Mathematical Prerequisites

This is a summary of basic mathematical notation and concepts which you need to be familiar with in order to take this course. Most of the material here should be familiar either from high school mathematics or first-year discrete math, and it will not be covered in lectures. A good reference for most of these topics is Rosen's Discrete Mathematics and Its Applications (or any other textbook that you used for EECS 1019).

1 Logic

Propositions and Predicates

A *proposition* is a statement that is either true or false. For example, "$3 + 3 = 7$" and "Joseph T. Smith felt ill on December 23, 2008" are propositions. "How are you?" is not a proposition because it is a question, not a statement. "$3 + 3$" is not a proposition because it lacks a verb and is therefore not a statement.

A *predicate* is like a proposition, except some of the nouns in the statement have been replaced by variables. For example "$x = y + 1$" is a predicate with two variables, x and y. Let's use $P(x, y)$ to denote the predicate "$x = y + 1$". Whether $P(x, y)$ is true or false depends on the values filled in for x and y. For example $P(3, 2)$ stands for the statement "$3 = 2 + 1$", which is true. $P(2, 3)$ stands for the statement "$2 = 3 + 1$", which is false.

Note that we can have predicates with any number of variables. A proposition can be thought of as a predicate with 0 variables.

Often, we assume that the variables in a predicate stand for some particular type of object. For example, in the predicate "$x = y + 1$", we might assume that x and y stand for integers. To be clear, it is usually best to state what type of object each variable represents. Thus, we might say: If x and y are integers, let $P(x, y)$ represent the statement "$x = y + 1$". In this case, the set of all integers is called the *domain* of variables x and y.

Boolean Operators

Propositions and predicates can be joined together to make more complex statements using the connectives $\wedge$ (and), $\vee$ (or), $\Rightarrow$ (implies) and $\Leftrightarrow$ (if and only if). The following tables specify whether the complex statements are true or false, depending on the truth of their component parts.

A	B	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

Note that $A \vee B$ is the "inclusive or": $A \vee B$ is true in the case where both A and B are true.

The statement $A \Rightarrow B$ is read "A implies B" or "if A then B".

The Boolean operator $\neg$ (not) is used to negate the value of a statement.

A	$\neg A$
T	F
F	T

We can build up more and more complicated statements using connectives repeatedly. The following statements give some examples.

- "Neither Bob nor Alice went to the store" has the logical form $\neg(B \vee A)$ (or, equivalently, $\neg B \wedge \neg A$) where B represents the statement "Bob went to the store" and A represents the statement "Alice went to the store".

- "Bob and Alice did not both go to the store" can be represented as $\neg(B \wedge A)$, using the same meaning for A and B as in the previous example.

- "Bob did not go to the store but Alice did" can be represented as $\neg B \wedge A$, using the same meaning for A and B as in the first example. Notice that in English, the word "but" used as a conjunction is logically equivalent to "and". The statement $\neg B \wedge A$ is equivalent to $(\neg B) \wedge A$ because of the precedence of Boolean operators, and it means something quite different from $\neg(B \wedge A)$ of the previous example.

- "If 7 provincial legislatures representing at least 50% of Canada's population approve a constitutional amendment A and the House of Commons also approves A and the Senate also approves A then the amendment A will become law" has the logical form $(W \wedge X \wedge Y) \Rightarrow Z$.

Quantifiers

If $S(x)$ is a predicate, the statement $\exists x\ S(x)$ means there exists at least one x such that $S(x)$ is true. If $S(x)$ is a predicate, the statement $\forall x\ S(x)$ means for all x, $S(x)$ is true. Generally, the domain of x should be specified in order for these statements to make sense.

For example, let x represent an integer. Then $\exists x\ x > 2$ is true because some integer x (namely, 536) is greater than 2. But $\forall x\ x > 2$ is false since not every integer is greater than 2 (in particular, -7 is not greater than 2).

Quantifiers can be nested and combined with Boolean operators, and it is important for you to be able to interpret the meaning of statements formed in this way and to be able to write them down yourself. Examples:

- "Every dog is a mammal" can be written "$\forall x$ (x is a dog $\Rightarrow$ x is a mammal)". The domain of x here is the set of all animals.

- "Some dog is a mammal" can be written "$\exists x$ (x is a dog $\wedge$ x is a mammal)". The domain of x is again the set of all animals.

- Suppose x and y represent positive integers. The statements "$\forall x\ \exists y\ y > x$" and "$\exists y\ \forall x\ y > x$" are very different. The first is true and the second is false. Thus, the order of quantifiers is important. The first statement says that for each positive integer x, there is a larger positive integer y. In this statement each x can have a different y. The statement is true, because we can take $y = x + 1$. The second statement says there is a *single* positive integer y that is greater than *every* positive integer x. This statement is false: no matter which positive integer y you choose, it will not be greater than every x (in particular, it will not be greater than x when $x = y + 1$).

