Introduction to Languages and Computation

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

LSSU Math 400

Subsection 1

Automata, Computability and Complexity

Theory of Computation

- The theory of computation includes three central areas:
    - automata;
    - computability;
    - complexity.
- They are linked by the question:

> What are the fundamental capabilities and limitations of computers?

- The question goes back to the 1930s when mathematical logicians first began to explore the meaning of computation.
- Technological advances have increased computing capabilities and have made this question of significant practical concern.
- In each of the three areas the question is interpreted differently, and the answers vary according to the interpretation.
- This set of notes deals primarily with automata and computability; complexity is dealt with in another companion set.

Complexity Theory

- Some computer problems are easy, and some are hard.
  - The sorting problem is an easy one: Even a small computer can sort a million numbers very quickly.
  - Scheduling problems are hard: Finding a schedule of classes for the entire university to satisfy some reasonable constraints (e.g., no two classes take place in the same room at the same time) seems to be much harder than the sorting problem. Finding the best schedule may require centuries, even with a supercomputer.
- What makes some problems computationally hard and others easy? This is the central question of complexity theory.
- Even though the answer to this question is still not known, researchers have discovered an elegant scheme for classifying problems according to their computational difficulty.
- This may be viewed as analogous to the periodic table for classifying elements according to their chemical properties.

Hard Problems: Options and Advantages

- If a problem appears to be computationally hard:
  - By understanding which aspect of the problem is at the root of the difficulty, alter it so that the problem is more easily solvable.
  - Settle for less than a perfect solution to the problem, e.g., an approximate rather than a perfect solution.
  - If the problem is hard only in the worst case situation, but easy most of the time, depending on the application, devise a procedure that is occasionally slow but usually runs quickly.
  - Consider alternative types of computation, such as randomized computation, that can speed up certain tasks.
- Complexity theory has affected the ancient field of cryptography:
  - In most fields, an easy computational problem is preferable to a hard one because easy ones are cheaper to solve.
  - Cryptography, on the contrary, requires computational problems that are hard, rather than easy, because secret codes should be hard to break without the secret key or password. Complexity theory has pointed cryptographers in the direction of computationally hard problems around which they have designed revolutionary new codes.

Computability Theory

- During the first half of the twentieth century, mathematicians such as Kurt Gödel, Alan Turing, and Alonzo Church discovered that certain basic problems cannot be solved by computers.
  - One example of this phenomenon is the problem of determining whether a mathematical statement is true or false. It seems like a natural for solution by computer because it lies strictly within the realm of mathematics. But no computer algorithm can perform this task.
- This result led to the development of theoretical models of computers that, later, resulted in the construction of actual computers.
- The theories of computability and complexity are closely related.
  - In complexity theory, the objective is to classify problems as easy ones and hard ones;
  - In computability theory the classification of problems is by those that are solvable and those that are not.
  - Computability theory introduces several of the concepts used in complexity theory.

Automata Theory

- Automata theory deals with the definitions and properties of mathematical models of computation.

  These models play a role in several applied areas of computer science.
  - One model, called the finite automaton, is used in text processing, compilers, and hardware design.
  - Another model, called the context-free grammar, is used in programming languages and artificial intelligence.
- The theories of computability and complexity require a precise definition of a computer and automata theory provides a good starting point.

Subsection 2

Mathematical Notions and Terminology

Sets

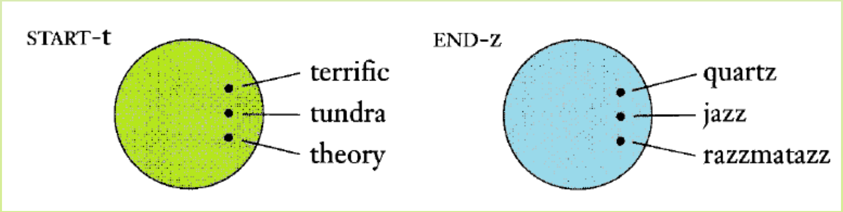
- A **set** is a group of objects represented as a unit.
- The objects in a set are called its **elements** or **members**.
- One way to describe a set is by listing its elements inside braces.
- Example: The set $\{7, 21, 57\}$ contains the elements 7, 21, and 57.
- The symbols $\in$ and $\notin$ denote set membership and nonmembership.
- Example: We write $7 \in \{7, 21, 57\}$ and $8 \notin \{7, 21, 57\}$.
- For two sets A and B, we say that A is a **subset** of B, written $A \subseteq B$, if every member of A also is a member of B.
- We say that A is a **proper subset** of B, written $A \subsetneq B$, if A is a subset of B and not equal to B.
- Order of describing a set or repetitions do not matter.
- Example: We get the same set by writing $\{57, 7, 7, 7, 21\}$.
- If we do want to take the number of occurrences of members into account we call the group a **multiset** instead of a set.
- Example: $\{7\}$ and $\{7, 7\}$ are different as multisets but identical as sets.

