

## Week 2 at a glance

**Textbook reading: Sections 1.1, 1.2**

Before Monday, read pages 41-43 (Figures 1.18, 1.19, 1.20) for examples of automata and languages.

Before Wednesday, read pages 48-50 (Figures 1.27, 1.29) which introduces nondeterminism.

Before Friday, read pages 45-46 (Theorem 1.25) that we'll refer to as a "closure proof".

For Week 3 Monday: Theorem 1.47 + 1.48, Theorem 1.39 "Proof Idea", Example 1.41, Example 1.56.

**We will be learning and practicing to:**

- Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.
  - Give examples of sets that are regular (and prove that they are).
    - \* **State the definition of the class of regular languages**
    - \* **Give examples of regular languages, using each of the three equivalent models of computation for proving regularity.**
  - Describe and use models of computation that don't involve state machines.
    - \* **Given a DFA or NFA, find a regular expression that describes its language.**
    - \* **Given a regular expression, find a DFA or NFA that recognizes its language.**
  - Use precise notation to formally define the state diagram of finite automata.
  - Use clear English to describe computations of finite automata informally.
    - \* **Design an automaton that recognizes a given language**
    - \* **Specify a general construction for DFA based on parameters**
    - \* **Design general constructions for DFA**
    - \* **Motivate the use of nondeterminism**
    - \* **State the formal definition of NFA**
    - \* **Trace the computation(s) of a NFA on a given string using its state diagram**
    - \* **Determine if a given string is in the language recognized by a NFA**

**TODO:**

#FinAid Assignment on Canvas (complete as soon as possible) and read syllabus on Canvas

Schedule your Test 1 Attempt 1, Test 2 Attempt 1, Test 1 Attempt 2, and Test 2 Attempt 2 times at PrairieTest (<http://us.prairietest.com>)

Homework 1 submitted via Gradescope (<https://www.gradescope.com/>), due Tuesday 10/8/2024

Review Quiz 2 on PrairieLearn (<http://us.prairielearn.com>), complete by Sunday 10/13/2024

## Monday: Finite automaton constructions

**Review:** Formal definition of DFA:  $M = (Q, \Sigma, \delta, q_0, F)$

- Finite set of states  $Q$
- Alphabet  $\Sigma$
- Transition function  $\delta$
- Start state  $q_0$
- Accept (final) states  $F$

Quick check: In the state diagram of  $M$ , how many outgoing arrows are there from each state?

**Note:** We'll see a new kind of finite automaton. It will be helpful to distinguish it from the machines we've been talking about so we'll use **Deterministic Finite Automaton** (DFA) to refer to the machines from Section 1.1.

$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_0\})$  where  $\delta$  is (rows labelled by states and columns labelled by symbols):

$\delta$	$a$	$b$
$q_0$	$q_1$	$q_1$
$q_1$	$q_2$	$q_2$
$q_2$	$q_0$	$q_0$

The state diagram for  $M$  is

Give two examples of strings that are accepted by  $M$  and two examples of strings that are rejected by  $M$ :

A regular expression describing  $L(M)$  is

A state diagram for a finite automaton recognizing

$$\{w \mid w \text{ is a string over } \{a, b\} \text{ whose length is not a multiple of } 3\}$$

Extra example: Let  $n$  be an arbitrary positive integer. What is a formal definition for a finite automaton recognizing

$$\{w \mid w \text{ is a string over } \{0, 1\} \text{ whose length is not a multiple of } n\}?$$

Consider the alphabet  $\Sigma_1 = \{0, 1\}$ .

A state diagram for a finite automaton that recognizes  $\{w \mid w \text{ contains at most two 1's}\}$  is

A state diagram for a finite automaton that recognizes  $\{w \mid w \text{ contains more than two 1's}\}$  is

**Strategy:** Add “labels” for states in the state diagram, e.g. “have not seen any of desired pattern yet” or “sink state”. Then, we can use the analysis of the roles of the states in the state diagram to work towards a description of the language recognized by the finite automaton.

Or: decompose the language to a simpler one that we already know how to recognize with a DFA or NFA.

Textbook Exercise 1.14: Suppose  $A$  is a language over an alphabet  $\Sigma$ . If there is a DFA  $M$  such that  $L(M) = A$  then there is another DFA, let's call it  $M'$ , such that  $L(M') = \overline{A}$ , the complement of  $A$ , defined as  $\{w \in \Sigma^* \mid w \notin A\}$ .

**Proof idea:**

A useful bit of terminology: the **iterated transition function** of a finite automaton  $M = (Q, \Sigma, \delta, q_0, F)$  is defined recursively by

$$\delta^*(q, w) = \begin{cases} q & \text{if } q \in Q, w = \varepsilon \\ \delta(q, a) & \text{if } q \in Q, w = a \in \Sigma \\ \delta(\delta^*(q, u), a) & \text{if } q \in Q, w = ua \text{ where } u \in \Sigma^* \text{ and } a \in \Sigma \end{cases}$$

Using this terminology,  $M$  accepts a string  $w$  over  $\Sigma$  if and only if  $\delta^*(q_0, w) \in F$ .

**Proof:**

## Wednesday: Nondeterministic automata

We saw that whenever a language is recognized by a DFA, its complement is also recognized by some (other) DFA.

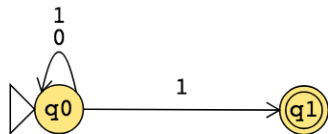
Another way to say this is that the collection of languages that are each recognizable by a DFA is **closed** under complementation.

