Pick up a handout from the back table near the east door

MA/CSSE 474

Theory of Computation

FSM: What's it all about?

Disclaimer: Slide sources:
Many slides from this course were provided by Elaine Rich in conjunction with our textbook.
Some others are based on slides by Jeffrey Ullman at Stanford University
Slides from both sources adapted by Claude Anderson
Additional slides by Claude Anderson

Instructor/Course Intro Tomorrow

- …along with roll call and other "first day" stuff
- Today: A look at DFSMs to give you the flavor for some major course ingredients
- Turn in your reading quiz. Another one due tomorrow at start of class.
- Feel free to ask questions/make comments at any point. Don't wait until I stop and ask
- Optional Q&A session Tuesday afternoon: Answers to your questions about the review material. 4:20 PM in G210

DFSM* Overview/Review

- MA/CSSE 474 *vs.* MA375
  – Same concept
  – Some different perspectives
  – Several different notations

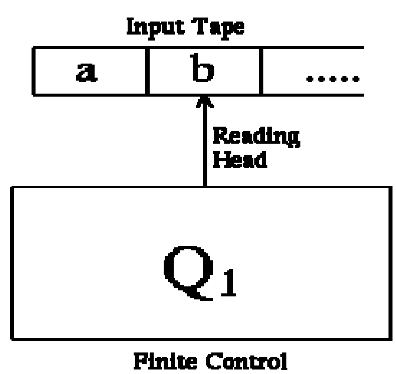
We'll provide more context, formalisms, and "***why's***" later.

- DFSM: a formal mathematical model of computation
  – A DFSM can remember only a ***fixed amount*** of info
  – That info is represented by the DFSM's current ***state***
  – Its state changes in response to ***input symbols***
  – A ***transition function*** describes how the state changes

*** DFSM stands for Deterministic Finite State Machine, a.k.a. Deterministic Finite Automaton (DFA)**

"Physical" DFSM Model

Input is finite, head moves right after reading each input symbol

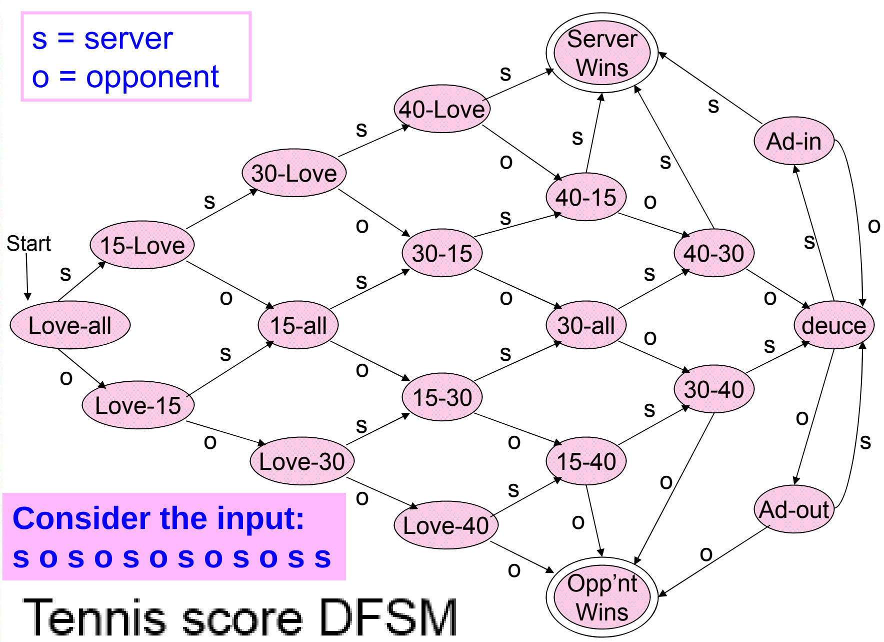


Scoring Tennis

- One person serves throughout a game
- To win, you must score at least 4 points
- You also must win by at least 2 points
- Inputs are:
  - s means “server wins a point” and
  - o means “opponent wins a point”
- State names are pairs of scores (the names the scores are called in tennis: love, 15, 30, 40, …)

5

s = server
o = opponent



Consider the input:
s o s o s o s o s o s s

Tennis score DFSM

from J. Ullman

Accepting states (a.k.a. Final states)

Notation: Alphabet, String, Language

- *Alphabet* Σ: finite set of symbols. **Examples:**
  – ASCII, Unicode, signals, {0, 1}, {a, b, c}, {s, o}
- *String* over an alphabet Σ: a *finite* sequence of symbols. **Examples:** 011, abc, sooso, ε
  – Note: 0 as string, 0 as symbol look the same
  – Context determines the type
  – ε is the empty string (some authors use λ)
- *Σ**: the set of *all* strings over the alphabet Σ
- A *language* over Σ is any subset of Σ*