Infinite Sets and Properties Defining Sets

- An **infinite set** contains infinitely many elements.
- Since we cannot list all the elements of an infinite set, we sometimes use the "..." notation to mean "continue the sequence forever".
- Example: The set of **natural numbers** $\mathbb{N}$ is written $\mathbb{N} = \{1, 2, 3, \ldots\}$. The set of **integers** $\mathbb{Z}$ is written $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
- The set with 0 members is called the **empty set** and is written $\emptyset$.
- When we want to describe a set containing elements according to some rule, we write $\{n : \text{rule about } n\}$.
- Example: $\{n : n = m^2 \text{ for some } m \in \mathbb{N}\}$ means the set of perfect squares.

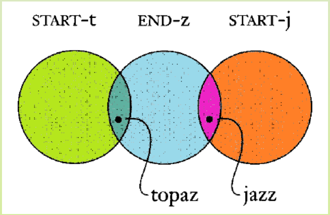
Union, Intersection and Complements of Sets

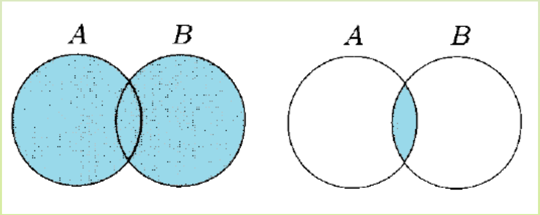
- If we have two sets A and B, the **union** of A and B, written $A \cup B$, is the set consisting of all the elements in A or in B.
- The **intersection** of A and B, written $A \cap B$, is the set of elements that are in both A and B.
- The **complement** of A, written $\overline{A}$, is the set of all elements under consideration that are not in A.
- To illustrate concepts involving sets, we use **Venn diagrams**.
- Example: Let START-t be the set of all English words that start with the letter "t" and END-z the set of English words that end with "z":



Venn Diagrams

- To represent both sets in the same Venn diagram we must draw them so that they overlap, indicating that they share some elements, e.g., the word “topaz” is in both sets.
- The figure also contains a circle for the set START-j. This circle does not overlap the circle for START-t because no word lies in both sets.





- The two Venn diagrams on the right depict the union and intersection of sets A and B.

Sequences and Tuples

- A **sequence** of objects is a list of these objects in some order. We usually designate a sequence by writing the list within parentheses.
- Example: The sequence 7, 21, 57 would be written $(7, 21, 57)$.
- Unlike in a set, in a sequence the order does matter.
- Example: The sequence $(7, 21, 57)$ is not the same as $(57, 7, 21)$.
- Similarly, repetition does matter in a sequence, but it does not matter in a set.
- Example: $(7, 7, 21, 57)$ is different from both $(7, 21, 57)$ and $(57, 7, 21)$, whereas the set $\{7, 21, 57\}$ is identical to the set $\{7, 7, 21, 57\}$.
- Finite sequences often are called **tuples**. A sequence with k elements is a k-**tuple**.
- Example: $(7, 21, 57)$ is a 3-tuple. A 2-tuple is also called a **pair**.
- Sets and sequences may appear as elements of other sets and sequences.

Power Sets and Cartesian Products

- The **power set** of A is the set of all subsets of A.
- Example: If $A = \{0, 1\}$, the power set of A is $\{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.
- If A and B are two sets, the **Cartesian product** or **cross product** of A and B, written $A \times B$, is the set of all pairs with first element in A and second element in B.
- Example: If $A = \{1, 2\}$ and $B = \{x, y, z\}$, $A \times B = \{(1, x), (1, y), (1, z), (2, x), (2, y), (2, z)\}$.
- The **Cartesian product** of k sets, $A_1, A_2, \ldots, A_k$, written $A_1 \times A_2 \times \cdots \times A_k$, is the set of all k-tuples $(a_1, a_2, \ldots, a_k)$, where $a_i \in A_i$.
- Example: If A and B are as above,
$$\begin{aligned} A \times B \times A \quad = \quad & \{(1, x, 1), (1, x, 2), (1, y, 1), (1, y, 2), (1, z, 1), (1, z, 2), \\ & (2, x, 1), (2, x, 2), (2, y, 1), (2, y, 2), (2, z, 1), (2, z, 2)\}. \end{aligned}$$
- If we have the Cartesian product of A with itself (k factors), we use the shorthand $A \times A \times \cdots \times A = A^k$.
- Example: The set $\mathbb{N}^2$ equals $\mathbb{N} \times \mathbb{N}$. It consists of all pairs of natural numbers. We also may write it as $\{(i, j) : i, j \geq 1\}$.

Functions

- A **function** is an object that sets up an input-output relationship; it takes an input and produces a unique output.
- $f(a) = b$ means that f produces output b when the input is a.
- A function is also called a **mapping**, and, if $f(a) = b$, we say that f **maps** a **to** b and write $a \overset{f}{\mapsto} b$.
- Example: The **absolute value** function abs takes a number x as input and returns x, if x is positive, and $-x$, if x is negative. Thus $\text{abs}(2) = \text{abs}(-2) = 2$.
- Example: **Addition** is a function, written add. The input to the addition function is a pair of numbers, and the output is the sum of those numbers.
- The set of possible inputs to the function is called its **domain**.
- The outputs of a function come from a set called its **range**.
- The notation for saying that f is a function with domain D and range R is $f : D \to R$.

