

# CS21 Decidability and Tractability

Lecture 2  
January 7, 2015

January 7, 2015

CS21 Lecture 2

1

## Outline

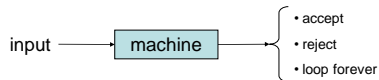
- Finite Automata
- Nondeterministic Finite Automata
- Closure under regular operations
- NFA, FA equivalence

January 7, 2015

CS21 Lecture 2

2

## What is computation?



- the set of strings that lead to “accept” is the language recognized by this machine
- if every other string leads to “reject”, then this language is decided by the machine

January 7, 2015

CS21 Lecture 2

3

## Terminology

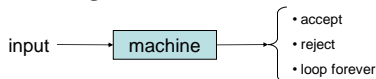
- finite alphabet  $\Sigma$  : a set of symbols
- language  $L \subseteq \Sigma^*$  : subset of strings over  $\Sigma$
- a machine takes an input string and either
  - accepts, rejects, or
  - loops forever
- a machine recognizes the set of strings that lead to accept
- a machine decides a language 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

January 7, 2015

CS21 Lecture 2

4

## What goes inside the box?



- We want the simplest mathematical formalization of computation possible.
- Strategy:
  - endow box with a feature of computation
  - try to characterize the languages decided
  - identify language we “know” real computers can decide that machine cannot
  - add new feature to overcome limits

January 7, 2015

CS21 Lecture 2

5

## Finite Automata

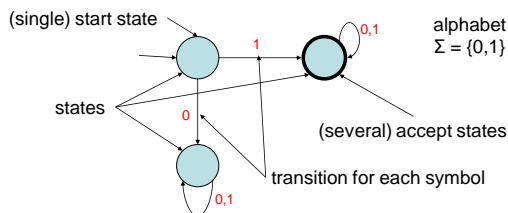
- simple model of computation
- reads input from left to right, one symbol at a time
- maintains state: information about what seen so far (“memory”)
  - finite automaton has finite # of states: cannot remember more things for longer inputs
- 2 ways to describe: by diagram, or formally

January 7, 2015

CS21 Lecture 2

6

## FA diagrams



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

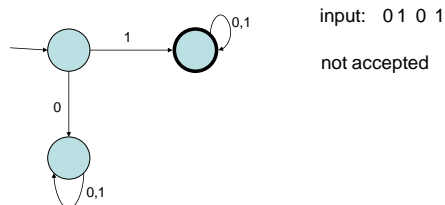
January 7, 2015

CS21 Lecture 2

7

## FA operation

- Example of FA operation:



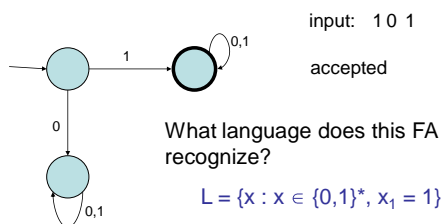
January 7, 2015

CS21 Lecture 2

8

## FA operation

- Example of FA operation:



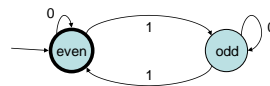
$$L = \{x : x \in \{0,1\}^*, x_1 = 1\}$$

January 7, 2015

CS21 Lecture 2

9

## Example FA



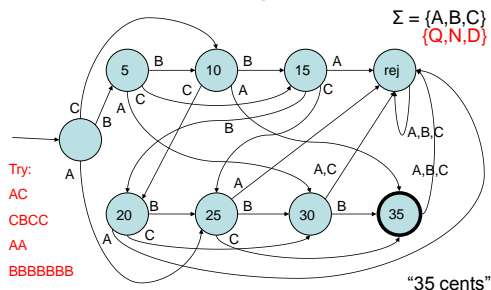
- What language does this FA recognize?  
 $L = \{x : x \in \{0,1\}^*, x \text{ has even \# of 1s}\}$
- illustrates fundamental feature/limitation of FA:
  - “tiny” memory
  - in this example only “remembers” 1 bit of info.

