The Incomplete Codex of Basic Mathematics for Computer Scientists

From Programmers to Hackers: Mathematical Basis to Computer Science

None(@n0n3x1573n7), jh05013(@jhznktrkstlhknm)

January 16, 2019

Chapter 1

Introduction

Let's face it: mathematics is hard.

But as a computer scientist, you need to know the principle of mathematics. And we know, it's not easy. The many mathematical principles are dispersed throughout many areas of mathematics, whether it is number theory, calculus, analysis, or statistics.

This book aims to give some help to computer scientists who are tired of searching the highly dispersed information on the net or in the books. This includes the theoretical parts of computer science, such as graph, language, and complexity theories.

In the first part, mathematical preliminaries, we see the important parts from many parts of mathematics as mentioned above. This may not be directly related to any algorithms, but this will serve as a basis for many theoretical parts of computer science.

In the second part, theory-heavy parts of computer science are described as mathematically precise as possible.

Contents

Part I

Mathematical Preliminaries

Chapter 2

Logic

Chapter 3

Algebraic Structures

3.1 Algebraic Structures

3.1.1 Sets

Definition 1 (Set)
A set is a collection of distinct objects.

To see some traits on sets, we literally start from nothing:

Axiom 2 (Empty Set Axiom)
There is a set containing no members, that is:

$$\exists B \text{ such that } \forall x, (x \notin B)$$

We call this set the empty set, and denote it by the symbol $\emptyset$.

We now have $\emptyset$; we now write down a few rules for how to manipulate sets.

Axiom 3 (Axiom of Extensionality)
Two sets are equal if and only if they share the same elements, that is:

$$\forall A, B[\forall z, ((z \in A) \Leftrightarrow (z \in B)) \Rightarrow (A = B)]$$

Axiom 4 (Axiom of Pairing)
Given any two sets A and B, there is a set which have the members just A and B, that is:

$$\forall A, B \exists C \forall x[x \in C \Leftrightarrow ((x = A) \vee (x = B))]$$

If A and B are distinct sets, we write this set C as $\{A, B\}$; if $A = B$, we write it as $\{A\}$.

Axiom 5 (Axiom of Union, simple version)
Given any two sets A and B, there is a set whose members are those sets belonging to either A or B, that is:

$$\forall A, B \exists C \forall x[x \in C \Leftrightarrow ((x \in A) \vee (x \in B))]$$

We write this set C as $A \cup B$.

In the simplified version of Axiom of Union, we take union of only two things, but we sometimes we want to take unions of more than two things or even more than finitely many things. This is given by the full version of the axiom:

Axiom 6 (Axiom of Union, full version)
Given any set A, there is a set C whose elements are exactly the members of the members of A, that is:

$$\forall A \exists C[x \in C \Leftrightarrow (\exists A'(A' \in A) \wedge (x \in A'))]$$

We denote this set C as

$$\bigcup_{A' \in A} A'$$

Axiom 7 (Axiom of Intersection, simple version)
Given any two sets A and B, there is a set whose members are member of both A and B, that is:

$$\forall A, B \exists C \forall x[(x \in C) \Leftrightarrow ((x \in A) \wedge (x \in B))]$$

Sometimes as union, we would want to take intersection of more than finitely many things. This is given by the full version of the axiom:

Axiom 8 (Axiom of Intersection, full version)
Given any set A, there is a set C whose elements are exactly the members of all members of A, that is:

$$\forall A \exists C \forall x[(x \in C) \Leftrightarrow (\forall A'((A' \in A) \Rightarrow (x \in A')))]$$

We denote this set C as

$$\bigcap_{A' \in A} A'$$

Axiom 9 (Axiom of Subset)
For any two sets A and B, we say that $B \subset A$ if and only if every member of B is a member of A, that is:

$$(B \subseteq A) \Leftrightarrow (\forall x(x \in B) \Rightarrow (x \in A))$$

By the Axiom of Subset we can define the power set of an any given set:

Definition 10 (Power Set)
For any set A, the power set of the set A, denoted $P(A)$, whose members are precisely the collection of all possible subsets of A, that is:

$$\forall A \exists P(A) \forall B((B \subseteq A) \Leftrightarrow (B \in P(A)))$$

Definition 11 (Equivalence Relation)
Let S be a set. An Equivalence Relation on S is a relation, denoted by ~, with the following properties, $\forall a, b, c \in S$:

- **Reflexivity** $a \sim a$
- **Symmetry** $a \sim b \Leftrightarrow b \sim a$
- **Transitivity** $(a \sim b) \wedge (b \sim c) \Rightarrow (a \sim c)$

Definition 12 (Setoid)
A setoid is a set in which an equivalence relation is defined, denoted $(S, \sim)$.

Definition 13 (Equivalence Class)
The equivalence class of $a \in S$ under ~, denoted $[a]$, is defined as $[a] = \{b \in S | a \sim b\}$.

Definition 14 (Order)
Let S be a set. An order on S is a relation, denoted by $<$, with the following properties:

- If $x \in S$ and $y \in S$ then one and only one of the following statements is true:

$$x < y, x = y, y < x$$

- For $x, y, z \in S$, if $x < y$ and $y < z$, then $x < z$.

Remark

- It is possible to write $x > y$ in place of $y < x$
- The notation $x \leq y$ indicates that $x < y$ or $x = y$.

Definition 15 (Ordered Set)
An ordered set is a set in which an order is defined, denoted $(S, <)$.

Definition 16 (Bound)
Suppose S is an ordered set, and $E \subset S$.
If there exists $\beta \in S$ such that $x \leq \beta$ for every $x \in E$, we say that E is bounded above, and call β an upper bound of E. If there exists $\alpha \in S$ such that $x \geq \alpha$ for every $x \in E$, we say that E is bounded below, and call α a lower bound of E.

Definition 17 (Least Upper Bound)
Suppose that S is an ordered set, and $E \subset S$. If there exists a $\beta \in S$ with the following properties:

- β is an upper bound of E
- If $\gamma < \beta$, then γ is not an upper bound of E

Then β is called the Least Upper Bound of E or the supremum of E, denoted

$$\beta = sup(E)$$

Definition 18 (Greatest Lower Bound)
Suppose that S is an ordered set, and $E \subset S$. If there exists a $\alpha \in S$ with the following properties:

- α is a lower bound of E
- If $\gamma < \alpha$, then γ is not an lower bound of E

Then α is called the Greatest Lower Bound of E or the infimum of E, denoted

$$\beta = inf(E)$$

Definition 19 (least-upper-bound property)
An ordered set S is said to have the least-upper-bound property if the following is true:
if $E \subset S$, E is not empty, and E is bounded above, then $sup(E)$ exists in S.

Definition 20 (greatest-lower-bound property)
An ordered set S is said to have the greatest-lower-bound property if the following is true:
if $E \subset S$, E is not empty, and E is bounded below, then $inf(E)$ exists in S.

Theorem 21
Suppose S is an ordered set with the least-upper-bound property, $B \subset S$, B is not empty, and B is bounded below.
Let L be the set of all lower bounds of B. Then

$$\alpha = sup(L)$$

exists in S, and $\alpha = inf(B)$.

Proof. Note that $\forall x \in L, y \in B, x \leq y$.
L is nonempty as B is bounded below.
L is bounded above since $\forall x \in S \backslash L, \forall y \in L, x > y$.
Since S has the least-upper-bound property and $L \subset S$, $\exists \alpha = sup(L)$.
The followings hold:

- α is a lower bound of B.
  ($\because$) $\forall \gamma \in B, \gamma > \alpha$
- β with $\beta > \alpha$ is not a lower bound of B
  ($\because$) Since α is an upper bound of L, $\beta \notin L$.

Hence $\alpha = inf(B)$. □

Corollary 22
For all ordered sets, the Least Upper Bound property and the Greatest Lower Bound Porperty are equivalent.

3.1.2 Group

Definition 23 (Group)
A group is a set G with a binary operation $\cdot$, denoted $(G, \cdot)$, which satisfies the following conditions:

- **Closure:** $\forall a, b \in G, a \cdot b \in G$
- **Associativity:** $\forall a, b, c \in G, (a \cdot b) \cdot c = a \cdot (b \cdot c)$
- **Identity:** $\exists e \in G, \forall a \in G, a \cdot e = e \cdot a = a$
- **Inverse:** $\forall a \in G, \exists a^{-1} \in G, a \cdot a^{-1} = a^{-1} \cdot a = e$

Definition 24 (Semigroup)
A semigroup is $(G, \cdot)$, which satisfies Closure and Associativity.

Definition 25 (Monoid)
A monoid is a semigroup $(G, \cdot)$ which also has identity.

Definition 26 (Abelian Group)
An Abelian Group or Commutative Group is a group $(G, \cdot)$ with the following property:

- **Commutativity:** $\forall a, b \in G, a \cdot b = b \cdot a$

3.1.3 Ring

Definition 27 (Ring)
A Ring is a set R with two binary operations $+$ and $\cdot$, often called the addition and multiplication of the ring, denoted $(R, +, \cdot)$, which satisfies the following conditions:

- $(R, +)$ is an abelian group
- $(R, \cdot)$ is a semigroup
- **Distribution:** $\cdot$ is distributive with respect to $+$, that is, $\forall a, b, c \in R$:
  - $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
  - $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$

The identity element of $+$ is often noted 0.

Definition 28 (Ring with identity(1))
A Ring with identity is a ring $(R,+,\cdot)$ of which $(R,\cdot)$ is a monoid. The identity element of $\cdot$ is often noted 1.

Definition 29 (Commutative Ring)
A commutative ring is a ring $(R,+,\cdot)$ of which $\cdot$ is commutative.

Definition 30 (Zero Divisor)
For a ring $(R,+,\cdot)$, let 0 be the identity of $+$.
$a,b \in R$, $a \neq 0$ and $b \neq 0$, if $a \cdot b = 0$, a,b are called the zero divisors of the ring.

Definition 31 (Integral Domain)
An integral domain is a commutative ring $(R,+,\cdot)$ with 1 which does not have zero divisors.

3.1.4 Field

Definition 32 (Field)
A Field is a set F with two binary operations $+$ and $\cdot$, often called the addition and multiplication of the field, denoted $(R,+,\cdot)$, which satisfies the following conditions:

- $(F,+,\cdot)$ is a ring
- $(F\backslash\{0\},\cdot)$ is a group

Alternatively, a Field may be defined with a set of Field Axioms listed below:

(A) **Axioms for Addition**

(A1) **Closed under Addition**
$\forall a,b \in F, a+b \in F$

(A2) **Addition is Commutative**
$\forall a,b \in F, a+b = b+a$

(A3) **Addition is Associative**
$\forall a,b,c \in F, (a+b)+c = a+(b+c)$

(A4) **Identity of Addition**
$\exists 0 \in F, \forall a \in F, 0+a = a$

(A5) **Inverse of Addition**
$\forall a \in F, \exists -a \in F, a+(-a) = 0$

(M) **Axioms for Multiplication**

(M1) **Closed under Multiplication**
$\forall a,b \in F, a \cdot b \in F$

(M2) **Multiplication is Commutative**
$\forall a,b \in F, a \cdot b = b \cdot a$

(M3) **Multiplication is Associative**
$\forall a,b,c \in F, (a \cdot b) \cdot c = a \cdot (b \cdot c)$

(M4) **Identity of Multiplication**
$\exists 1 \in F, \forall a \in F, 1 \cdot a = a$

(M5) **Inverse of Multiplication**
$\forall a \in F \backslash \{0\}, \exists a^{-1} \in F, a \cdot a^{-1} = 1$

(D) **Distributive Law**
$\forall a, b, c \in F, (a + b) \cdot c = a \cdot c + b \cdot c$
where $\cdot$ takes precedence over $+$.

Definition 33 (Ordered Field)
An ordered field is a field F which is an ordered set, such that the order is compatible with the field operations, that is:

- $x + y < x + z$ if $x, y, z \in F$ and $y < z$
- $xy > 0$ if $x, y \in F$, $x > 0$ and $y > 0$

3.1.5 Polynomial Ring

Definition 34 (Polynomial over a Ring)
A polynomial $f(x)$ over the ring $(R, +, \cdot)$ is defined as

$$f(x) = \sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x^1 + \cdots, a_i \in R$$

where $a_i = 0$ for all but finitely many values of i.
The degree of the polynomial $\deg(f)$ is defined as $\deg(f) = \max\{n | n \in \mathbb{N}, a_n \neq 0\}$.
The leading coefficient of the polynomial is defined as $a_{\deg(f)}$.