More on Functions

- Example:
  - In the case of the function abs, if we are working with integers, the domain and the range are $\mathbb{Z}$, so we write abs : $\mathbb{Z} \to \mathbb{Z}$.
  - In the case of the addition function for integers, the domain is the set of pairs of integers $\mathbb{Z} \times \mathbb{Z}$ and the range is $\mathbb{Z}$, so we write add : $\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$.
- Note that a function may not necessarily use all the elements of the specified range. The function abs never takes on the value -1 even though $-1 \in \mathbb{Z}$.
- A function that does use all the elements of the range is said to be **onto** the range.

Using Tables to Describe Functions

- One way to describe a function is with a procedure for computing an output from a specified input.
- Another way is with a table that lists all possible inputs and gives the output for each input.
- Example: Consider the function $f : \{0,1,2,3,4\} \to \{0,1,2,3,4\}$:

n	$f(n)$
0	1
1	2
2	3
3	4
4	0

  This function adds 1 to its input and outputs the result modulo 5.
- A number **modulo** m is the remainder after division by m. When we do modular arithmetic we define $\mathbb{Z}_m = \{0,1,2,\ldots,m-1\}$. With this notation, the aforementioned function f has the form $f : \mathbb{Z}_5 \to \mathbb{Z}_5$.

Two-Dimensional Tables

- Sometimes a two-dimensional table is used if the domain of the function is the Cartesian product of two sets.
- Example: Consider a function $g : \mathbb{Z}_4 \times \mathbb{Z}_4 \to \mathbb{Z}_4$. The entry at the row labeled i and the column labeled j in the table is the value of $g(i,j)$.

g	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

The function g is the addition function modulo 4.

k-ary Functions

- When the domain of a function f is $A_1 \times \cdots \times A_k$ for some sets $A_1, \ldots, A_k$, the input to f is a k-tuple $(a_1, a_2, \ldots, a_k)$ and we call the a_i the **arguments** to f.
- A function with k arguments is called a k-**ary function**, and k is called the **arity** of the function.
- If $k = 1$, f has a single argument and f is called a **unary function**.
- If $k = 2$, f is a **binary function**.
- Certain familiar binary functions are written in a special **infix notation**, with the symbol for the function placed between its two arguments, rather than in **prefix notation**, with the symbol preceding.
- Example: The addition function add is usually written in infix notation, with the $+$ symbol between its two arguments, as in $a + b$, instead of in prefix notation add(a, b).

Predicates (or Properties) and Relations

- A **predicate** or **property** is a function whose range is $\{\text{TRUE}, \text{FALSE}\}$.
- Example: Let *even* be a property that is TRUE if its input is an even number and FALSE if its input is an odd number. Thus, $even(4) = \text{TRUE}$ and $even(5) = \text{FALSE}$.
- A property whose domain is a set of k-tuples $A \times \cdots \times A$ is called a **relation**, a **k-ary relation**, or a **k-ary relation on A**.
- A common case is a 2-ary relation, called a **binary relation**. When writing an expression involving a binary relation, we customarily use infix notation.
- Example: "less than" is a relation, usually written with the infix operation symbol $<$. "Equality", written with the $=$ symbol, is another familiar relation.
- If R is a binary relation, $a\ R\ b$ means $a\ R\ b = \text{TRUE}$.
- If R is a k-ary relation, $R(a_1, \ldots, a_k)$ means $R(a_1, \ldots, a_k) = \text{TRUE}$.

Scissors-Paper-Stone

- In Scissors-Paper-Stone, two players simultaneously select a member of the set $\{\text{SCISSORS}, \text{PAPER}, \text{STONE}\}$ and indicate their selections with hand signals. If the two selections are the same, the game starts over. If the selections differ, one player wins, according to the relation *beats*.

beats	SCISSORS	PAPER	STONE
SCISSORS	FALSE	TRUE	FALSE
PAPER	FALSE	FALSE	TRUE
STONE	TRUE	FALSE	FALSE

From this table we determine that SCISSORS *beats* PAPER is TRUE and that PAPER *beats* SCISSORS is FALSE.

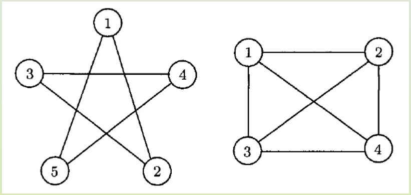
Sometimes describing predicates with sets instead of functions is more convenient. The predicate $P : D \to \{\text{TRUE}, \text{FALSE}\}$ may be written (D, S), where $S = \{a \in D : P(a) = \text{TRUE}\}$, or, simply S, if the domain D is obvious from the context. Hence *beats* may be written $\{(\text{SCISSORS}, \text{PAPER}), (\text{PAPER}, \text{STONE}), (\text{STONE}, \text{SCISSORS})\}$.

