CSC165 Pre-Class Notes

Avery Ng

2024.12.23

1 Mathematical Expression

1.1 Sets

Definition 1.1. Sets are a collection of distinct **elements** (finite or infinite).

The size of a finite set A is denoted by $|A|$.

The **empty set** is denoted by $\emptyset$.

Cardinality of a set is how many elements are in the set.[1]

[1] It is important to note that the hierarchy of cardinalities is $\#\mathbb{N} = \#\mathbb{Z} = \#\mathbb{Q} < \#\mathbb{R} = \#\mathbb{C}$.

Examples

Finite sets $\{a, b, c, d\}$, $\{2, 4, 10, 11\}$

Set of tuples $\{(AvaDoe, \$700, 50), (Donald, \$670, 30\}$

Infinite sets

$\mathbb{R}$ = real numbers

$\mathbb{Q}$ = rational numbers

$\mathbb{Z}$ = integers : $\{\ldots, 2, -1, 0, 1, 2, \ldots\}$

$\mathbb{Z}^+$ = positive integers : $\{1, 2, \ldots\}$

$\mathbb{N}$ = all natural numbers : $\{0, 1, 2, \ldots\} = \mathbb{Z}^+ \cap \emptyset$

A string of length 0 is called the *empty string* and is denoted by ϵ.

A set of all natural numbers greater or equal to five can be denoted by:

$$\{x \mid x \in \mathbb{N} \text{ and } x \geq 5\}$$

(left side: description, right side: condition)

The set of all rational numbers can be denoted by [2]

$$\mathbb{Q} = \left\{ \frac{p}{q} \;\middle|\; p, q \in \mathbb{Z} \; and \; q \neq 0 \right\}$$

[2] Some definitions on binary operators and relations:

$=$	equals
$\in$	in
$\notin$	not in
$\subseteq$	subset
$\cap$	intersection
$\cup$	union
$\setminus$	set difference
$\times$	Cartesian product
$\mathcal{P}(\mathbb{A})$	power set

Operations on sets

The size of a set, $\mid A \mid$, is an example of a set operation. There are many other common set operations.[3]

Examples

Returns booleans

element of A: $x \in A$, $y \notin A$

subset of: $A \subseteq B$

contains same elements: $A = B$

Returns sets

union: $A \cap B$, intersection: $A \cup B$

difference of A and B: $A \setminus B$ $\{x \mid x \in A \text{ and } x \notin B\}$

Cartesian product: $A \times B$, power set of A: $\mathcal{P}(\mathbb{A})$

[3] Cartesian products are all *pairs* (a, b) where a and b are elements of their respective sets.

Power sets are sets containing *all* subsets of A. If $A = 1, 2, 3$, then

$$\mathcal{P}(\mathbb{A}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

1.2 Functions

Definition 1.2. Let A and B be sets. A **function** $f : A \to B$ is a mapping from elements in A to elements in B.

A is the **domain** of the function.

B is the **codomain** of the function.

Range of f is the set containing all possible outputs $\{f(x) | x \in A\}$

Example

Predecessor function $Pred : \mathbb{Z} \to \mathbb{Z}$

Given $Pred(x) = x - 1$: $Pred(A) = \{\ldots, (-2, -3), (-1, -2), (0, -1), (1, 0), (2, 1), \ldots\}$

Arity function: k-ary function[4] $f : A_1 \times A_2 \times \ldots \times A_k \to B$

Addition operator (binary) takes 2 real numbers and return sum $+ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

Predicate function (codomain is True and False) $f : \mathbb{N} \to \{\text{True, False}\}$

Given *any* predicate $P : A \to \{T, F\}$, construct a *specific* subset $S_P \subseteq A$ as follows:

$$S_P = \{x \in A \mid P(x) = T\}.$$

Given *any* subset $S \subseteq A$, construct a *specific* predicate $P_S : A \to \{T, F\}$ as follows:

$$P_S(x) = \begin{cases} T, & \text{if } x \in S, \\ F, & \text{if } x \notin S. \end{cases}$$

Predicates and sets are closely related.