Definition 35 (Addition and Multiplication of Polynomials)
Let $f(x) = \sum_{i=0}^{\infty} a_i x^i$, $g(x) = \sum_{i=0}^{\infty} b_i x^i$, $a_i, b_i \in R$ be a polynomial over the ring $(R, +, \cdot)$. Define:

$$f(x) + g(x) = \sum_{i=0}^{\infty} (a_i + b_i) x^i$$

$$f(x)g(x) = \sum_{k=0}^{\infty} (c_k) x^k \text{ where } c_k = \sum_{i+j=k} a_i b_j$$

Definition 36 (Polynomial Ring)
The set of polynomials over the ring $(R, +, \cdot)$, $R[x] = \{f(x) | f(x)$ is a polynomial over $R\}$ is called the Polynomial Ring(or Polynomials) over R.

Theorem 37 (Degree of Polynomial on Addition and Multiplication)
Let $f(x), g(x) \in R[x]$ with $\deg(f) = n$, $\deg(g) = m$.

- $0 \leq \deg(f + g) \leq \max(\deg(f), \deg(g))$
- $\deg(fg) \leq \deg(f) + \deg(g)$.
  If $(R, +, \cdot)$ is an integral domain, $\deg(fg) = \deg(f) + \deg(g)$

Theorem 38 (Relationship between a Ring and its Polynomial Ring)
Let $(R, +, \cdot)$ be a ring and $R[x]$ the polynomials over R.

1. If $(R, +, \cdot)$ is a commutative ring with 1, then $(R[x], +, \cdot)$ is a commutative ring with 1.
2. If $(R, +, \cdot)$ is a integral domain, then $(R[x], +, \cdot)$ is a integral domain.

Theorem 39 (Division Algorithm for Polynomials over a Ring)
Let $(R, +, \cdot)$ be a commutative ring with 1.
Let $f(x), g(x) \in R[x]$, $g(x) \neq 0$ with the leading coefficient of $g(x)$ being invertible.
Then, $\exists! q(x), r(x) \in R[x]$ such that

$$f(x) = q(x)g(x) + r(x)$$

where either $r(x) = 0$ or $\deg(r) < \deg(g)$.

Proof. Use induction on $\deg(f)$.
1. $f(x) = 0$ or $\deg(f) < \deg(g)$: $q(x) = 0, r(x) = f(x)$
2. $\deg(f) = \deg(g) = 0$: $q(x) = f(x) \cdot g(x)^{-1}, r(x) = 0$
3. $\deg(f) \geq \deg(g)$:

1) Existence
Let $\deg(f) = n$, $\deg(g) = m$, $n > m$.
Suppose the theorem holds for $\deg(f) < n$.
Let $f(x) = a_0 + a_1x^1 + \cdots + a_nx^n$, $g(x) = b_0 + b_1x^1 + \cdots + b_mx^m$.
Choose $f_1(x) = f(x) - (a_nb_m^{-1})x^{n-m}g(x) \in R[x]$.
Since $\deg(f_1) < n$, $\exists q(x), r(x) \in R[x]$ so that $f_1(x) = g(x)q(x) + r(x)$, where $r(x) = 0$ or $\deg(r) < \deg(g)$.
$f_1(x) = f(x) - (a_nb_m^{-1})x^{n-m}g(x) = g(x)q(x) + r(x)$
$f(x) = g(x)((a_nb_m^{-1})x^{n-m} + q(x)) + r(x)$
Hence such pair exists.

2) Uniqueness
Suppose $f(x) = g(x)q_1(x) + r_1(x) = g(x)q_2(x) + r_2(x)$.
$g(x)(q_1(x) - q_2(x)) = r_2(x) - r_1(x)$
If $r_1 \neq r_2$, $\deg(g) > \deg(r_2 - r_1) = \deg(g(q_1 - q_2))$.
Since $\deg(g(q_1 - q_2)) \geq \deg(g)$ if $q_1 - q_2 \neq 0$, $q_1 = q_2$, but if so, $r_1 = r_2$.
If $r_1 = r_2$, trivially $q_1 = q_2$.
Hence they exist uniquely. □

Chapter 4

Number Theory

4.1 Arithmetic

4.1.1 Integer Arithmetic

Theorem 40 (Division Algorithm)

Definition 41 (Divisibility)

Theorem 42 (Euclidean Algorithm)

Theorem 43 (Extended Euclidean Algorithm)

Definition 44 (Linear Diophantine Equation)

Theorem 45 (Solutions for Linear Diophantine Equation)

4.1.2 Modular Arithmetic

Definition 46 (Modulus)

Chapter 5

Analysis

Chapter 6

Linear Algebra

Chapter 7

Calculus

7.1 Limits

You may have seen an equation of the form $\lim_{x\to a} f(x) = L$. Intuitively, it means that as x *approaches* a, $f(x)$ goes *arbitrarily close* to L. But no, this "intuition" is not how mathematics works. What do you mean by "approaches?" What do you mean by "arbitrarily close?" How are you going to prove any theorem with this "definition?"

Let's give a precise definition of a limit. $f(x)$ goes arbitrarily close to L, but how close does that mean? It can go closer than any positive number. That means for any $\epsilon > 0$, $f(x)$ can go closer to L than ϵ. That is, $|f(x) - L| < \epsilon$.

Next, x approaches a, but how close does it approach a? How much should x approach a so that $f(x)$ goes arbitrarily close to L, in other words, $|f(x) - L| < \epsilon$? Well, close enough. When x is closer to a than some threshold, say δ, we would have $|f(x) - L| < \epsilon$. But it doesn't need to exactly be a. Expressing this mathematically, we get $0 < |x - a| < \delta$.

Combine those two inequalities, and presto! We have this definition of a limit.

Definition 47 (Limit at a)
Let f be a function defined on some open interval that contains a, except possibly at a itself. Then we say $\lim\limits_{x\to a} f(x) = L$ if for every number $\epsilon > 0$ there is a number $\delta > 0$ such that $0 < |x - a| < \delta$ implies $|f(x) - L| < \epsilon$.

Similarly, we can define left-hand limits, right-hand limits, and limits at infinity.

Definition 48
.

This allows us to prove the theorems involving limits.

Theorem 49
.

Definition 50 (Continuous function)
.

7.2 Differentiation

Definition 51 (Derivative)
The derivative of a function f at a, denoted $f'(a)$, is $f'(a) = \lim\limits_{h\to 0} \dfrac{f(a+h) - f(a)}{h}$,

if this limit exists. f is differentiable at a if $f'(a)$ exists.

Theorem 52
If f is differentiable at a, then f is continuous at a.

Proof. . □

7.3 Derivative Formulae

Theorem 53
Let f and g be differentiable functions and c be a constant.

1. $c' = 0$.
2. $(cf)' = c(f')$.
3. $(f+g)' = f' + g'$.
4. $(f-g)' = f' - g'$.
5. $(fg)' = fg' + gf'$.
6. $(\frac{f}{g})' = \frac{gf' - fg'}{g^2}$, where $g(x) \neq 0$.
7. $(x^c)' = cx^{c-1}$, where c is a rational number. (It also holds for real numbers, but we won't prove it here.)

Proof. . □

Theorem 54

1. $(\sin x)' = \cos x$.
2. $(\cos x)' = -\sin x$.
3. $(\tan x)' = \sec^2 x$.
4. $(\csc x)' = -\csc x \cot x$.
5. $(\sec x)' = \sec x \tan x$.
6. $(\cot x)' = -\csc^2 x$.

Proof. . □

Theorem 55 (Chain rule)
If g is differentiable at x and f is differentiable at $g(x)$, then $F = f \circ g$ defined by $F(x) = f(g(x))$ is differentiable at x and $F'(x) = f'(g(x))g'(x)$.

Proof. . □

7.4 Integration

Chapter 8

Statistics

Chapter 9

From $\mathbb{N}$ to $\mathbb{R}$

9.1 $\mathbb{N}$: The set of Natural Numbers

9.1.1 Construction of $\mathbb{N}$

We start from the Axioms of Set[2,3,4,5,6,9], the definition of power set[10], the definition of equivalence relation and class[11,13] and the following definitions:

Definition 56 (Successor)
For any set x, the successor of x, denoted $\sigma(x)$, is defined as the following set:

$$\sigma(x) = x \cup \{x\}$$

Let us define $0 = \emptyset$, $1 = \sigma(\emptyset) = \sigma(0)$. Using the definition of successors, and following the pattern, $2 = \sigma(1)$, $3 = \sigma(2)$, and so on. Basically we can make any finite number using the definition of successor and the Axioms of Set, but actually getting all of the natural numbers at once(or any infinitely large set, since only the empty set is guaranteed to exist by the axioms) is not possible with our axioms. We define the concept of Inductive Sets and make another Axiom for this purpose:

Definition 57 (Inductive Set)
A set A is called inductive if it satisfies the following two properties:

- $\emptyset \in A$
- $(x \in A) \Rightarrow (\sigma(x) \in A)$

Axiom 58 (Axiom of Infinity)
There is an inductive set, that is:

$$\exists A(\emptyset \in A) \wedge (\forall x \in A, \sigma(x) \in A)$$

Theorem 59
Take any two inductive sets, S and T. Then, $S \cap T$ is also an inductive set.

Proof. Let $U = S \cap T$.

1. $\emptyset \in U$
   $\emptyset \in S$ and $\emptyset \in T$ since S and T are both inductive.
2. $(x \in U) \Rightarrow (\sigma(x) \in U)$
   $\forall x \in U, (x \in S) \wedge (x \in T)$.
   Since S and T are both inductive, $(\sigma(x) \in S) \wedge (\sigma(x) \in T)$
   Therefore $\sigma(x) \in U$.

Therefore U is inductive. □

Corollary 60
An intersection of any number of inductive sets is inductive.

Theorem 61
For any inductive set S, define N_S as follows:

$$N_S = \bigcap_{\substack{A \subseteq S \\ A \text{ is inductive}}} A$$

Take any two inductive sets, S and T. Then $N_S = N_T$.

Proof. Suppose not; WLOG, $\exists x$ such that $x \in N_S$ and $x \notin N_T$.
Let $X = N_S \cap N_T$. Then X is inductive, $X \subset N_S$, and $x \notin X$.
Since by the definition of N_S, $N_S = X \cap N_S$, but $x \notin X \cap N_S$ hence the RHS and the LHS are different.
Therefore the assumption is wrong; therefore $N_S = N_T$. □

Using this theorem, we can finally define the set of natural numbers:

Definition 62 (The Set (N) of natural numbers)
Take any inductive set S, and let

$$N = \bigcap_{\substack{A \subseteq S \\ A \text{ is inductive}}} A$$

This set is the natural numbers, which we denote as $\mathbb{N}$.

9.1.2 Operations on $\mathbb{N}$

We now define two operations on $\mathbb{N}$, addition$(+)$ and multiplication$(\cdot)$.

Definition 63 (Addition and Multiplication on $\mathbb{N}$)
The operation of addition, denoted by $+$, is defined by following two recursive rules:

1. $\forall n \in \mathbb{N}, n + 0 = n$
2. $\forall n, m \in \mathbb{N}, n + \sigma(m) = \sigma(n + m)$

Similarly the operation of multiplication, denoted by $\cdot$, is defined by following two recursive rules:

1. $\forall n \in \mathbb{N}, n \cdot 0 = 0$
2. $\forall n, m \in \mathbb{N}, n \cdot \sigma(m) = n \cdot m + n$

Lemma 64 (Operations on 0)
$\forall x \in \mathbb{N}$

- $x + 0 = 0 + x$
- $x \cdot 0 = 0 \cdot x$

Proposition 65 (Properties of $+$ and $\cdot$)
$\forall x, y, z \in \mathbb{N}$,

- **Associativity of Addition** $x + (y + z) = (x + y) + z$
- **Commutativity of Addition** $x + y = y + x$

- **Associativity of Multiplication** $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
- **Commutativity of Multiplication** $x \cdot y = y \cdot x$
- **Distributive Law** $x \cdot (y + z) = x \cdot y + x \cdot z$
- **Cancellation Law for Addition** $x + z = y + z \Rightarrow x = y$

9.1.3 Ordering on $\mathbb{N}$

Definition 66 (Ordering on $\mathbb{N}$)
For $n, m \in \mathbb{N}$, we say that n is less than m, written $n \geq m$, if there exists a $k \in \mathbb{N}$ such that $m = n + k$. We also write $n < m$ if $k \neq 0$.

Theorem 67
$(N, <)$ is an ordered set[15].