Equivalence Relations

- A special type of binary relation, called an equivalence relation, captures the notion of "two objects being equal in some feature".
- A binary relation R is an **equivalence relation** if R satisfies three conditions:
  1. R is **reflexive** if, for every x, $x \ R \ x$;
  2. R is **symmetric** if, for every x, y, $x \ R \ y$ implies $y \ R \ x$;
  3. R is **transitive** if, for every x, y, z, $x \ R \ y$ and $y \ R \ z$ implies $x \ R \ z$.
- Example: Define an equivalence relation on the natural numbers, written $\equiv_7$. For $i, j \in \mathbb{N}$, say that $i \equiv_7 j$, if $i - j$ is a multiple of 7. This is an equivalence relation because it satisfies the three conditions:
  1. It is reflexive, as $i - i = 0$, which is a multiple of 7.
  2. It is symmetric, as $j - i$ is a multiple of 7 if $i - j$ is a multiple of 7.
  3. It is transitive, as whenever $i - j$ is a multiple of 7 and $j - k$ is a multiple of 7, then $i - k = (i - j) + (j - k)$ is the sum of two multiples of 7 and, hence, also a multiple of 7.

Graphs

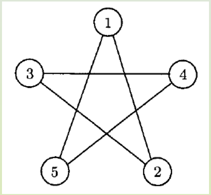
- An **undirected graph**, or simply a **graph**, is a set of points with lines connecting some of the points.
  The points are called **nodes** or **vertices**, and the lines are called **edges**. The number of edges at a particular node is the **degree** of that node.



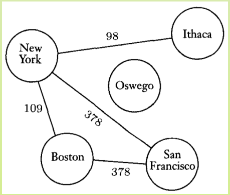
- Example: In the graph on the left, all the nodes have degree 2. In the graph on the right all the nodes have degree 3.
- No more than one edge is allowed between any two nodes.
- In a graph G that contains nodes i and j, the pair (i,j) represents the edge that connects i and j. The order of i and j does not matter in an undirected graph, i.e., (i,j) and (j,i) represent the same edge. Sometimes we describe edges with sets, as in $\{i,j\}$, instead of pairs, if the order of the nodes is unimportant.

Representing Graphs

- If V is the set of nodes and E of edges of G, we say $G = (V, E)$.
- We can describe a graph formally by specifying V and E.
- Example: A formal description of the graph on the right is: $(\{1,2,3,4,5\}, \{(1,2),(2,3),(3,4),(4,5),(5,1)\})$.

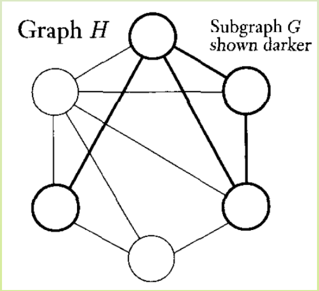


- Graphs frequently are used to represent data.
  - Nodes might be cities and edges the connecting highways;
  - Nodes might be electrical components and edges wires between them.
- Sometimes, for convenience, we label the nodes and/or edges of a graph, which then is called a **labeled graph**. On the right is a graph whose nodes are cities and whose edges are labeled with the cost of the cheapest nonstop air fare, if flying nonstop between the cities is possible.



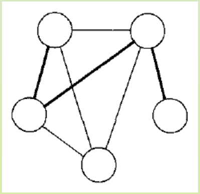
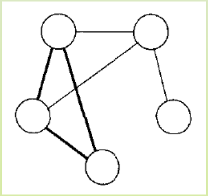
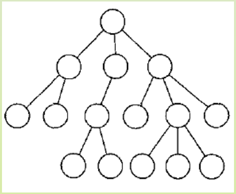
(Induced) Subgraphs

- We say that graph G is a (**induced**) **subgraph** of graph H if the nodes of G form a subset of the nodes of H, and the edges of G are the edges of H on the corresponding nodes.
- The following figure shows a graph H and a subgraph G.



Paths, Cycles and Trees

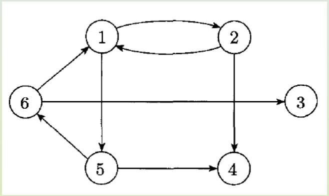
- A **path** in a graph is a sequence of nodes connected by edges.
- A **simple path** is a path that does not repeat any nodes.
- A graph is **connected** if every two nodes have a path between them.
- A path is a **cycle** if it starts and ends in the same node.
- A **simple cycle** is one that contains at least three nodes and repeats only the first and last nodes.

- A graph is a **tree** if it is connected and has no simple cycles. A tree may contain a specially designated node called the **root**. The nodes of degree 1 in a tree, other than the root, are called the **leaves** of the tree.

Directed Graphs

- If a graph has arrows instead of lines, the graph is a **directed graph**



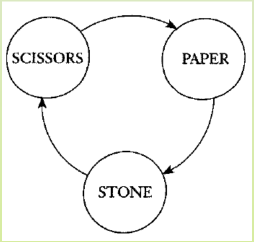
The number of arrows pointing from a particular node is the **outdegree** of that node, and the number of arrows pointing to a particular node is the **in-degree**.

