CS21
Decidability and Tractability

Lecture 3

January 9, 2015

Outline

- NFA, FA equivalence
- Regular Expressions
- FA and Regular Expressions
- Pumping Lemma

NFA formal definition

A nondeterministic FA

(Q, Σ, δ

– Q is a finite set called

– Σ is a finite set called the alphabet

– $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \wp(Q)$ is a function called the transition function

– q_0 is an element of Q called the start state

– F is a subset of Q called the accept states

transit
labeled
alpha
symbols or ε

"powerset of Q": the set of all subsets of Q

Formal description of NFA operation

NFA M = $(Q, \Sigma, \delta, q_0, F)$

accepts a string $w = w_1w_2w_3\ldots w_n \in \Sigma^*$

if w can be written (by inserting ε's) as:

$$y = y_1y_2y_3\ldots y_m \in (\Sigma \cup \{\varepsilon\})^*$$

and ∃ sequence $r_0, r_1, \ldots, r_m$ of states for which

– $r_0 = q_0$

– $r_{i+1} \in \delta(r_i, y_{i+1})$ for i = 0,1,2, …, m-1

– $r_m \in F$

NFA, FA equivalence

Theorem: a language L is recognized by a FA if and only if L is recognized by a NFA.

Must prove *two* directions:

(⇒) L is recognized by a FA implies L is recognized by a NFA.

(⇐) L is recognized by a NFA implies L is recognized by a FA.

(usually one is easy, the other more difficult)

NFA, FA equivalence

(⇒) L is recognized by a FA implies L is recognized by a NFA

Proof: a finite automaton *is* a nondeterministic finite automaton that happens to have no ε-transitions, and for which each state has exactly one outgoing transition for each symbol.

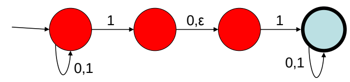
NFA, FA equivalence

(⇐) L is recognized by a NFA implies L is recognized by a FA.

<u>Proof</u>: we will build a FA that *simulates* the NFA (and thus recognizes the same language).

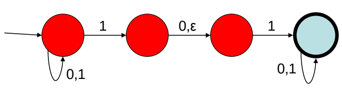
- alphabet will be the same
- what are the states of the FA?

NFA, FA equivalence



- given NFA $M = (Q, \Sigma, \delta, q_0, F)$
- construct FA $M' = (Q', \Sigma', \delta', q_0', F')$
- same alphabet: $\Sigma' = \Sigma$
- states are subsets of M's states: $Q' = \wp(Q)$
- if we are in state $R \in Q'$ and we read symbol $a \in \Sigma'$, what is the new state?

NFA, FA equivalence



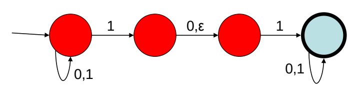
- given NFA $M = (Q, \Sigma, \delta, q_0, F)$
- construct FA $M' = (Q', \Sigma', \delta', q_0', F')$

Helpful def'n: $E(S) = \{q \in Q : q$ reachable from S by traveling along 0 or more ε-transitions$\}$

- new transition fn: $\delta'(R, a) = \cup_{r \in R} E(\delta(r, a))$

= "all nodes reachable from R by following an a-transition, and then 0 or more ε-transitions"

NFA, FA equivalence



- given NFA $M = (Q, \Sigma, \delta, q_0, F)$
- construct FA $M' = (Q', \Sigma', \delta', q_0', F')$
- new start state: $q_0' = E(\{q_0\})$
- new accept states:

$F' = \{R \in Q' : R$ contains an accept state of $M\}$

NFA, FA equivalence

- We have proved (⇐) by construction.

Formally we should also prove that the construction works, by induction on the number of steps of the computation.

- at each step, the state of the FA M' is exactly the set of reachable states of the NFA M…

So far…

<u>Theorem</u>: the set of languages recognized by NFA is closed under union, concatenation, and star.

<u>Theorem</u>: a language L is recognized by a FA if and only if L is recognized by a NFA.

<u>Theorem</u>: the set of languages recognized by FA is closed under union, concatenation, and star.