Convention: Strings and Symbols

- … u, v, w, x, y, z will usually represent strings
- a, b, c,… will usually represent single input symbols
- When we write **w=ua**, we mean that
  – a is the last symbol of string w, and that
  – u is the substring (a.k.a. ***prefix*** of w) consisting of everything in w that comes before that a

8

DFSM - formal definition

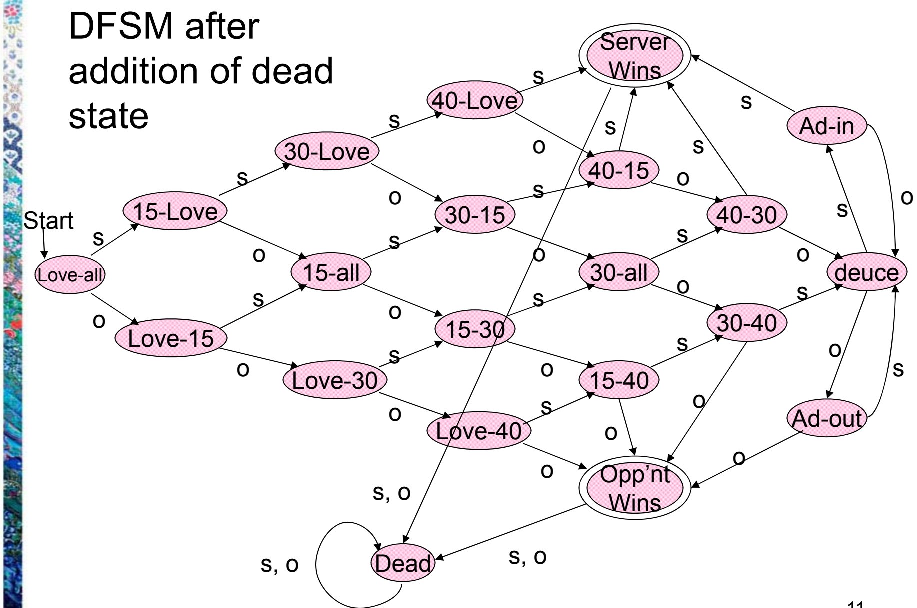
- $M = (K, \Sigma, \delta, s, A)$
  - Σ is the (finite) *alphabet*
  - K is the (finite) set of *states* (some authors: Q)
  - $s \in K$ is the *start state*
  - $A \subseteq K$ is the set of *accepting states* (some authors use F, for *final states*)
  - $\delta: K \times \Sigma \rightarrow K$ is the *transition function*
    Usually specified as a table or a graph

More on the Transition Function

- δ takes two arguments:
  a state and an input symbol
- $\delta(q, a)$ = the state that the DFSM goes to next after it is in state *q* and the tape head reads input *a*.
- *Note:* there is always a next state – wherever there is no explicitly shown transition, we assume a transition to a dead state.

Example on next slide

Tennis score DFSM after addition of dead state



11

Example: words ending in "ing"

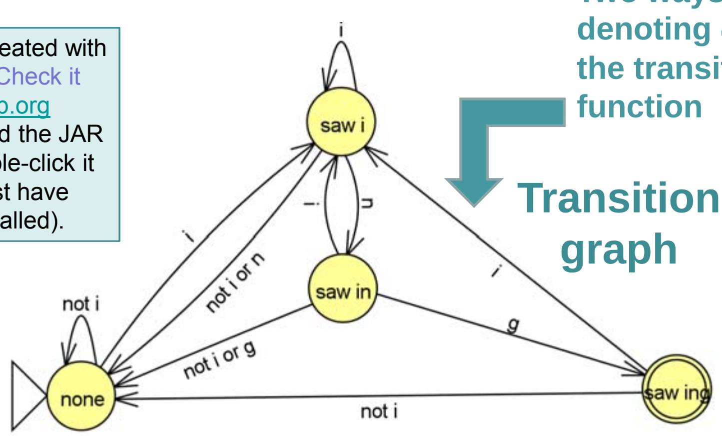
	i	n	g	other
none	saw i	none	none	none
saw i	saw i	saw in	none	none
saw in	saw i	none	saw ing	none
saw ing	saw i	none	none	none

Transition table

Two ways of denoting δ, the transition function

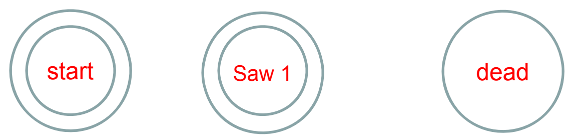
Transition graph

Graph created with JFLAP. Check it out! Jflap.org Download the JAR file, double-click it (you must have Java installed).