- In a directed graph we represent an edge from i to j as a pair (i, j).
- The formal description of a directed graph G is (V, E), where V is the set of nodes and E is the set of edges.
- Example: The formal description of the graph above is
$$(\{1, 2, 3, 4, 5, 6\}, \{(1, 2), (1, 5), (2, 1), (2, 4), (5, 4), (5, 6), (6, 1), (6, 3)\}).$$
- A path in which all the arrows point in the same direction as its steps is called a **directed path**.
- A directed graph is **strongly connected** if every two nodes are connected by a directed path.

Directed Graphs and Relations

- Directed graphs are a handy way of depicting binary relations.
- If R is a binary relation whose domain is $D \times D$, a labeled graph $G = (D, E)$ represents R, where $E = \{(x, y) : x \ R \ y\}$.
- The binary relation *beats* recalled on the left is represented (or can be depicted) by the graph on the right:

beats	SCISSORS	PAPER	STONE
SCISSORS	FALSE	TRUE	FALSE
PAPER	FALSE	FALSE	TRUE
STONE	TRUE	FALSE	FALSE



Alphabets and Strings

- An **alphabet** is any nonempty finite set. The members of the alphabet are the **symbols** of the alphabet.
- Capital Greek letters Σ and Γ are used to designate alphabets and a typewriter font to designate symbols from an alphabet.
- Example: The following are a few examples of alphabets.

$$\begin{aligned}\Sigma_1 &= \{\mathtt{0},\mathtt{1}\};\\ \Sigma_2 &= \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h},\mathtt{i},\mathtt{j},\mathtt{k},\mathtt{l},\mathtt{m},\\ &\qquad \mathtt{n},\mathtt{o},\mathtt{p},\mathtt{q},\mathtt{r},\mathtt{s},\mathtt{t},\mathtt{u},\mathtt{v},\mathtt{w},\mathtt{x},\mathtt{y},\mathtt{z}\};\\ \Gamma &= \{\mathtt{0},\mathtt{1},\mathtt{x},\mathtt{y},\mathtt{z}\}.\end{aligned}$$

- A **string over an alphabet** is a finite sequence of symbols from that alphabet, written next to one another and not separated by commas.
- Example: If $\Sigma_1 = \{\mathtt{0},\mathtt{1}\}$, then `01001` is a string over Σ_1. If $\Sigma_2 = \{\mathtt{a},\mathtt{b},\mathtt{c},\ldots,\mathtt{z}\}$, then `abracadabra` is a string over Σ_2.

Length, Reverse and Substrings of Strings

- If w is a string over Σ, the **length** of w, written $|w|$, is the number of symbols that it contains.
- The string of length zero is called the **empty string** and is written ε. The empty string plays the role of 0 in a number system.
- If w has length n, we can write $w = w_1 w_2 \ldots w_n$, where each $w_i \in \Sigma$.
- The **reverse** of w, written $w^{\mathcal{R}}$, is the string obtained by writing w in the opposite order, i.e., $w^{\mathcal{R}} = w_n w_{n-1} \ldots w_1$.
- String z is a **substring** of w if z appears consecutively within w.
- Example: `cad` is a substring of `abracadabra`.

Concatenation, Lexicographic Ordering and Languages

- If we have string x of length m and string y of length n, the **concatenation** of x and y, written xy, is the string obtained by appending y to the end of x, as in $x_1 \ldots x_m y_1 \ldots y_n$.
- To concatenate a string with itself many times we use the superscript notation $\overbrace{xx \cdots x}^{k \text{ times}} = x^k$.
- The **lexicographic ordering** of strings is the same as the familiar dictionary ordering, except that shorter strings precede longer strings.
- Example: The lexicographic ordering of all strings over the alphabet $\{0, 1\}$ is $(\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots)$.
- A **language** is a set of strings.

Boolean Values

- **Boolean logic** is built around the two values TRUE and FALSE.
- It was originally conceived of as a pure mathematical construct to model the laws of thought.
- Later, it became the foundation of digital electronics and computer design.
- The values TRUE and FALSE are called the **Boolean values** and are often represented by the values 1 and 0.
- We use Boolean values in situations with two possibilities, such as
  - a wire that may have a high or a low voltage;
  - a proposition that may be true or false;
  - a question that may be answered yes or no.
- We can manipulate Boolean values with specially designed operations, called the **Boolean operations**.

Boolean Operations

- The **negation** or **NOT** operation, designated with the symbol $\neg$. The negation of a Boolean value is the opposite value.
- The **conjunction**, or **AND**, operation is designated with the symbol $\wedge$. The conjunction of two Boolean values is 1 if both of those values are 1.
- The **disjunction**, or **OR**, operation is designated with the symbol $\vee$. The disjunction of two Boolean values is 1 if either of those values is 1.

