Before we start

We are committed to fostering a learning environment for this course that supports a diversity of thoughts, perspectives and experiences, and respects your identities (including race, ethnicity, heritage, gender, sex, class, sexuality, religion, ability, age, educational background, etc.). Our goal is to create a diverse and inclusive learning environment where all students feel comfortable and can thrive.

If you or someone you know is suffering from food and/or housing insecurities there are UCSD resources here to help:

Basic Needs Office: https://basicneeds.ucsd.edu/

Triton Food Pantry (in the old Student Center) is free and anonymous, and includes produce:

https://www.facebook.com/tritonfoodpantry/

Mutual Aid UCSD: https://mutualaiducsd.wordpress.com/

Financial aid resources, the possibility of emergency grant funding, and off-campus housing referral resources are available. See CAPS and your college dean.

If you find yourself in an uncomfortable situation, ask for help. We are committed to upholding University policies regarding nondiscrimination, sexual violence and sexual harassment.

Counseling and Psychological Services (CAPS) at 858 5343755 or http://caps.ucsd.edu

OPHD at (858) 534-8298, ophd@ucsd.edu , http://ophd.ucsd.edu. CARE at Sexual Assault Resource Center at 858 5345793 sarc@ucsd.edu http://care.ucsd.edu

Please do not come to class if you are sick or even think you might be sick. We will continue to follow the campus guidelines as updated on https://returntolearn.ucsd.edu/ .

Please reach out (minnes@ucsd.edu or dgrier@ucsd.edu) if you need support with extenuating circumstances.

Introductions

Class website: https://canvas.ucsd.edu/courses/45073

Instructor for A00: Prof. Mia Minnes "Minnes" rhymes with Guinness, minnes@ucsd.edu, http://cseweb.ucsd.edu/ minnes

Instructor for B00: Prof. Daniel Grier, dgrier@ucsd.edu, http://danielgrier.com/

Our team: Two instructors + four TAs and nine tutors + all of you

Fill in contact info for students around you, if you'd like:

study groups!

Homework groups!
(You can move around in different
homework groups throughout the
quarter)

Note: midterm exam   May 5, 2023 in class
      final   exam   June 14, 2023 (see webreg)

On a typical week: **MWF** Lectures + review quizzes, **Tu** Homework due, **WF** Discussion. Office hours (hosted by instructors and TAs and tutors, where you can come to talk about course concepts and ask for help as you work through sample problems) and Q+A on Piazza available throughout the week. All dates are on Canvas (click for link) and details are on the Syllabus page.

Welcome to CSE 105: Introduction to Theory of Computation in Spring 2023!

CSE 105's Big Questions

- What problems are computers capable of solving?

- What resources are needed to solve a problem?

- Are some problems harder than others?

In this context, a **problem** is defined as: "Making a decision or computing a value based on some input"

Consider the following problems:

- Find a file on your computer *Input: filename*
- Determine if your code will compile *Input: code*
- Find a run-time error in your code *Input: code and description of valid input*
- Certify that your system is un-hackable

Which of these is hardest? *... stay tuned ☺*

In Computer Science, we operationalize "hardest" as "requires most resources", where resources might be memory, time, parallelism, randomness, power, etc.

To be able to compare "hardness" of problems, we use a consistent description of problems

Input: String

Output: Yes/ No, where Yes means that the input string matches the pattern or property described by the problem.

Monday

Some terminology we'll use:
Set, sequence, finite

We will use vocabulary that should be familiar from your discrete math and introduction to proofs classes. Some of the notation conventions may be a bit different: we will use the notation from this class' textbook[1].

Write out in words the meaning of the symbols below:

$$\{a, b, c\}$$

The set whose elements are a, b, and c.

Note: $\{a, b, c\} = \{c, b, a\}$

$$|\{a, b, a\}| = 2$$

The cardinality of the set whose elements are a, b and a is 2.

$$|aba| = 3$$

The length of the string aba is 3.

$$(a, 3, 2, b, b)$$

The 5-tuple whose first component is a, whose second component is 3, whose third component is 2, whose fourth component is b, and whose fifth component is also b.