Example: strings without any 11 substrings

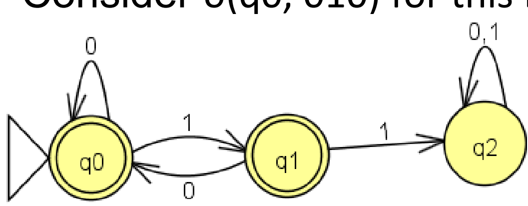
- {w∈{0,1}* : w does not have two consecutive 1's}
- Can you draw the state diagram?



- This example and the following slides were inspired by Jeffrey Ullman; significantly modified by CWA.

Extending the δ function

- If we consider (as in Python) a character to be a string of length 1, we can extend δ to δ: K × Σ* → K as follows
  - δ(q, ε) = q for every state q
  - If u is a string and a is a single symbol, δ(q, ua) = δ(δ(q, u), a)
- Consider δ(q0, 010) for this DFSM:



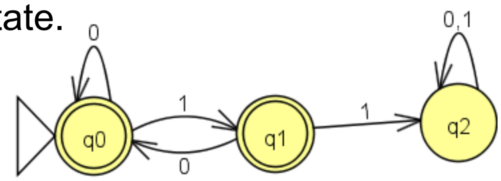
Example:
δ(q0, 010) =
δ(δ(q0, 01), 0) =
δ(δ(δ(q0, 0), 1), 0) =
δ(δ(q0, 1), 0) =
δ(q1, 0) =
q0

The Language of a DFSM

- If M is an automaton (any variety of automaton), **L(M)** means
  "the language accepted by M."
- If M=(K, Σ, δ, s, A) is a DFSM, then

$$L(M) = \{w \in \Sigma^* : \delta(s, w) \in A\}$$

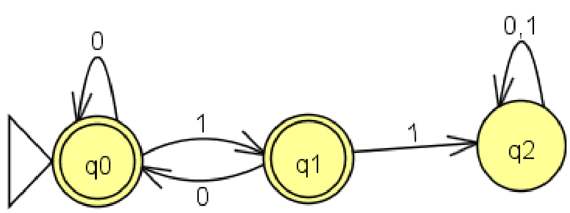
i.e., the set of all input strings that take the machine from its start state to an accepting state.



If this is M, what is L(M)?

Proving Set Equivalence

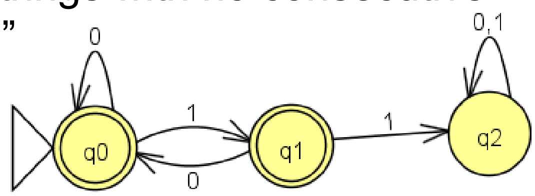
- Often, we need to prove that two sets S and T are in fact the same set. What is the general approach?
- Here, S is "the language accepted by this DFSM," and Tis "the set of strings of 0's and 1's with no consecutive pair of 1's."



16

Details of proof approach

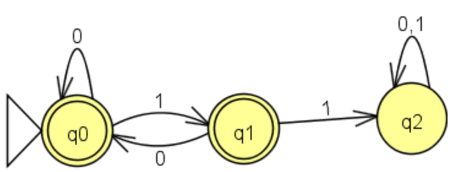
- In general, to prove S = T, we need to prove: S ⊆ T and T ⊆ S. That is:
  - A. If a string *w* is in S, then *w* is in T.
  - B. If *w* is in T, then w is in S.
  - C. Those are usually two separate proofs.
- Here, S = the language of our DFSM, and T = "strings with no consecutive pair of 1's."



17

Part A: S ⊆ T

- **To prove:** if w is accepted by M, then w does not have consecutive 1's.
- Proof is by induction on |w|, the length of w.
- **Important trick:** Expand the inductive hypothesis to be more general than the statement you are trying to prove.



18

Prove S⊆T by induction on |w| :

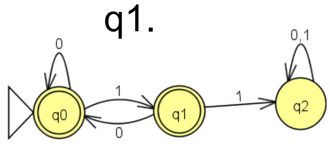
More general statement that we will prove:

Both of the following statements are true:

1. If δ(q0, w) = q0, then w does not end in 1 and w has no pair of consecutive 1's.
2. If δ(q0, w) = q1, w ends in 1 and w has no pair of consecutive 1's.

Can you see that (1) and (2) imply S⊆T?

- **Base case:** |w| = 0; i.e., w = ε.
  - (1) holds since ϵ has no 1's at all.
  - (2) holds ***vacuously***, since δ(q0, ϵ) is not q1.

