Let's get started

We want you to be successful.

We will work together to build an environment in CSE 105 that supports your learning in a way that respects your perspectives, experiences, and identities (including race, ethnicity, heritage, gender, sex, class, sexuality, religion, ability, age, educational background, etc.). Our goal is for you to engage with interesting and challenging concepts and feel comfortable exploring, asking questions, and thriving.

If you are skipping and stretching meals, or having difficulties affording or accessing food, you may be eligible for CalFresh, California's Supplemental Nutrition Assistance Program, that can provide up to $292 a month in free money on a debit card to buy food. Students can apply at `benefitscal.com/r/ucsandiegocalfresh`. The Hub Basic Needs Center empowers all students by connecting them to resources for food, stable housing and financial literacy. Visit their site at `basicneeds.ucsd.edu`.

Financial aid resources, the possibility of emergency grant funding, and off-campus housing referral resources are available: see your College Dean of Student Affairs.

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. Here are some campus contacts that could provide this help: Counseling and Psychological Services (CAPS) at 858 534-3755 or http://caps.ucsd.edu; OPHD at 858 534-8298 or ophd@ucsd.edu , http://ophd.ucsd.edu; CARE at Sexual Assault Resource Center at 858 534-5793 or sarc@ucsd.edu , http://care.ucsd.edu.

Please reach out (minnes@ucsd.edu) if you need support with extenuating circumstances affecting CSE 105.

Introductions

Class website on Canvas https://canvas.ucsd.edu

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

Our team: One instructor + two TAs and five tutors + all of you

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

Welcome to CSE 105: Introduction to Theory of Computation in Winter 2025!

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   output: yes/no
- Determine if your code will compile   input: binary file   output: yes/no
- Find a run-time error in your code   input: binary file.   output: yes/no
- * Certify that your system is un-hackable   defining unhackable
  defining input   output: yes/no.

Which of these is hardest?

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   finite list of characters

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

Weeks 0 and 1 at a glance

Textbook reading: Chapter 0, Sections 1.3, 1.1

Before Monday, review class syllabus on Canvas (https://canvas.ucsd.edu/).

Before Wednesday, read Example 1.51.

Notice: we are jumping to Section 1.3 and then will come back to Section 1.1 next week.

Before Friday, read Definition 1.52 (definition of regular expressions) on page 64.

For Week 2 Monday: Figure 1.4 and Definition 1.5 (definition of finite automata) on pages 34-35.

Textbook references: Within a chapter, each item is numbered consecutively. Example 1.51 is the fifty-first numbered item in chapter one.

We will be learning and practicing to:

- Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.
  - Translate a decision problem to a set of strings coding the problem.
    - **Distinguish between alphabet, language, sets, and strings**
  - Use regular expressions and relate them to languages and automata.
    - **Write and debug regular expressions using correct syntax**
    - **Determine if a given string is in the language described by a regular expression**

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)

Review Quiz 1 on PrairieLearn (http://us.prairielearn.com), complete by 1/15/25

Create a homework group, possibly by using the Piazza (https://piazza.com/) find-a-teammate tool

Homework 1 submitted via Gradescope (https://www.gradescope.com/), due 1/16/25

Week 1 Monday: Terminology and Notation

The CSE 105 vocabulary and notation build on discrete math and introduction to proofs classes. Some of the conventions may be a bit different from what you saw before so we'll draw your attention to them.

For consistency, we will use the notation from this class' textbook[1].

These definitions are on pages 3, 4, 6, 13, 14, 53.

Notice? Wonder ?

Term	Typical symbol or Notation	Meaning
Alphabet	Sigma Gamma 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) empty string	ε	The (only) string of length 0
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
(The) empty language	$\emptyset$	The empty set, i.e. the set that has no strings (and no other elements either)
The power set of a set X	$\mathcal{P}(X)$	The set of all subsets of X
(The set of) natural numbers	$\mathcal{N}$	The set of positive integers
(Some) finite set		The empty set or a set whose distinct elements can be counted by a natural number
(Some) infinite set		A set that is not finite.
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

[1] Page references are to the 3rd edition of Sipser's Introduction to the Theory of Computation, available through various sources for approximately \$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@ucsd.edu)

Write out in words the meaning of the symbols below:

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

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

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

The number of distinct elements in the set {a, b, a} is 2.

$$|aba| = 3$$

strings

The length of string aba is 3.

Circle the correct choice:

A **string** over an alphabet Σ is an element of Σ^* ~~OR a subset of Σ^*~~.

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

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 $\Gamma = \{0, 1, x, y, z\}$

~~**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

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 $0 < 1$:

ε, 0, 1, 00, 01

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

ε, a, b, c, d

How do you look for a file?