Term	Typical symbol	Meaning
Alphabet	Σ, Γ	A non-empty finite set
Symbol over Σ	σ, b, x	An element of the alphabet Σ
String over Σ	u, v, w	A finite list of symbols from Σ
The set of all strings over Σ	Σ^*	The collection of all possible strings formed from symbols from Σ
(Some) language over Σ	L	(Some) set of strings over Σ
Empty string	ε	The string of length 0
Empty set	$\emptyset$	The empty language
The power set of a set X	$\mathcal{P}(X)$	The set of all subsets of X
Natural numbers	$\mathcal{N}$	The set of positive integers
Finite set		The empty set or a set whose distinct elements can be counted by a natural number
Infinite set		A set that is not finite.
Pages 3, 4, 6, 13, 14, 53		

---

[1]Page references are to the 3rd edition (International) of Sipser's Introduction to the Theory of Computation, available through various sources for under $30. You may be able to opt in to purchase a digital copy through Canvas. Copies of the book are also available for those who can't access the book to borrow from the course instructor, while supplies last (minnes@eng.ucsd.edu)

Term	Notation	Meaning
Reverse of a string w	$w^{\mathcal{R}}$	write w in the opposite order, if $w = w_1 \cdots w_n$ then $w^{\mathcal{R}} = w_n \cdots w_1$. Note: $\varepsilon^{\mathcal{R}} = \varepsilon$
Concatenating strings x and y	xy	take $x = x_1 \cdots x_m$, $y = y_1 \cdots y_n$ and form $xy = x_1 \cdots x_m y_1 \cdots y_n$
String z is a substring of string w		there are strings u, v such that $w = uzv$
String x is a prefix of string y		there is a string z such that $y = xz$
String x is a proper prefix of string y		x is a prefix of y and $x \neq y$
Shortlex order, also known as string order over alphabet Σ		Order strings over Σ first by length and then according to the dictionary order, assuming symbols in Σ have an ordering.

Pages 13, 14

Circle the correct choice:

Recall: Σ^* is the set of all strings over the alphabet Σ.

A **string** over an alphabet Σ is (an element of Σ^*) ~~a subset of Σ^*~~

A **language** over an alphabet Σ is ~~an element of Σ^*~~ (a subset of Σ^*).

Extra examples for practice:

With $\Sigma_1 = \{0, 1\}$ and $\Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$ and $\Gamma = \{0, 1, x, y, z\}$

An example of a string of length 3 over Σ_1 is ___000___

example of a substring of $w = 000$ is $z = 00$

let $u = 0$, $v = \varepsilon$ so $w = uzv$.

An example of a string of length 1 over Σ_2 is _____

The number of distinct strings of length 2 over Γ is _____
(review CSE21

The other examples of substrings of 000 are 0, 000, and ε.

An example of a language over Σ_1 of size 1 is ___{000}___

An example of an infinite language over Σ_1 is ___Σ_1^*___

An example of a finite language over Γ is ___$\emptyset$___

True or **False**: $\varepsilon \in \Sigma_1$

True or **False**: ε is a string over Σ_1

True or **False**: ε is a language over Σ_1

Note because $000 = \varepsilon \cdot 000 \cdot \varepsilon$ then 000 is a substring of 000.

True or **False**: ε is a prefix of some string over Σ_1

True or **False**: There is a string over Σ_1 that is a proper prefix of ε

The first five strings over Σ_1 in string order, using the ordering $0 < 1$:

The first five strings over Σ_2 in string order, using the usual alphabetical ordering for single letters:

Review: Week 1 Monday

1. Please complete the beginning of the quarter survey linked from our #FinAid Assignment on Canvas https://canvas.ucsd.edu/courses/45073/assignments/609253?wrap=1

2. We want you to be familiar with class policies and procedures so you are ready to have a successful quarter. Please take a look at the syllabus on our class website https://canvas.ucsd.edu/courses/45073 and answer the questions about it on Gradescope.

A summary of the terminology we will use is on page 16 in the textbook.

Pre class reading for next time: Example 1.51, Definition 1.52

Notice: we are jumping to Section 1.3 Regular Expressions and then will come back to Section 1.1 Finite Automata on Friday.

Week 1 Wednesday

Our motivation in studying sets of strings is that they can be used to encode problems. To calibrate how difficult a problem is to solve, we describe how complicated the set of strings that encodes it is. How do we define sets of strings?