Set	Predicate
$\{x \mid x \in A \text{ and } P(x) = True\}$.	$P : A \to \{\text{True, False}\}$
$B \subseteq A$	$P : A \to \{\text{True, False}\}$ by $P(x) = \text{True}$ if $x \in A$
$\{0, 2, 4, \ldots\}$	$Even : \mathbb{N} \to \{\text{True, False}\}$

[4] k-ary refers to terms like unary, binary, ternary functions that take one, two, and three inputs respectively.

1.3 Summation and product notation

Definition 1.3. The **summation notation** is used to express sums of terms where each term follows a pattern.

$$\sum_{i=1}^{100} \frac{i + i^2}{3 + i}$$

i is the *index of summation*, 1 and 100 are the *lower* and *upper bounds* of the summation.

$$\sum_{i=j}^{k} f(i) = f(j) + f(j+1) + f(j+2) + \cdots + f(k).$$

Definition 1.4. The **product notation** is similar to the summation notation, but is used to abbreviate multiplication instead.

$$\prod_{i=j}^{100} f(i) = f(j) \times f(j+1) \times \ldots \times f(k)$$

The lower bound can be greater than its upper bound, in which case it is an *empty* summation or product[5]

- Summations have a sum of 0: When $j > k$, $\sum_{i=j}^{k} f(i) = 0$.
- Products have a product of 1: When $j > k$, $\prod_{i=j}^{k} f(i) = 1$.

[5] These values are chosen so that the overall value of the expression is not changed when adding an empty summation or multiplying by an empty product.

Theorem 1.1

For all $n \in \mathbb{N}$, the following formulas hold:

1. For all $c \in \mathbb{R}$, $\sum_{i=1}^{n} c = c \cdot n$ (sum with constant terms).
2. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ (sum of consecutive numbers).
3. $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ (sum of consecutive squares).
4. For all $r \in \mathbb{R}$, if $r \neq 1$, then $\sum_{i=0}^{n-1} r^i = \frac{r^n-1}{r-1}$ (sum of powers).
5. For all $r \in \mathbb{R}$, if $r \neq 1$, then $\sum_{i=0}^{n-1} i \cdot r^i = \frac{nr^n}{r-1} - \frac{r(r^n-1)}{(r-1)^2}$ (arithmetico-geometric series).

Theorem 1.2

$$\sum_{i=m}^{n}(a_i + b_i) = \left(\sum_{i=m}^{n} a_i\right) + \left(\sum_{i=m}^{n} b_i\right) \quad \text{(separating sums)}$$

$$\prod_{i=m}^{n}(a_i \cdot b_i) = \left(\prod_{i=m}^{n} a_i\right) \cdot \left(\prod_{i=m}^{n} b_i\right) \quad \text{(separating products)}$$

$$\sum_{i=m}^{n} c \cdot a_i = c \cdot \left(\sum_{i=m}^{n} a_i\right) \quad \text{(factoring out constants, sums)}$$

$$\prod_{i=m}^{n} c \cdot a_i = c^{n-m+1} \cdot \left(\prod_{i=m}^{n} a_i\right) \quad \text{(factoring out constants, products)}$$

$$\sum_{i=m}^{n} a_i = \sum_{i'=0}^{n-m} a_{i'+m} \quad \text{(change of index } i' = i - m\text{)}$$

$$\prod_{i=m}^{n} a_i = \prod_{i'=0}^{n-m} a_{i'+m} \quad \text{(change of index } i' = i - m\text{)}$$

1.4 Inequalities

Theorem 1.3

For all real numbers a, b, and c, the following are true:[6]

(a) If $a \leq b$ and $b \leq c$, then $a \leq c$.

(b) If $a \leq b$, then $a + c \leq b + c$.

(c) If $a \leq b$ and $c > 0$, then $ac \leq bc$.

(d) If $a \leq b$ and $c < 0$, then $ac \geq bc$.

(e) If $0 < a \leq b$, then $\frac{1}{a} \geq \frac{1}{b}$.

(f) If $a \leq b < 0$, then $\frac{1}{a} \geq \frac{1}{b}$.