- "An integer x is even if and only if it is twice as big as some integer" can be written "x is even $\Leftrightarrow \exists y\ x = 2y$". (The domain of x and y is the set of all integers.)

- "A positive integer x is prime if and only if the only positive integers of which x is a multiple are 1 and x" can be written "x is prime $\Leftrightarrow \forall y((\exists z\ x = yz) \Rightarrow (y = 1 \vee y = x))$". (Here, the domain of x, y and z is the set of positive integers.) The way to come up with the logical form of a complex statement like this is to first write down the logical form of pieces of the statement and then combine the pieces: for example, to say x is a multiple of y, we can write "$\exists z\ x = yz$". Similarly, if you see a complicated formula, you should first understand what each piece means and then see how those pieces are connected to figure out the meaning of the whole statement.

2 Set Theory

The Basics

We talk about sets *a lot* in this course, so it is important to be comfortable working with sets.

Intuitively, a set is a collection of objects. (The objects are called the *members* or *elements* of the set.) This is not a proper definition of "set" because "collection" is just a different word for "set", but the notion of a set is so basic that it is difficult to give a proper definition of what it means. The sets we deal with in this course will be fairly simple (as far as sets go), so this intuitive description of what a set is will be sufficient for our purposes.

If a set is finite, we can specify it by listing its elements explicitly. For example, the set of the smallest five positive integers can be written down as $\{1, 2, 3, 4, 5\}$.

Some important infinite sets are:

- $\mathbb{N}$: the set of all natural numbers, $\{0, 1, 2, \dots\}$
- $\mathbb{Z}$: the set of all integers, $\{\dots, -2, -1, 0, 1, 2, \dots\}$
- $\mathbb{Z}^+$: the set of all positive integers, $\{1, 2, 3, \dots\}$
- $\mathbb{Q}$: the set of all rational numbers
- $\mathbb{R}$: the set of all real numbers

Both finite and infinite sets can be specified using set-builder notation. Suppose $S(x)$ is a predicate with one variable, x. (Recall that, for each possible value we can assign to x, $S(x)$ becomes a statement that is either true or false.) Then, the notation $\{x : S(x)\}$ denotes the set of all elements x for which the statement $S(x)$ is true. In this notation, x is a dummy variable. This means that $\{y : S(y)\}$ is exactly the same set as $\{x : S(x)\}$.

Examples:

- $\{x : x \text{ is one of the five smallest positive integers}\} = \{1, 2, 3, 4, 5\}$
- $\{x : x \text{ is an integer and } x^2 = 4\} = \{-2, 2\}$
- $\{x : x \text{ is an integer and } x^2 > 4\}$ is an infinite set containing all of the integers, except $-2, -1, 0, 1$ and 2.

If x is an element of set A, we write $x \in A$.

When using set-builder notation, we often give the domain (*i.e.*, the set of possible values the variable can take) before the colon. For example,

- $\{x \in \mathbb{N} : x^2 = 4\} = \{2\}$ and

- $\{x \in \mathbb{Z} : x^2 = 4\} = \{-2, 2\}$.

Two sets A and B are considered equal if every element of A is in B and every element of B is in A. In particular, this means that the ordering of elements in the set is not important. For example, $\{1, 2, 3, 4, 5\} = \{4, 3, 2, 5, 1\}$. Also repetitions of an element in a set are not important. For example, $\{1, 1, 2, 2, 3, 3, 4, 4, 5, 5\} = \{1, 2, 3, 4, 5\}$.

Notation

For any sets A and B, we will use the following standard notation.

- $\varnothing$ or $\{\}$: "the empty set"; *i.e.*, the set that contains no elements.
- $x \in A$: "x is an element of A".
- $x \notin A$: "x is not an element of A".
- $A \subseteq B$: "A is a subset of B"; *i.e.*, every element of A is also in B.
- $A = B$: "A equals B"; *i.e.*, $A \subseteq B$ and $B \subseteq A$.
- $A \cap B$: "A intersect B"; *i.e.*, the set $\{x : x \in A \text{ and } x \in B\}$.
- $A \cup B$: "A union B"; *i.e.*, the set $\{x : x \in A \text{ or } x \in B\}$.
- $A \setminus B$ or $A - B$: "A minus B" (*set* difference); *i.e.*, the set $\{x : x \in A \text{ and } x \notin B\}$.
- $\overline{A}$: "the complement of A". This assumes there is some domain D from which the items in A are chosen; then $\overline{A} = D - A$.
- $|A|$: "cardinality of A" (intuitively, the number of elements in A; we shall talk about what this means for infinite sets later in the course.)
- $\mathscr{P}(A)$: "the power set of A"; *i.e.*, the set of all subsets of A; $\mathscr{P}(A) = \{B : B \subseteq A\}$.