P	$\neg P$
0	1
1	0

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

P	Q	$P \vee Q$
0	0	0
0	1	1
1	0	1
1	1	1

Operands and Functionality

- We use Boolean operations for combining simple statements into more complex Boolean expressions, just as we use the arithmetic operations $+$ and $\times$ to construct complex arithmetic expressions.
- Example: If P is the Boolean value representing the truth of the statement "the sun is shining" and Q represents the truth of the statement "today is Monday", we may write:
    - $P \wedge Q$ to represent the truth value of the statement "the sun is shining and today is Monday";
    - $P \vee Q$ to represent the truth value of the statement "the sun is shining or today is Monday"
- The values P and Q are called the **operands** of the operation.

Additional Boolean Operations

- Several other Boolean operations occasionally appear.
  - The **exclusive or**, or **XOR**, operation is designated by the $\oplus$ symbol. It is 1 if either but not both of its two operands are 1.
  - The **equivalence** or **equality operation**, written with the symbol $\leftrightarrow$. It is 1 if both of its operands have the same value.
  - The **implication operation** is designated by the symbol $\rightarrow$. It is 0 if its first operand is 1 and its second operand is 0; otherwise $\rightarrow$ is 1.

P	Q	$P \oplus Q$
0	0	0
0	1	1
1	0	1
1	1	0

P	Q	$P \leftrightarrow Q$
0	0	1
0	1	0
1	0	0
1	1	1

P	Q	$P \rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

Logical Equivalences

- We can express all Boolean operations in terms of the AND and NOT operations using some equivalences.
- Each row in the following table of equivalences expresses the operation in the left in terms of operations above it and AND and NOT:

$$\begin{aligned} P \vee Q &\equiv \neg(\neg P \wedge \neg Q) \\ P \rightarrow Q &\equiv \neg P \vee Q \\ P \leftrightarrow Q &\equiv (P \rightarrow Q) \wedge (Q \rightarrow P) \\ P \oplus Q &\equiv \neg(P \leftrightarrow Q) \end{aligned}$$

- The distributive law for AND and OR comes in handy in manipulating Boolean expressions

$$\begin{aligned} P \wedge (Q \vee R) &\equiv (P \wedge Q) \vee (P \wedge R) \\ P \vee (Q \wedge R) &\equiv (P \vee Q) \wedge (P \vee R). \end{aligned}$$

Subsection 3

Definitions, Theorems and Proofs

Definitions, Statements, Proofs, Theorems

- **Definitions** describe the objects and notions that we use. Precision is essential to any mathematical definition. When defining some object we must make clear what constitutes that object and what does not.
- After we have defined various objects and notions, we usually make **mathematical statements** about them. The statement may or may not be true, but like a definition, it must be precise.
- A **proof** is a convincing logical argument that a statement is true. In mathematics an argument must be convincing in an absolute sense. In everyday life or in the law, the standard of proof is lower.
- A murder trial demands proof "beyond any reasonable doubt." A mathematician demands proof beyond any doubt.
- A **theorem** is a mathematical statement proved true.
- A statement that assists in the proof of another, more significant statement is called a **lemma**.
- A statement whose truth follows easily from a theorem or its proof is called a **corollary**.

Some Strategies for Producing Proofs I

- Carefully read the statement to be proven.
- Break it down and consider each part separately.
  - A common multipart statement has the form "P if and only if Q", often written "P iff Q", where both P and Q are mathematical statements. This notation is shorthand for a two-part statement.
    - The first part is "P only if Q", which means: If P is true, then Q is true, written $P \Rightarrow Q$.
    - The second is "P if Q", which means: If Q is true, then P is true, written $P \Leftarrow Q$.

    The first of these parts is the forward direction of the original statement and the second is the reverse direction. We write "P if and only if Q" as $P \Leftrightarrow Q$. To prove a statement of this form we must prove each of the two directions.
  - Another type of multipart statement states that sets A and B are equal.
    - The first part states that A is a subset of B;
    - the second part states that B is a subset of A.

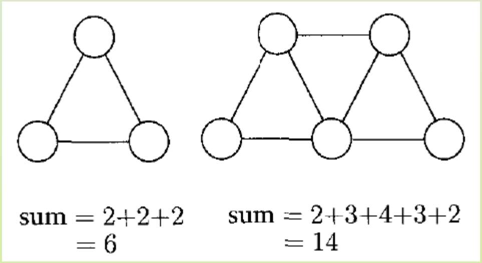
    One way to prove that $A = B$ is to prove that every member of A is also a member of B and that every member of B is also a member of A.

Some Strategies for Producing Proofs II

- Try to get an intuitive feeling of why a statement should be true. Experimenting with examples is especially helpful.
  - If the statement says that all objects of a certain type have a particular property, pick a few objects of that type and observe that they actually do have that property.
  - After doing so, try to find an object that fails to have the property, called a **counterexample**.
    - If the statement is actually true, we will not be able to find a counterexample.
    - Seeing where we run into difficulty when attempting to find a counterexample can help in understanding why the statement is true.

Comprehending a Mathematical Statement

- Suppose we want to prove that, for every graph G, the sum of the degrees of all the nodes in G is an even number.
- First, pick a few graphs and observe this statement in action.



- Next, try to find a counterexample, that is, a graph in which the sum is an odd number.
- Can you see why the statement is true and how to prove it?