Proposition 68
The followings are true:

- If $n \neq 0$, then $0 < n$.
- Let $x, y, z \in \mathbb{N}$. Then the followings are true:
  - $(x \leq y) \wedge (y < z) \Rightarrow (x < z)$
  - $(x < y) \wedge (y \leq z) \Rightarrow (x < z)$
  - $(x \leq y) \wedge (y \leq z) \Rightarrow (x \leq z)$
  - $(x < y) \Rightarrow (x + z < y + z)$
  - $(x < y) \Rightarrow (xz < yz)$
- $\forall n \in \mathbb{N}, n \neq n + 1$
- $\forall n, k \in \mathbb{N}, k \neq 0, n \neq n + k$

Definition 69 (Least Element)
Let $S \subset \mathbb{N}$. An element $n \in S$ is called a least element if $\forall m \in S, n \leq m$

Proposition 70 (Uniqueness of the Least Element)
Let $S \subset \mathbb{N}$. Then if S has a least element, then it is unique.

Theorem 71 (Well-Ordering Property)
Let S be a nonempty subset of $\mathbb{N}$. Then S has a least element.

Note
The well-ordering property states that the set of natural numbers $\mathbb{N}$ has the greatest lower bound property[20] and thereby theorem 21, has the least upper bound property[19].

9.1.4 Properties of $\mathbb{N}$

Many of the mathematics book defines the set of Natural Numbers as the set satisfying the Peano Axioms.

Proposition 72 (Peano Axioms)

1. 0, which we defined as the empty set $\emptyset$, is a natural number.
2. There exist a distinguished set map $\sigma : \mathbb{N} \to \mathbb{N}$
3. σ is injective

4. There does not exist an element $n \in \mathbb{N}$ such that $\sigma(n) = 0$

5. (Principle of Induction) If $S \in N$ is inductive, then $S = N$.

Proposition 73
Suppose that a is a natural number, and that $b \in a$. Then $b \subseteq a$, $a \nsubseteq b$.

Proposition 74
For any two natural numbers $a, b \in \mathbb{N}$, if $\sigma(a) = \sigma(b)$, then $a = b$.

Lemma 75
If $n \in \mathbb{N}$ and $n \neq 0$, then there exists $m \in \mathbb{N}$ such that $\sigma(m) = n$.

9.2 $\mathbb{Z}$: The set of Integers

9.2.1 Construction of $\mathbb{Z}$

We now have the set of natural numbers, and starting there, we construct the set of integers.

Proposition 76
Define a relation $\equiv$ on $\mathbb{N} \times \mathbb{N}$ by $(a, b) \equiv (c, d)$ iff $a + d = b + c$. This relation is an equivalence relation on $\mathbb{N} \times \mathbb{N}$.

Let $\mathbb{Z}$ be the set of equivalence classes under this relation, and the equivalence class containing (a, b) be denoted by $[a, b]$.

9.2.2 Operations on $\mathbb{Z}$

Definition 77 (Addition and Multiplication on $\mathbb{Z}$)
Addition and multiplication on $\mathbb{Z}$ are defined by:

- $[a, b] + [c, d] = [a + c, b + d]$
- $[a, b] \cdot [c, d] = [ac + bd, ad + bc]$

Definition 78 (Subtraction on $\mathbb{Z}$)
Subtraction on $\mathbb{Z}$ is defined by:

$$[a, b] - [c, d] = [a, b] + [d, c]$$

9.2.3 Ordering on $\mathbb{Z}$

Definition 79 (Ordering on $\mathbb{Z}$)
Let $[a, b], [c, d] \in \mathbb{Z}$. $[a, b] < [c, d]$ iff $a + d < b + c$.

9.2.4 Property of $\mathbb{Z}$

Theorem 80 (Arithmetic Properties of $\mathbb{Z}$)

1. Addition and multiplication are well-defined.
2. Addition and multiplication have identity elements $[n, n]$ and $[n, n + 1]$, respectively.
3. Addition and multiplication are commutative and associative.
4. The distributive law holds.
5. Each element $[a, b]$ has an additive inverse $[b, a]$.

We can treat $\mathbb{N}$ to be a subset of $\mathbb{Z}$ by identifying the number n with the class $[0,n]$. Since $[0,a]+[0,b]=[0,a+b]$ and $[0,a]\cdot[0,b]=[0,ab]$, these operations mirror the corresponding operation in $\mathbb{N}$.

Given $n\in\mathbb{N}$, we write $-n$ for $[n,0]$, 0 for $[n,n]$, and 1 for $[n,n+1]$. By the fifth arithmetic property of $\mathbb{Z}$[80], this defines $-n$ to be the additive inverse of n. We also use the minus sign for subtraction; it is therefore natural to write $[a,b]$ as $b-a$.

Proposition 81
For $a,b\in\mathbb{N}$, let $-b$, a, and b be defined in $\mathbb{Z}$ as above. Then

$$a-b=a+(-b) \text{ and } -(-b)=b$$

9.3 $\mathbb{Q}$: The set of Rational Numbers

We construct the set of rational numbers from the set of integers as follows:

9.3.1 Construction of $\mathbb{Q}$

Proposition 82
Define a relation $\equiv$ on $\mathbb{Z}\times(\mathbb{Z}\backslash\{0\})$ by $(a,b)\equiv(c,d)$ iff $ad=bc$. This relation is an equivalence relation on $\mathbb{Z}\times(\mathbb{Z}\backslash\{0\})$.

Let $\mathbb{Q}$ be the set of equivalence classes under this relation, and the equivalence class containing (a,b) is denoted by a/b or $\frac{a}{b}$, and $\frac{a}{b}=\frac{c}{d}$ mean that (a,b) and (c,d) belong to the same equivalence class. Especially we write 0 and 1 to denote $\frac{0}{1}$ and $\frac{1}{1}$, respectively.

9.3.2 Operations on $\mathbb{Q}$

Definition 83 (Addition and Multiplication on $\mathbb{Q}$)
The sum and product of $\frac{a}{b},\frac{c}{d}\in\mathbb{Q}$ are defined by

$$\frac{a}{b}+\frac{c}{d}=\frac{ad+bc}{bd} \text{ and } \frac{a}{b}\frac{c}{d}=\frac{ac}{bd}$$

Definition 84 (Subtraction on $\mathbb{Q}$)
Subtraction on $\mathbb{Z}$ is defined by:

$$\frac{a}{b}-\frac{c}{d}=\frac{ad-bc}{bd}$$

Definition 85 (Division on $\mathbb{Q}$)
Division on $\mathbb{Z}$ is defined by:

$$\frac{a}{b}\div\frac{c}{d}=\frac{ad}{bc}$$

9.3.3 Ordering on $\mathbb{Q}$

Definition 86 (Ordering on $\mathbb{Q}$)
Let $\frac{a}{b},\frac{c}{d}\in\mathbb{Q}$. $\frac{a}{b}<\frac{c}{d}$ iff $(bd>0\wedge ad<bc)\vee(bd<0\wedge ad>bc)$.

9.3.4 Property of $\mathbb{Q}$

Theorem 87 (Arithmetic Properties of $\mathbb{Q}$)

1. Addition and multiplication are well-defined.

2. Addition and multiplication have identity elements 0 and 1, respectively.

3. Addition and multiplication are commutative and associative.

4. The distributive law holds.

Theorem 88
$(\mathbb{Q}, +, \cdot)$ forms an ordered field.

9.4 $\mathbb{R}$: The set of Real Numbers

9.4.1 Construction of $\mathbb{R}$

One simple way to construct $\mathbb{R}$ is by proving the following theorem:

Theorem 89 (Existence of $\mathbb{R}$)
There exists an ordered field $\mathbb{R}$ containing $\mathbb{Q}$ as a subfield which has the least-upper-bound property.

But where's the fun in that? We will be constructing the field of real numbers using Cauchy sequences[**??**], starting with the following proposition:

Theorem 90
Define a relation $\equiv$ on the set S of Cauchy sequences of rational numbers as follows:

$$\{a_n\} \equiv \{b_n\} \text{ iff } (a_n - b_n) \to 0$$

This relation is an equivalence relation.

Now let us define $\mathbb{R}$ as the set of equivalence classes of S under the relation $\equiv$.

9.4.2 Operations on $\mathbb{R}$

Before the definition of operations on $\mathbb{R}$, we need to find out whether if the Cauchy sequences of rational numbers are closed under addition and multiplication, and it turns out they do, as stated in the following proposition:

Proposition 91
The set S of Cauchy sequences of rational numbers is closed under addition, multiplication, and scalar multiplication, that is:

1. If $\{a_n\} \in S$ and $\{b_n\} \in S$, then $\{a_n + b_n\} \in S$

2. If $\{a_n\} \in S$ and $\{b_n\} \in S$, then $\{a_n b_n\} \in S$

3. If $\{a_n\} \in S$ and $c \in \mathbb{Q}$, then $\{ca_n\} \in S$

We can finally go on to defining the operations on $\mathbb{R}$.

Definition 92 (Addition and Multiplication on $\mathbb{R}$)
Let $\{a_n\}$ and $\{b_n\}$ be sequences contained in the real numbers α, β, respectively. Then the sum and product of α and β are defined by:

$$\alpha + \beta = \{a_n + b_n\} \text{ and } \alpha\beta = \{\alpha\beta\}$$

We can define subtraction and division on $\mathbb{R}$ similar to addition and multiplication, by term-by-term calculation on each term of the Cauchy sequence.

9.4.3 Ordering on $\mathbb{R}$

Definition 93 (Ordering on $\mathbb{R}$)
Let $\alpha = \{a_n\}, \beta = \{b_n\} \in \mathbb{R}$. $\alpha < \beta$ iff $\exists N \in \mathbb{N}, \forall n \geq N, a_n < b_n$.

9.4.4 Property of $\mathbb{R}$

Theorem 94 (Arithmetic Properties of $\mathbb{R}$)

1. Addition and multiplication are well-defined.
2. Addition and multiplication have identity elements $\{0\}$ and $\{1\}$, respectively.
3. Addition and multiplication are commutative and associative.
4. The distributive law holds.
5. Each element $\{a_n\}$ has an additive inverse $\{-a_n\}$.

Theorem 95
$(\mathbb{R}, +, \cdot)$ forms an ordered field.

We now define an extension to $\mathbb{R}$ as follows:

Definition 96 (Extended Real Number System)
The extended real number system, denoted $\mathbb{R}^+$, $[-\infty, \infty]$, or $\mathbb{R} \cup \{-\infty, \infty\}$, consists of the real field $\mathbb{R}$ and two symbols, $+\infty$ and $-\infty$. We preserve the original order in $\mathbb{R}$, and define $\forall x \in \mathbb{R}$,

$$-\infty < x < \infty$$

Remark
The extended real number system does not form a field.

9.5 $\mathbb{C}$: The set of Complex Numbers

We construct the set of complex numbers from $\mathbb{R}$. Unlike the previous constructions, we do not construct it using equivalence class. Instead the construction is done by considering the quotient ring of polynomial ring over $\mathbb{R}$ modulo $i^2 + 1$.

Definition 97
Complex number is defined as the quotient ring $\mathbb{R}[i]/(i^2 + 1)$, with operations defined as normal.

Theorem 98
$(\mathbb{C}, +, \cdot)$ forms a field.

Part II

Applications to Computer Science

Chapter 10

Language Theory

Automaton is defined as a machine or control mechanism designed to automatically follow a predetermined sequence of operations, or respond to predetermined instructions. Theoretically, they all can be considered as the simplest form of algorithm, whether it is finite state automaton, push down automaton, or Turing machine. They all accept an input, and produce output; usually the output is *accept* or *reject*, but in the case of Turing machines, the output may be something different.

Before we start talking about the machines however we need to define what "Language" is.

Definition 99 (Language)
A (formal) language L over an alphabet Σ is a subset of Σ^*, that is, a set of words over that alphabet.

In this section, we explore Regular Language[101], Context-free Language[107], Decidable Language[114], and Recognizable Language[115] and the mechanisms, or machines, that are related those languages.

10.1 Regular Language

10.1.1 Regular Expression

If you have studied regular expression using some programming languages, then you might have easier time understanding the following definition. The regular expression used in real life is much more powerful than the regular expression mentioned below, as more special characters and syntaxes are allowed. However the following regular expression consists of the "basics" of regular expression, and is used in language theories as the regular expression:

Definition 100 (Regular Expression(RE))
Given a finite alphabet Σ, the following constants are defined as regular expressions:

- **Empty set**: $\emptyset$, denoting the set $\emptyset$.
- **Empty string**: ϵ, denoting the set containing only the "empty" string, which has no characters at all.
- **Literal character**: $a \in \Sigma$, denoting the only character a.

And when given regular expressions R and S, the following operations over them produce regular expressions:

- **Concatenation**: RS, denoting the concatenation of strings in R and S, in that order.

  R^n denotes the concatenation of R, n times: Specifically, $R^0 = \{\epsilon\}$.

- **Alternation**: $R|S$, denoting the set union of the strings in R and S.

- **Kleene star**: R^*, denoting $\bigcup_{i\in\mathbb{N}} R^i$.

Definition 101 (Regular Languages)
Regular Languages are languages that can be represented with regular expressions.