Cartesian Products

An ordered pair is a sequence of two objects in a specified order. We write an ordered pair consisting of the objects a and b (in that order) as (a, b). Two ordered pairs (a, b) and (a', b') are equal if and only if $a = a'$ and $b = b'$.

If A and B are sets, the set of all ordered pairs whose first element is an element of x and whose second element is an element of B is called the Cartesian product of A and B, written $A \times B$. Thus, $A \times B = \{(a, b) : a \in A \text{ and } b \in B\}$.

More generally, if k is a positive integer, a k-tuple is a sequence of k objects in a specified order. (Thus, a 2-tuple is just an ordered pair.) We use similar notation for k-tuples as for ordered pairs. For example, $(3, 2, 1, 7)$ is an ordered 4-tuple. Two k-tuples $(a_1, a_2, \ldots, a_k)$ and $(a'_1, a'_2, \ldots, a'_k)$ are equal if and only if for every i in $\{1, 2, \ldots, k\}$, $a_i = a'_i$. The Cartesian product of k sets is a set of k-tuples: $A_1 \times A_2 \times \cdots \times A_k = \{(a_1, a_2, \ldots, a_k) : a_i \in A_i \text{ for } 1 \leq i \leq k\}$.

3 Functions

Formally, a function $f : A \to B$ is a set $f \subseteq A \times B$, where, for each $x \in A$, there is exactly one element $y \in B$ such that $(x, y) \in f$. If $(x, y) \in f$, we use the notation $f(x)$ to denote y. This means that f associates with each value $x \in A$ a unique value $f(x)$. A is called the *domain* of the function f. Note that the domain can be a Cartesian product of other sets: in this case we use the notation $f(x, y)$ instead of $f((x, y))$.

Properties of Functions

A function $f : A \to B$ is called

- *surjective* (or *onto*) if, for every $z \in B$, there is some $x \in A$ such that $f(x) = z$.
- *injective* (or *one-to-one*) if $x \neq y$ implies $f(x) \neq f(y)$.
- *bijective* (or *a one-to-one correspondence*) if it is surjective and injective.

4 Binary Notation

A *binary number* is a sequence of bits $a_k \cdots a_1 a_0$ where each bit a_i is either 0 or 1. Every binary number represents a natural number in the following way:

$$a_k \cdots a_1 a_0 \text{ represents } \sum_{i=0}^{k} a_i 2^i = a_k 2^k + \cdots + a_1 2 + a_0.$$

For example, 1001 represents $1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 1 = 9$, and 01110 represents $8 + 4 + 2 = 14$. Each positive integer has a unique binary representation whose leftmost bit is 1.

5 Proofs

The main objective of EECS 2001 is to learn how to think about computing in a formal way, so that mathematics can be applied to computing. The most important aspect of this is proving statements about mathematical models of computing. This means that an important component of the course is devoted to understanding and writing mathematical proofs. It is okay if you are not an expert at writing proofs now; that is a skill that you will develop during this course. However, you should be familiar with basic proof techniques from EECS 1019.

If you do not feel too comfortable about this part of your background for the course, Daniel Solow's *How to Read and Do Proofs* (available from the library) is a very good introduction.

You should be familiar with the following proof techniques.

- Proof by cases: this just means that to prove a statement A is true under all possible circumstances, you may have to use different proofs for different circumstances. For example, to prove that for all real numbers x and y, $|x + y| \leq |x| + |y|$, you might want to consider several cases (for example, you could consider three cases: x and y are both positive, or both negative or one is positive and one is negative). When using this proof technique, you must be sure that you have covered all possible cases.
- Proof by contradiction: to prove A, derive a contradiction from $\neg A$.

- Direct proof of an implication: to prove $A \Rightarrow B$, assume A as a hypothesis and prove B.
- Proof of an implication by proving contrapositive: to prove $A \Rightarrow B$, prove $\neg B \Rightarrow \neg A$.
- Proof by mathematical induction. To prove a statement of the form $\forall n \in \mathbb{N}\ P(n)$, it is sufficient to prove

  1. $P(0)$, and
  2. $\forall n \geq 1, (P(n-1) \Rightarrow P(n))$.

- Proof by strong mathematical induction. To prove a statement of the form $\forall n \in \mathbb{N}\ P(n)$, it is sufficient to prove

  1. $P(0)$, and
  2. $\forall n \geq 1, ((\forall x < n\ P(x)) \Rightarrow P(n))$.

- Proof of existence by construction: to prove $\exists x\ P(x)$, explicitly construct an x and show that $P(x)$ is true for it.
- Proof of universally quantified statement by generalization: to prove $\forall x\ P(x)$, start with a generic element x of the domain and show that $P(x)$ holds. Note that your proof must apply to every x; it cannot use any properties of particular x's.

You should be able to combine proof techniques above, for example to prove "if and only if" statements or statements with nested quantifiers.