CS21
Decidability and Tractability

Lecture 2

January 8, 2014

Outline

- Finite Automata
- Nondeterministic Finite Automata
- Closure under regular operations
- NFA, FA equivalence

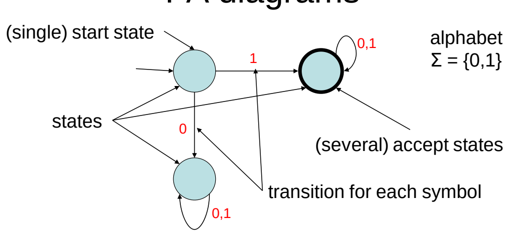
Terminology

- finite alphabet Σ : a set of symbols
- language $L \subseteq \Sigma^*$: subset of strings over Σ
- a machine takes an input string and either
  - accepts, rejects, or
  - loops forever
- a machine recognizes the set of strings that lead to accept
- a machine decides a language L if it accepts $x \in L$ and rejects $x \notin L$

Finite Automata

- simple model of computation
- reads input from left to right, one symbol at a time
- maintains state: information about what seen so far ("memory")
  - finite automaton has finite # of states: cannot remember more things for longer inputs
- 2 ways to describe: by diagram, or formally

FA diagrams



- read input one symbol at a time; follow arrows; accept if end in accept state

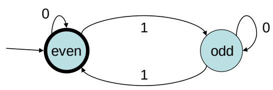
FA formal definition

A finite automaton is a 5-tuple

$(Q, \Sigma, \delta, q_0, F)$

- Q is a finite set called the states
- Σ is a finite set called the alphabet
- $\delta: Q \times \Sigma \to Q$ 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

FA formal definition



- Specification of this FA in formal terms:
  - $Q = \{\text{even}, \text{odd}\}$
  - $\Sigma = \{0,1\}$
  - $q_0 = \text{even}$
  - $F = \{\text{even}\}$

function δ:

δ(even, 0) = even
δ(even, 1) = odd
δ(odd, 0) = odd
δ(odd, 1) = even

Formal description of FA operation

finite automaton

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

accepts a string

$$w = w_1 w_2 w_3 \ldots w_n \in \Sigma^*$$

if ∃ sequence $r_0, r_1, r_2, \ldots, r_n$ of states for which

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0,1,2, \ldots, n-1$
- $r_n \in F$

What now?

- We have a model of computation
  (Maybe this is it. Maybe everything we can do with real computers we can do with FA…)
- try to characterize the languages FAs can recognize
  - investigate closure under certain operations
- show that some languages not of this type

Characterizing FA languages

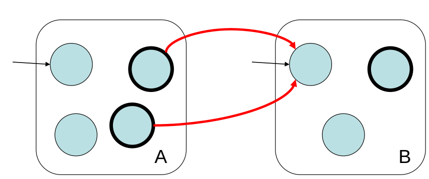
- We will show that the set of languages recognized by FA is closed under:
  - union "$C = (A \cup B)$"
  - concatenation "$C = (A \circ B)$"
  - star "$C = A^*$"
- Meaning: if A and B are languages recognized by a FA, then C is a language recognized by a FA

Characterizing FA languages

- union "$C = (A \cup B)$"
  $(A \cup B) = \{x : x \in A \text{ or } x \in B \text{ or both}\}$
- concatenation "$C = (A \circ B)$"
  $(A \circ B) = \{xy : x \in A \text{ and } y \in B\}$
- star "$C = A^*$" (note: ε always in A^*)
  $A^* = \{x_1 x_2 x_3 \ldots x_k : k \geq 0 \text{ and each } x_i \in A\}$

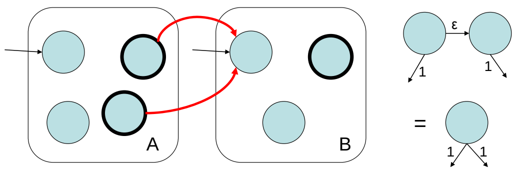
Concatenation attempt

$(A \circ B) = \{xy : x \in A \text{ and } y \in B\}$



What label do we put on the new transitions?

Concatenation attempt

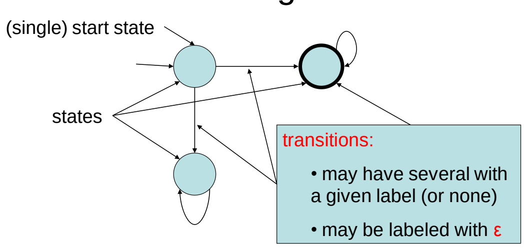


- Need it to happen "for free": label with ε (?)
- allows construct with multiple transitions with the same label (!?)

Nondeterministic FA

- We will make life easier by describing an additional feature (nondeterminism) that helps us to "program" FAs
- We will prove that FAs with this new feature can be simulated by ordinary FA
  - same spirit as programming constructs like procedures
- The concept of nondeterminism has a significant role in TCS and this course.

