = Λ (δ (q_0, 1) ∪ δ (p, 1) ∪ δ (t, 1))
  - = Λ (∅ ∪ {r} ∪ {t})
  - = {r, t}
  - δ^* (q_0, 10) = δ^* (q_0, 10)
  - = Λ (δ (r,0) ∪ δ (t,0))
  - = Λ ({s} ∪ {u})
  - = {s, u, w, q_0, p, t}

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Example: find $\delta^*(q_0, 101)$
  - $\delta^*(q_0, 101) = \delta^*(q_0, 10\ 1)$
  - $= \Lambda(\delta(s,1) \cup \delta(u,1) \cup \delta(w,1) \cup \delta(q_0,1) \cup \delta(p,1) \cup \delta(t,1))$
  - $= \Lambda(\{s\} \cup \varnothing \cup \varnothing \cup \varnothing \cup \{r\} \cup \{t\})$
  - $= \Lambda(\{s,r,t\})$
  - $= \{s, w, q_0, p, t, r\}$

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

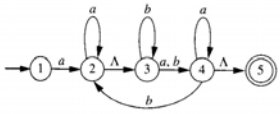
- Accepting a string
  - A string x is accepted if running the machine on input x, considering all paths, including the use of Λ transitions, puts the machine into one of the accepting states
  - Formally:
    - $x \in \Sigma^*$ is accepted by M if
    - $\delta^*(q_0, x) \cap A \neq \varnothing$

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Accepting a string: example
  - In our example
    - $A = \{w\}$
    - $\delta^*(q_0, 101) = \{s, w, q_0, p, t, r\}$
    - $A \cap \delta^*(q_0, 101) = \{w\}$
  - 101 is accepted

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Are the following strings accepted by the NDFA- Λ below:
  - aba
  - abab
  - aaabbb



Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

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

Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

- Language accepted by M
  - The language accepted by M
    - $L(M) = \{ x \in \Sigma^* \mid x \text{ is accepted by M} \}$
- If L is a language over Σ, 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 Λ transitions (NDFA- Λ)

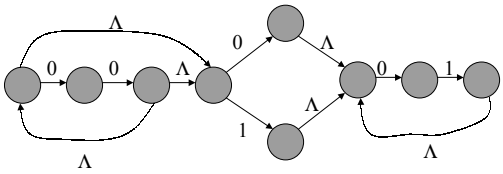
- Why they're a good idea
  - Given a regular expression, it is far easier to create an NDFA- Λ for the language described by the expression than it is to create a plain old FA.

RE -> NDFA- Λ

- Intuitive notions
  - a union in an RE implies a branch in an NDFA- Λ
  - Concatenation in an RE implies a move from one state to the next
  - Concatenation in an RE implies a loop in the NDFA- Λ .

RE -> NDFA- Λ

- Example:
  - $(00)^*(1 + 0)(01)^*$



NDFA- Λ Summary

- Nondeterministic Finite Automata with Λ transitions (NDFA-Λ)
  - Machine can move without reading a symbol.
  - δ is a function from Q x $\{\Sigma \cup \{\Lambda\}\}$ to 2^Q
  - Λ closure
  - A string is accepted if there is at least one sequence of moves on input x, including Λ transitions, placing the machine into an accepting state.

Lecture Summary

- Two variants of the FA
  - Nondeterministic Finite Automata (NDFA)
  - Nondeterministic Finite Automata with Λ transitions (NDFA- Λ)

Next Time

- Though it may seem counter-intuitive, each of the FAs we've looked are equivalent w.r.t the languages they accept.
- Next time, we will prove this.

Questions?

- Remember homework is due next class