Theorem 102 (Pumping Lemma for Regular Languages)
Let L be a regular language. Then, there exists an integer $p \geq 1$, depending only on L, such that every string $w \in L$ of length at least p, called the pumping length, can be written as $w = xyz$ (i.e. w can be divided into three substrings), satisfying the following conditions:

- $|y| \geq 1$
- $|xy| \leq p$
- $\forall n \geq 0, xy^n z \in L$

10.1.2 Deterministic Finite State Automaton

Definition 103 (Deterministic Finite Automaton(DFA))
A DFA M is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of:

- Finite set of states Q;
- Finite set of input symbols called the alphabet Σ;
- Transition function $\delta : Q \times \Sigma \to Q$;
- Initial state $q_0 \in Q$;
- Set of accepting states $F \subseteq Q$.

Let $w = a_1 a_2 \ldots a_n$ be a string over the alphabet Σ. DFA M accepts the string w if a sequence of states, $r_0, r_1, \ldots, r_n \in Q$ exists with the following conditions:

- $r_0 = q_0$
- $r_{i+1} = \delta(r_i, a_{i+1}), i = 0, \ldots, n-1$
- $r_n \in F$

Theorem 104
DFAs recognize exactly the set of regular languages.

10.1.3 Nondeterministic Finite Automaton

Definition 105 (Nondeterministic Finite Automaton(NFA))
A NFA M is a 5-tuple, $(Q, \Sigma, \Delta, q_0, F)$, consisting of:

- Finite set of states Q;
- Finite set of input symbols called the alphabet Σ;
- Transition function $\Delta : Q \times \Sigma \to P(Q)$ where P is the powerset function;

- Initial state $q_0 \in Q$;
- Set of accepting states $F \subseteq Q$.

Sometimes the transition function Δ is represented as the transition relation, $\Delta \subseteq Q \times \Sigma \times Q$.

Let $w = a_1 a_2 \dots a_m$ be a string over the alphabet Σ, where $a_i \in \Sigma_\epsilon$. NFA M accepts the string w if a sequence of states, $r_0, r_1, \dots, r_n \in Q$ exists with the following conditions:

- $r_0 = q_0$
- $r_{i+1} \in \Delta(r_i, a_{i+1})$, or in relation form, $(r_i, a_{i+1}, r_{i+1}) \in \Delta$, $i = 0, \dots, n-1$
- $r_n \in F$

Theorem 106
NFAs recognize exactly the set of regular languages.

10.2 Context-Free Language

10.2.1 Context-free Grammar

Definition 107 (Context-Free Grammar(CFG))
A CFL is a 4-tuple (V, Σ, R, S) where:

- V is the set of nonterminal variables;
- Σ is the set of terminal characters;
- R is the set of rules, where each rules are in the form of $A \to w, A \in V, w \in (\Sigma \cup V)^*$
- S is the starting variable.

Definition 108 (Context-free Languages)
Context-free Languages are languages that can be represented with context-free grammars.

Theorem 109 (Pumping Lemma for Context-free Languages)
Let L be a regular language. Then, there exists an integer $p \geq 1$, depending only on L, such that every string $s \in L$ of length at least p, called the pumping length, can be written as $s = uvwxy$(i.e. w can be divided into five substrings), satisfying the following conditions:

- $|vx| \geq 1$
- $|vwx| \leq p$
- $\forall n \geq 0, uv^n wx^n y \in L$

10.2.2 Push-down Automaton

Similar to Finite Automatons, Push-down automatons have deterministic version and nondeterministic version; Only the nondeterministic version is shown here as similar method can be used to convert it into a deterministic version.

Definition 110 (Push-down Automaton(PDA))
A PDA is a 6-tuple $(Q, \Sigma, \Gamma, q_0, \Delta, F)$ where:

- Q is the set of states;
- Σ is the set of input alphabet;
- Γ is the set of stack alphabet;
- q_0 is the starting state;
- Δ is the transition relation of $Q \times \Sigma_\epsilon \times \Gamma_\epsilon \times Q \times \Gamma_\epsilon$
- F is the set of accepting states

Δ is often written as a transition function of $Q \times \Sigma_\epsilon \times \Gamma_\epsilon \times \rightarrow P(Q \times \Gamma_\epsilon)$ where P is the powerset function.
Sometimes the last element of the relation is extended to Γ_ϵ^*; in that case, when inserting into the stack, insert the last element first. Let $w = w_1 w_2 \dots w_m$ be a string over the alphabet Σ, where $w_i \in \Sigma_\epsilon$. NFA M accepts the string w if a sequence of states, $r_0, r_1, \dots, r_n \in Q$, and a sequence of stack strings $s_0, s_1, \dots, s_n \in \Gamma^*$ exists with the following conditions:

- $r_0 = q_0$, $s_0 = \epsilon$
- $(r_i, w_{i+1}, a, r_{i+1}, b) \in \Delta$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\epsilon$, and $t \in \Gamma^*$.
  If $b = \epsilon$, then it is a pop-operation. If $a = \epsilon$, then it is a push-operation.
- $r_m \in F$, $s_m = \epsilon$

Theorem 111
PDAs recognize exactly the set of CFLs.

Proof. The proof is quite tedious; so only a partial proof is given: we are going to convert any given CFG into PDA. Suppose a CFG $(V_0, \Sigma_0, R_0, S_0)$ is given.
We can construct a new PDA $(Q, \Sigma, \Gamma, q_0, \Delta, F)$ from the given CFL s.t.

- $Q = \{Q_S, Q_M, Q_F\}$
- $\Sigma = \Sigma_0$
- $\Gamma = V_0 \cup \Sigma_0$
- $q_0 = Q_S$
- $F = Q_F$
- $\Delta =$
  $\{(Q_S, \epsilon, \epsilon, Q_M, S\$)\} \cup$
  $\{(Q_M, \epsilon, \epsilon, X, Q_M, W) | X- > W \in R\} \cup$
  $\{(Q_M, a, a, Q_M, \epsilon) | a \in \Sigma_0\} \cup$
  $\{(Q_M, \epsilon, \$, Q_F, \epsilon)\}$

This exactly simulates the parse tree of the CFL. $\square$

10.3 Turing Machines

Definition 112 (Turing Machine)
A TM is a 7 tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where:

- Q is the set of states;
- Γ is the set of tape alphabet;
- $b \in \Gamma$ is the blank symbol, the only symbol allowed to occur infinitely often at any step of the computation;
- $\Sigma \subseteq \Gamma \backslash \{b\}$ is the set of input symbols, that is, the set of symbols allowed to appear in the initial tape contents;
- $q_0 \in Q$ is the starting state;
- $F \subseteq Q$ is the set of accepting states, and the initial tape contents is said to be accepted by M if it eventually halts in a state from F;
- δ is a partial function called the transition function of $(Q \backslash F) \times \gamma \to Q \times \gamma \times \{L, R\}$, where L and R signifies left and right shifts of the tape. If δ is undefined on the current state and the current tape symbol, then the machine halts.

A Turing machine

The definition of a Turing Machine is not unique. Some definitions use multiple tapes, using one of them as the input tape that can't be modified and another as the output tape. Some has more than one halting states. Some include the "starting symbol" in the alphabet. Some include N in the final output of the transition function. But in general, a Turing machine starts from one state, follows the decision function every step, and halts at the halting state. Some of the many variations on the Turing machine are mentioned in 10.5.2.

In fact, the different definitions of a Turing machine turns out to be the same, in the sense that a function $f : \{0,1\}^* \to \{0,1\}$ is computable using one definition of a Turing machine iff it is computable using another definition of a Turing Machine.

We now give the following thesis from the creator of the λ-calculus, Alonzo Church and Alan Turing.

Thesis 113 (Church-Turing Thesis)
A function on Natural Numbers which is computable by a human being following an algorithm, ignoring resource limitations, if and only if it is computable by a Turing Machine.

10.4 Decidable and Recognizable Languages

Definition 114 (Decidable Languages)
Decidable Languages are languages that can be represented with decidable Turing machines; that is, the set of Turing machines that always accepts accepting words and rejects others.

Definition 115 (Recognizable Languages)
Recognizable Languages are languages that can be represented with recognizable Turing machines; that is, the set of Turing machines that always accepts accepting words.

Decidable and Recognizable Turing Machines seem similar; however recognizable machines does not have to reject a non-accepting word; it may instead loop infinitely.

10.5 Equivalences to Turing Machine

The followings can be shown to be computationally equivalent to a Turing machine; however no proofs are given since they are usually long and arduous.

10.5.1 Push-down Automaton with Two Stacks

The simplest version that is equivalent to a Turing Machine would be a PDA which has two stacks. The two stacks can simulate the tape of the Turing machine by pushing and popping.

10.5.2 Variations on the Turing Machine

The following variations on the Turing machine are equivalent to the original Turing machine:

- Variations on the Definition
  - Allowing N, "no shift", in the movement rules;
  - Having a single accepting state, say q_{accept} and a single rejecting state, say q_{reject}, and forcing the transition function δ to be a function. In this variant, the machine accepts iff it ends in q_{accept}, and rejects iff it ends in q_{reject}.
- Variations on the Form of the Machine
  - Tape is infinite only in one direction;
  - Tape is infinite in both directions;
  - Tape is 2-dimensional;
  - There exists multiple tapes that the machine can access concurrently.

There are many more variations other than these.

10.5.3 General Recursive Functions

Definition 116 (General Recursive Functions)
General Recursive Functions, otherwise known as μ-recursive functions, is a set of functions $\forall n \in \mathbb{N}, f : \mathbb{N}^n \to \mathbb{N}$ that includes the three "Initial", or "Basic" functions, and closed under three operators:

- Initial Functions
  - **Constant Function:** $\forall n, k \in \mathbb{N}, f(x_1, \ldots, x_k) = n$
    Alternative definition use a Zero function: $\forall k \in \mathbb{N}, Z(x_1, \ldots, x_k) = 0$
  - **Successor Function** S**:** $S(x) = x + 1$
  - **Projection Function** P_i^k**:**
    This is also called the Identity Function I_i^k
- Operators

- **Composition Operator** $\circ$: Given an m-ary function $h(x_1,\ldots,x_m)$ and m k-ary functions $g_1(x_1,\ldots,x_k),\ldots,g_m(x_1,\ldots,x_k)$:

$$h \circ (g_1,\ldots,g_m) = f \text{ where } f(x_1,\ldots,x_k) = h(g_1(x_1,\ldots,x_k),\ldots,g_m(x_1,\ldots,x_k))$$

  This is also called the Substitution Operator.

- **Primitive Recursion Operator** ρ: Given the k-ary function $g(x_1,\ldots,x_k)$ and (k+2)-ary function $h(y,z,x_1,\ldots,x_k)$:

$$\begin{aligned}\rho(g,h) &= f \text{ where}\\ f(0,x_1,\ldots,x_k) &= g(x_1,\ldots,x_k)\\ f(y+1,x_1,\ldots,x_k) &= h(y,f(y,x_1,\ldots,x_k),x_1,\ldots,x_k)\end{aligned}$$

- **Minimization Operator** μ: Given a (k+1)-ary total function $f(y,x_1,\ldots,x_k)$:

$$\begin{aligned}\mu(f)(x_1,\ldots,x_k) = z \Leftrightarrow & f(z,x_1,\ldots,x_k) = 0 \text{ and}\\ & f(i,x_1,\ldots,x_k) > 0 \text{ for } i = 0,\ldots,z-1\end{aligned}$$

  Intuitively, this operator seeks the smallest argument that causes the function to return 0; if none exists, the search never ends and therefore cannot return.

10.5.4 Lambda Calculus

Lambda Calculus, first defined by Alonzo Church, is a formal system of mathematical logic for expressing computation based on function-like objects.

Definition 117 (Lambda Expression)
Lambda expressions are composed of:

- Variables, $v_1,\ldots,v_n,\ldots$
- The abstraction symbols lambda λ and dot .
- Parentheses ()

For some applications, terms for logical and mathematical constants and operation may be included.

The set of lambda expressions, Λ, can be defined inductively:

- If x is a variable, then $x \in \Lambda$
- If x is a variable and $M \in \Lambda$, then $(\lambda x.M) \in \Lambda$

  This rule is also known as Abstractions.
- If $M,N \in \Lambda$, then $(MN) \in \Lambda$

  This rule is also known as Application.

Though only the definition is given, This Wikipedia article can be helpful to understand how lambda calculus works.