Looking at a Special Case of a Statement

- If we get stuck trying to prove a statement, we may try something easier.
- Attempt to prove a special case of the statement.

  For example, if we are trying to prove that some property is true for every $k > 0$, we first try to prove it for $k = 1$. If we succeed, we try for $k = 2$, and so on until we can understand the more general case.
- If a special case is hard to prove, we try a different special case or perhaps a special case of the special case.
- Finally, when we believe that we have found the proof, we **must write it up properly**. A well-written proof is a sequence of statements, wherein each one follows by simple reasoning from previous statements in the sequence.
- Carefully writing a proof is important, both to enable a reader to understand it and to make sure that it is free from errors.

Tips for Producing a Proof

- Be patient: Finding proofs takes time. If we do not see how to do it right away, we should not worry.
- Come back to it: Look over the statement to be proved, think about it a bit, leave it, and then return a few minutes or hours later. Give the unconscious, intuitive part of the mind a chance to work.
- Be neat: When building an intuition for the statement we are trying to prove, we should use simple, clear pictures and/or text. In trying to develop an insight into the statement, sloppiness gets in the way of insight. Furthermore, when we are writing a solution for another person to read, neatness will help that person understand it.
- Be concise: Brevity helps express high-level ideas without getting lost in details. Good mathematical notation is useful for expressing ideas concisely. But we must include enough of the reasoning when writing up a proof so that the reader can easily understand the argument.

A Proof of DeMorgan's Law

Theorem (DeMorgan's Law)

For any two sets A and B,
$$\overline{A \cup B} = \overline{A} \cap \overline{B}.$$

- We must show that the two sets $\overline{A \cup B}$ and $\overline{A} \cap \overline{B}$ are equal. We may prove that two sets are equal by showing that every member of one set is also a member of the other and vice versa.
  - Suppose that x is an element of $\overline{A \cup B}$. Then x is not in $A \cup B$ from the definition of the complement of a set. Therefore x is not in A and x is not in B, from the definition of the union of two sets. In other words, x is in $\overline{A}$ and x is in $\overline{B}$. Hence, by the definition of the intersection of two sets, x is in $\overline{A} \cap \overline{B}$.
  - For the other direction, suppose that x is in $\overline{A} \cap \overline{B}$. Then x is in both $\overline{A}$ and $\overline{B}$. Therefore x is not in A and x is not in B, and, thus, not in the union of these two sets. Hence x is in the complement of the union of these sets; in other words, x is in $\overline{A \cup B}$.

The Sum of Degrees of Vertices

Theorem

For every graph G, the sum of the degrees of all the nodes in G is an even number.

- Every edge in G is connected to two nodes. Each edge contributes 1 to the degree of each node to which it is connected. Therefore, each edge contributes 2 to the sum of the degrees of all the nodes. Hence, if G contains e edges, then the sum of the degrees of all the nodes of G is $2e$, which is an even number.

Subsection 4

Types of Proof

Proof by Construction

- Many theorems state that a particular type of object exists.
- One way to prove such a theorem is by demonstrating how to construct the object.
- This technique is a **proof by construction**.

Definition (k-Regular Graph)

A graph is called k-**regular** if every node in the graph has degree k.

Theorem

For each even number $n > 2$, there exists a 3-regular graph with n nodes.

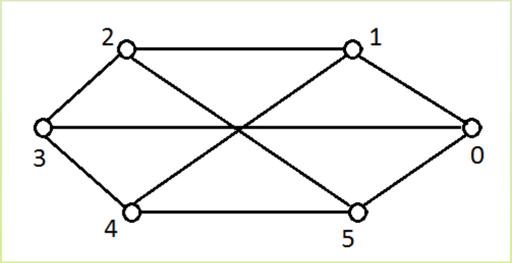
- Let n be an even number greater than 2. Construct graph $G = (V, E)$ with n nodes as follows:
  - The set of nodes of G is $V = \{0, 1, \ldots, n-1\}$;
  - The set of edges of G is the set

$$\begin{aligned} E &= \{\{i, i+1\} : 0 < i < n-2\} \cup \{\{n-1, 0\}\} \\ &\quad \cup \{\{i, i+\tfrac{n}{2}\} : 0 < i < \tfrac{n}{2} - 1\}. \end{aligned}$$

Illustration of the Proof

- Picture the nodes of this graph written consecutively around the circumference of a circle.
  - The edges described in the top line of E go between adjacent pairs around the circle.
  - The edges described in the bottom line of E go between nodes on opposite sides of the circle.

This mental picture clearly shows that every node in G has degree 3.



Proof by Contradiction

- In one common form of argument for proving a theorem, we assume that the theorem is false and then show that this assumption leads to an obviously false consequence, called a **contradiction**.
- Example: Jack sees Jill, who has just come in from outdoors. On observing that she is completely dry, he knows that it is not raining. His "proof" that it is not raining is:
  - If it were raining (the assumption that the statement is false),
  - Jill would be wet (the obviously false consequence).
  - Therefore it must not be raining.

