CS 3313
Foundations of Computing:

Non-Deterministic Finite Automata (NFA)

http://gw-cs3313-2021.github.io

Deterministic Finite Automata - Review

- **Definition:** A *deterministic finite automaton (DFA)* is defined as a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where:
  1. δ: a ***transition function*** from Q X Σ to Q – can be represented as a graph
  - Language accepted by a DFA L(M)
  - States – summarize events (i.e, input processed thus far)
- DFA and Translation to an Algorithm:
  - At each step M reads one symbol from input and goes to next state/step

Automata to Code…

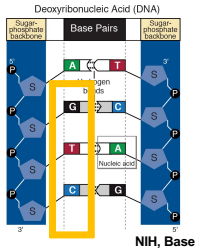
- Code:
  1. Read next input
  2. Decide on the next state
  3. Jump to beginning of the code for that state

```
1: ….
2: / * read i */
    c = getNextInput();
    if (c== ‘n’) goto 3;
     elseif (c == ‘I’) goto 2
     else goto 1;
3: /* ‘n’ is the input */
```

3

Exercise 2 - Recall: DNA Sequence Matching Problem

- *Problem*: Suppose a certain genetic disease *D* is characterized by the pattern *p* = **ATCG** presented in a strand of DNA sequence.
- p is consisted of *M* base pairs (bps) of 4 kinds.
  - For *p* = **ATCG**, *M* = 4 bps.



NIH, Base Pair
[https://www.genome.gov/genetics-glossary/Base-Pair]

- *Question*: given a DNA sequence sample, does the patient have *D*?
  - *Assume* only need to find the pattern once.
  - DNA Sample: … AACGAC~~CAAT~~CGTGGA …
  - However, the DNA sample has a length of *N* bps, where *N* >> *M*.
- Naïve Algorithm: Given sequence of length N and disease sequence of length M, we have an O(MN) algorithm.

4

Exercise 2: Apply DFA Problem Solving technique to the DNA sequence matching problem

- Why study DFA model of 'computing' (processing)?
- Now that you know how a DFA works, can you construct a more efficient solution for the DNA sequence matching problem ?
  - What is the time complexity of your solution ?
- Work in groups and submit……One per table

5

Exercise 2 - Recall: DNA Sequence Matching Problem

- *Problem*: Suppose a certain genetic disease *D* is characterized by the pattern *p* = **ATCG** presented in a strand of DNA sequence.
- p is consisted of *M* base pairs (bps) of 4 kinds.
  - For *p* = **ATCG**, *M* = 4 bps.
- *Question*: given a DNA sequence sample of length N, does the patient have *D?*
  - *Assume* only need to find the pattern once.
  - DNA Sequence of length N: … AACGACCAATCGTGGA …
  - DNA sequence has a length of *N* bps, where *N* >> *M*.
- Naïve Algorithm: Given sequence of length N and disease sequence of length M, we have an O(MN) algorithm.
- Design a more efficient solution – what is the time complexity of your algo?
  - For general case of DNA sequence length N and disease sequence length M

6

Next……Non-determinism

- We want to add features to the base machine (DFA) in an attempt to increase its ”power”
  - Power = more problems it can solve ?
  - Or is it efficiency (time) ?
  - Or is it expressive power (i.e., easier to write/develop a solution) ?
- Next topic: non-deterministic finite automaton

7

Non-determinism

- Nondeterminism: useful concept with big impact on theory of computation
- What is it ?.....a theoretical (problem solving) concept
  - All physical machines are built to be deterministic
- Basic idea: allow the machine (algorithm) to explore multiple (allowed) paths at each step
- This ”feature” can increase the expressive power (make it easier to solve problems)
  - Does this feature allow the machine to solve more problems ?

8

Parallel Execution…..Non-determinism?

```
int a=0;
while (!EOF){
      getInput(x);
      if (x==0)
        { a++ ;}

      elseif (x==1)
           { a--; }
```

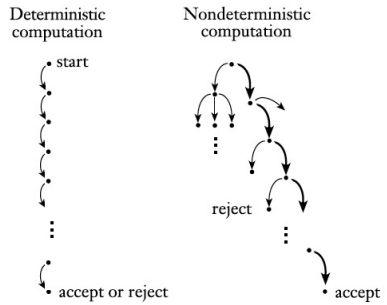
9

Non-determinism

- Nondeterminism: useful concept with big impact on theory of computation
- A "parallel" computation wherein multiple independent "*processes*" (or "*threads*") can be running concurrently and

  if at least one of these processes accepts (i.e., computes

  result) then process accepts



10

Multithreading, Parallelism and Non-determinism

11

Non-Deterministic Finite Automata (NFA)

- Define Non-Deterministic Finite Automata (NFA) Model
  - Simplified form first (without transitions on empty string)
  - Extend to include moves on empty string (no input)
    - Warning: the textbook jumps straight to this model of NFAs
  - Acceptance by NFA
  - Examples & Exercises
- Next week – Are NFAs more powerful than DFAs ? Can they solve problems that a DFA cannot ?