Chapter 11

Complexity Theory

11.1 Complexity

(TODO: Write something about asymptotic notation here)

Definition 118 (Asymptotic notation)
Let f and g be two functions from $\mathbb{N}$ to $\mathbb{N}$. Then we say:

- $f = O(g)$ if there is a constant c such that $f(n) \leq c \cdot g(n)$ for every sufficiently large n. That is, $n > N$ for some N.
- $f = \Omega(g)$ if $g = O(f)$.
- $f = \Theta(g)$ if $f = O(g)$ and $g = O(f)$.
- $f = o(g)$ if for every constant $c > 0$, $f(n) < c \cdot g(n)$ for every sufficiently large n.
- $f = \omega(g)$ if $g = o(f)$.

Definition 119 (P, NP, EXP)

- **P** is the set of boolean functions computable with a deterministic Turing machine in time $O(n^c)$ for some constant $c > 0$.
- **NP** is the set of boolean functions computable with a non-deterministic Turing machine in time $O(n^c)$ for some constant $c > 0$.
- **EXP** is the set of boolean functions computable with a deterministic Turing machine in time $O(2^{n^c})$ for some constant $c > 0$.

Theorem 120
P $\subseteq$ **NP** $\subseteq$ **EXP**.

Proof. . □

11.2 Reduction

Is there a polynomial-time algorithm for a given decision problem? Computer scientists are interested in this question because if there is one, it is usually a small-degree polynomial like $O(n^2)$ or $O(n^5)$. Some problems have a special property that if the problem has a polynomial-time algorithm, then several other problems do.

Definition 121 (Polynomial-time Karp reduction)
A problem $A \subseteq \{0,1\}^*$ is polynomial-time Karp reducible to $B \subseteq \{0,1\}^*$, denoted $A \leq_p B$, if there is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $x \in A$ iff $f(x) \in B$.

The intuitive meaning is that a problem of A can be "reduced" to a problem of B, and if we can solve B in polynomial-time, then we can solve A in polynomial-time too.

Definition 122 (NP-complete)
A problem A is NP-hard if every problem in **NP** is polynomial-time reducible to A, and NP-complete if A is NP-hard and NP.

Theorem 123

1. If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$.
2. An NP-complete problem A is in **P** iff **P** = **NP**.
3. If $A \leq_p B$ and A is NP-hard, then B is NP-hard.

Proof. (1) Let f be a reduction from A to B with polynomial time $p(n)$, and g from B to C with $q(n)$. Then $g \circ f$ is a reduction from A to C with polynomial time $q(p(n))$.
(2) Suppose A is NP-complete and in **P**. Then any problem B in **NP** can be polynomial-time reduced to A, so transitivity implies that B is polynomial-time computable. The converse is trivial.
(3) Any problem C in **NP** can be polynomial-time reduced to A. Transitivity implies that C can be polynomial-time reduced to B. □

Now the obvious question is, does such a strong problem actually exist? The answer is yes, and a lot of important problems are NP-complete.
(TODO: SAT)
Having proven that SAT is NP-hard, more problems can be proven NP-hard if we can reduce SAT to those problems in polynomial-time. Here are only a tiny fraction of the NP-complete problems:

Definition 124 (NP-complete problems)

- The 3-SAT problem is a SAT problem where each clause contains exactly 3 variables.
- Given a graph G and an integer $0 \leq k \leq |V(G)|$, the clique problem asks whether there is a complete induced subgraph of G with size at least k.
- The independent set problem asks whether there is a subset S of $V(G)$ with size at least k such that no two vertices in S are adjacent, and 0 otherwise.
- The vertex cover problem asks whether there is a subset S of $V(G)$ with size at most k such that each edge is adjacent to at least one vertex in S.
- The chromatic number problem asks whether G is 3-colorable.
- Given a set S of integers and an integer k, the subset sum problem asks whether there is a subset of S whose sum of elements equals k.
- Given an $n \times m$ matrix A and an $n \times 1$ matrix b of integers, the integer programming problem asks whether there is an $m \times 1$ matrix x of integers such that each element of $Ax + b$ is non-negative.

Theorem 125
All problems in [124] are NP-complete.

Proof. Clearly all problems described are NP. We will only show that they are all NP-hard.

If we can reduce SAT to 3-SAT in polynomial time, then [123] will show that 3-SAT is NP-hard. To do this, note that

- x is equivalent to $x \vee x \vee x$,
- $x_1 \vee x_2$ is equivalent to $x_1 \vee x_2 \vee x_2$,
- $x_1 \vee \cdots \vee x_n$ is equivalent to $(x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee y_2) \wedge \cdots (\neg y_{n-4} \vee x_{n-2} \vee y_{n-3}) \wedge (\neg y_{n-3} \vee x_{n-1} \vee x_n)$, where $n \geq 4$ and $y_1, \cdots, y_{n-3}$ are new variables unused in the original SAT formula.

Next, we reduce 3-SAT to a clique problem. (TODO)

G has a clique of size k iff $\bar{G}$ has an independent set of size k. This shows that clique and independent set are polynomial-time reducible to each other.

G has an independent set of size k iff G has a vertex cover of size $|V(G)|-k$, by taking the complement of the independent set. Therefore independent set and vertex cover are polynomial-time reducible to each other.

We reduce 3-SAT to a chromatic number problem. (TODO)

We reduce 3-SAT to a subset sum problem. (TODO)

Finally, we reduce 3-SAT to an integer programming problem. Given a 3-SAT formula with n variables, set $0 \leq x_i \leq 1$ for $i = 1, \cdots, n$, and convert the clause $(x_a \vee x_b \vee x_c)$ into $x_a + x_b + x_c \geq 1$. If the clause contains $\neg x_a$, convert it to $1 - x_a$. This system of inequalities can easily be converted to the matrix form.

□

Chapter 12

Graph Theory

12.1 Basic Graph Definitions

12.2 Degrees

12.3 Trees

One of the important classes of graphs is a tree. There are many ways to define a tree. First we will state one definition, and then prove that other definitions are equivalent.

Definition 126 (Tree)
A forest is a simple graph without any cycle. A tree is a connected forest. A leaf of a forest is a vertex with degree 1.

Theorem 127
The following statements are equivalent for a simple graph G:

1. G is a tree.
2. For any two vertices u and v of G, there is exactly one path connecting them.
3. G is connected, and for any e of G, $G \backslash e$ is disconnected.
4. G has no cycle, and for any two vertices u and v not having an edge between them, $G + uv$ has a cycle.
5. G is connected, and $|E| = |V| - 1$.

Proof. TODO

□

12.3.1 Spanning Trees

Definition 128 (Spanning Subgraph)
A spanning subgraph of a graph G is a subgraph of G such that its vertex set equals $V(G)$. A spanning tree is a spanning graph that is a tree.

Theorem 129
A connected graph G has a spanning tree.

Proof. Let $m = |E|$, and label the edges as e_0, $\cdots$, e_m, arbitrarily. Define the subsets E_0, $\cdots$, E_m of E, as

$$\begin{cases} E_0 = \varnothing \\ E_i = E_{i-1} \cup \{e_i\} & \text{if the spanning subgraph of } G \\ & \text{with } E = E_{i-1} \cup \{e_i\} \text{ has no cycle} \\ E_i = E_{i-1} & \text{otherwise.} \end{cases}$$

Let H be the spanning subgraph of G with $E = E_m$. Clearly, H has no cycle. If $e_i \notin E_m$ and $H + e_i$ has no cycle, then E_i would contain e_i, contradiction. From [127], H is a tree. □

(TODO: minimum spanning tree)

There are other minimum spanning tree algorithms like Prim's algorithm or Borůvka's algorithm.

12.4 Planar Graphs

Definition 130 (Planar Graph)
A plane graph is a graph G where:

- $V \subseteq \mathbb{R}^2$;
- every edge is an arc between two endpoints;
- the interior of each edge contains no vertex and no point of any other edge.

The connected components of $\mathbb{R}^2 \backslash G$ are called faces of G. Since G is contained in a sufficiently large disc, exactly one face is unbounded; that face is called the outer face of G. All other faces are called inner faces of G. A graph H is planar if it is isomorphic to some plane graph.

Theorem 131 (Euler's Formula)
If G is a connected plane graph, and the number of faces of G is F, then

$$|V| - |E| + F = 2.$$

Proof. Induction on $|E|$. The base case is when G has no edges, one vertex, and one face; the formula clearly holds.

Pick any edge e. If e is a loop, removing it reduces $|E|$ and F by one. Otherwise, contracting it reduces $|V|$ and $|E|$ by one. Either way the result follows by induction. □

Theorem 132
If G is simple and planar, and $|V| \geq 3$, then $|E| \leq 3|V| - 6$. If in addition G has no triangles (i.e. K_3 as a subgraph), then $|E| \leq 2|V| - 4$.

Proof. Count the number N of pairs (f, e) where the face f and the edge e are incident. For each face, there are at least 3 edges incident to it, for otherwise there would be parallel edges or loops. Therefore $N \geq 3F$. On the other hand, each edge is incident to exactly two faces, so $N = 2|E|$. This gives $3F \leq 2|E|$. From [131], $3F = 6 - 3|V| + 3|E| \leq 2|E|$, and the first result follows.

The second result can be proved in the exactly same way, using $N \geq 4F$. □

Corollary 133
K_5 and $K_{3,3}$ are not planar.

Proof. K_5 has 5 vertices and 10 edges. $K_{3,3}$ has no triangles, 6 vertices, and 9 edges. The result follows from [132]. □

It clearly follows that any subdivision of K_5 or $K_{3,3}$ are not planar. Surprisingly, those two graphs are the only graphs that "need to be checked" to determine if a given graph is planar. The proof requires several more lemmas and theorems, so we have moved the proof to the appendix.

Theorem 134 (Kuratowski's Theorem)
A graph G is planar if and only if it does not have K_5 or $K_{3,3}$ as a topological minor.

12.5 Coloring

Definition 135 (Coloring)
A k-coloring of a graph G is a function $c : V(G) \to \{1, 2, \cdots, k\}$ such that if u and v are adjacent vertices, then $c(u) \neq c(v)$. G is k-colorable if there is a k-coloring of G. The chromatic number $\chi(G)$ of G is the smallest integer k such that G is k-colorable.

Perhaps the most famous theorem about graph coloring is the four-color theorem. (TODO: write something)

Theorem 136 (Four-color Theorem)
If G is planar, then $\chi(G) \leq 4$.

Unfortunately, the proof is too long and complicated to contain in the codex. We prove a weaker result:

Theorem 137 (Five-color Theorem)
If G is planar, then $\chi(G) \leq 5$.

Proof. Induction on $|V|$. For $|V| \leq 5$, the theorem is trivial.

From [132], G has a vertex v of degree at most 5. If $deg_G(v) < 5$, then inductively find a 5-coloring of $G - v$, and color v by some color in $\{1, 2, 3, 4, 5\}$ not appearing in the neighbors of v. If $deg_G(v) = 5$ and not all colors are used in the neighbors of v, then the same argument applies.

Now suppose all 5 colors are used. Denote the neighbors of v as u_1, u_2, u_3, u_4, u_5, in clockwise order. Without loss of generality, we will assume that $c(u_i) = i$.

The main idea of the rest of the proof is that we want to change the color of one of the neighbors, say change $c(u_i)$ to k. This is impossible if u_i has a neighbor of color k, in which case we want to also change the color of that neighbor, to k'. But then that neighbor might have yet another neighbor of color k', and this continues to form a chain. Hence we introduce the Kempe chain, named after Alfred Kempe.

Let V_{ij} be the set of vertices w in G such that there is a path from u_i to w consisting of vertices of color i or j. Note that if we switch the colors of the vertices in V_{ij} (i.e. change i to j and j to i), and leave everything else the same, then the result is still a coloring.

If V_{13} does not contain u_3, then switch the colors of the vertices in V_{13} and color v by 1. Otherwise, V_{24} does not contain u_4; switch the colors of the vertices in V_{24} and color v by 2. This gives a 5-coloring of G.

(TODO: picture) □

Fun fact: In 1879, the Kempe chain method was used to "prove" the four-color theorem by Alfred Kempe. No one noticed that this "proof" had an error until eleven years later when Percy Heawood found the error. What we saw above is the modification of the incorrect proof to prove the weaker theorem. The correct proof of four-color theorem was completed in 1976 by Kenneth Appel and Wolfgang Haken.

Chapter 13

Cryptosystem

13.1 Basic Terminology

Definition 138 (Basic Terminology on Cryptosystems)

- **Plaintext:** The text before encryption
- **Ciphertext:** The text after encryption
- **Cryptosystems:** Encryption and decryption algorithms, see definition below for more

  Encryption: Using some sort of algorithm to change the content of a message so that it is unrecognizable.

  Decryption: Processing the encrypted message to change it back to the message.
- **Key:** A value required to encrypt or decrypt.

  Encryption Key: The key for encryption.

  Decryption Key: The key for decryption.
- **Cryptanalysis:** Decrypting the ciphertext without any prior knowledge(i.e. key).