January 7, 2015

CS21 Lecture 2

10

## Example FA



January 7, 2015

CS21 Lecture 2

11

## FA formal definition

A finite automaton is a 5-tuple

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

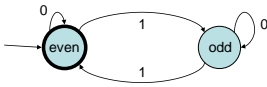
- $Q$  is a finite set called the **states**
- $\Sigma$  is a finite set called the **alphabet**
- $\delta: Q \times \Sigma \rightarrow Q$  is a function called the **transition function**
- $q_0$  is an element of  $Q$  called the **start state**
- $F$  is a subset of  $Q$  called the **accept states**

January 7, 2015

CS21 Lecture 2

12

## 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  $\delta$ :

$\delta(\text{even}, 0) = \text{even}$   
 $\delta(\text{even}, 1) = \text{odd}$   
 $\delta(\text{odd}, 0) = \text{odd}$   
 $\delta(\text{odd}, 1) = \text{even}$

January 7, 2015

CS21 Lecture 2

13

## Formal description of FA operation

finite automaton

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

accepts a string

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

if  $\exists$  sequence  $r_0, r_1, r_2, \dots, r_n$  of states for which

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

January 7, 2015

CS21 Lecture 2

14

## 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

January 7, 2015

CS21 Lecture 2

15

## 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

January 7, 2015

CS21 Lecture 2

16

## 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:  $\epsilon$  always in  $A^*$ )  
 $A^* = \{x_1 x_2 x_3 \dots x_k : k \geq 0 \text{ and each } x_i \in A\}$

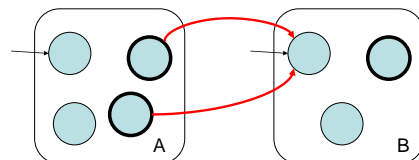
January 7, 2015

CS21 Lecture 2

17

## Concatenation attempt

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



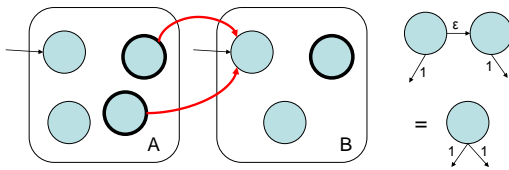
What label do we put on the new transitions?

January 7, 2015

CS21 Lecture 2

18

## Concatenation attempt



- Need it to happen “for free”: label with  $\epsilon$  (?)
- allows construct with multiple transitions with the same label (!?)

January 7, 2015

CS21 Lecture 2

19

## Nondeterministic FA

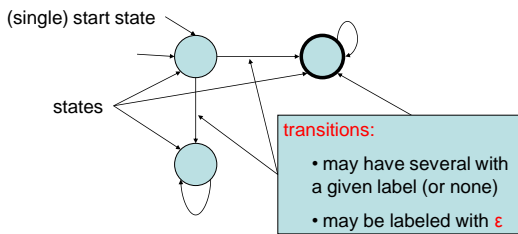
- We will make life easier by describing an additional feature (**nondeterminism**) that helps us to “program” FAs
- We will **prove** that FAs with this new feature can be **simulated** by ordinary FA
  - same spirit as programming constructs like procedures
- The concept of **nondeterminism** has a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

January 7, 2015

CS21 Lecture 2

20

## NFA diagrams



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

January 7, 2015

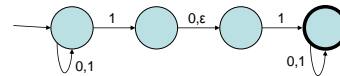
CS21 Lecture 2

21

## NFA operation

- Example of NFA operation:

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



input: 0 1 0

not accepted

January 7, 2015

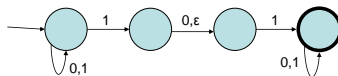
CS21 Lecture 2

22

## NFA operation

- Example of NFA operation:

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



input: 1 1 0

accepted

January 7, 2015

CS21 Lecture 2

23

## NFA operation

- One way to think of NFA operation:

- string  $x = x_1x_2x_3\dots x_n$  accepted if and only if
  - **there exists** a way of inserting  $\epsilon$ 's into  $x$ 

$$x_1\epsilon\epsilon x_2x_3\dots\epsilon x_n$$
  - so that **there exists** a path of transitions from the start state to an accept state

January 7, 2015

CS21 Lecture 2

24

## NFA formal definition

A nondeterministic FA

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

- $Q$  is a finite set called the **states**
- $\Sigma$  is a finite set called the **alphabet**
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$  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**

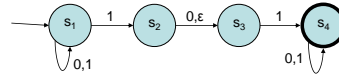
"powerset of  $Q$ ":  
the set of all  
subsets of  $Q$