Defining finite sets : listing elements (roster method)

Define sets with properties (set builder)

$$\{ x \in \underline{\quad} \mid \underline{\qquad\qquad} \}$$

Define special sets by naming them

$$\Sigma^*$$

Also, recursive definitions.

$\emptyset$

How would you describe the language that has no elements at all?

$\emptyset$

How would you describe the language that has all strings over $\{0, 1\}$ as its elements?

When $\Sigma = \{0, 1\}$ is the alphabet then the language that has all strings over this Σ^* or $\{0, 1\}^*$ alphabet is

Regular expression describing this set

is $L\left((0 \cup 1)^* \right) = \left\{ \underline{\%_1} \underline{\%_1} \cdots \underline{\%_1} \mid k \geq 0 \right\}$

k

Definition 1.52: A **regular expression** over alphabet Σ is a syntactic expression that can describe a language over Σ. The collection of all regular expressions is defined recursively:

R

Basis steps of recursive definition

a is a regular expression, for $a \in \Sigma$

ε is a regular expression

$\emptyset$ is a regular expression

Recursive steps of recursive definition

$(R_1 \cup R_2)$ is a regular expression when R_1, R_2 are regular expressions

$(R_1 \circ R_2)$ is a regular expression when R_1, R_2 are regular expressions

(R_1^*) is a regular expression when R_1 is a regular expression

The *semantics* (or meaning) of the syntactic regular expression is the **language described by the regular expression**. The function that assigns a language to a regular expression over Σ is defined recursively, using familiar set operations:

$L(R)$

set of strings.

Basis steps of recursive definition

The language described by a, for $a \in \Sigma$, is $\{a\}$ and we write $L(a) = \{a\}$

The language described by ε is $\{\varepsilon\}$ and we write $L(\varepsilon) = \{\varepsilon\}$

The language described by $\emptyset$ is $\{\}$ and we write $L(\emptyset) = \emptyset$.

Recursive steps of recursive definition

When R_1, R_2 are regular expressions, the language described by the regular expression $(R_1 \cup R_2)$ is the union of the languages described by R_1 and R_2, and we write

$$L((R_1 \cup R_2)) = L(R_1) \cup L(R_2) = \{w \mid w \in L(R_1) \vee w \in L(R_2)\}$$

When R_1, R_2 are regular expressions, the language described by the regular expression $(R_1 \circ R_2)$ is the concatenation of the languages described by R_1 and R_2, and we write

$$L((R_1 \circ R_2)) = L(R_1) \circ L(R_2) = \{uv \mid u \in L(R_1) \wedge v \in L(R_2)\}$$

When R_1 is a regular expression, the language described by the regular expression (R_1^*) is the **Kleene star** of the language described by R_1 and we write

$$L((R_1^*)) = (L(R_1))^* = \{w_1 \cdots w_k \mid k \geq 0 \text{ and each } w_i \in L(R_1)\}$$

For the following examples assume the alphabet is $\Sigma_1 = \{0, 1\}$:

The language described by the regular expression 0 is $L(0) = \{0\}$

The language described by the regular expression 1 is $L(1) = \{1\}$

The language described by the regular expression ε is $L(\varepsilon) = \{\varepsilon\}$

Version April 2, 2023 (8)

The language described by the regular expression $\emptyset$ is $L(\emptyset) = \emptyset$

The language described by the regular expression $((0 \cup 1) \cup 1)$ is $L(\,((0 \cup 1) \cup 1)\,) =$

$$L\left(\,(\,(0 \cup 1)\, \cup 1)\,\right) = L((0 \cup 1)) \cup L(1)$$
$$= \{0\} \cup \{1\} \cup \{1\}$$
$$= \{0,1\} \cup \{1\} = \{0,1\}$$

The language described by the regular expression 1^+ is $L(\,(1)^+\,) = \{w_1 \cdots w_k \mid w_i \in L(1), k > 0\}$

Examples of strings in this language $= \{\underbrace{1 \cdots 1}_{k} \mid k > 0\}$

1
111   111111 $L((1)^+) = \{1^k \mid k > 0\}$.