12

Recall Definition: DFA

- **Definition:** *deterministic finite automaton (DFA)* defined as M= (Q, Σ, δ, q_0 ,F) where δ: a ***transition function*** from Q X Σ to Q
- Can extend transition function to apply over strings. δ(q,w)
- For a DFA M, L(M) is the set of strings labeling paths from the start state to a final state, i.e., L(M) = the set of strings w such that $\delta(q_0, w)$ is in F.

  $L(M) = \{ w \mid \delta(q_0, w) \in F \}$
- DFAs accept a family of languages collectively known as *regular languages*.

13

Non-Deterministic Finite Automaton (NFA)

- An NFA can be in several states at once or viewed another way

  it can "guess" which state to go to next

- In terms of an algorithm: it explores all parallel paths at the same time
  - If any one of them leads to a final state then the machine "accepts" the input
- since at each step, it can "go" to several states: after reading an input string w it can be in set of states – subset of Q

14

Formal Definition: NFA

- $M = (Q, \Sigma, \delta, q_0, F)$
- A finite set of states, typically Q.
- An input alphabet, typically Σ.
- A transition function, typically δ from $Q \times \Sigma$ to 2^Q
- A start state in Q: q_0.
- A set of final states $F \subseteq Q$.
- transition function reads input a in state q and goes to a subset of states in Q
  - Ex: $Q = \{ q_0, q_1, q_2 \}$ $\Sigma = \{ 0, 1 \}$ $F = \{q_0\ q_4\}$
    where the transition function is given by
    $\delta(q_0, 0) = \{q_0\}$ $\delta(q_0, 1) = \{q_0, q_1\}$
    ……

15

DFA vs NFA

- Like the DFA, a nondeterministic finite automaton, or NFA, has one start state, where computation begins, and reads one input symbol at each step

  and goes to a set of states

- The intuition is that the NFA is allowed to guess which way to go, but it is able always to guess right, since all the guesses are followed in parallel and the NFA gets credit for the right guesses, no matter how many wrong guesses it also makes
  - . Machine *eventually follows one sequence of options/choices*

16

Language of an NFA

- The NFA can have any number of final states, and **an input is accepted if any sequence of choices leads from the start state to some final state.**

- A string w is accepted by an NFA if $\delta(q_0, w)$ contains at least one final state.

$$L(M) = \{ w \mid \delta(q_0, w) \cap F \neq \emptyset \}$$

The language of the NFA is the set of strings it accepts.

17

Extended Transition Function of an NFA

- $\delta(q, a)$ is a set of states.
- Extend to strings as follows:
- Basis: $\delta(q, \varepsilon) = \{q\}$
- Induction: $\delta(q, wa)$ = the union over all states p in $\delta(q, w)$ of $\delta(p, a)$

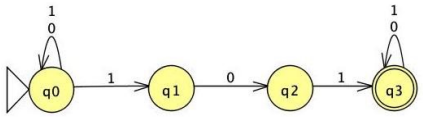
$$\delta(q, wa) = \bigcup_{p \in \delta(q, w)} \delta(p_i, a)$$

18

NFA Example 1: L= { w | w contains 101 }

- We've designed a DFA already….so how does the NFA design process work ?
- Suppose a string w is in L – what does this string look like ?
  - $w = u\ 101\ v$ and u,v can be any strings over $\{0,1\}$
- Algorithm:

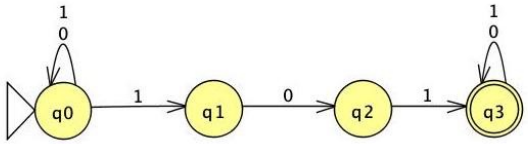
1. Read 0 stay in 1. Read 1: stay in 1 OR go to 2
2. Read 0 go to 3
3. Read 1 go to 4
4. Read any input, stay in 4. Accept if in step 4.



