Theory of Computation

- Studying various computing devices and comparing their powers
  - It answers questions like what can be computed (by what devices) and what cannot.
- Space and Time: Measure them qualitatively, not quantitatively.
  - Do they need a finite amount of space?
  - Does the computing ever stop?
- Looking for the most powerful computing devices
- Identifying problems which can or cannot be computed (by what devices)

How to Represent a Problem?

- Computing involves different inputs, e.g., integers, reals, characters, strings, …
- The presentation needs to be simple and expressive.
- Anything can be represented by a string of characters
- Strings and numbers can be represented by each other.
  - each number is entered by a sequence of key strokes;
  - each string is stored as a binary number in a computer.
- Representation of integers:
  - Unary (e.g., 0^5 for 5)
  - Binary (e.g., 101 for 5)
Characters and Strings

\( \Sigma \) — a non-empty, finite set of (abstract) symbols, called the alphabet, whose elements are also called letters (or characters)

\( \Sigma^* \) — (infinite) set of all finite sequences over \( \Sigma \), called words (or strings)

A word \( w \in \Sigma^* \) has length, \( \text{len}(w) \), the number of letters in the sequence \( w \) (0, 1, 2, …)

\( \varepsilon \) denotes the empty (or null) string with \( \text{len}(\varepsilon) = 0 \).

\( 0^n \) denotes \( n \) copies of 0; \( 0^0 = \varepsilon \).

Formal Language Operations

A formal language \( L \) is a subset of \( \Sigma^* \), \( L \subseteq \Sigma^* \)

The empty set is written as \( \emptyset \).

\( \varepsilon \), \( \emptyset \), and \( \{\varepsilon\} \) are all distinct entities

Languages can be combined with the familiar set operations: union, intersection, and complement

Languages can also be concatenated by forming all the possible combinations of concatenations of their member strings. For \( L_1, L_2 \subseteq \Sigma^* \),

\( L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\} \)

Power of Languages

Language concatenation behaves as a “multiplicative” operator

\( \emptyset \) is a “zero” and

\( \{\varepsilon\} \) is an “identity”.

In this sense we define powers of a language.

For \( L \subseteq \Sigma^* \) define \( L^0 = \{\varepsilon\} \), and inductively for \( n \geq 0 \)

\( L^{n+1} = L^n \cdot L \). Language powers satisfy the usual laws of exponents — \( L^n \cdot L^m = L^{n+m} \).
Kleene Closure

In terms of these powers we define two other language operations. The Kleene closure (or star) of $L \subseteq \Sigma$ is

$$L^* = \bigcup_{n=0}^{\infty} L^n$$

The positive closure is

$$L^+ = \bigcup_{n=1}^{\infty} L^n$$

Listing Strings in $\Sigma^*$

If you write a program to list all the strings in $\Sigma^*$ how can you print everyone without missing any?

- **Dictionary order:**
  - Eg., $\varepsilon$, 0, 00, 000, .., 0000001, .., 1, 10, 100, ..

- **Canonical Order:**
  - List the shortest strings first;
  - For the strings of the same length, list them in the dictionary order.

Countably Infinite Sets

- The size of a set is called the **cardinality** of the set.
- The cardinality of a finite set is the number of elements in the set, also called **size**.
- Two sets have the same cardinality if there is a one-to-one correspondence.
- A set is **countably infinite** if its elements can be enumerated one by one.
- The set $N$ of natural numbers are countably infinite.
Countably Infinite Sets

- Theorem: A set is countable infinite iff it has the same cardinality as the set $\mathbb{N}$ of natural numbers.
- The following sets are countably infinite:
  - The set of even numbers
  - Any infinite subset of $\mathbb{N}$
  - The set of all integers
  - $\{ a \}^*$
  - $\{ a, b \}^*$
  - For any alphabet $\Sigma, \Sigma^*$

Non-Countably Infinite Sets

- The set of binary strings with infinite length
- The set of real numbers in $[0, 1)$
- The power set of $\mathbb{N}$
- The set of all functions on $\mathbb{N}$

The “Language” of Regular Expressions

The regular expressions over an alphabet $\Sigma$ comprise a collection of formal notations. Each notation represents a formal language.

- (Meta) Language of regular expressions:
  - Syntax: rules of formation
  - Semantics: the associated language $L \subseteq \Sigma^*$. 

**Regular Expression Syntax**

Regular expressions involve the symbols from $\Sigma$ plus several auxiliary symbols, and are defined inductively.

**Basis case(s):**
(a) $\emptyset$ is a regular expression
(b) $\epsilon$ is a regular expression
(c) each $\lambda \in \Sigma$ is a regular expression

**Inductive case(s):**
If $\alpha$ and $\beta$ are regular expressions, then so are:
(a) $(\alpha + \beta)$
(b) $(\alpha \cdot \beta)$
(c) $(\alpha^*)$

**Regular Expression Semantics**

The meaning of each regular expression $\alpha$ is a language $\mu(\alpha) \subseteq \Sigma^*$ whose definition parallels the inductive definition of the syntax.