The language described by the regular expression Σ_1^*1 is $L(\,\Sigma_1^*1\,) = \{\underbrace{\%\,\%\,\%\,\%\,\%}\,1\,\}$ (informal)
$$= \{w \in \Sigma_1^* \mid w \text{ ends in } 1\}$$
$$= \{u1 \mid u \in \Sigma_1^*\}.$$

$\Sigma_1 = \{0,1\}$

Examples of strings in this language
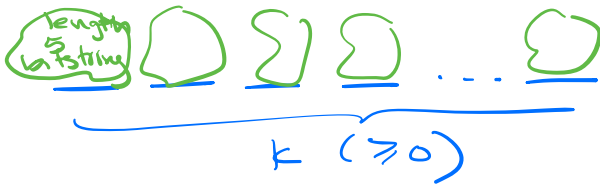
01   001   111   1 ← because $\varepsilon \in \Sigma^*$

The language described by the regular expression $(\Sigma_1\Sigma_1\Sigma_1\Sigma_1\Sigma_1)^*$ is $L(\,(\Sigma_1\Sigma_1\Sigma_1\Sigma_1\Sigma_1)^*\,) =$

↑↑↑↑ concatenation $\{w \in \Sigma_1^* \mid \text{length of } w \text{ is divisible by } 5\}$

length 5 list string

$k \, (\geq 0)$

$L(\Sigma_1) = \{0,1\}$ $L(\Sigma_1\Sigma_1) = \{00,01,10,11\}$
$L(\Sigma_1\Sigma_1\Sigma_1\Sigma_1\Sigma_1) = \{u \in \Sigma_1^* \mid \text{length of } u \text{ is } 5\}$

Examples of strings in this language

0000000000 1^{15} 00101 ε
$k=2$ $k=3$ $k=1$ $k=0$

00001 11011
$k=2$

A regular expression that describes the language $\{0^n1 \mid n \text{ is even}\}$ is

$$(00)^*1$$

string of 0's can be grouped into pairs, w/out any stragglers
00 00 00 _ _ _

Shorthand and conventions

$(01)^2 = (01) \circ (01) = 0101$

Assuming Σ is the alphabet, we use the following conventions	
Σ	regular expression describing language consisting of all strings of length 1 over Σ
$*$ then $\circ$ then $\cup$	precedence order, unless parentheses are used to change it
R_1R_2	shorthand for $R_1 \circ R_2$ (concatenation symbol is implicit)
R^+	shorthand for $R^* \circ R$ $L(R^+) = \{w_1 \cdots w_k \mid w_i \in R, k > 0\}$.
R^k	shorthand for R concatenated with itself k times, where k is a natural number
Pages 63 - 65	

Caution: k needs to be declared or specified ↑

Regular expressions are everywhere (once you start looking for them).

Software tools and languages often have built-in support for regular expressions to describe **patterns** that we want to match (e.g. Excel/ Sheets, grep, Perl, python, Java, Ruby).

Under the hood, the first phase of **compilers** is to transform the strings we write in code to tokens (keywords, operators, identifiers, literals). Compilers use regular expressions to describe the sets of strings that can be used for each token type.

Next time: we'll start to see how to build machines that decide whether strings match the pattern described by a regular expression.

Extra examples for practice:

Which regular expression(s) below describe a language that includes the string a as an element?

a^*b^*

$a(ba)^*b$

$a^* \cup b^*$

$(aaa)^*$

$(\varepsilon \cup a)b$

Review: Week 1 Wednesday

Please complete the review quiz questions on Gradescope about strings, languages, and regular expressions.

Recall: Review quizzes based on class material are assigned each day. These quizzes will help you track and confirm your understanding of the concepts and examples we work in class. Quizzes can be submitted on Gradescope as many times (with no penalty) as you like until the quiz deadline: the three quizzes each week are all due on Friday (with no penalty late submission open until Sunday).

The definition of the union, concatenation, and star operations for languages is given as Definition 1.23 on page 44 and a useful example is Example 1.24.

Pre class reading for next time: Figure 1.4, Definition 1.5

Notice: On Friday we are going back to the start of Chapter 1 and will be discussing finite automata and their computations.