Definition (Rational and Irrational Numbers)

A number is **rational** if it is a fraction $\frac{m}{n}$, where m and n are integers, with $n \neq 0$, i.e., a rational number is the ratio of integers m and n. A number is **irrational** if it is not rational.

Example of Proof by Contradiction

Theorem

$\sqrt{2}$ is irrational.

- Assume that $\sqrt{2}$ is rational. Thus $\sqrt{2} = \frac{m}{n}$, where both m and n are integers. If both m and n are divisible by the same integer greater than 1, divide both by that integer. Doing so does not change the value of the fraction. Now, at least one of m and n must be an odd number. We multiply both sides of the equation by n and obtain $n\sqrt{2} = m$. We square both sides and obtain $2n^2 = m^2$. Because m^2 is 2 times the integer n^2, we know that m^2 is even. Therefore m too, is even, as the square of an odd number always is odd. So we can write $m = 2k$ for some integer k. Then, substituting $2k$ for m, we get $2n^2 = (2k)^2 = 4k^2$. Dividing both sides by 2 we obtain $n^2 = 2k^2$. But this result shows that n^2 is even and, hence, that n is even. Thus we have established that both m and n are even. We had earlier reduced m and n so that they were not both even, a contradiction.

Proof by Induction

- **Proof by induction** is an advanced method used to show that all elements of an infinite set have a specified property.
- Take the infinite set to be the natural numbers $\mathbb{N} = \{1, 2, 3, \ldots\}$, and say that the property is called $\mathcal{P}$. Our goal is to prove that $\mathcal{P}(k)$ is true for each natural number k, i.e., we want to prove that $\mathcal{P}(1)$ is true, as well as $\mathcal{P}(2), \mathcal{P}(3), \mathcal{P}(4)$, and so on.
- A proof by induction consists of two parts, the basis and the induction step:
  - The basis proves that $\mathcal{P}(1)$ is true.
  - The induction step proves that for each $i \geq 1$, if $\mathcal{P}(i)$ is true, then so is $\mathcal{P}(i + 1)$.
- There are variations and generalizations of the same idea:
  - E.g., the basis does not necessarily need to start with 1; it may start with any value b. In that case the induction proof shows that $\mathcal{P}(k)$ is true for every k that is at least b.

Writing Induction Proofs

- In the induction step the assumption that $\mathcal{P}(i)$ is true is called the **induction hypothesis**.
- Sometimes having the stronger induction hypothesis that $\mathcal{P}(j)$ is true for every $j < i$ is useful.
- The format for writing down a proof by induction is as follows:
  - Basis: Prove that $\mathcal{P}(1)$ is true.
  - Induction step: For each $i \geq 1$:
    - Assume that $\mathcal{P}(i)$ is true;
    - Use this assumption to show that $\mathcal{P}(i+1)$ is true.

Monthly Payments of Home Mortgages I

- Set up the names and meanings of several variables:
  - Let P be the principal, the amount of the original loan.
  - Let $I > 0$ be the yearly interest rate of the loan, where $I = 0.06$ indicates a 6% rate of interest.
  - Let Y be the monthly payment.
  - Define another variable M from I, for the monthly multiplier. It is the rate at which the loan changes each month due to the interest on it. Assuming monthly compounding, we get $M = 1 + \frac{I}{12}$.
- Two things happen each month.
  - First, the amount of the loan tends to increase because of the monthly multiplier.
  - Second, the amount tends to decrease because of the monthly payment.
- Let P_t be the outstanding amount of the loan after the t-th month.
  - Then $P_0 = P$ is the amount of the original loan;
  - $P_1 = MP_0 - Y$ is the amount of the loan after one month;
  - $P_2 = MP_1 - Y$ is the amount of the loan after two months, and so on.

Monthly Payments of Home Mortgages II

Theorem

For each $t \geq 0$,
$$P_t = PM^t - Y\left(\frac{M^t - 1}{M - 1}\right).$$

- Basis: The formula is true for $t = 0$. If $t = 0$, then the formula states that $P_0 = PM^0 - Y\left(\frac{M^0-1}{M-1}\right)$. We can simplify the right-hand side to get $P_0 = P$, which holds because we have defined P_0 to be P.
- Induction step: For each $k \geq 0$ assume that the formula is true for $t = k$ and show that it is true for $t = k+1$. Induction hypothesis: $P_k = PM^k - Y\left(\frac{M^k-1}{M-1}\right)$. We need: $P_{k+1} = PM^{k+1} - Y\left(\frac{M^{k+1}-1}{M-1}\right)$.
  - From the definition of P_{k+1}, from P_k, we know that $P_{k+1} = P_k M - Y$;
  - Therefore, using the induction hypothesis to calculate P_k, $P_{k+1} = \left[PM^k - Y\left(\frac{M^k-1}{M-1}\right)\right]M - Y$.
  - Multiplying through by M and rewriting Y yields $P_{k+1} = PM^{k+1} - Y\frac{M^{k+1}-M}{M-1} - Y\frac{M-1}{M-1} = PM^{k+1} - Y\left(\frac{M^{k+1}-1}{M-1}\right)$.