

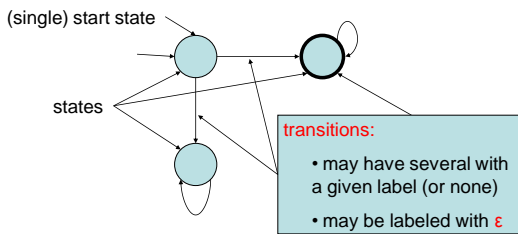
CS21 Decidability and Tractability

Lecture 3
January 9, 2012

Outline

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

NFA diagrams



- At each step, **several** choices for next state

NFA formal definition

A nondeterministic FA 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**
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$ 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**

Formal description of NFA operation

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

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

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

$$y = y_1y_2y_3\dots y_m \in (\Sigma \cup \{\epsilon\})^*$$

and \exists sequence r_0, r_1, \dots, r_m of states for which

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

Closures

- Recall: we showed the set of languages recognized by NFA is **closed** under:

– **union** " $C = (A \cup B)$ "

– **concatenation** " $C = (A \circ B)$ "

– **star** " $C = A^*$ "

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)

January 9, 2012

CS21 Lecture 3

7

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.

January 9, 2012

CS21 Lecture 3

8

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).

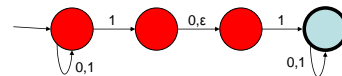
- alphabet will be the same
- what are the states of the FA?

January 9, 2012

CS21 Lecture 3

9

NFA, FA equivalence



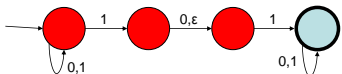
- given NFA $M = (Q, \Sigma, \delta, q_0, F)$
- construct FA $M' = (Q', \Sigma', \delta', q_0', F')$
- same alphabet: $\Sigma' = \Sigma$
- states are **subsets** of M's states: $Q' = \wp(Q)$
- if we are in state $R \in Q'$ and we read symbol $a \in \Sigma'$, what is the new state?

January 9, 2012

CS21 Lecture 3

10

NFA, FA equivalence



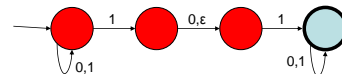
- given NFA $M = (Q, \Sigma, \delta, q_0, F)$
- construct FA $M' = (Q', \Sigma', \delta', q_0', F')$
- Helpful def'n:** $E(S) = \{q \in Q : q \text{ reachable from } S \text{ by traveling along 0 or more } \epsilon\text{-transitions}\}$
- new transition fn: $\delta'(R, a) = \cup_{r \in R} E(\delta(r, a))$
- = "all nodes reachable from R by following an a-transition, and then 0 or more ϵ -transitions"

January 9, 2012

CS21 Lecture 3

11

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

January 9, 2012

CS21 Lecture 3

12

NFA, FA equivalence

- We have proved (\Leftrightarrow) 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 ...

January 9, 2012

CS21 Lecture 3

13

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.

January 9, 2012

CS21 Lecture 3

14

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.

January 9, 2012

CS21 Lecture 3

15

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** $L(R)$.

January 9, 2012

CS21 Lecture 3

16

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 $R = (0\Sigma)^*$

January 9, 2012

CS21 Lecture 3

17

Some examples

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

January 9, 2012

CS21 Lecture 3

18

Manipulating regular expressions

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

January 9, 2012

CS21 Lecture 3

19

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.

January 9, 2012

CS21 Lecture 3

20

Regular expressions and FA

(\Leftarrow) 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 $L(R)$.

January 9, 2012

CS21 Lecture 3

21

Regular expressions and FA

- R is a regular expression if R is

$- a$, for some $a \in \Sigma$



$- \epsilon$, the empty string



$- \emptyset$, the empty set



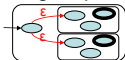
January 9, 2012

CS21 Lecture 3

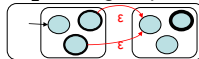
22

Regular expressions and FA

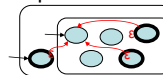
$- (R_1 \cup R_2)$, where R_1 and R_2 are reg. exprs.



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



$- (R_1^*)$, where R_1 is a regular expression



January 9, 2012

CS21 Lecture 3

23

Regular expressions and FA

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

Proof: given FA M that recognizes L , we will

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

January 9, 2012

CS21 Lecture 3

24

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 $w = w_1 w_2 w_3 \dots w_k$ 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 $w_i \in L(R)$

January 9, 2012

CS21 Lecture 3

25

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

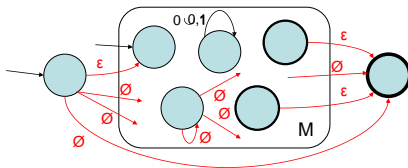
January 9, 2012

CS21 Lecture 3

26

Regular expressions and FA

- converting our FA M into GNFA in normal form:



January 9, 2012

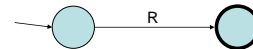
CS21 Lecture 3

27

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

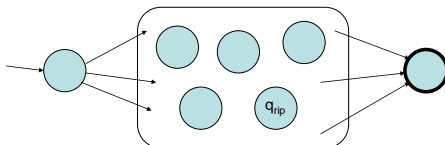
January 9, 2012

CS21 Lecture 3

28

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.

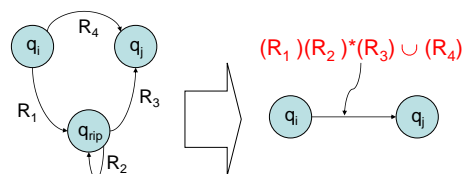
January 9, 2012

CS21 Lecture 3

29

Regular expressions and FA

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



January 9, 2012

CS21 Lecture 3


30

Regular expressions and FA

– summary:

FA $M \rightarrow$ k -state GNFA \rightarrow $(k-1)$ -state GNFA
 \rightarrow $(k-2)$ -state GNFA $\rightarrow \dots \rightarrow$ 2-state GNFA \rightarrow 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 GNFA equivalent to R 