**Nondeterministic finite automaton** (Sipser Page 53) Given as  $M = (Q, \Sigma, \delta, q_0, F)$

Finite set of states $Q$	Can be labelled by any collection of distinct names. Default: $q_0, q_1, \dots$
Alphabet $\Sigma$	Each input to the automaton is a string over $\Sigma$ .
Arrow labels $\Sigma_\epsilon$	$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ . Arrows in the state diagram are labelled either by symbols from $\Sigma$ or by $\epsilon$
Transition function $\delta$	$\delta : Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$ gives the <b>set of possible next states</b> for a transition from the current state upon reading a symbol or spontaneously moving.
Start state $q_0$	Element of $Q$ . Each computation of the machine starts at the start state.
Accept (final) states $F$	$F \subseteq Q$ .

$M$  accepts the input string  $w \in \Sigma^*$  if and only if **there is** a computation of  $M$  on  $w$  that processes the whole string and ends in an accept state.

The formal definition of the NFA over  $\{0, 1\}$  given by this state diagram is:



The language over  $\{0, 1\}$  recognized by this NFA is:

Change the transition function to get a different NFA which accepts the empty string (and potentially other strings too).

The state diagram of an NFA over  $\{a, b\}$  is below. The formal definition of this NFA is:



Suppose  $A_1, A_2$  are languages over an alphabet  $\Sigma$ . **Claim:** if there is a NFA  $N_1$  such that  $L(N_1) = A_1$  and NFA  $N_2$  such that  $L(N_2) = A_2$ , then there is another NFA, let's call it  $N$ , such that  $L(N) = A_1 \cup A_2$ .

**Proof idea:** Use nondeterminism to choose which of  $N_1, N_2$  to run.

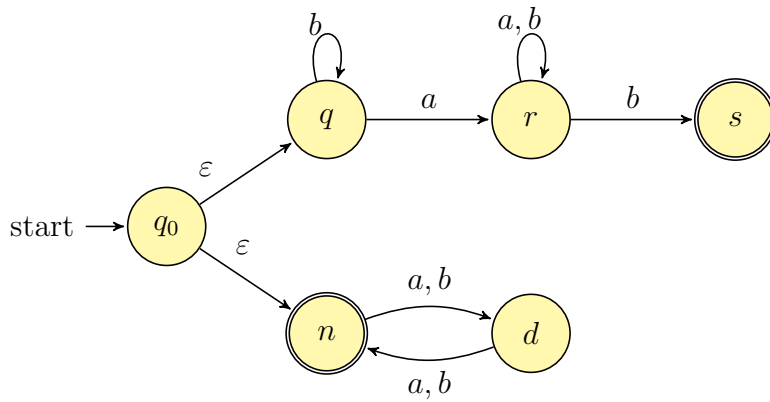
**Formal construction:** Let  $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$  and  $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$  and assume  $Q_1 \cap Q_2 = \emptyset$  and that  $q_0 \notin Q_1 \cup Q_2$ . Construct  $N = (Q, \Sigma, \delta, q_0, F_1 \cup F_2)$  where

- $Q =$
- $\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$  is defined by, for  $q \in Q$  and  $x \in \Sigma_\varepsilon$ :

*Proof of correctness would prove that  $L(N) = A_1 \cup A_2$  by considering an arbitrary string accepted by  $N$ , tracing an accepting computation of  $N$  on it, and using that trace to prove the string is in at least one of  $A_1, A_2$ ; then, taking an arbitrary string in  $A_1 \cup A_2$  and proving that it is accepted by  $N$ . Details left for extra practice.*

## Friday: Automata constructions

**Review:** The language recognized by the NFA over  $\{a, b\}$  with state diagram



is:

So far, we know:

- The collection of languages that are each recognizable by a DFA is **closed** under complementation.  
*Could we do the same construction with NFA?*

- The collection of languages that are each recognizable by a NFA is **closed** under union.  
*Could we do the same construction with DFA?*

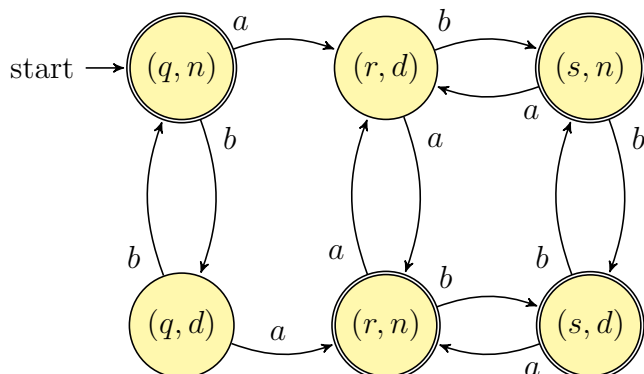
Happily, though, an analogous claim is true!

Suppose  $A_1, A_2$  are languages over an alphabet  $\Sigma$ . **Claim:** if there is a DFA  $M_1$  such that  $L(M_1) = A_1$  and DFA  $M_2$  such that  $L(M_2) = A_2$ , then there is another DFA, let's call it  $M$ , such that  $L(M) = A_1 \cup A_2$ .  
*Theorem 1.25 in Sipser, page 45*

**Proof idea:**

**Formal construction:**

**Example:** When  $A_1 = \{w \mid w \text{ has an } a \text{ and ends in } b\}$  and  $A_2 = \{w \mid w \text{ is of even length}\}$ .



Suppose  $A_1, A_2$  are languages over an alphabet  $\Sigma$ . **Claim:** if there is a DFA  $M_1$  such that  $L(M_1) = A_1$  and DFA  $M_2$  such that  $L(M_2) = A_2$ , then there is another DFA, let's call it  $M$ , such that  $L(M) = A_1 \cap A_2$ .  
*Footnote to Sipser Theorem 1.25, page 46*

**Proof idea:**

**Formal construction:**