19

NFA Example 1: L= { w | w contains 101 }

- Assume Trap state = q_5
- Behavior of M on reading input 1101
  - After reading 1, (all possible) states= $\{q_0, q_1\}$
  - After reading second 1, states = $\{q_0, q_1, q_5\}$
  - After reading 0, states = $\{q_0, q_2, q_5\}$
  - After reading last 1, states = $\{q_0, q_3, q_5\}$

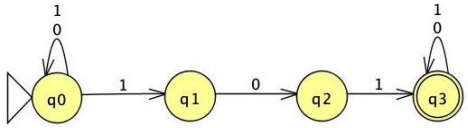


20

Extended Transition Function of an NFA

- $\delta(q, a)$ is a set of states.
- Extend to strings as follows:
- Basis: $\delta(q, \varepsilon) = \{q\}$
- Induction: $\delta(q, wa)$ = the union over all states p in $\delta(q, w)$ of $\delta(p, a)$

$$\delta(q, wa) = \cup_{p \in \delta(q, w)} \delta(p_i , a)$$



21

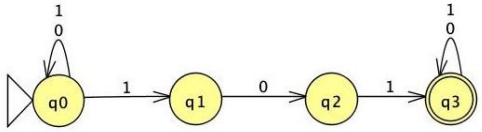
Extended Transition Function of an NFA

- $\delta(q_0 , 10)$ =

  $\delta((q_0 , 10) = \cup_{p \in \delta(q0 , 1)} \delta(p , 0)$

  $\delta(q_0 , 1) = \{ q_0 , q_1\}$

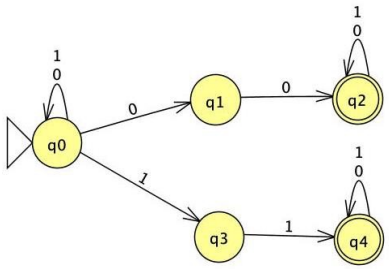
  $\delta((q_0 , 10) = \delta(q_0 , 0) \cup \delta(q_1 , 0) = \{q_0, q_2\}$
- $\delta(q_0 , 11) = \delta(q_0 , 1) \cup \delta(q_1 , 1) = \{q_0,q_1\} \cup \emptyset = \{q_0, q_1\}$
- Note: If we had a trap state q_4 then $\delta(q_1 , 1) = \{ q_4 \}$ and $\delta(q_0 , 11) = \{ q_0, q_1, q_4\}$
  - if $\delta(q , a) = \emptyset$ that is equivalent of a trap state



22

NFA Exercise 1 – Work at your table and submit one per table

- Consider this NFA
  - Ignore trap state; i.e., $\delta(q_1, 1) = \emptyset$ etc.
- Show the states the machine can be in after reading:
  a) 101 is this accepted by the NFA ?
  b) 0010 is this accepted by the NFA ?
- Describe the language accepted by the NFA



23

Breakout Group Exercises: Submit with names of all group members on the solution

- Design an NFA N that accepts the language

  L = { w | w is a string in $\{0,1\}^*$ and w contains the substring 101 with *at most* 1-bit position of mis-match.

24

Extending the NFA : NFA's With λ-Transitions

- Add more "features" to the NFA by allowing the Machine to change states without reading any input symbol
  - allow state-to-state transitions on empty string input λ (ϵ) .
- These transitions are done spontaneously, without looking at the input string.
  - Allowing λ-transitions can make it easier to define and build the automaton
- Analogous to program going to several next states before reading the next input

25

25

Parallel Execution….. NFA with λ-Transitions ?

- Find all documents that contain "Kyle" or "Oliver"

```
int a=0;
For each file Fi {
      a= search(Fi, 'Kyle');
      if (a==1)
       printf("Found File i");
 }
```

```
int a=0;
For each file Fi {
      a= search(Fi, 'Oliver');
      if (a==1)
       printf("Found File i");
  }
```

26

Formal Definition: NFA (with λ moves)

- M= (Q, Σ, δ, q_0 ,F)
- A finite set of states, typically Q.
- An input alphabet, typically Σ.
- A transition function, typically δ from Q X {Σ ∪ {λ }} to 2^Q
- A start state in Q: q_0.
- A set of final states F ⊆ Q.
- transition function reads input a or λ in state q and goes to a subset of states in Q
  - Ex: Q = { q_0, q_1, q_2 } Σ = { 0, 1 } F = {q_0 q_1 }
    where the transition function is given by
    δ(q_0, λ) = {q_0 , q_2} δ(q_0, 1) = {q_0 ,q_1 } δ(q_0, 0) = {q_0 }
    ……

27

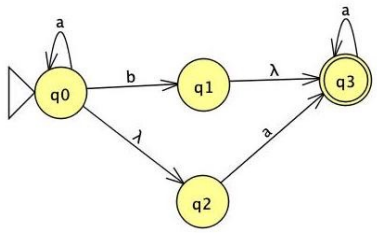
Extended transition function for NFA (with λ moves)

Intuition: δ(q, w) is the set of states you can reach from q following a path labeled w.