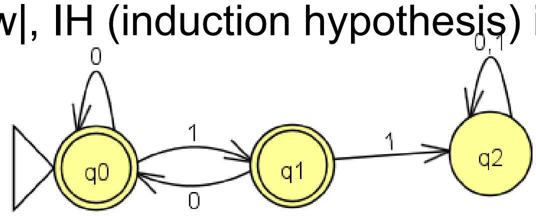


Important logic rule:
If the "if" part of any "if..then" statement is false, the whole statement is true.

19

Inductive Step for S ⊆ T

- Let |w| be ≥ 1, and assume (1) and (2) are true for all strings shorter than w.
- Because w is not empty, we can write w = ua, where *a* is the last symbol of w, and u is the string that precedes that last a.
- Since |u| < |w|, IH (induction hypothesis) is true for u.



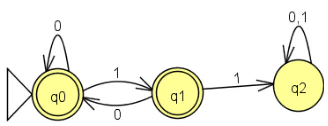
Reminder: What we are proving by induction:

1. If δ(q0, w) = q0, then w has no consecutive 1's and does not end in 1.
2. If δ(q0, w) = q1, then w has no consecutive 1's and ends in 1.

20

Inductive Step: S ⊆ T (2)

- Need to prove (1) and (2) for w = ua, assuming that they are true for u.
- (1) for w is: If δ(q0, w) = q0, then w has no consecutive 1's and does not end in 1. **Show it:**
- Since δ(q0, w) = q0, δ(q0, u) must be q0 or q1, and *a* must be 0 (look at the DFSM).
- By the IH, u has no 11's. The a is a 0.
- Thus, w has no 11's and does not end in 1.

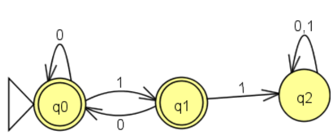


1. If δ(q0, w) = q0, then w has no consecutive 1's and does not end in 1.
2. If δ(q0, w) = q1, then w has no consecutive 1's and ends in 1.

21

Inductive Step : S ⊆ T (3)

- Now, prove (2) for w = ua: If δ(q0, w) = q1, then w has no 11's and ends in 1.
- Since δ(q0, w) = q1, δ(q0, u) must be q0, and *a* must be 1 (look at the DFSM).
- By the IH, u has no 11's and does not end in 1.
- Thus, w has no 11's and ends in 1.



1. If δ(q0, w) = q0, then w has no consecutive 1's and does not end in 1.
2. If δ(q0, w) = q1, then w has no consecutive 1's and ends in 1.

22

Part B: $T \subseteq S$

X

- Now, we must prove: if w has no 11's, then w is accepted by M

Y

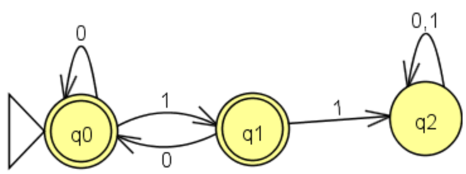
- *Contrapositive* : If w is *not* accepted by M then w has 11 as a substring.

Key idea: contrapositive of "if X then Y" is the equivalent statement "if not Y then not X."

23

Using the Contrapositive

- ***Contrapositive*** : If w is *not* accepted by M then w has 11 as a substring.
- **Base case** is again vacuously true.
- Because there is a unique transition from every state on every input symbol, each w gets the DFSM to exactly one state.
- The only way w can not be accepted is if it takes the DFSM M to q2. How can this happen?

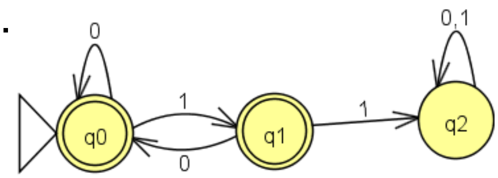


24

Using the Contrapositive – (2)

Looking at the DFSM, there are two possibilities: (recall that w=ua)

1. δ(q0,u) = q1 and a is 1. We proved earlier that if δ(q0,u) = q1, then u ends in 1. Thus w ends in 11.
2. δ(q0,u) = q2. In this case, the IH says that u contains 11 as a substring. So does w=ua.



25

Your 474 HW induction proofs

- Can be slightly less detailed
  - Many of the details here were about how the proof process works in general, rather than about the proof itself.
  - You can assume that the reader knows the proof *techniques*.
- Must always make it clear what the IH is, and where you apply it.
  - When in doubt about whether to include a detail, include it!
- Well-constructed proofs often contain more words than symbols.

This Proof as a 474 HW Problem

- An example of how I would write up this proof if it was a 474 HW problem will be linked from the schedule page this afternoon.
- You do not need to copy it exactly in your proofs, but it gives an idea of the kinds of things to include or not include.
- Also, I will post another version of the slides that includes the parts that I wrote on the board today.