NFA diagrams

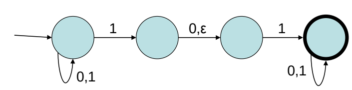


- At each step, several choices for next state

NFA operation

- Example of NFA operation:

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



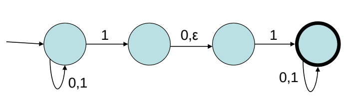
input: 0 1 0

not accepted

NFA operation

- Example of NFA operation:

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



input: 1 1 0

accepted

NFA operation

- One way to think of NFA operation:
- string $x = x_1x_2x_3\ldots x_n$ accepted if and only if
  - there exists a way of inserting ε's into x

    $x_1\varepsilon\varepsilon x_2\, x_3\ldots\varepsilon x_n$
  - so that there exists a path of transitions from the start state to an accept state

NFA formal definition

A nondeterministic FA

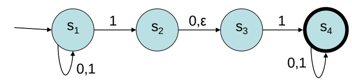
$(Q, \Sigma, \delta$

transit
labeled
alpha
symbols or ε

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

- 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

NFA formal definition



- Specification of this NFA in formal terms:
  - $Q = \{s_1, s_2, s_3, s_4\}$
  - $\Sigma = \{0,1\}$
  - $q_0 = s_1$
  - $F = \{s_4\}$

$\delta(s_1, 0) = \{s_1\}$
$\delta(s_1, 1) = \{s_1, s_2\}$
$\delta(s_1, \varepsilon) = \{\ \}$
$\delta(s_2, 0) = \{s_3\}$
$\delta(s_2, 1) = \{\ \}$
$\delta(s_2, \varepsilon) = \{s_3\}$

$\delta(s_3, 0) = \{\ \}$
$\delta(s_3, 1) = \{s_4\}$
$\delta(s_3, \varepsilon) = \{\ \}$
$\delta(s_4, 0) = \{s_4\}$
$\delta(s_4, 1) = \{s_4\}$
$\delta(s_4, \varepsilon) = \{\ \}$

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

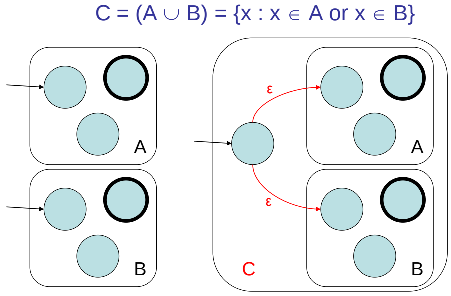
- $r_0 = q_0$
- $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0,1,2, \ldots, m-1$
- $r_m \in F$

Closures

- Recall: want to show the set of languages recognized by NFA is closed under:
  - union "$C = (A \cup B)$"
  - concatenation "$C = (A \circ B)$"
  - star "$C = A^*$"

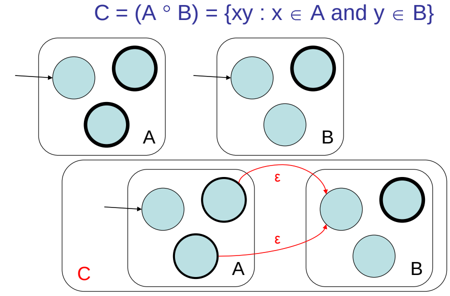
Closure under union

$C = (A \cup B) = \{x : x \in A \text{ or } x \in B\}$



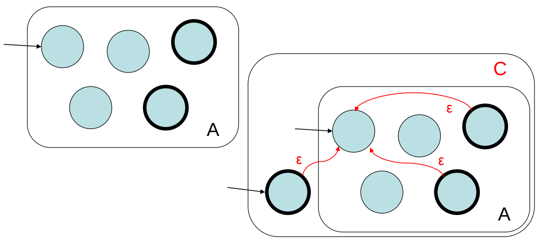
Closure under concatenation

$C = (A \circ B) = \{xy : x \in A \text{ and } y \in B\}$



Closure under star

$C = A^* = \{x_1x_2x_3\ldots x_k : k \geq 0 \text{ and each } x_i \in A\}$



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:

($\Rightarrow$) L is recognized by a FA implies L is recognized by a NFA.

($\Leftarrow$) L is recognized by a NFA implies L is recognized by a FA.

(usually one is easy, the other more difficult)

NFA, FA equivalence

($\Rightarrow$) 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.

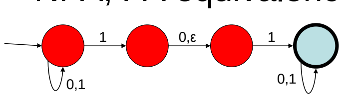
NFA, FA equivalence

($\Leftarrow$) L is recognized by a NFA implies L is recognized by a FA.

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

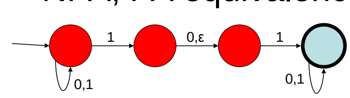
- alphabet will be the same
- what are the states of the FA?

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



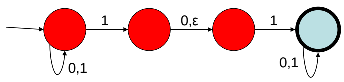
- given NFA M = $(Q, \Sigma, \delta, q_0, F)$
- construct FA M' = $(Q', \Sigma', \delta', q_0', F')$

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, Σ, δ, q_0, F)
– construct FA M' = (Q', Σ', δ', q_0', F')

– new start state: q_0' = E({q_0})
– new accept states:
F' = {R ∈ 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…

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

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

Theorem: 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 ∈ Σ
  – ε, the empty string
  – Ø, 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 ∪ 1)
  – if Σ = {0,1} then use "Σ" as shorthand for R
- example: R = 0 ° Σ*
  – shorthand: omit "°" R = 0Σ*
  – precedence: *, then °, then ∪, unless override by parentheses
  – in example R = 0(Σ*), not R = (0Σ)*