Will define it more formally later

δ(q_0, a) =

δ(q_0, ab) =



28

How does this model help ?...Revisit Example

- NFA for L = { w | w is a string in {0,1}* and w contains (a) the substring 101 or (b) substring 010 }
- NFA for property (a) {w | w contains 010 } and NFA for property (b)

DFA M1 for Property (a)

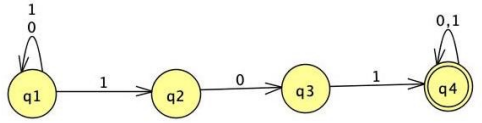
DFA M2 for Property (b)

29

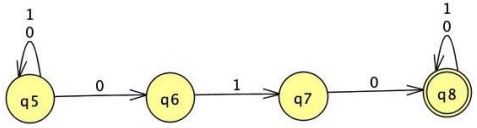
How does this model help ?...Revisit Example

- NFA for L = { w | w is a string in {0,1}* and w contains (a) the substring 101 or (b) substring 010 }
- NFA for property (a) {w | w contains 010 } and NFA for property (b)

DFA M1 for Property (a)

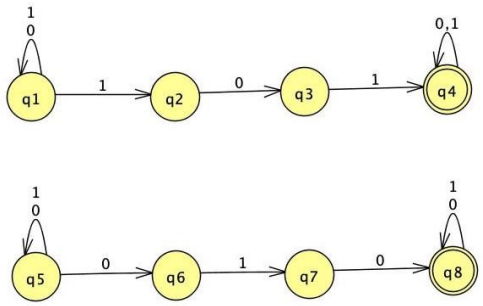


DFA M2 for Property (b)



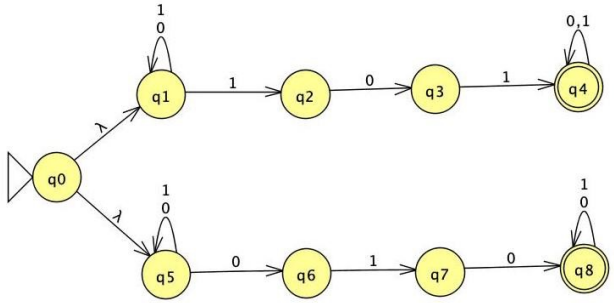
30

So how do we combine them ?



31

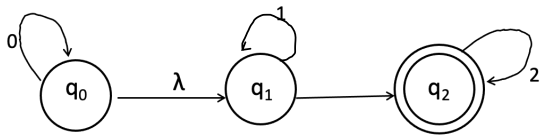
So how do we combine them ?



32

Example :

L = {w | w has *i* 0's followed by *j* 1's followed by *k* 2's, *i,j,k >= 0* }



33

Any advantage to NFA model with empty string input ?

- w is a string in L1 or L2: Construct M1 for L1 and M2 for L2, then add new start state and on λ start (go to start states) both M1 and M2; if either accepts then accept
  - –Run machines/programs in parallel
- w is a string x.y where x is in L1 and y is in L2: Construct M1 for L1 and M2 for L2, start in M1 and if it goes to final state then start M2.
  - – run machines in sequence (without reading input before starting second machine)
  - •Ex: {xy | x has substring 00 and y has substring 11 }
- What are we doing here…..simplification of the language/problem

34

Next : Are NFAs more powerful than DFAs ?
Is there another way to define Regular Languages

- NFAs can be easier to design…so more expressive power
- Is NFA with λ transition more powerful than NFA … NO!
- Can NFAs solve problems that cannot be solved by DFAs…NO!

  NFA and DFA are equivalent (in solving power) !
- Proof ? … We look into a constructive proof:

  Provide an algorithm that takes any NFA as input and generates an equivalent DFA
- Allows us to design NFAs and let the algorithm generate the DFA!
- How do we describe regular languages….Regular Expressions!
- End goal: Write a regular expression to describe a language and an algorithm will generate a DFA for that language !

35

NFA Exercise 2: Work in groups

- Provide an NFA with λ moves) that accepts the language L over alphabet {0,1,2} where:

L = { *w* | (a) *w* has two consecutive 0's or (b) *w* has substring 101 and ends with two 2's }

Ex: 0120012 is in L

010201222 is in L

02010220 is not in L

36