Definition 139 (Cryptosystem)
A cryptosystem is defined as a set of three algorithms, (G, E, D);

G The key generation algorithm, sometimes abbreviated as KeyGen, chooses the encryption key k_1 and the decryption key k_2 from the set of possible keys. The set of possible keys is called the key space. Usually each key from the key space is chosen at uniformly random probability.

E The Encryption Algorithm, sometimes abbreviated as Enc, uses the encryption key k_1, takes the plaintext m as an input, and produces the ciphertext c. This is usually denoted as follows:

$$E_{k_1}(m) = c$$

D The Decryption Algorithm, sometimes abbreviated as Dec, uses the decryption key k_2, takes the ciphertext c as an input, and gains the plaintext m. This is usually denoted as follows:

$$D_{k_2}(c) = m$$

For a cryptosystem to be valid, by encrypting the plaintext m and decrypting the ciphertext, we must be able to get m, that is;

$$D_{k_2}(E_{k_1}(m)) = m$$

A cryptosystem is classified into two categories; if the encryption key is the same as the decryption key, it is called a Symmetric Key Algorithm; if not, it is called an Asymmetric Key Algorithm or a Public Key Algorithm. A symmetric key algorithm is again classified into two categories; Block Cipher and Stream Cipher.

Definition 140 (Kerckhoffs' Principle)
Kerckhoffs' Principle states that a cryptosystem must be secure even if everything about the cryptosystem except for the key is exposed.

Kerckhoffs' Principle says that the cryptosystem's security must depend only on the secrecy of the key. Its core comes from the idea that "The enemy knows the system". In some, "Security through obscurity"(i.e. hiding the cryptosystem itself) holds but Kerckhoffs' Principle has its value for the following reasons:

1. Storing a smaller sized key is easier than hiding the entire cryptosystem. Also the cryptosystem is not safe from reverse engineering, but keys are, as they are usually a random number.
2. If the key is exposed, it is easier to change only the key, not the entire cryptosystem.
3. A cryptosystem is often used for many users, and everybody using the same cryptosystem allows for more efficient usage of space.
4. If the cryptosystem itself is kept a secret, if a problem arises(i.e. reverse engineering) to expose the cryptosystem, then the entire thing must be redesigned. This takes a lot of knowledge and time.
5. A cryptosystem is made weak by a small mistake; these mistakes are not found before the cryptosystems are analyzed fully, which is most easily done by making the system public. If they are indeed made public, the cryptosystem can be checked for security, allowing for a more secure system.

13.2 Encryption of Arbitrary Length Message

13.2.1 Padding

When using a block cipher, we need the length of the message to be an exact multiple of the length of the block used in the block cipher. If not, we use padding to make the message longer to make it an exact multiple. There are many ways to do so, but:

- Zero Padding, otherwise known as Null Padding

  Pad the message with zero(00) bytes to make the length be an exact multiple of the cipher block length. This may cause a problem if the last bytes of the message are 00.

- Bit Padding

  Pad the message with $10|00^n$, so that we can know the start of padding. We must pad the message even if its length is a multiple of the cipher block length.

- Byte Padding

  Same as zero padding, except the last byte is equal to the length of padding, that is; if we require four more bytes, the padding is $00|00|00|04$. We must pad the message even if its length is a multiple of the cipher block length.

- PKCS#7 Padding

  Similar to byte padding, except every byte of the padding is equal to the length of padding, that is; if we require four more bytes, the padding is $04|04|04|04$.

13.2.2 Modes of Operation

Sometimes we are required to encrypt a longer message than the length of the block. The plaintext are first padded using one of the techniques above, and the padded plaintext P is separated into blocks of padding length, $P_1, P_2, \cdots, P_N$. They are then encrypted using the key K, sometimes with the help of the initialization vector IV, and produces the ciphertexts $C_1, C_2, \cdots, C_N$. There are five major ways to do this; ECB, CBC, CFB, OFB, and CTR.

Electronic Code Book(ECB)

ECB mode is the simplest mode of them all. They simply take each blocks and encrypt them separately. In equation:

- **Encryption** $C_i = E_K(P_i)$
- **Decryption** $P_i = D_K(C_i)$

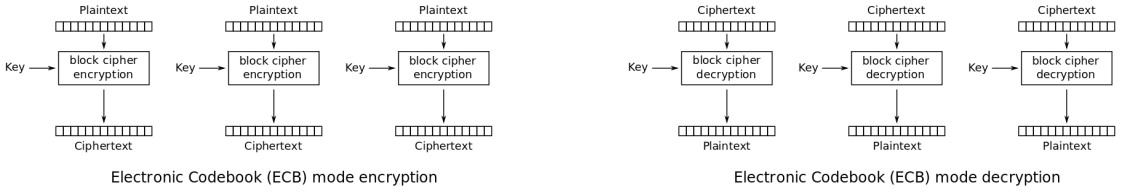

Figure 13.1: ECB Mode

Since same plaintext blocks are encrypted into same ciphertext block, the blocks can be copied, or replayed, to change the message easily. This is called the Block Replay Attack.

Cipher Block Chaining(CBC)

CBC takes the previous ciphertext block and XOR($\bigoplus$) it with the plaintext before encryption. The first block has no previous ciphertext block, hence it is XOR-ed with the IV. In equation:

- **Encryption** $C_0 = IV,\ C_i = E_K(P_i \bigoplus C_{i-1}), i = 1, 2, 3, \cdots, N$
- **Decryption** $C_0 = IV,\ C_i = D_K(C_i) \bigoplus C_{i-1}, i = 1, 2, 3, \cdots, N$

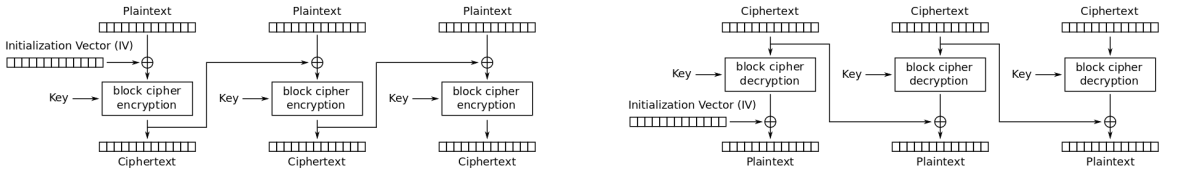

Figure 13.2: CBC Mode

Cipher Feedback(CFB)

CFB can be used to encrypt a block even smaller than the size of the encryption block, and can be used to make a stream cipher out of block cipher. In the diagram given below, original block sizes are used. In equation:

- **Encryption** $C_0 = IV,\ C_i = E_K(P_i \bigoplus C_{i-1}), i = 1, 2, 3, \cdots, N$
- **Decryption** $C_0 = IV,\ C_i = D_K(C_i) \bigoplus C_{i-1}, i = 1, 2, 3, \cdots, N$

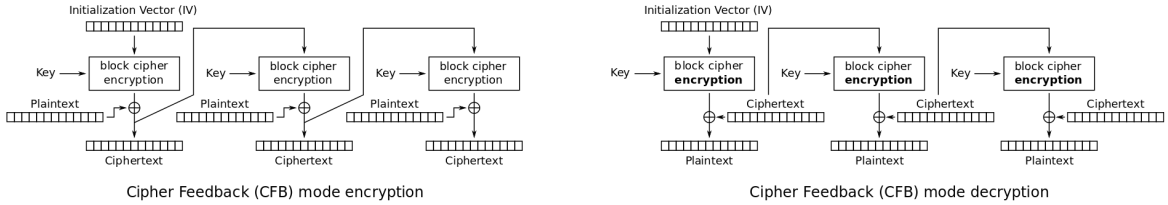

Figure 13.3: CFB Mode

By altering the equation to the following we have the "stream cipherized" version, where << is the shift operation, $head(a, x)$ is the first x bits of a, and n is the size of the IV:

- **Shift Register** $S_0 = IV,\ S_i = ((S_i << x) + C_i) \mod 2^n$
- **Encryption** $C_i = head(E_K(S_{i-1}), x) \bigoplus P_i$
- **Decryption** $P_i = head(E_K(S_{i-1}), x) \bigoplus C_i$

Output Feedback(OFB)

OFB can be used to encrypt a block even smaller than the size of the encryption block, and can be used to make a stream cipher out of block cipher.

- **Input and Output** $I_0 = IV,\ I_j = E_K(I_{j-1}), j = 1, 2, 3, \cdots, N$
- **Encryption** $C_j = P_j \bigoplus I_j, i = 1, 2, 3, \cdots, N$
- **Decryption** $P_j = C_j \bigoplus I_j, i = 1, 2, 3, \cdots, N$

We can similarly alter the equation as OFB so that it can be used as a stream cipher.

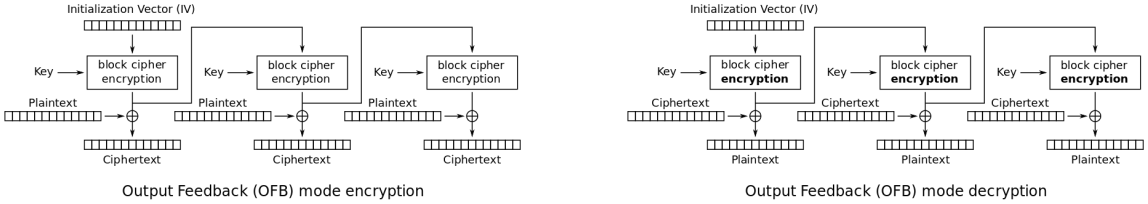

Figure 13.4: OFB Mode

Counter(CTR)

CTR can be used to encrypt a block even smaller than the size of the encryption block, and can be used to make a stream cipher out of block cipher. It encrypts the counter value instead of the plaintext, and XORs the value to gain the ciphertext.

- **Encryption** $C_i = P_i \bigoplus E_K(Counter), i = 1, 2, 3, \cdots, N$
- **Decryption** $P_i = C_i \bigoplus E_K(Counter), i = 1, 2, 3, \cdots, N$

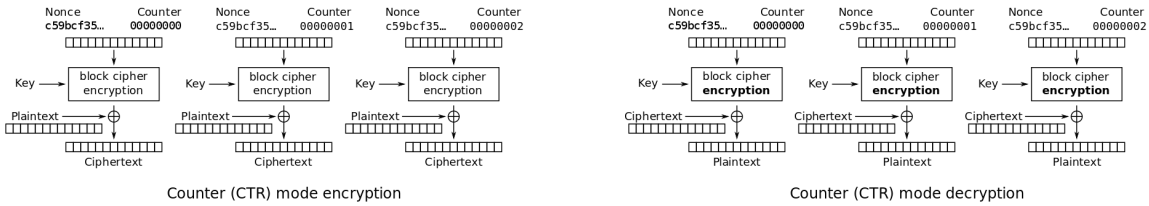

Figure 13.5: CTR Mode

We can similarly alter the equation as OFB so that it can be used as a stream cipher.

Characteristics

Table 13.1 shows the characteristics for each modes of operation.

- Block Pattern: Whether if the overall pattern is kept after encryption
- Preprocessing: Whether if preprocessing is possible on encryption and decryption
- Parallel Processing: Whether if parallel processing is possible on encryption or decryption
- Error Propagation: If there is an error in the encryption/decryption process, whether if the error spreads through other blocks
- Encryption Unit: The minimum requirement byte for encryption

		ECB	CBC	CFB	OFB	CTR
Block Pattern		O	X	X	X	X
Preprocessing		X	X	X	O	O
Parallel Processing	Encryption	O	X	X	O	O
	Decryption	O	O	O	O	O
Error Propagation		X	(P_i, P_{i+1})	$\lceil \frac{n}{r} \rceil$ blocks	X	X
Encryption Unit		n	n	$r(\leq n)$	$r(\leq n)$	$r(\leq n)$

Table 13.1: Characteristics for Each Modes of Operation

13.3 Types of Attack

13.3.1 Attacking Classical Cryptosystems

Classical Cryptosystems are typically a substitution cipher and/or a transposition cipher. Since most, if not all, the classical cryptosystems are broken, the two valid ways to attack any classical cryptosystems is given here.

- Brute Force Attack

  When the attacker gains the ciphertext c, the attacker uses every key possibility to try to gain m. This is otherwise known as the Exhaustive Key Search Attack. Theoretically this can be done to any symmetric-key cipher; but this is inapplicable to most modern cryptosystems as they have an extremely large key space.

- Frequency Analysis

  The plaintext having some pattern, such as the alphabet 'e' appearing with the most frequency, will help the attacker gain knowledge on the plaintext just by seeing the ciphertext.

13.4 Cryptographic Hash Functions

A general hash function has the following properties:

- They take an arbitrary size of data as input, and;
- They produce a constant and fixed length data as output.