Next…

- Describe the set of languages that can be built up from:
  - unions
  - concatenations
  - star operations
- Called "patterns" or regular expressions
- **Theorem**: a language L is recognized by a FA if and only if L is described by a regular expression.

Regular expressions

- R is a regular expression if R is
  - a, for some $a \in \Sigma$
  - ε, the empty string
  - $\emptyset$, the empty set
  - $(R_1 \cup R_2)$, where R_1 and R_2 are reg. exprs.
  - $(R_1 \circ R_2)$, where R_1 and R_2 are reg. exprs.
  - $(R_1{}^*)$, where R_1 is a regular expression

A reg. expression R describes the language L(R).

Regular expressions

- example: $R = (0 \cup 1)$
  - if $\Sigma = \{0,1\}$ then use "Σ" as shorthand for R
- example: $R = 0 \circ \Sigma^*$
  - shorthand: omit " $\circ$ " $R = 0\Sigma^*$
  - precedence: $*$, then $\circ$, then $\cup$, unless override by parentheses
  - in example $R = 0(\Sigma^*)$, not $R = (0\Sigma)^*$

Some examples

alphabet $\Sigma = \{0,1\}$

- {w : w has at least one 1}
  $= \Sigma^*1\Sigma^*$
- {w : w starts and ends with same symbol}
  $= 0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$
- $\{w : |w| \leq 5\}$
  $= (\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)$
- {w : every 3rd position of w is 1}
  $= (1\Sigma\Sigma)^*(\varepsilon \cup 1 \cup 1\Sigma)$

Manipulating regular expressions

- The empty set and the empty string:
  - $R \cup \emptyset = R$
  - $R\varepsilon = \varepsilon R = R$
  - $R\emptyset = \emptyset R = \emptyset$
  - $\cup$ and $\circ$ behave like +, x; $\emptyset$, ε behave like 0,1
- additional identities:
  - $R \cup R = R$ (here + and $\cup$ differ)
  - $(R_1{}^*R_2)^*R_1{}^* = (R_1 \cup R_2)^*$
  - $R_1(R_2R_1)^* = (R_1R_2)^*R_1$

Regular expressions and FA

- **Theorem**: a language L is recognized by a FA if and only if L is described by a regular expression.

Must prove *two* directions:

($\Rightarrow$) L is recognized by a FA implies L is described by a regular expression

($\Leftarrow$) L is described by a regular expression implies L is recognized by a FA.

Regular expressions and FA

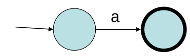
(⇐) L is described by a regular expression implies L is recognized by a FA

Proof: given regular expression R we will build a NFA that recognizes L(R).

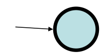
then NFA, FA equivalence implies a FA for L(R).

Regular expressions and FA

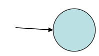
- R is a regular expression if R is
  - a, for some $a \in \Sigma$



  - ε, the empty string

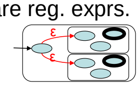


  - Ø, the empty set

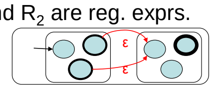


Regular expressions and FA

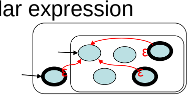
– $(R_1 \cup R_2)$, where R_1 and R_2 are reg. exprs.



– $(R_1 \circ R_2)$, where R_1 and R_2 are reg. exprs.



– $(R_1{}^*)$, where R_1 is a regular expression



Regular expressions and FA

(⇒) L is recognized by a FA implies L is described by a regular expression

Proof: given FA M that recognizes L, we will

1. build an equivalent machine "Generalized Nondeterministic Finite Automaton" (GNFA)
2. convert the GNFA into a regular expression

Regular expressions and FA

- GNFA definition:
  - it is a NFA, but may have regular expressions labeling its transitions
  - GNFA accepts string $w \in \Sigma^*$ if can be written

$$w = w_1w_2w_3\ldots w_k$$

where each $w_i \in \Sigma^*$, and there is a path from the start state to an accept state in which the i^{th} transition traversed is labeled with R for which $w_i \in L(R)$