[6] For all non-negative real numbers a and b, and all strictly increasing functions $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$, if $a \leq b$, then $f(a) \leq f(b)$. This also follows the strict equality substitution: if $a < b$, then $f(a) < f(b)$.

If any of the above equalities are replaced with a strict equality, then the corresponding "then" equality will also be strict.[7]

[7] A strict equality is expressed by $>$ or $<$.

The implications of the inequalities is that adding or multipying by positive numbers preserves inequalities, while multiplying by negative numbers or taking reciprocals reverses inequalities.

1.5 Propositional logic

Definition 1.5. A **proposition** is a statement that is either True or False.

Examples

- $2 + 1 = 3$
- $4 + 3 < 8$
- Every even number greater than 2 is the sum of two prime numbers
- Python's `list.sort` is correct on every input list

Propositional variables are used to represent propositions. The variable names start at p by convention.

propositional/logical operator is a predicate whose arguments must be either True or False.

propositional formula is an expression built up from propositional variables by applying the following operators.

Basic operators

Negation	**NOT**	$\neg$	Unary negation operator that flips the truth value.
Conjunction	**AND**	$\wedge$	Binary conjunction operator that returns True when both arguments are True.
Disjunction	**OR**	$\vee$	Binary disjunction operator that returns True when at least one argument is True.

These are some common operators on propositional variables.[8]

[8] There are two different *or's*, the *exclusive* and the *inclusive*. Exclusive is having one but not the other, inclusive is having either or both. Usually, the inclusive *or* is used. The exclusive *or* is denoted by $\oplus$.

Implication operator

Definition 1.6. The **implication** $p \implies q$ asserts that whenever p is True, q must be True.

The statement p is called the **hypothesis** of the implication.

The statement q is called the **conclusion** of the implication.

The **converse** of an implication switches the hypothesis and conclusion. The converse of $p \implies q$ is $q \implies p$. These two equations are not logically equivalent.

p	q	$p \Rightarrow q$
False	False	True
False	True	True
True	False	False
True	True	True

The two cases where $p \implies q$ is True even though p is False are called **vacuous truth** cases. These are True because the statement does not say anything about the behaviour of q (conclusion) when p (hypothesis) is False. If the premise p doesn't occur, the "promise" doesn't need to be upheld. The conclusion q becomes irrelevant, so the statement is considered true. When p is False, $p \implies q$ is always true because the condition under which it would be False (True premise with a False conclusion) never arises.

The following two formulas are equivalent to $p \implies q$:

$\neg p \vee q$ This uses the vacuous truth cases.
$\neg q \Rightarrow \neg p$ This is the **contrapositive** case.

Example[9]
If $p \implies q$ is "If it rains, then the ground will be wet."
$\neg p \vee q$ would be "It is not raining or the ground is wet."
$\neg q \implies \neg p$ would be "It is not raining, then the ground is not wet."

Biconditional operator

Definition 1.7. The **biconditional** operator denoted by $p \iff q$ returns True when both $p \implies q$ and $q \implies p$ are True. In other words, $p \iff q$ abbreviates $(p \Rightarrow q) \wedge (q \Rightarrow p)$.

p	q	$p \iff q$
False	False	True
False	True	False
True	False	False
True	True	True

This condition can be phrased any of the below:

"If p then q, and if q then p."
"p if and only if q."
"p if q."

Definition 1.8. A **tautology** is a formula that is always True for every possible assignment of values to its propositional variables. E.g. $(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$, $(\neg(p \vee q)) \Leftrightarrow (\neg p \wedge \neg q)$.

1.6 Predicate logic

Predicates, we know they have a codomain of True or False. We can extend this definition to be "A statement whose truth depends on one or more variables from any set."[9]

When we substitute values into a predicate, we obtain a proposition: $P(x, y)$ is the statement $x^2 = y$ $P(5, 25)$ is True, $P(5, 24)$ is False.

We can complete the definiton of the above statements by giving the domain of the prediate. In fact, **it is not a predicate if we do not give the domain.**

$P(x)$: "x is a power of 2," where x$\in \mathbb{N}$

[9] Examples of predicates: the operators = and < both return True or False based on operands

Quantification of variables

Truth aggregation is when we want to express a predicate's truth values for all elements of its domain, like the inequality $x^2 - 2x + 1 > 0$.

