Today's lecture: Regular Languages

I. Finite Automata with Multiple Input Tapes
II. Non-deterministic Finite Automata **III.** Concatenation and Asteration of Languages

Reading:

Today: Non-Determinism – Read: *Kozen* lecture 5 or *Sipser* 1.2.

September 16: Equivalence of DFAs and NFAs – Read: *Kozen* lecture 6 (or *Sipser* 1.2. as before).

September 19: Regular Expressions – Read: *Kozen* lectures 7 & 8 or *Sipser* 1.3.

September 21: Equivalence of Regular expressions and Finite Automata Read: *Kozen* lecture 9 (or *Sipser* 1.3. as before).

September 23: Nonregular Languages – Read: *Kozen* lecture 12 or *Sipser* 1.4.

September 26: More examples of the pumping Lemma Read: *Kozen* lecture 13 or *Sipser* 1.4.

September 28: Applications of finite automata

September 30: More applications

October 26: Midterm: In class.

I. Finite Automata with Multiple Input Tapes

A. Historical note:
The input to an automaton is often referred to as an "input tape." This is because mechanical tabulation machines that used punched cards or paper tapes for input were in use when the discipline of automata theory was first developed. The idea of a "tape" seemed to go better with strings than punched cards; so, the earlier researchers refered to the input as being on a "tape." The terminology stuck, and we still call it a "tape" today.

B. What can we do with two input tapes?

1. We can formalize their definition:

a. Let Σ_1 and Σ_2 denote the alphabets of the two tapes.

b. Let Q denote the finite set of machine states as before. Let q_0 be the initial state, and F be the set of accepting states as before.

c. At each step, the automaton reads a symbol from each tape, and makes a transition. Thus, $\delta : Q \times \Sigma_1 \times \Sigma_2 \to Q$.

2. Does the second tape give us any new kinds of languages?

a. No.

b. We could define an ordinary finite automaton whose input alphabet $\Sigma = \Sigma_1 \times \Sigma_2$.

c. δ, Q, q_0, and F remain the same.

d. This produces a machine that recognizes the same language as the two-input-tape machine

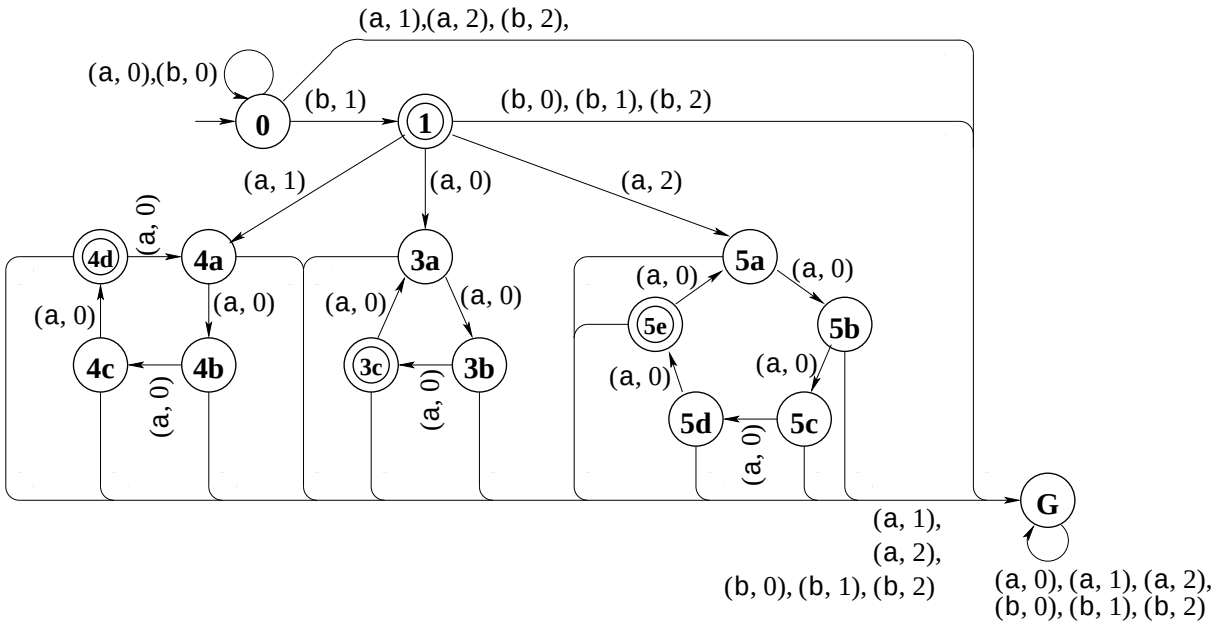

Figure 1: A Two-Input-Tape Finite Automaton

3. We do get something "interesting" if we quantify over the second tape.

a. Let M_2 be a two-input-tape machine as described above.

i.. We'll write $L(M_2)$ to denote the two-input-tape language accepted by M_2.

ii.. For two strings, s_1 and s_2 with $|s_1| = |s_2|$, define $\mathsf{weave}(s_1, s_2)$ as shown below:

$$\begin{array}{rcll} \mathsf{weave}(\epsilon, \epsilon) & = & \epsilon & \\ \mathsf{weave}(x_1\mathtt{c}_1, x_2\mathtt{c}_2) & = & \mathsf{weave}(x_1, x_2)(\mathtt{c}_1, \mathtt{c}_2), & \text{where } \mathtt{c}_1 \text{ and } \mathtt{c}_2 \text{ are individual symbols} \end{array}$$

b. Let $L_\exists(M_2) = \left\{ s_1 \in \Sigma_1^* \;\middle|\; \exists s_2 \in \Sigma_2^{|s_1|}.\ \mathsf{weave}(s_1, s_2) \in L(M_2) \right\}$