Week 1 Friday

Review: Determine whether each statement below about regular expressions over the alphabet $\{a, b, c\}$ is true or false:

True or False: $a \in L(\ (a \cup b) \ \cup c)$

True or False: $ab \in L(\ (a \cup b)^* \ )$

True or False: $ba \in L(\ a^*b^* \ )$

True or False: $\varepsilon \in L(a \cup b \cup c)$

True or False: $\varepsilon \in L(\ (a \cup b)^* \ )$

True or False: $\varepsilon \in L(\ a^*b^* \ )$

From the pre-class reading, pages 34-36: A deterministic finite automaton (DFA) is specified by $M = (Q, \Sigma, \delta, q_0, F)$. This 5-tuple is called the **formal definition** of the DFA. The DFA can also be represented by its state diagram: with nodes for the state, labelled edges specifying the transition function, and decorations on nodes denoting the start and accept states.

Finite set of states Q can be labelled by any collection of distinct names. Often we use default state labels $q0, q1, \ldots$

The alphabet Σ determines the possible inputs to the automaton. Each input to the automaton is a string over Σ, and the automaton "processes" the input one symbol (or character) at a time.

The transition function δ gives the next state of the DFA based on the current state of the machine and on the next input symbol.

The start state q_0 is an element of Q. Each computation of the machine starts at the start state.

The accept (final) states F form a subset of the states of the DFA, $F \subseteq Q$. These states are used to flag if the machine accepts or rejects an input string.

The computation of a machine on an input string is a sequence of states in the machine, starting with the start state, determined by transitions of the machine as it reads successive input symbols.

The DFA M accepts the given input string exactly when the computation of M on the input string ends in an accept state. M rejects the given input string exactly when the computation of M on the input string ends in a nonaccept state, that is, a state that is not in F.

The language of M, $L(M)$, is defined as the set of all strings that are each accepted by the machine M. Each string that is rejected by M is not in $L(M)$. The language of M is also called the language recognized by M.

What is **finite** about all deterministic finite automata? (Select all that apply)

☐ The size of the machine (number of states, number of arrows)

☐ The number of strings that are accepted by the machine
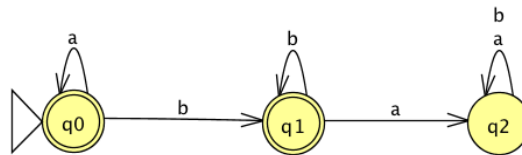
☐ The length of each computation of the machine
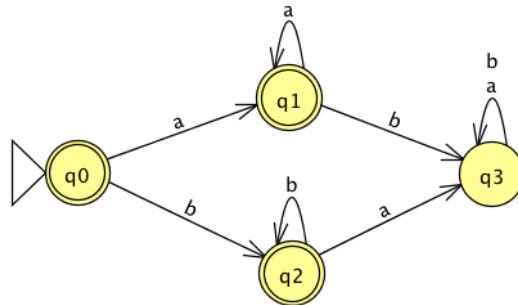
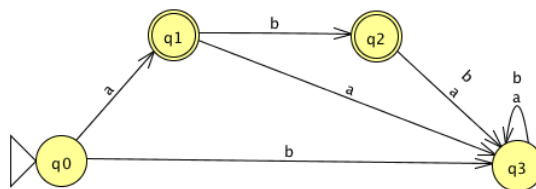The formal definition of this DFA is

Classify each string $a, aa, ab, ba, bb, \varepsilon$ as accepted by the DFA or rejected by the DFA.

Why are these the only two options?

The language recognized by this DFA is

The language recognized by this DFA is

The language recognized by this DFA is

Version April 2, 2023 (14)

Review: Week 1 Friday

Please complete the review quiz questions on Gradescope about DFA and the strings they accept and reject.

The definition of the union, concatenation, and star operations for languages is given as Definition 1.23 on page 44 and a useful example is Example 1.24.

The first homework assignment for CSE 105 this quarter is due next week. Find it on the class website: in the Calendar section of the website, click on the link under "Due Dates" for Week 2, or find the detailed assignment in the Assignments section of the website. Due: 4/11/23 at 5pm (no penalty late submission until 8am next morning)

We encourage you to work on homework in groups of up to three CSE 105 classmates. To find group members: reach out to people sitting around you in class, in discussion section, or during office hours. The pre-survey also asks if you want help finding group members: the CSE 105 instructional team can help you connect with other students. Working within the campus safety guidelines, you may choose to meet with your group mates in person or remotely. We highly recommend meeting synchronously so that you can work through the homework problems *together*.

Pre class reading for next time: pages 41-43 (figures 1.18, 1.19, 1.20)