There are two ways we can express this truth aggregation using quantifiers, which modify predicates by specifying how a variable should be interpreted.

Definition 1.9. The **existential quantifier**, $\exists$, abbreviates "there exists an element in the domain that satisfies the given predicate."

Examples[10]

$\exists x \in \mathbb{N}$, $x \geq 0$	There exists a natural number x that is greater than or equal to zero.
$\exists y \in \mathbb{N}$, $y = 2^a$	There exists a natural number y that is a power of 2.

[10] These examples are stating that there must be *at least* one of the variable that satisfies the condition — a continuous OR operation.

Definition 1.10. The **universal quantifier**, $\forall$, abbreviates "*every* element in the domain satisfies the given predicate."

Examples[11]

$\forall x \in \mathbb{N}$, $x \geq 0$	Every natural number is x that is greater than or equal to zero.
$\forall y \in \mathbb{N}$, $y = 2^a$	Every natural number is y that is a power of 2.

[11] In contrast to the above, this asks for *all* elements to meet the condition, like a continuous AND operation.

Understanding multiple quantifiers

It is important to note that the ordering of quantifiers *do* matter in some cases.

For commutative operators like addition and multiplication, order does not matter. The universal operator is commutative if used consecutively:

$$\forall x \in S_1, \forall y \in S_2, P(x, y)$$

$$\forall y \in S_2, \forall x \in S_1, P(x, y)$$

These formulas are equivalent. In fact, we can combine these quantifications since the variables have the same range!

$$\forall x, y \in S, P(x, y)$$

This reads, "every x and y in S follows $P(x, y)$."

The same follows for consecutive existential quantifiers:

$$\exists x \in S_1, \exists y \in S_2, P(x, y) \qquad \exists y \in S_2, \exists x \in S_1, P(x, y)$$

This can be summed up as $\exists x, y \in S, P(x, y)$, read as "there is at least one pair of elements x and y that satisfy $P(x, y)$."

This is *not* the case for alternating quantifiers however.

Example

$$\forall a \in A, \exists b \in B, Likes(a, b)$$

"For every person a in A, there exists a person b in B, that a likes b."

$$\exists b \in B, \forall a \in A, Likes(a, b)$$

"There exists a person b in B, where for every person a in A, a likes b"

In both these cases, the first variable is *independent* of the second variable. Read nested quantifiers from left to right!

Sentences in predicate logic

With quantifiers, propositional operators, and predicates, we can represent statements using **sentences**.

Definiton 1.11. A **sentence** is a formula with no unquantified variable.[12] This ensures that the formula has a fixed truth value.

[12] Sometimes a quantified variable will be refered to as a **bound** variable, and an unquantified variable a **free** variable.

Examples

$\forall x \in \mathbb{N}, x^2 > y$	Not a sentence since y is not bound.
$\forall x, y \in \mathbb{N}, x^2 > y$	Is a sentence since both variables are bound.

Manupulating negation

For any formula, we can state its negation by preceding it by a $\neg$ symbol.

$\forall x \in \mathbb{N}, x \geq 0$ $\qquad$ $\neg(\forall x \in \mathbb{N}, x \geq 0)$

Though, sometimes it is hard to transliterate the formula. Instead, there are *simplification rules*[13]

$\neg(\neg p)$	p
$\neg(p \vee q)$	$(\neg p) \wedge \neg(q)$
$\neg(p \wedge q)$	$(\neg p) \vee (\neg q)$
$\neg(p \Rightarrow q)$	$p \wedge (\neg q)$[14]
$\neg(p \Leftrightarrow q)$	$(p \wedge (\neg q)) \vee ((\neg p) \wedge q)$
$\neg(\exists x \in S, P(x))$	$\forall x \in S, \neg P(x)$
$\neg(\forall x \in S, P(x))$	$\exists x \in S, \neg P(x)$

[13] Note that many of these rules lead into switching from *and* to *or*, and $\forall$ to $\exists$, and vice versa. Try not to memorize these, but to understand them.

[14] As a reminder, $p \Rightarrow q$ is equivalent to $p \wedge \neg q$.

