CSE 105 — Theory of Computation

Notes taken by Nolan Chai

Winter 2023

Contents

1 Lecture 1 **5**
1.1 CSE 105's Big Questions 5
1.1.1 Computational Problems 5
1.1.2 Computation Models 5
1.1.3 Vocabulary Review 5
1.2 Language Operations 6
1.2.1 Union 6
1.2.2 Concatenate 6
1.2.3 Kleene Star 6
1.3 Regular Expressions 7

2 Lecture 2 **8**
2.1 Regular Expressions (Cont.) 8
2.2 The Alphabet 9

3 Review Quiz 1 **10**

Preface

These are a collection of notes personally taken by me, specifically for readings and allotted content for UCSD's CSE 105 Theory of Computation taken in Winter 2023. These notes are not endorsed by the lecturers nor staff, and I have modified them (often significantly) over random periods of time. They may become nowhere near accurate representations of what was actually lectured, or written in the books, and are simply to aid in my own understanding. In particular, all errors are almost surely mine.
Notes will be reviewed, updated, and revised within 48 hours of each lecture.

Introduction

Course covered by Miles Jones, more geared towards theory and mathematics rather than programming. In general, the major course topics and contents covered by these notes will include finite automata, pushdown automata, and Turing machines.

Logistics

All lectures will be posted before class on the content calendar tab of the canvas site (also podcasted / available via Canvas).
There will be **short review quizzes due every Sunday at midnight**, with a total of 8.

Exams

- Two tests on the Fridays of Week 4 and 8, being asynchronous.
- One midterm on Thursday of Week 6
- One Final Exam

For each exam, you have the option of taking a retest with unlimited atttempts, potentially earning 1/3rd of the points you missed on the exam. Official exam score

$$\text{MAX}(\text{Test}, 2 * \text{Test} + \text{ReTest})/3$$

Discussions are not mandatory, but highly recommended, and are on Wednesdays 2-2:50 PM.
Office hours will become essential at one point. Office hours include both regular office hours and 1-1s that you can sign up for. Office hours are usually for what's coming up, while 1-1s are for catching up.
Three Grading Options:
Option 1:
Review Quizzes: 5%, Homework: 30%, Each Exam: 10%, Midterm: 15%, Final: 30%
Option 2:
Review Quizzes: 5%, Homework: 30%, Each Exam: 10%, Midterm: 0%, Final: 45%
Option 3:
Review Quizzes: 5%, Homework: 30%, Each Exam: 10%, Midterm: 15%, Final: 40%
Jflap and flap.js are useful for drawing automata. Also `https://www3.nd.edu/~kogge/courses/cse30151-fa17/Public/other/tikz_tutorial.pdf` to draw automata.

1 Lecture 1

1.1 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?

1.1.1 Computational Problems

- Find a file on your computer
- Determine if your code will compile
- Find a run-time error in your code
- Certify that your system is un-hackable

Which one of these problems is the hardest?
We can quantify how hard an operation is - referred to as the 'hardness' - based on how many resources it uses, which is part of what we will focus on in this class.

1.1.2 Computation Models

A computational model has the following structure: You input something, like a string, and it will output a yes or a no in response. More specifically, our output will be based on whether or not the input string matches a specific pattern. Assume that whatever input you have has already been encoded.

1.1.3 Vocabulary Review

- Set: a collection of "objects"
- Character/Symbol: a, b, c, 0, 1, @, ...
- Alphabet: A non-empty set of symbols (usually denoted with Σ).
- Word/String: A sequence of symbols
  - 00010110
  - Aazya00
- Language: A set of strings

Examples:

$\{a, b, c, d, e\}$	The set whose elements are a,b,c,d,e
$\lvert ababab \rvert = 6$	The length of the string ababab is 6
$\lvert a, b, c, d, e \rvert = 5$	The size/cardinality of the set a, b, c, d, e is 5

1.2 Language Operations

1.2.1 Union

Let r_1 and r_2 be languages over the alphabet Σ.
$r_1 \cup r_2$ is the language that includes any string that is either in r_1 or r_2.
Formally, this can be represented as:

$$r_1 \cup r_2 = \{w | w \in r_1 \vee w \in r_2\}$$

Example. What is $\{aa, ab, ba, a\} \cup \{bb, b, ba, \epsilon\}$?

$$\begin{aligned} &\{aa, ab, ba, a\} \cup \{bb, b, ba, \epsilon\} \\ &= |\{aa, ab, ba, a, bb, b, ba, \epsilon\}| \\ &= |\{aa, ab, ba, a, bb, b, ba\}| \\ &= 7 \end{aligned}$$

1.2.2 Concatenate

Let r_1 and r_2 be languages over the alphabet Σ.
$r_1 \circ r_2$ is the language that includes any string from r_1 concatenated with any string from r_2.
Formally, this can be represented as:

$$r_1 \circ r_2 = \{w_1 \circ w_2 | w_1 \in r_1 \wedge w_2 \in r_2\}$$

Example. What is $\{ab, ba, a\} \circ \{bb, b, \epsilon\}$? (Reminder that $\Sigma = \{a, b\}$

$$\{ab, ba, a\} \circ \{bb, b, \epsilon\} = |\{abbb, abb, ab, babb, bab, ba, a\}| = 7$$