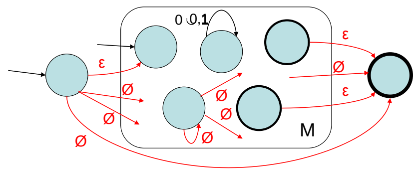
Regular expressions and FA

- Recall step 1: build an equivalent GNFA

- Our FA M *is* a GNFA.
- We will require "normal form" for GNFA to make the proof easier:
  - *single* accept state q_{accept} that has all possible incoming arrows
  - every state has all possible outgoing arrows; exception: start state q_0 has no self-loop

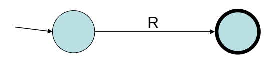
Regular expressions and FA

- converting our FA M into GNFA in normal form:



Regular expressions and FA

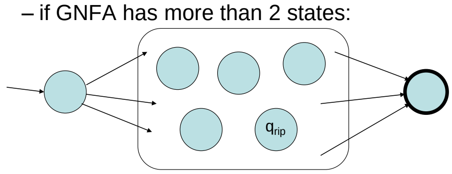
- On to step 2: convert the GNFA into a regular expression
  - if normal-form GNFA has two states:



the regular expression R labeling the single transition describes the language recognized by the GNFA

Regular expressions and FA

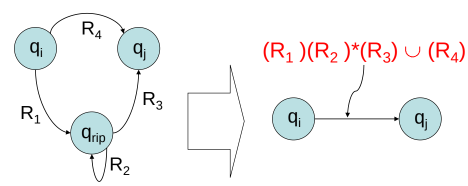
- if GNFA has more than 2 states:



- select one "q_{rip}"; delete it; repair transitions so that machine still recognizes same language.
- repeat until only 2 states.

Regular expressions and FA

- how to repair the transitions:
- for *every* pair of states q_i and q_j do



Regular expressions and FA

- summary:

FA M → k-state GNFA → (k-1)-state GNFA → (k-2)-state GNFA →…→ 2-state GNFA → R

- want to *prove* that this procedure is correct, i.e. L(R) = language recognized by M

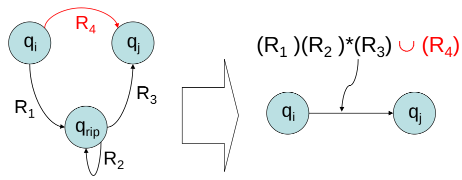
  - FA M equivalent to k-state GNFA 
  - i-state GNFA equivalent to (i-1)-state GNFA (we will prove…)
  - 2-state GFNA equivalent to R 

Regular expressions and FA

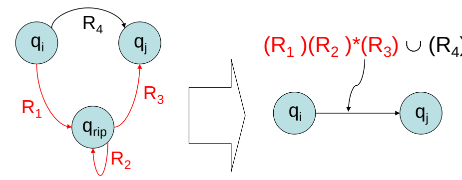
- **Claim**: i-state GNFA G equivalent to (i-1)-state GNFA G' (obtained by removing q_{rip})
- **Proof**:
  - if G accepts string w, then it does so by entering states: $q_0, q_1, q_2, q_3, \ldots, q_{accept}$
  - if none are q_{rip} then G' accepts w (see slide)
  - else, break state sequence into runs of q_{rip}:

    $q_0q_1\ldots q_iq_{rip}q_{rip}\ldots q_{rip}q_j\ldots q_{accept}$
  - transition from q_i to q_j in G' allows all strings taking G from q_i to q_j using q_{rip} (see slide)
  - thus G' accepts w

Regular expressions and FA



Regular expressions and FA



Regular expressions and FA

- **Proof** (continued):
  - if G' accepts string w, then every transition from q_i to q_j traversed in G' corresponds to

    either

    a transition from q_i to q_j in G

    or

    transitions from q_i to q_j via q_{rip} in G
  - In both cases G accepts w.
  - Conclude: G and G' recognize the same language.

Regular expressions and FA

- **Theorem**: a language L is recognized by a FA iff L is described by a regular expr.
- Languages recognized by a FA are called regular languages.
- Rephrasing what we know so far:
  - regular languages closed under 3 operations
  - NFA recognize exactly the regular languages
  - regular expressions describe exactly the regular languages