January 7, 2015

CS21 Lecture 2

25

## NFA formal definition



- Specification of this NFA in formal terms:

- $Q = \{s_1, s_2, s_3, s_4\}$
- $\Sigma = \{0,1\}$
- $q_0 = s_1$
- $F = \{s_4\}$

- |  |                                   |
|--|-----------------------------------|
| $\delta(s_1, 0) = \{s_1\}$             | $\delta(s_3, 0) = \{s_3\}$        |
| $\delta(s_1, 1) = \{s_2\}$             | $\delta(s_3, 1) = \{s_4\}$        |
| $\delta(s_1, \epsilon) = \{s_1, s_2\}$ | $\delta(s_3, \epsilon) = \{s_3\}$ |
| $\delta(s_2, 0) = \{s_3\}$             | $\delta(s_4, 0) = \{s_4\}$        |
| $\delta(s_2, 1) = \{s_3\}$             | $\delta(s_4, 1) = \{s_4\}$        |
| $\delta(s_2, \epsilon) = \{s_3\}$      | $\delta(s_4, \epsilon) = \{s_4\}$ |

January 7, 2015

CS21 Lecture 2

26

## 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  $\epsilon$ '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  $i = 0, 1, 2, \dots, m-1$
- $r_m \in F$

January 7, 2015

CS21 Lecture 2

27

## 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** " $C = A^*$ "

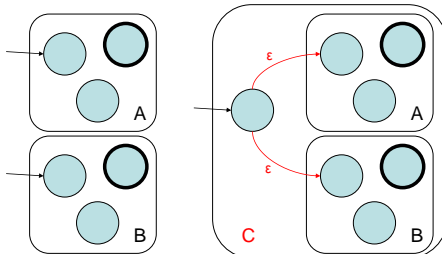
January 7, 2015

CS21 Lecture 2

28

## Closure under union

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



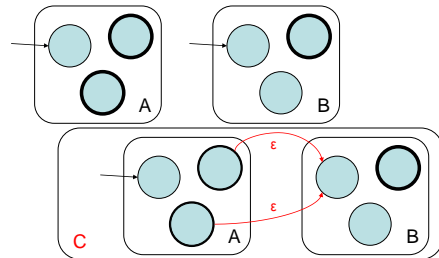
January 7, 2015

CS21 Lecture 2

29

## Closure under concatenation

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



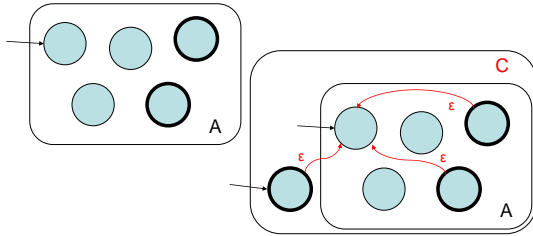
January 7, 2015

CS21 Lecture 2

30

## Closure under star

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



January 7, 2015

CS21 Lecture 2

31

## 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 7, 2015

CS21 Lecture 2

32

## 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  $\epsilon$ -transitions, and for which each state has exactly one outgoing transition for each symbol.

January 7, 2015

CS21 Lecture 2

33

## 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

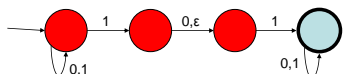
– what are the states of the FA?

January 7, 2015

CS21 Lecture 2

34

## 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:  $Q' = \wp(Q)$

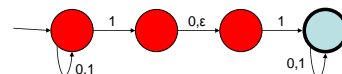
– if we are in state  $R \in Q'$  and we read symbol  $a \in \Sigma'$ , what is the new state?

January 7, 2015

CS21 Lecture 2

35

## NFA, FA equivalence



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

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

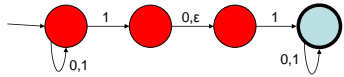
- new transition fn:  $\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$
- = “all nodes reachable from  $R$  by following an  $a$ -transition, and then 0 or more  $\epsilon$ -transitions”

January 7, 2015

CS21 Lecture 2

36

## 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 7, 2015

CS21 Lecture 2

37

## NFA, FA equivalence

- We have proved  $(\Leftarrow)$  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 7, 2015

CS21 Lecture 2

38