1.2.3 Kleene Star

The Kleene star is made up of all concatentations within itself. Formally, the Kleene Star on the set S, S^* is the set of all finite strings made up of concatenations of elements of S.
Given the set $S = \{a, b\}$, a, b^* would include the following:

- The empty string ϵ
- a, aa, aaa
- b, bb, bbb
- ab, ababab, aaaaaaabbb, and so forth

The set would have infinitely many different elements (for most cases), but does **not** include infinite sequences of a's and b's (i.e., a^∞, b^∞).
Formally, this can be represented as:

$$S^* = \{w_1 \circ w_2 \circ ... \circ w_n | w_1, w_2, ..., w_n \in S \wedge n \geq 0\}$$

For two cases, the Kleene star could produce a finite language. When the language is empty, and a language with just an empty string g will produce a finite language instead of an infinite language.

1.3 Regular Expressions

Regular expressions give us syntax to describe languages. For all isntances, suppose that the Alphabet here is $\{0, 1\}$.
The basic symbols in regular expressions are: $\emptyset, \epsilon, 0, 1$

- $\emptyset$ means $\emptyset$
- ϵ means $\{\epsilon\}$
- 0 means $\{0\}$
- 1 means $\{1\}$

Basic operations include: $\cup$, $circ$, $*$, with parentheses to group.
Note: Often, $\circ$ is not used. We'd often rather write aab rather than $a \circ a \circ b$, or $(a \cup b)(c \cup b)$ rather than $(a \cup b) \circ (c \cup b)$.

Example. How would you write these languages using regular expressions?

- $\{a, b, c\} = L(a \cup b \cup c)$
- $\{a^n | n \geq 0\} = \{\epsilon, a, aa, aaa, ...\} = L(a^*) = L(aa^*)$
- $\{a^n | n \geq 1\} = \{a, aa, aaa...\} = L(a^* \circ a) = L(a^*a)$
- $\{a, ab, abb, abbb, b, bb, bbb\} = L(a \cup ab \cup ...) = L((a \cup \epsilon) \circ (b \cup bb \cup bbb) \cup a)$

2 Lecture 2

2.1 Regular Expressions (Cont.)

A regular expression is a string of symbols that *describes* a language. You can think of stuff like single characters, empty string, the empty set, as atomic elements, strung together and recursively built up using our main three operators - union, concatenation, and Kleene star. Formally, if R_1 and R_2 are regular expressions, then

- $R_1 \cup R_2$ is a regular expression that *describes* the union of the languages described by R_1, R_2.
- $R_1 \circ R_2$ (or more commonly written as $R_1 R_2$ is a regular expression that *describes* the concatenation of the languages described by R_1, R_2.
- R_1^* is a regular expression that *describes* the Kleene star of the language described by R_1.

R is defined as a regular expression over the alphabet Σ if

1. $R = a$, where $a \in \Sigma$
2. $R = \epsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$, where R_1, R_2 are themselves regular expressions.
5. $R = (R_1 \circ R_2)$, where R_1, R_2 are themselves regular expressions.
6. (R_1^*), where R_1 is a regular expression.

Notes: Σ is shorthand for $(0 \cup 1)$ if $\Sigma = 0, 1$. Parentheses may be omitted. R^k means R concatenated with itself k times.

So one of our first big questions we're going to try to tackle is that:
We know how to build these regular expressions. We've seen that they can describe languages. We've seen some languages that they can describe. How far can we push it? What are the languages that we can describe by regular expressions? Can you describe it? Any languages? Any language? Are there languages out there that are too complicated to be described by regular expressions? So kinda keep this in your mind as we go.

Example. Let's say we have the set $A = \{1, 01, 001, ...\}$.
Then 0^*1 is the *regular expression* that describes A. It is clear that 0^*1 is a regular expression because $*$ only acts on sets. However, we **DO NOT SAY THAT** $0^*1 = A$. Instead, we say that $\boxed{L(0^*1) = A}$, where L stands for *"The Language of..."*.

Note: the author of the Theory of Computation textbook that we are using doesn't do this and is a bit more loose with their writing. However, we will be more stringent with this kind of notation.

Furthermore, there is an order of operations to the regular expressions described.

1. Kleene Star
2. Concatenation
3. Union

Example. So if we had the following two sets B and C, what can you say about them? Are they the same or different? If they are different, then how do they differ?

$$\text{Let } B = L(00^*1) \text{ and let } C = L((00)^*1)$$

If we were to expand these expressions, we'd have:

$$B = \{01, 001, 0001, 00001, ...\}, C = \{1, 001, 00001, 0000001\}$$

As you can see, the string 1 is contained in C, but not contained in B, and some strings from B are not contained in C.

2.2 The Alphabet

Often, we will use the symbol Σ to be the alphabet that we are using. *This can differ from problem to problem*, so we will define it each time.

The most common alphabets you will see in this class are: $\{0, 1\}$ or $\{a, b\}$. If Σ is an alphabet, then Σ^* is the set of all finite strings using symbols from the alphabet Σ.

Exercise. If $\Sigma = \{1, a, \$\}$, then what is Σ^*?

$$Sigma^* = \{\epsilon, 1, a, \$, a\$, ...\}$$

3 Review Quiz 1

Highly recommended to complete the quiz before reading.