Week 1 Wednesday: Regular expressions

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?

list all and only elements of finite sets

Roster $\{ x_1, x_2, x_3 \}$

give rules for membership

Set builder $\{ x \mid P(x) \}$

property

$\{ f(x) \mid P(x) \}$

result of function applied to x — property

Set operations : $\cup$, $\cap$, $\times$, $\mathcal{P}(\ )$, *

Named sets: $\mathbb{R}$ $\mathbb{N}$

set of real numbers — set of positive integers

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

set

Empty set

$\{ \}$

$\{ x \mid x \neq x \}$

$\emptyset$ \emptyset

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

alphabet

$\{ w \mid w \text{ is a string of length } n \text{ over } \{0,1\},\ n \geq 0,\ n \text{ integer} \}$

$\{0,1\}^* = \{ w_1 \cdots w_k \mid k \geq 0,\ k \text{ int},\ w_i \in \{0,1\} \}$

element of $\{0,1\}$

$\underset{\text{slot 1}}{\underline{0/1}}$ $\underset{\text{slot 2}}{\underline{0/1}}$ $\cdots$ $\underset{\text{slot } k}{\underline{0/1}}$

Note: $k = 0$ slots means $\varepsilon \in \{0,1\}^*$.

Which sets are "simple" to define ?

This definition was in the pre-class reading **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 over Σ is defined recursively:

Sets $\qquad \Sigma = \{0,1\}$

$\{0\} \qquad \{1\}$

$\{\varepsilon\}$

$\emptyset$

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:

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

union symbol

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

union operation

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

gluing together strings

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

setwise concatenation symbols

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

Kleene star operation

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

Kleene star symbol

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

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 $L(\varepsilon) = \{\varepsilon\}$

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

The language described by the regular expression $1^* \circ 1$ is $L(1^* \circ 1) =$

The language described by the regular expression 0 is the set whose only element is the string 0

$$L(1^* \circ 1) = L(1^*) \circ L(1)$$
$$= \{1\}^* \circ \{1\}$$
$$= \{w \mid w \text{ is formed only of 1s}\} \circ \{1\}$$
$$= \{1^i \mid i \text{ int}, i \geq 1\}$$
$$= \{1^{i+1} \mid i \text{ int}, i \geq 0\}.$$

The language described by the regular expression $((0 \cup 1) \circ (0 \cup 1) \circ (0 \cup 1))^*$ is $L(((0 \cup 1) \circ (0 \cup 1) \circ (0 \cup 1))^*) =$?

regular expression

the language being described by this regular expression

$L(0) = \{0\}$ ε is not an element of $\{0\}$

$L(1) = \{1\}$

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

$$L((0 \cup 1) \circ (0 \cup 1)) = \{uv \mid u \in \{0,1\}, v \in \{0,1\}\} =$$
$$\{11, 01, 00, 10\} = \{w \mid w \text{ is a length 2 string over } \{0,1\}\}$$

ε is not an element of this set

$$L((0 \cup 1) \circ (0 \cup 1) \circ (0 \cup 1)) = \{z \mid z \text{ is a length 3 string over } \{0,1\}\}$$

example elements are 000, 001

$$L(((0 \cup 1) \circ (0 \cup 1) \circ (0 \cup 1))^*) = \{w \mid w \text{ is a string over } \{0,1\} \text{ whose length is a multiple of 3}\}$$

example elements are 000, 110011, ε

length 3 string

Week 1 Friday: Regular expressions conventions

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

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

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

~~True~~ or False: $\varepsilon \in L(a \cup b \cup c)$ = $\{a, b, c\}$

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

~~True~~ or False: $\varepsilon \in L(\ aa^* \cup bb^*\ )$

$L((a \cup b)^*) = \{w \mid w \text{ is a string over } \{a,b\}\}$

$L(a^*b^*)$ is set of strings where have some # of a's followed by some # of b's

$L(aa^* \underline{\cup} bb^*)$

union

a^i, $i > 0$

b^j, $j > 0$

Shorthand and conventions (Sipser pages 63-65)

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 $R^* \circ R$ $\quad R \circ R^*$
R^k	shorthand for R concatenated with itself k times, where k is a (specific) natural number

Theorem: $L(R^+) \stackrel{\text{def}}{=} L(R^* \circ R) = L(R \circ R^*)$

Caution: many programming languages that support regular expressions build in functionality that is more powerful than the "pure" definition of regular expressions given here.

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.

Practice with the regular expressions over $\{a, b\}$ below.

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

a^*b^*

$a \in L(a^*b^*)$

$a(ba)^*b$

$a \notin L(a(ba)^*b)$

$a^* \cup b^*$

$a \in L(a^* \cup b^*)$

$(aaa)^*$

ε

aaa

$aaaaaa$

$a \notin L((aaa)^*)$

$(\varepsilon \cup a)b$

$a \notin L((\varepsilon \cup a)b)$

 Version January 4, 2025