A cryptographic hash function, in addition to the properties above, must have the following properties:

- Preimage Resistance

  If the hash value y is given, it must be hard to find an x such that $h(x) = y$, that is, the hash function must have one-wayness.

- Second Preimage Resistance

  If the message x is given, it must be hard to find an $x' \neq x$ such that $h(x) = h(x')$.

- Collision Resistance

  It must be hard to find $x \neq x'$ such that $h(x) = h(x')$. The pair (x, x') is called the collision pair.

13.5 Attacking the Cyrptosystems

Attacks on cryptosystems are classified into passive and active attack. Passive attacks simply eavesdrops the transmission, and gains what the attacker wants without modification of the message. This type of attacking includes eavesdropping, of which the attacker intercepts the message in the middle to check the plaintext. This type of attacker is often referred to as ”Eve”(as in eavesdropping) in theories. Active attackers will modify the message, which includes Modification, Deletion, Impersonation, and Replay. This type of attacker can also be referred to as ”Eve”, but sometimes is referred to as ”Mallory”, for malicious user.

- Modification: Changes the order of the message or changes a part of it to alter the meaning.
- Deletion: Intercepts the message and does not send it, interrupting the communication.
- Impersonation: Fakes their own identity to be identified as a correct user.
- Replay: Send a message again after eavesdropping, expecting some kind of result.

The four methods above are just the general ways to attack. We need to attack the system itself to know how to attack it. There are four methods of attack on system:

- **Ciphertext Only Attack**

  The attacker knows only the ciphertext.

- **Known Plaintext Attack**

  The attacker knows a list of (message, ciphertext) pair, and attempts to crack a ciphertext not in the list.

- **Chosen Plaintext Attack**

  The attacker has access to an oracle that can encrypt the message, and attempts to crack a ciphertext.

- **Chosen Ciphertext Attack**

  The attacker has access to an oracle that can decrypt a ciphertext, except for the target ciphertext.

There are three important properties to encryption schemes:

- Semantic Security

  A semantically secure encryption scheme is infeasible for any computationally bounded adversary to derive a significant information about the original plaintext when given only its ciphertext and the corresponding public key if any. This can be represented as a game between the oracle and the adversary, as below:

  1. The oracle generates a key for the challenge.
  2. The adversary is given the encryption oracle(or the public key, in the case of public key cryptosystem).
  3. The adversary can perform any number of polynomially bounded number of encryptions or operations.

4. The adversary generates two equal-length messages m_0 and m_1, and transmits it to the oracle.
5. The oracle randomly chooses $b \in \{0,1\}$ to encrypt the message m_b to C.
6. The adversary, upon receiving C, guesses b.

If the adversary cannot guess b correctly with significantly greater than 50% probability, then the scheme is said to be semantically secure under CPA.

- Indistinguishability

  If a cryptosystem is indistinguishable, then an adversary would not be able to distinguish pairs of ciphertexts based on the message they encrypt. There are three types: IND-CPA, IND-CCA1, and IND-CCA2. They can be represented as a game between the oracle and the adversary. In both cases, they are said to be secure if the adversary does not have a clear advantage.

  – IND-CPA

  1. The oracle generates a key for the challenge.
  2. The adversary is given the encryption oracle(or the public key, in the case of public key cryptosystem).
  3. The adversary can perform any number of polynomially bounded number of encryptions or operations.
  4. The adversary generates two distinct equal-length messages m_0 and m_1, and transmits it to the oracle.
  5. The oracle randomly chooses $b \in \{0,1\}$ to encrypt the message m_b to C.
  6. The adversary, upon receiving C, performs polynomially bounded encryptions or operations, and guesses b.

  – IND-CCA

  1. The oracle generates a key for the challenge.
  2. The adversary is given the decryption oracle and the public key, in the case of public key cryptosystem.

     Note that in the case of the public key cryptosystem, the encryption oracle is also given.
  3. The adversary can perform any number of polynomially bounded number of decryptions or operations.
  4. The adversary generates two distinct equal-length messages m_0 and m_1, and tramsmits it to the oracle.
  5. The oracle randomly chooses $b \in \{0,1\}$ to encrypt the message m_b to C.
  6. The adversary, upon receiving C, performs polynomially bounded operations.

     In the case of IND-CCA1, the adversary may not make further calls to the decryption oracle.

     In the case of IND-CCA2, the adversary may make further calls to the decryption oracle, but may not submit C.
  7. The adversary guesses b.

This can be said with a random oracle. In that case, the adversary submits only one message and the oracle returns the encryption of the message or the random string equal to the length of the encryption with a fair chance. The adversary then guesses whether if the message is randomly generated or encrypted.

- Non-malleability

  Cryptosystems are called ”malleable” if it is possible to transform a ciphertext into another ciphertext which decrypts to a related plaintext. Cryptosystems that are not malleable are called non-malleable. These, similar to indistinguishability, can be represented as a game between the oracle and the adversary, and are called NM-CPA, NM-CCA1, NM-CCA2.

Theorem 141
The following relations for each security properties hold:

- IND-CPA $\Leftrightarrow$ Semantic security under CPA
- NM-CPA $\Rightarrow$ IND-CPA
- NM-CCA2 $\Leftrightarrow$ IND-CCA2
- NM-CPA does not necessarily imply IND-CCA2.

13.6 Digital Signatures

Digital signatures are used in pair with the public key cryptosystems to verify the sender of the messages. When attacking, there are three major methods:

- **Key-Only Attack**

  The attacker only has access to the digital signature algorithm and the public key of the signer, pk_A. This is similar to the Ciphertext Only attack.

- **Known Message Attack**

  The attacker has access to the digital signature algorithm, the public key of the signer, and a list of (message, signature) pairs. This is similar to the Known Plaintext attack.

- **Chosen Message Attack**

  The attacker has access to the digital signature algorithm, the public key of the signer, and an oracle that takes a message as an input and returns signature as an output.

The attacker can have three different purposes:

- **Total Break**

  The attacker wants to gain the private key of the signer.

- **Selective Forgery**

  The attacker wants to generate a valid signature for a message the attacker wants(i.e. any message for that matter).

- **Existential Forgery**

  The attacker wants to generate a valid (message, signature) pair for any message.

It is said that an attack is valid if the attack succeeds with a non-negligible probability.

13.7 Zero-Knowledge Authentication

Three major ways to authenticate a user is using password, challenge-response, and zero-knowledge authentication. Passwords must be sent through network, thereby they are susceptible to interception. Challenge-response can be abused by malicious users to crack the secret key. That is where the concept of zero-knowledge interactive proof comes in.

An interactive proof system can be described as a communication between the verifier and the prover. They exchange messages to check whether if the statement is true or false. In here, the prover is assumed to have unlimited calculating power but cannot be trusted; the verifier has bounded computation power but is assumed to be always honest. Messages are sent between the prover and the verifier until the verifier has an answer to the problem and has convinces itself that the answer is correct. Any interactive proof system must have the following properties:

- **Completeness**: If the statement is true, the honest verifier will be convinced of this fact by an honest prover.
- **Soundness**: If the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability.

In authentication, if the proof is only interactive, a malicious verifier may abuse the protocol to reveal the "knowledge"(in the case for cryptosystems, private keys) only the prover knows. This is where the concept of "Zero-knowledgeness" comes in.

- **Zero-knowledgeness**: If the statement is true, no verifier can learn anything apart from the fact that the statement is true.

The best way to describe this is by an analogy of a colorblind person. Suppose the person has two balls that looks the exactly same for them. Their friend, as a non-colorblind person, want to prove that the two balls are of different color. The colorblind person resumes the role of verifier and the non-colorblind friend the prover. Here is an example protocol on how the fact can be proven:

1. Verifier shows you a ball.
2. Prover memorize it.
3. Verifier then hides both balls, and choose to keep the ball shown before or change the ball.
4. Verifier shows the newly chosen ball.
5. Prover tell verifier whether if the ball has been changed or not.
6. If the prover is wrong, the prover has told a lie; end the protocol.
7. If the prover is right, the prover may be telling the truth; continue the protocol until convinced.

If the statement('The two balls are of different color') is false, then the prover(in this case, cheating) cannot tell whether if the ball has been changed; therefore their guess is right for 50% of the time. n consecutive application of the protocol gives $\frac{1}{2^n}$ chance of success, and as the number of trials increase, the less the cheating prover will be able to pass the protocol.

If the statement, on the other hand, is indeed true, then the prover can tell whether if the ball has been switched every time. In the verifier's point of view, the prover's n-th consecutive success for verification proves that they are lying at $\frac{1}{2^n}$ probability; their improbable probability of success at lying will thereby prove their honesty.

13.8 RSA Cryptosystem and Signature

13.8.1 Keygen

1. Choose two primes p and q.
2. Let $n = p \cdot q$.
3. Choose e such that $(e, \phi(n)) = 1$
4. Find d such that $e \cdot d \equiv 1 \mod \phi(n)$

Public Key: (n, e)
Private Key: d or (p, q, d), depending on the method.

13.8.2 Cryptosystem

Encryption

$C \equiv M^e \mod n$

Decryption

- **Basic Method**
  $C^d \equiv (M^e)^d \equiv M^{\phi(n) \cdot k + 1} \equiv M \mod n$
- **Chinese Remainder Theorem**
  Split $C^d \mod n$ into two congruences: $C^d \mod p$ and $C^d \mod q$.
  Using Euler's Theorem(If $(a, n) = 1$, $a^{\phi(n)} \mod n$.), reduce d to reduce the number of multiplication.

13.8.3 Signature

Signing

$S \equiv M^d \mod n$

Verifying

Compare $S^e \mod n$ to M. If equal, accept; otherwise reject.

13.8.4 Attacking the Cryptosystem

On the Case of Exposed Private Key e

For the public key, $n = pq$ where p and q are primes.
Then, $\phi(n) = (p-1)(q-1)$.
We know that $ed \equiv 1 \mod \phi(n)$.
By the definition of modular, $ed - 1 = k\phi(n)$ for some k.
For a large enough $n = pq$, $\frac{\phi(n)}{n} = \frac{(p-1)(q-1)}{pq} = 1 - \frac{1}{p} - \frac{1}{q} + \frac{1}{pq} \simeq 1$.
We can find k by dividing both sides of the equation $ed - 1 = k\phi(n)$ by n, since $\frac{ed-1}{n} = k\frac{\phi(n)}{n} \simeq k$.

We can then find $\phi(n) = \frac{ed-1}{k}$.
Since $n = pq$ and $\phi(n) = (p-1)(q-1) = pq - (p+q) + 1 = n - (p+q) + 1$, $p+q = n - \phi(n) + 1$.
Then the quadratic equation $(x-p)(x-q) = x^2 - (p+q)x + pq = x^2 - (n - \phi(n) + 1)x + n = 0$ can be solved to yield p and q.

Chosen Ciphertext Attack

1. Alice sends $C \equiv M^e \mod n$ to Bob
2. Eve intercepts Alice's transmission; chooses x s.t. $(x, n) = 1$ (and therefore $x^{-1} \mod n$ exists) to send $C' = Cx^e \mod n$ to Bob.
3. Bob decrypts C' as $(C')^d \equiv (Cx^e)^d \equiv C^d x^{ed} \equiv Mx \mod n$
4. Eve intercepts Bob's decryption result, Mx, and multiplies x^{-1} modulo n to gain M.

Coppersmith Attack

Theorem 142 (Coppersmith)
Let $n \in \mathbb{Z}$ and $f \in \mathbb{Z}[x]$ be a monic polynomial (i.e. leading coefficient of f is 1) of degree d over integer.
Set $X = n^{1/d - \epsilon}$ for $1/d > epsilon > 0$. Then given n and f, the attacker, using the LLL Algorithm, can efficiently find all integer $x_0 < X$ such that $f(x_0) \equiv 0 \mod n$.

Note
In the case of RSA, Finding M when given $C \equiv M^e \mod n$ can be interpreted as finding the solution of the equation $f(x) \equiv x^e - C \mod n$. This attack's strength is the ability to find all small roots of the polynomials modulo a composite N.

Håstad's Broadcast Attack

This attack is viable if the value of e is fixed and is small, and the same message is broadcast without padding.

Suppose the same plaintext M is encrypted to multiple people, each using same e and different moduli, say N_i. If Eve successfully intercepts e or more messages, say $C_1, C_2, \cdots, C_e$, $C_i \equiv M^e \mod N_i$. We may assume $(N_i, N_j) = 1$ for $i \neq j$, otherwise the attacker will be able to factorize some N_i by finding their GCD. Using the Chinese Remainder Theorem on the e congruences, the attacker may compute $C \in Z^*_{\Pi N_i}$ such that $C_i \equiv C \mod N_i$. Then, $C \equiv M^e \mod \Pi N_i$; however since $M < N_i$ for each i, $M^e < \Pi N_i$; thus $C = M^e$ holds over the integers, and the attacker can easily find the message M.