4. An example:

a. Figure 1 shows a state-transition diagram for a two-input-tape finite automaton where:

$$\begin{array}{rcl} \Sigma_1 & = & \{\mathtt{a}, \mathtt{b}\} \\ \Sigma_2 & = & \{\mathtt{0}, \mathtt{1}, \mathtt{2}\} \end{array}$$

b. This automaton recognizes strings where the last '`b`' is followed by n '`a`'s where n is any multiple of 3, 4, or 5.

c. This language can be recognized by a deterministic finite automaton (that's the kind we've studied so far), but the smallest such deterministic automaton has 61 states. This machine only has 15.

d. Consider what the machine does on input $s_1 =$ "`ababaaaaaaaa`". The machine accepts, with string "`000010000000`" as the existential witness for s_2.

0	$\xrightarrow{(\mathtt{a},\mathtt{0})}$	**0**	$\xrightarrow{(\mathtt{b},\mathtt{0})}$	**0**	$\xrightarrow{(\mathtt{a},\mathtt{0})}$	**0**	$\xrightarrow{(\mathtt{b},\mathtt{1})}$
1	$\xrightarrow{(\mathtt{a},\mathtt{0})}$	**4a**	$\xrightarrow{(\mathtt{a},\mathtt{0})}$	**4b**	$\xrightarrow{(\mathtt{a},\mathtt{0})}$	**4c**	$\xrightarrow{(\mathtt{a},\mathtt{0})}$
4d	$\xrightarrow{(\mathtt{a},\mathtt{0})}$	**4a**	$\xrightarrow{(\mathtt{a},\mathtt{0})}$	**4b**	$\xrightarrow{(\mathtt{a},\mathtt{0})}$	**4c**	$\xrightarrow{(\mathtt{a},\mathtt{0})}$
4d							

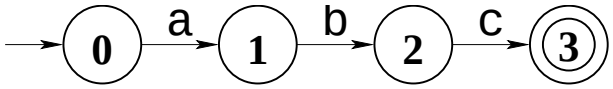

Figure 2: An NFA that recognizes "abc'

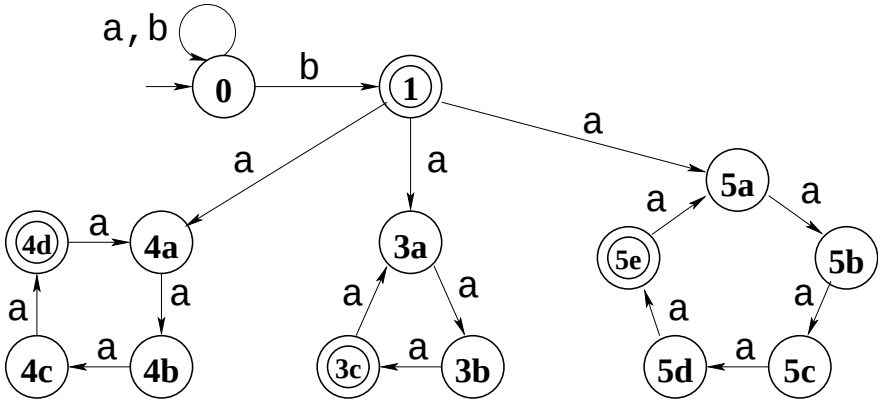

Figure 3: A Simplified Version of the Automaton from figure 1

e. Note that for some states (for example, state **1** in figure 1), the existentially quantified tape, s_2, guides the machine to the right state so that it will accept the string. You can think of s_2 as a collection of hints that the machine uses to find its way through a maze from the start state to an accepting state. If there is any string for s_2 for which machine M accepts s_1, then we say that $s_1 \in L_\exists(M)$.

II. Non-Deterministic Finite Automata (NFAs)

A. Simplifying Existentially Quantified, Two-Input-Tape, Finite Automata

1. Because s_2 is existentially quantified, it doesn't really matter which symbols in Σ_2 are used for which labels. All that matters is for each state $q \in Q$ and each symbol $\mathtt{c} \in \Sigma_1$, what states are reachable from q with input c and some symbol from Σ_2 for s_2. So, we drop the labels for the s_2 string.

2. The machine will find a path to an accepting state if one exists. Thus, we don't need to show transitions to terminally non-accepting states (i.e. state **G** in figure 1). For example, figure 2 shows a transition diagram for an automaton that recognizes the language $\{\mathtt{abc}\}$.

3. Figure 3 shows a transition diagram for the machine from figure 1 with these two simplifications.

B. NFAs formally defined

A NFA is a 5-tuple, $(Q, \Sigma, \Delta, Q_0, F)$ where

1. The ingredients of an NFA

a. Q is the set of states.

b. Σ is the input alphabet.

c. Each state has multiple possible successors. Thus, we have a state transition *relation* instead of the function that we had for a DFA (deterministic finite automaton). In particular, $\Delta \subseteq Q \times \Sigma \times Q$, $(q, \mathtt{c}, q') \in \Delta$ means that the machine can transition from state q to state q' when reading input symbol c.

d. We extend the generality of having a next state relation to allowing multiple initial states. $Q_0 \subseteq Q$ is the set of initial states.

e. F is the set of accepting states.

2. The acceptance condition

a. Define $\hat{\Delta}$
b. Define the acceptance condition
c. An example

C. Adding ϵ transitions