**Basis case(s):**
(a) $\mu(\emptyset) = \emptyset$
(b) $\mu(\epsilon) = \{\epsilon\}$
(c) $\mu(\lambda) = \{\lambda\}$

**Inductive case(s):**
If $\alpha$ and $\beta$ are regular expressions, then:
(a) $\mu(\alpha + \beta) = \mu(\alpha) \cup \mu(\beta)$
(b) $\mu(\alpha \cdot \beta) = \mu(\alpha) \cdot \mu(\beta)$
(c) $\mu(\alpha^*) = (\mu(\alpha))^*$

**Regular Languages**

$L \subseteq \Sigma$ is a regular language if there exists a regular expression $\alpha$ so that $L = \mu(\alpha)$

Regular expressions $\alpha$ and $\beta$ are equivalent, $\alpha = \beta$, provided that $\mu(\alpha) = \mu(\beta)$

To avoid excessive parentheses in regular expressions, we assign precedence to the operations: $\ast$ highest, $\cdot$ intermediate, and $+$ lowest; also, $\cdot$ may be omitted. For instance, the regular expression $(0^*1^*)$ is written $01^*$. 
Example 1.1.3 (h) — 0 *(10*10*)*
The language described by this regular expression contains the strings:
\( \epsilon, 0, 00, 000, \ldots \)
11, 110, 101, 1100, 1001, 1010, 11000, …
011, 0110, 0101, 01100, 01001, …
Informally, all binary strings with an even number of '1's.

Example 1.1.3 (g) — 00+11+101
This regular expression describes \{00, 11, 101\}.

Example 1.1.3 (i) — 0 *1*
The meaning in this case is the language \( \{\epsilon, 0, 1, 00, 01, 11, 000, 001, 011, 111, \ldots \} = \{0^{p+q} \mid p,q \geq 0\} \).
Briefly, and informally, all strings where '1' is followed by '1' or nothing.

The language described by this regular expression contains the strings:
\( \epsilon, 0, 00, 000, \ldots \)
11, 110, 101, 1100, 1001, 1010, 11000, …
011, 0110, 0101, 01100, 01001, …
Informally, all binary strings with an even number of '1's.

Theorem 1.1.1: For any regular expressions \( \alpha, \beta \) and \( \gamma \),
(i) \( \alpha + \beta = \beta + \alpha \),
(ii) \( (\alpha + \beta) + \gamma = \alpha + (\beta + \gamma) \),
(iii) \( \emptyset + \alpha = \alpha + \emptyset = \alpha \),
(iv) \( (\alpha \beta) \gamma = \alpha (\beta \gamma) \),
(v) \( \alpha (\beta + \gamma) = \alpha \beta + \alpha \gamma \),
(vi) \( (\alpha + \beta) \gamma = \alpha \gamma + \beta \gamma \),
(vii) \( \epsilon \alpha = \alpha \epsilon = \alpha \),
(viii) \( \emptyset \alpha = \emptyset = \emptyset \),
(ix) \( \emptyset^* = \epsilon \),
(x) \( (\alpha + \epsilon)^* = \alpha^* \),
(xi) \( \alpha (\beta \alpha)^* = (\alpha \beta)^* \alpha \),
(xii) \( (\alpha^*)^* = \alpha^* \),
(xiii) \( (\alpha^+)^* = (\alpha + \beta)^* \),
(xiv) \( (\alpha^*)^* = \epsilon + (\alpha + \beta)^* \).

Assertion: The regular languages are the smallest family of languages containing the finite languages and closed under union, concatenation, and star.
- each finite language is regular
- regular languages are “closed under” union, concatenation, and star
- any family of languages that contains the finite languages and is closed under union, concatenation and star also contains all the regular languages

Assertion: A regular expression denotes an infinite language only if it includes *, and \( \mu(\alpha^*) \) is infinite unless \( \alpha = \emptyset \) or \( \alpha = \epsilon \).
Language Transformations
Let \( \Sigma \) and \( \Delta \) be alphabets. A function \( h: \Sigma \rightarrow \Delta^* \) is a homomorphism.

A homomorphism may be extended uniquely as a function \( h: \Sigma^* \rightarrow \Delta^* \):
\[
h(\varepsilon) = \varepsilon, \text{ and } h(x \lambda) = h(x) h(\lambda) \text{ for all } x \in \Sigma^* \text{ and } \lambda \in \Sigma.
\]

Finally, homomorphism may be applied to languages, \( h: p(\Sigma^*) \rightarrow p(\Delta^*) \), by the element-wise extension (\( p(S) = \text{power set of } S \)):
\[
h(L) = \{h(x) | x \in L\} \text{ for each } L \subseteq \Sigma^*.
\]

Let \( \Sigma \) and \( \Delta \) be alphabets. A function \( s: \Sigma \rightarrow p(\Delta^*) \) is a substitution (i.e., each letter of \( \Sigma \) is associated with a language over \( \Delta \)). A substitution is called regular if each of the languages \( s(\lambda) \) is regular.

A substitution may be also extended as a function \( s: \Sigma^* \rightarrow p(\Delta^*) \) as inductively defined by \( s(\varepsilon) = \{\varepsilon\} \), and \( s(x \lambda) = s(x) s(\lambda) \) for all \( x \in \Sigma^* \) and \( \lambda \in \Sigma \).

Finally, substitutions may be applied to languages, \( s: p(\Sigma^*) \rightarrow p(\Delta^*) \), by element-wise extension:
\[
s(L) = \bigcup_{x \in L} s(x).
\]

Every homomorphism \( h \) can be regarded as a substitution \( h' \) where if \( h(\lambda) = w \), \( h'(\lambda) = \{w\} \).

Homeomorphisms map a letter to one string.

Substitutions map a letter to set of strings.

The latter may be regarded as “non-deterministic” homomorphisms.

Every homomorphism has a finite description, but a substitution need not have a finite description. If the languages associated with letters are infinite, a finite description of the substitution may be impossible.
Theorem 1.1.3: For each regular language \( R \subseteq \Sigma^* \) and each regular substitution \( s \), \( s(R) \) is regular.

Corollary 1.1.4: For each regular language \( R \subseteq \Sigma^* \) and each homomorphism \( h \), \( h(R) \) is regular.

Closure Properties of Languages

A set of languages is said to be closed for an operation if the result of the operation applying to members of the set is also in the set.

Theorem: the set of regular languages is closed under union, (set) concatenation, Kleene closure (star), regular substitution, and homomorphism.