For more generalized version, the following theorem is available:

Theorem 143 (Håstad)
Suppose $N_1, \cdots, N_k$ are relatively prime integers and set $N_{\min} = \min_i\{N_i\}$. Let $g_i(x) \in \mathbb{Z}/N_i[x]$ be k polynomials of maximum degree q. Suppose there exists a unique $M < N_{\min}$ satisfying $g_i(M) \equiv 0 \mod N_i \forall i \in \{1, \cdots, k\}$. Furthermore, suppose $k > q$. Then there is an efficient algorithm which, given $\langle N_i, g_i(x) \rangle \forall i$, computes M.

This theorem can be used in the following way:
Suppose the i-th plaintext is padded with the polynomial $f_i(x)$. Let $g_i(x) = (f_i(x))^{e_i} - C_i \mod N_i$. Then $g_i(M) \equiv 0 \mod N_i$ is true, and the Coppersmith's Attack[13.8.4] can be used.

Franklin-Reiter Related Message Attack

This attack is viable if the value of e is fixed and is small, and the same message is broadcast with padding.

Theorem 144
Let (n, e) be the public key of RSA, and e is small. Let $f(x) = ax + b \in Z_n[x]$, $b \neq 0$; i.e. f is the padding function.
Suppose that $M_1 \neq M_2$ and $M_1 \equiv f(M_2) \mod n$.
Then, given the quintuplet (n, e, C_1, C_2, f), M_1 and M_2 can be recovered in $O((log_2 n)^2)$

Proof.
$C_1 \equiv M_1^e \mod n$
$C_2 \equiv M_2^e \mod n$
$M_1 \equiv f(M_2) \equiv aM_2 + b \mod n$

Let $g_2(x) = x^e - C_2 \mod n$, and $g_1(x) = (ax + b)^e - C_1 \mod n$

$g_1(x) = (ax + b)^e - C_1$
$= (ax + b)^e - M_1^e$
$= (ax + b)^e - (aM_2 + b)^e$
$= ((ax + b) - (aM_2 + b))Q(x)$
$= a(x - M_2)Q(x)$

$g_2(x) = x^e - C_2$
$= x^e - M_2^e$
$= (x - M_2)Q'(x)$

$\rightarrow (x - M_2)|(g_1(x), g_2(x)$
Using the euclidean algorithm on the two polynomials g_1 and g_2, M_2 can be recovered. □

13.8.5 Forgeries of the Signature

Known Message Attack

Suppose (M_1, S_1) and (M_2, S_2) are both valid signatures.
Then, $(M_1 M_2, S_1 S_2)$ is also a valid signature.

Chosen Message Attack

Eve chooses M_1 and M_2 s.t. $M = M_1 M_2$.
Eve asks Alice to sign M_1 and M_2; let them be S_1 and S_2.
Then $S_1 S_2$ is a valid signature for M.

13.9 ElGamal Cryptosystem

13.9.1 Keygen

Choose a prime p. Note that $(Z_p^*, \times)$ is a cyclic group.

- Choose e_1 to be the primitive root of $(Z_p^*, \times)$
- Choose $d \in Z_p^*$ and compute $e_2 \equiv e_1^d \mod p$

In theory, p and e_1 can be shared as long as e_2 are kept distinct. **Public Key:** (e_1, e_2, p)
Private Key: d

13.9.2 Cryptosystem

Encryption

Randomly choose $r \in Z_p^*$. M is the message.

- $C_1 \equiv e_1^r \mod p$
- $C_2 \equiv M e_2^r \mod p$

Ciphertext: (C_1, C_2)

Decryption

$C_2(C_1^d)^{-1} \equiv M e_2^r (e_1^{rd})^{-1} \equiv M (e_1^d)^r (e_1^{rd})^{-1} \equiv M \mod p$

13.9.3 Signature

Signing

Randomly choose $r \in Z_p^*$. M is the message.

- $S_1 \equiv e_1^r \mod p$
- $S_2 \equiv (M - dS_1) r^{-1} \mod (p-1)$

Signature: (S_1, S_2)

Verifying

Calculate:

- $V_1 \equiv e_1^M \mod p$
- $V_2 \equiv e_2^{S_1} S_1^{S_2} \mod p$

Verify with:

- Check $0 < S_1 < p$, $0 < S_2 < p - 1$.
- Check $V_1 = V_2$

$V_2 \equiv e_2^{S_1} S_1^{S_2} \equiv (e_1^d)^{S_1} (e_1^r)^{S_2} \equiv e_1^{dS_1 + rS_2} \equiv e_1^M \equiv V_1 \mod p$

13.9.4 Attacking the Cryptosystem

Exposure of r

Since (C_1, C_2) and r are exposed, $M = C_2 (e_2^r)^{-1} \mod p$.

Baby step, Giant step

When the random number r is small, then the following meet-in-the-middle attack is possible:
$y = e_1^x \mod p$.
Let $m = \lceil \sqrt{p} \rceil$.
Then, $\exists q, r \in \mathbb{Z}$ such that $x = mq + r, 0 \le r \le m - 1$
$\Rightarrow y = e_1^x \equiv e_1^{mq+r} \mod p$
$\Rightarrow y(e_1^{-m})^q \equiv g^r \mod p$
Hence we can find r using the following protocol:

1. Construct the table with entries $(r, e_1^r \mod p), 0 \le r \le m - 1$: (Baby step table)
2. Compute the value $g^{-m} \mod p$: (Giant step value)
3. For q from 0 to $m - 1$, find q such that $y(g^{-m})^q \equiv g^r \mod p$ in the table.

Known Plaintext Attack

Suppose the random number r is reused to encrypt two distinct messages, M and M'.
Suppose M encrypted to (C_1, C_2); M' encrypted to (C_1', C_2').
Note that $C_1 = C_1' = e_1^r$, $C_2 = Me_2^r$, $C_2' = M'e_2^r$.
If we know M', then $\frac{C_2 \times M'}{C_2'} = \frac{Me_2^r \times M'}{M'e_2^r} = M$

13.9.5 Forgeries of the Signature

Constructing from Scratch: One Variable

Choose $1 < x < p-1$.

- $S_1 \equiv e_1^x e_2 \mod p$
- $S_2 \equiv -S_1 \mod (p-1)$
- $M \equiv xS_2 \mod (p-1)$

Constructing from Scratch: Two Variables

Choose $u, v \in Z_p^*$ such that $(v, p-1) = 1$ so that $\exists v^{-1} \mod (p-1)$

- $S_1 \equiv e_1^u e_2^v \mod p$
- $S_2 \equiv -S_1 v^{-1} \mod (p-1)$
- $M \equiv S_2 u \mod (p-1)$

Known Plaintext Attack

A valid signature $(M, (S_1, S_2))$ is given for $(M, p-1) = 1$ so that $\exists M^{-1} \mod (p-1)$.
Choose a message M'.
Set $u = M'M^{-1} \mod (p-1)$.
Compute $S_2 \equiv S_2 u \mod (p-1)$.
Solve the following set of linear congruences using CRT:

$$\begin{cases} S_1' = S_1 u \mod (p-1) \\ S_1' = S_1 \mod p \end{cases}$$

Then, $(M', (S_1', S_2'))$ is also a valid signature, if the range conditions are not checked.

13.10 Schnorr Digital Signature

Signatures based on cryptosystems have a weakness: they pose a threat to expose the secret key, or makes it easier to forge a specific message. Schnorr Digital Signature is a signature-only algorithm that helps solve this.

13.10.1 Keygen

- Choose a cryptographic hash function h.
- Choose a prime p.

- Choose a prime q such that:

  $q|p-1$, and;

  The size of q is the same as the hash output.

- Choose e_0 such that it is a generator in Z_p^*.
- Set $e_1 \equiv e_0^{(p-1)/q} \not\equiv 1 \mod p$.
- Choose d.
- Set $e_2 \equiv e_1^d \mod p$.

Public Key: (h, e_1, e_2, p, q)
Private Key: d

13.10.2 Signature

Signing

Choose $r \in Z_q^*$ at random.

- $S_1 = h(M||e_1^r \mod p)$ where $||$ is concatenation.
- $S_2 = r + dS_1 \mod q$

Signature: (S_1, S_2)

Verifying

Calculate $V = h(M||e_1^{S_2} e_2^{-S_1} \mod p)$.
If $V = S_1$, then M is accepted.
$e_1^{S_2} e_2^{-S_1} = e_1^{r+dS_1} e_1^{-dS_1} = e_1^r \mod p$

Part III

Appendix

Chapter 14

Appendix

14.1 Cook-Levin Theorem

In this section

14.2 Kuratowski Theorem

In this section we prove [134].

14.2.1 The Preparation

First, we show that a planar graph can be drawn so that an arbitrary vertex or an edge is incident to the outer face.

Lemma 145
If G is planar and $v \in V(G)$, then there is a planar embedding of G such that v is on the boundary of the outer face. The same can be done for $e \in E(G)$.

Proof. We use the stereographic projection. In $\mathbb{R}^3$, let $z = -1$ be the plane P and $x^2 + y^2 + z^2 = 1$ be the sphere S. $(0,0,1)$ is the "north pole" of S. Define the projection $\rho : S \backslash \{(0,0,1)\} \to P$ as follows: given (x,y,z) on S which is not the north pole, draw a straight line through $(0,0,1)$ and (x,y,z). There is a unique intersection of this line with P, denoted as $(X,Y,-1)$. Then $\rho(x,y,z) = (X,Y,-1)$. Clearly ρ is bijective.

Given an embedding of a planar graph G on P, ρ^{-1} gives an embedding of G on S. Rotate the embedding so that a face incident to v or e contains the north pole. ρ gives an embedding of G on P such that the face is the outer face. □

Next, we introduce the notion of connectivity. Although connectivity is a crucial part of graph theory, we didn't put this into the main part of the codex because of the length concerns.

Definition 146 (Connectivity)
A graph G is k-connected if $|V| > k$ and, for every $S \subset V$ with $|S| < k$, $G \backslash S$ is connected.

Theorem 147
If G is 3-connected with $|V(G)| \geq 5$, then there is an edge e such that G/e is 3-connected.

Proof. Let $e = xy$ and suppose G/e is not 3-connected. Then G/e has a cut set $\{v, z\}$. Since G is 3-connected, this set has a vertex, say v, which is the new vertex made by contracting e. That is, $\{x, y, z\}$ is a cut set of G.

Suppose that for every e, G/e is not 3-connected, so to every e corresponds a vertex z_e. Among all edges, take $e = xy$ and z_e such that $G - x - y - z$ has the largest component C, and denote another component as D. Each of x, y, z has neighbors in C and in D since G is 3-connected. Take a neighbor u of z in D and let $v = z_{zu}$.

If $v \in V(C) \cup \{x, y\}$, then $G - z - v$ is disconnected, contradicting the connectivity of G. Otherwise, $G - z - u - v$ has a component that contains all vertices in C and x and y in addition, contradicting the choice of C.

(TODO: picture) □

Then, we show the connection between minors and topological minors.

Lemma 148
$K_{3,3}$ is a topological minor of G iff $K_{3,3}$ is a minor of G.

Proof. A topological minor of G is also a minor of G. We just need to prove the other direction of the lemma.

□

Lemma 149
If K_5 is a minor of G, then $K_{3,3}$ or K_5 is a topological minor of G.

Proof. . □

14.2.2 The Proof

The last step is closely related to the Kuratowski's theorem.

Definition 150 (Convex Embedding)
A convex embedding of a planar graph G is a plane graph in which all edges are straight line segments and all face boundaries are convex polygons.

Lemma 151
If G is simple, 3-connected, and has no K_5 or $K_{3,3}$ as a minor, then G has a convex embedding on a plane, with no three vertices on a line.

Proof. TODO □

We are finally ready to prove the Kuratowski's theorem. For convenience, we will restate the theorem:

> A graph G is planar if and only if it does not have K_5 or $K_{3,3}$ as a topological minor.

Proof. Induction on $|V|$, with trivial base case $|V| \leq 4$.

If G is disconnected, from induction there is a planar embedding of each component. Since each embedding is bounded by a finite disc, their union can be drawn on a plane.

If G is connected but not 2-connected, then take a cut-vertex v. Let G_1, $\cdots$, G_n be the connected components of $G - v$, and H_i be the subgraph induced by $V(G_i) \cup \{v\}$. Take an embedding of each H_i such that v is in the outer face [145] and squeeze it into an angle $< 2\pi/n$ at the vertex v. Joining those embeddings together forms an embedding of G.

If G is 2-connected but not 3-connected, TODO

If G is 3-connected, the conclusion immediately follows from [151]. □