Avoid commas

Commas can cause ambiguity when connecting propositions.

$$P(x), Q(x)$$

Does this mean $P(x)$ *and* $Q(x)$? Or $P(x)$ *then* $Q(x)$?

We must never use commas to **connect propositions**.

Commas have two valid uses only:

- Immediately after variable quantification, or separating two variables with the same quantification
- Separating arguments to a predicate

Example

$$\forall x, y \in \mathbb{N}, \forall x \in \mathbb{R}, P(x, y) \Rightarrow Q(x, y, z)$$

Defining predicates

Definiton 1.12. Let $n, d \in \mathbb{Z}$. We want to say that d divides n, or n is divisible by d, when there exists a $k \in \mathbb{Z}$ such that $n = dk$. So, we will use the notation $d \mid n$ to represent "d divides n." This is a *binary* **divisibility predicate**.

Examples
Let us express the statement "For every integer x, if x divides 10, then it also divides 100" with the divisibility predicate, and without.

Without the predicate:

$$\forall x \in \mathbb{Z}, (\exists k \in \mathbb{Z}, 10 = kx) \Rightarrow (\exists k \in \mathbb{Z}, 100 = kx)$$

"For every integer x, if there exists an integer k such that $kx = 10$, then for another integer k, $kx = 100$."[15]

[15] Note that there are two different k variables, we could also express this using k_1 and k_2.

With the predicate:

$$\forall x \in \mathbb{Z}, x \mid 10 \Rightarrow x \mid 100$$

Much easier, isn't it?

We can use this definiton to formally define prime numbers.

Definition 1.13. Let $p \in \mathbb{N}$. A **prime** number is greater than 1, and the only natural numbers that divide it are 1 and itself. Primes are restricted to being positive.

Example
Let $Prime(p)$ denote that "p is a prime number."

$$Prime(p) : p > 1 \wedge (\forall d \in \mathbb{N}, d \mid p \Rightarrow d = 1 \vee d = p), \quad p \in \mathbb{N}$$

Let us express the property that "there are infinitely many primes."
How do we express *infinitely many*? Since we know that $\mathbb{N}$ is infinite, we can express the statement as "every natural number has a prime number larger than it."

$$\forall n \in \mathbb{N}, \exists p \in \mathbb{N}, p > n \wedge Prime(p)$$

Definition 1.14. Fermat's Last Theorem states that there are no three positive integers a , b, and c that satisfy $a^n + b^n = c^n$ for any integer $n > 2$.[16]

[16] First conjectured by Pierre de Fermat in 1637, he states that the margins of the text *Arithmetica* were too narrow to fit his proof!

Let us express this theorem using predicate logic.

Which of these variables are quantified? n is certainly bound to the range of for all $n > 2$ and n being an integer. a, b, and c are not specifically bound, but since the theorm states *None* of them satisfy the statement, we can say "there does not exist" instead.

$$\forall n \in \mathbb{N}, n > 2 \Rightarrow \neg(\exists a, b, c \in \mathbb{Z}^+, a^n + b^n = c^n)$$

Following negation rules, we can push this negative inwards closer to the predicates.

$$\forall n \in \mathbb{N}, n > 2 \Rightarrow (\exists a, b, c \in \mathbb{Z}^+, a^n + b^n \neq c^n)$$

Formula conventions

Operation precedence in decreasing order:

1. $\neg$
2. $\vee, \wedge$
3. $\Rightarrow, \Leftarrow$
4. $\forall, \exists$

Combinations of operations at the same level *must* be disambiguated using parentheses.

The $\vee$ and $\wedge$ operators are *associative*, meaning that their orders do not matter. But the implicator operator is *not associative*.

Variable naming conventions state that variables should have distinct names within the same formula.

$$(\forall x \in \mathbb{N}, f(x) \geq 5) \vee (\exists x \in \mathbb{N}, f(x) < 5)$$

Although the above is correct since variables only exist in the scope of their parentheses, we still prefer to use the following:

$$(\forall x \in \mathbb{N}, f(x) \geq 5) \vee (\exists y \in \mathbb{N}, f(y) < 5)$$